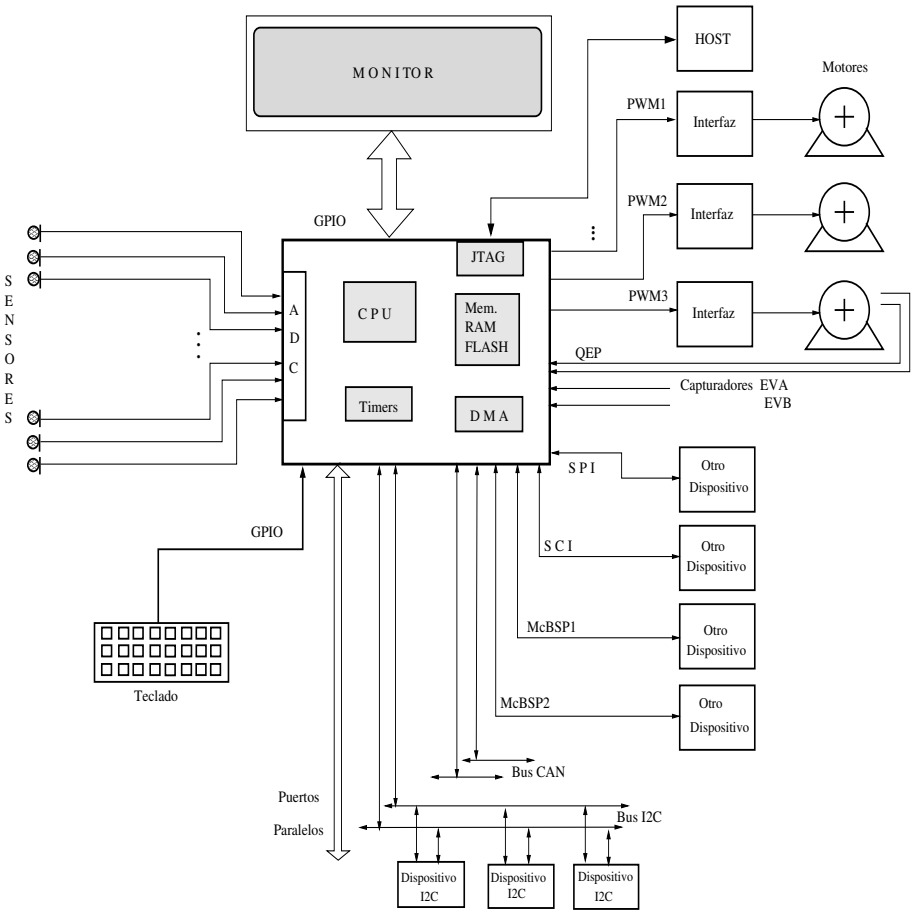
Arquitecturas de DSP TMS320F28xxx y aplicaciones



F.I. UNAM.
Larry Escobar

Índice general

1. Introducción **1**

2. Características generales de la familia F28xx **5**

3. Arquitectura de la familia F28xx **11**
3.1. La convolución y el núcleo de un DSP 11
3.2. Arquitectura general de los DSP C28x 13
3.3. Buses y registros del CPU 13
3.3.1. Descripción de registros 16
3.4. Unidad central de proceso 20
3.4.1. Registros de corrimiento 20
3.4.2. Unidad aritmético lógica (ALU) 22
3.4.3. Multiplicador 22
3.5. Pipeline 23

4. Memoria y modos de direccionamiento **27**
4.1. Mapa de memoria 27
4.1.1. Memoria flash y SARAM 29
4.1.2. Vectores de interrupción 30
4.1.3. Buses internos 30
4.1.4. Seguridad 31
4.1.5. Sintaxis de instrucciones 31
4.2. Modos de direccionamiento 35
4.2.1. Direccionamiento inmediato 35
4.2.2. Direccionamiento directo 36
4.2.3. Modo de direccionamiento por la pila 38
4.2.4. Modo de direccionamiento indirecto 38
4.2.5. Direccionamiento circular 43
4.2.6. Direccionamiento de registros 45
4.2.7. Modo de direccionamiento al espacio de dato, de programa y espacio I/O 46

4.2.8. Modo de direccionamiento byte 47

5. Unidad de control **49**
5.1. Registros de control 49
5.1.1. Contador de programa (PC) 49
5.1.2. Apuntador de pila (SP) 49
5.2. Instrucciones que cambian el flujo del programa 50
5.2.1. Saltos incondicionales 50
5.2.2. Saltos condicionales 51
5.2.3. Saltos dinámicos 51
5.2.4. Instrucciones de llamada a subrutina 54
5.2.5. Llamadas incondicionales a subrutina 54
5.2.6. Llamadas dinámicas a subrutina 55
5.2.7. Retorno de subrutinas 55
5.2.8. Instrucciones de repetición 55
5.3. Registros de estado 58
5.4. Módulo de código de seguridad (CSM) 61
5.5. Instrucciones 61
5.6. Ejemplos en lenguaje ensamblador 65
5.6.1. Suma varias constantes en modo inmediato 65
5.6.2. Suma varios datos en modo directo 66
5.6.3. Suma varios datos en modo indirecto 67
5.6.4. Decimación de una secuencia 69
5.6.5. Modo de direccionamiento de buffer circular 70
5.6.6. Producto punto entre dos vectores 70
5.6.7. Correlación y autocorrelación de señales discretas 74
5.6.8. Operación a 32 bits 75
5.6.9. Aplicaciones propuestas 77

6. Implementación de filtros digitales **79**
6.1. Filtros de respuesta finita al impulso 79
6.1.1. Estructuras de los filtros FIR 79
6.1.2. Implementación de líneas de retardo o buffer lineal 82
6.1.3. Implementación de filtros FIR 83
6.2. Filtros de respuesta infinita al impulso 89
6.2.1. Estructuras de filtros digitales IIR 89
6.2.2. Filtro IIR forma directa 90
6.3. Osciladores digitales 94
6.4. Generación de señales DTMF 97
6.5. Aplicaciones propuestas 103

7. Interrupciones **105**
7.1. Interrupciones del C28x 105
7.1.1. Operación y manejo de interrupciones 106
7.1.2. Interrupciones mascarables 109
7.1.3. Proceso de atención de interrupciones mascarables 110
7.1.4. Interrupciones no mascarables 111
7.1.5. Interrupciones por software 111
7.2. Módulo de expansión de interrupción de periféricos (PIE) 113
7.2.1. El watchdog o perro guardián 120

8. Periféricos **123**
8.1. Periféricos de las familias C28xxx 123
8.2. Entradas y salidas de propósito general 124
8.2.1. Registros de configuración de GPIO 124
8.2.2. Puertos GPIO del DSP Piccolo TMS320F28069 124
8.3. Sistema de reloj 132
8.3.1. Reloj de periféricos 133
8.4. Temporizadores del CPU de 32 bits 135
8.5. Convertidor análogo digital (ADC) 146
8.5.1. Principio de operación del secuenciador de autoconversión 148
8.5.2. Inicialización de registros 149
8.6. Convertidor ADC de la familia Piccolo 151
8.6.1. Principio de operación 153
8.6.2. Configuración 153
8.6.3. Conversión simple por disparo de SOC 154
8.6.4. Modo de muestreo simultáneo 154
8.6.5. Operación de interrupción y EOC 154
8.6.6. Secuencia de encendido de ADC 155
8.6.7. Voltajes convertidos con referencias interna y externa 155
8.6.8. Circuitos electrónicos externos para el ADC y DAC 156
8.7. Aplicaciones propuestas 163
8.8. Módulos manejadores de eventos 163
8.8.1. Temporizadores de propósito general para el módulo EV 165
8.9. Señales moduladas por ancho de pulso 169
8.9.1. Generador de señales PWM 170
8.9.2. Generador simétrico y asimétrico de PWM 171
8.9.3. Unidad de captura 174
8.9.4. Generador programable de banda muerta “dead band” 176
8.9.5. Módulo codificador en cuadratura de pulso QEP 178

9. Puertos seriales **183**
9.1. Interfaz de comunicación serial . . . 183
9.1.1. Características . . . 184
9.1.2. Comunicación por la interfaz SCI . . . 185
9.1.3. Interrupción a la interfaz SCI . . . 188
9.1.4. Comunicación multiprocesos por SCI . . . 189
9.2. Interfaz de puerto serie (SPI) . . . 191
9.2.1. Registros de SPI . . . 192
9.2.2. Operación de la interfaz SPI . . . 195
9.2.3. Interrupciones de SPI . . . 196
9.3. Controlador de red de área . . . 196
9.3.1. Módulo eCAN . . . 197
9.3.2. Funcionamiento del controlador eCAN . . . 198
9.4. Controlador I2C . . . 203
9.4.1. Funcionamiento . . . 205
9.4.2. Condiciones START y STOP . . . 206
9.4.3. Transmisión en formato libre . . . 207
9.4.4. Requerimiento de interrupciones . . . 207

10. Puerto serie multicanal buffereado **209**
10.1. Generalidades . . . 209
10.2. Compresión logarítmica . . . 210
10.3. Proceso de recepción de un puerto serie síncrono convencional . . . 212
10.3.1. Transmisión . . . 212
10.4. Puerto serie buffereado . . . 213
10.5. Puerto serie multiplexado por división de tiempo (TDM) . . . 214
10.6. Puerto serie multicanal buffereado (McBSP) . . . 215
10.6.1. Registros . . . 215
10.6.2. Configuración de puertos McBSP . . . 218
10.6.3. Registros de control de transmisión y recepción RCR(1,2) y XCR(1,2) 223
10.6.4. Registros generadores de razón de muestreo SRGR(1,2) . . . 227
10.6.5. Operación multicanal . . . 229

11.Transferencia por DMA **237**
11.1. Generalidades de operación del DMA . . . 237
11.2. Ejemplos de aplicación de DMA . . . 242
11.3. Módulo DMA del DSP C28x . . . 244
11.3.1. Descripción de registros de DMA . . . 245
11.3.2. Modos de transferencia . . . 247
11.3.3. Tabla de registros . . . 247

A. Formatos numéricos **251**
A.1. Errores numéricos 251
A.2. Formatos de punto fijo 252
A.2.1. Representación de números fraccionarios en punto fijo 253
A.2.2. Formatos numéricos de punto fijo Qi 254
A.2.3. Intervalos dinámicos y precisión numérica 256
A.2.4. Asignación de variables 258
A.2.5. Operación suma en punto fijo 259
A.2.6. Operación de multiplicación en punto fijo 260
A.3. Formato numérico de punto flotante 262
A.3.1. Suma de punto flotante 264
A.3.2. Multiplicación de punto flotante 264
A.3.3. División de punto flotante 265
A.4. Formatos estándares de punto flotante 265
A.4.1. Formato de punto flotante estándar IEEE 754 265
A.5. Formato de punto flotante de Microsoft 268
A.6. Otros formatos 268

B. Glosario **271**

Bibliografía **276**

Capítulo 1

Introducción

En la actualidad, las aplicaciones del procesamiento digital de señales (PDS) han ido avanzando a pasos agigantados y las podemos encontrar vinculadas a una gran cantidad de actividades de la vida moderna. Entre algunas están: las aplicaciones multimedia, computadoras personales, teléfonos celulares, comunicaciones vía internet, aplicaciones telefónicas, aplicaciones biomédicas, instrumentación, tabletas digitales, video juegos, televisión digital, etc. Por estas razones, un ingeniero que se dedique al área del PDS debe estar en constante actualización en algoritmos, nuevas tecnologías, aplicaciones, nuevos productos, entre otros.

El *objetivo* de este libro es de carácter académico ya que fomenta el aprendizaje de las familias de procesadores de señales digitales (DSP) TMS320C28xxx, con el fin de que los estudiantes de licenciatura y posgrado adquieran un conocimiento más rápido de estas arquitecturas y puedan en un futuro cercano desarrollar proyectos y aplicaciones en esta área.

Todo el material incluido en este trabajo es la recopilación, condensación, análisis y estructuración de la información de estos DSP que se puede encontrar en más de 30 manuales de Texas Instruments^{TM} (TI), algunos libros y artículos citados en la bibliografía, así como las metodologías empleadas por el autor en su labor académica.

La compañía TI, diseñadora y productora de circuitos integrados, se ha convertido en una empresa líder en la fabricación de DSP y aplicaciones en tiempo real. Esta compañía clasifica sus dispositivos en una serie de grupos denominados familias, cada una de ellas posee diferentes particularidades que los caracteriza. En las últimas décadas, TI ha ido orientando las familias de sus arquitecturas a tres grandes campos de aplicación de ingeniería, las cuales son:

- *Control e instrumentación:* familias TMS320C2xx, TMS320C28xxx y TMS320F28xxx de punto entero y algunas de punto flotante.

- *Comunicaciones, entretenimiento y aplicaciones de bajo consumo:* familias TMS320C5x, TMS320C54x, TMS320C54xx y TMS320C55xx de punto entero.

- *Alto desempeño y video:* familias TMS320C64xx de punto entero y TMS320C67xx de punto flotante.

La familia TMS320C28x (C28x) con tecnología CMOS y TMS320F28x (F28x) con memoria flash que se aborda en este trabajo, contiene una arquitectura interna de un procesador digital que integra un gran número de periféricos y memoria, es decir, que conjunta la potencialidad de un DSP con las prestaciones de un microcontrolador. Estas características lo convierten en una plataforma de alto desempeño en el diseño de aplicaciones de control de motores, instrumentación, soluciones embebidas en un sólo chip, manejo de potencia, comunicaciones y procesamiento digital de señales.

Los DSP C28x y F28x (genéricamente los llamaremos C28x) son arquitecturas de aritmética de 16 y 32 bits en punto entero y algunas de punto flotante, diseñados especialmente para aplicaciones de control de alto desempeño; tales como robótica, automatización industrial, dispositivos de almacenamiento masivo, control de iluminación, instrumentación, fuentes de potencia y otras aplicaciones que necesitan de un procesador simple que resuelva problemas complejos.

En la figura 1.1 se muestra un sistema embebido que integra un conjunto de periféricos externos, adquisición de varias señales, comunicaciones con periféricos y otros dispositivos, salidas, control de procesos, control de motores, etc., que son manejados por un sistema central en tiempo real, un dispositivo capaz de realizar todas estas acciones puede ser un DSP C28x. En la actualidad se requieren soluciones de esta naturaleza que involucre una gran cantidad de opciones en un sólo sistema, la figura 1.1 nos da una idea de la potencialidad del dispositivo a tratar.

Este trabajo no sólo se enfoca al estudio del hardware de las arquitecturas C28x, sino que considera aspectos teóricos del PDS donde se enfatiza la importancia de los DSP, además, se presentan muchos ejemplos para abundar en el software. Los ejemplos están elaborados en lenguaje ensamblador para profundizar en el manejo del hardware; sin embargo, estos dispositivos permiten programarse en lenguaje C y C++. El autor considera que cuando un diseñador tiene un conocimiento profundo del hardware, la migración a la programación de los DSP en lenguaje C es más fácil, por tanto se pueden optimizar las aplicaciones utilizando lenguaje ensamblador, lenguaje C o combinando ambos.

Los DSP de la familia C5x son dispositivos con aritmética de punto entero a 16 bits cuya versión más rápida alcanzó los 80 millones de instrucciones por segundo (MIPS), los DSP C54xx también son de punto fijo a 16 bits con bajo consumo de potencia con un desempeño de hasta 200 MIPS, mientras que el C6416 es un DSP tipo RISC con diseño tipo VLWI (very large word instruction), con aritmética de punto fijo a 32 bits que en la actualidad alcanzan desempeños mayores a los 8,000 MIPS. Además, TI también ha desarrollado arquitecturas llamadas plataformas abiertas para multimedia (OMAP) que integran DSP de la familia C55xx con arquitecturas ARM (Advanced RISC Machine), OMAP con TMS320C6748 y ARM, y arquitecturas muy específicas para video llamadas Davinci como el TMS320DM6467 que constan de un TMS320C6416 con un coprocesador de video integrado.

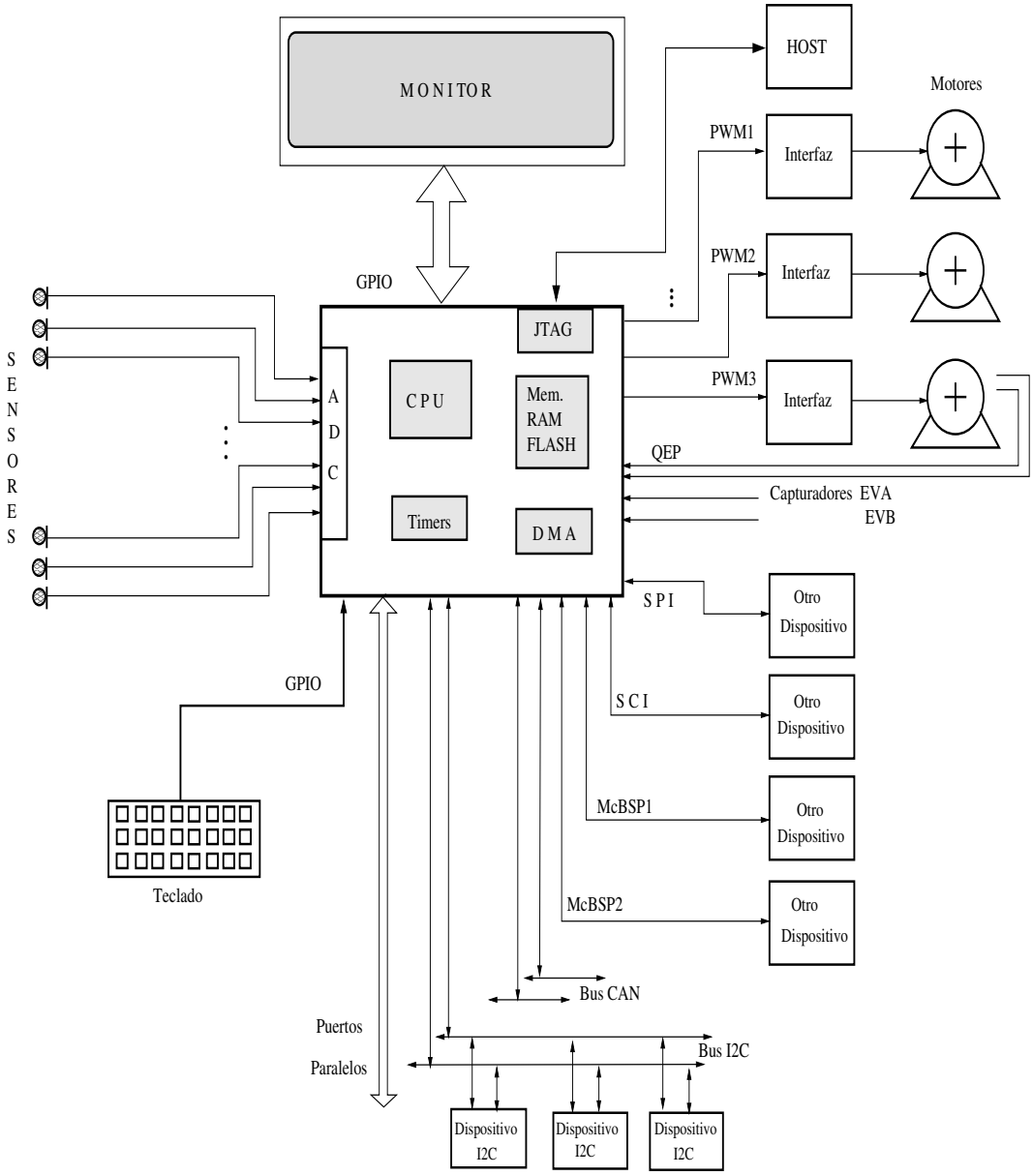


Figura 1.1. Sistema embebido utilizando un DSP C28x

Resumen de capítulos:
En el *capítulo dos*, se exponen las características generales más importantes de las familias C28x y F28x, partiendo desde las familias básicas hasta las de punto flotante. En el *capítulo tres*, se explican los bloques principales de la arquitectura del C28x, con el fin de ir conociendo sus partes y luego en los siguientes capítulos hacer uso de éstos. En el *capítulo cuatro*, nos adentramos en el conocimiento de la memoria, sus bloques principales y los modos de direccionamiento que se utilizan para la transferencia de datos en la arquitectura. Se presentan ejemplos en lenguaje ensamblador. En el *capítulo cinco*, el control de la máquina, se estudian los registros de control y estado, se muestran las instrucciones relacionadas con el cambio de flujo de un programa y se estructuran ejemplos de aplicaciones. En el *capítulo seis*, se presenta una breve introducción a los filtros digitales con varios ejemplos. En el *capítulo siete*, se analiza la forma de interactuar del CPU a través de interrupciones y se estudia la interfaz de expansión de interrupciones. En el *capítulo ocho*, se introduce a los periféricos y su interfaz con que interactúa el DSP. En el *capítulo nueve*, se presentan los diferentes puertos seriales que contiene esta familia de DSP. En el *capítulo diez*, el puerto multicanal serial (McBSP), un poderoso puerto serial que opera en varios modos y puede manejar hasta 128 canales de 32 bits en combinación con el DMA. En el *capítulo once*, la transferencia por acceso directo a memoria (DMA) como una técnica de procesamiento en paralelo muy poderosa en cuanto al manejo de información. Adicionalmente, se agrega un apéndice de formatos numéricos como una parte importante de las aplicaciones del PDS y los DSP, ya que es necesario conocer cómo se manejan los números en forma binaria.

El autor reconoce la valiosa colaboración que ha tenido el programa universitario de Texas Instruments en la donación de tarjetas al Laboratorio de Procesamiento Digital de Señales, esto ha contribuido a la enseñanza y aprendizaje de los DSP en la Facultad. También agradece a la Unidad de Apoyo Editorial de la Facultad de Ingeniería por su revisión y apoyo en la publicación de esta obra, así como a los ingenieros Agustín Silva y Samuel Vázquez por sus sugerencias y pruebas realizadas con estos DSP.

Larry Escobar
Profesor de la Facultad de Ingeniería, UNAM.
Ciudad Universitaria, México D. F., marzo de 2014.
e-mail: larryesc@gmail.com

Capítulo 2

Características generales de la familia F28xx

Como es sabido por los ingenieros, cuando se realiza un diseño, antes de llevarlo a su implementación es necesario conocer las características de los dispositivos que cumplan con las especificaciones, por tanto, en este capítulo se pretende dar a conocer las características generales de los DSP de la familia TMS320C2xxx C28x que genéricamente se le conoce como C28x, donde la letra C se refiere a la tecnología original CMOS, sin embargo, si hacemos referencia a las familias que contienen memoria flash se les nombra F28x, las familias con cuatro dígitos C28xx o F28xx y con cinco dígitos F28xxx

En este capítulo se abordan las características generales de los DSP familias F28x y F28xxx, se describe la arquitectura, sus rasgos más relevantes como su unidad de proceso, memoria, buses y periféricos. Se resaltan en particular las familias F280x y F281x, porque en este libro se desarrollarán ejemplos y prácticas con los DSP F2808, F2812 y eventualmente con el DSP Piccolo F28027 y F28069.

La generación de DSP C28x son un conjunto de dispositivos de la familia de DSP TMS320C2000 o también llamados controladores de señales digitales (DSC) de Texas Instruments (TI) con arquitectura Harvard modificada [18]. Estos dispositivos son de alta integración para aplicaciones de alto desempeño en control en instrumentación, en general, se puede decir que estas familias están constituidas por un DSP con un microcontrolador con características CISC (conjunto amplio de instrucciones) y ejecutan la mayoría de instrucciones en un ciclo de reloj. La unidad central de proceso (CPU) del C28x es muy eficiente en cuanto al uso de lenguajes como C y C++ que permite desarrollos en aplicaciones matemáticas en el procesamiento digital de señales (PDS), así como aplicaciones típicas de microcontroladores en sistemas de control.

En la figura 2.1 se ilustran los bloques principales de estas arquitecturas, donde podemos observar bloques como la unidad central de proceso CPU, la memoria, buses de datos y de programa, periféricos, un bloque manejador de interrupciones (PIE) y una interfaz de depuración y emulación JTAG (joint test action group) en tiempo real, todas estas partes se

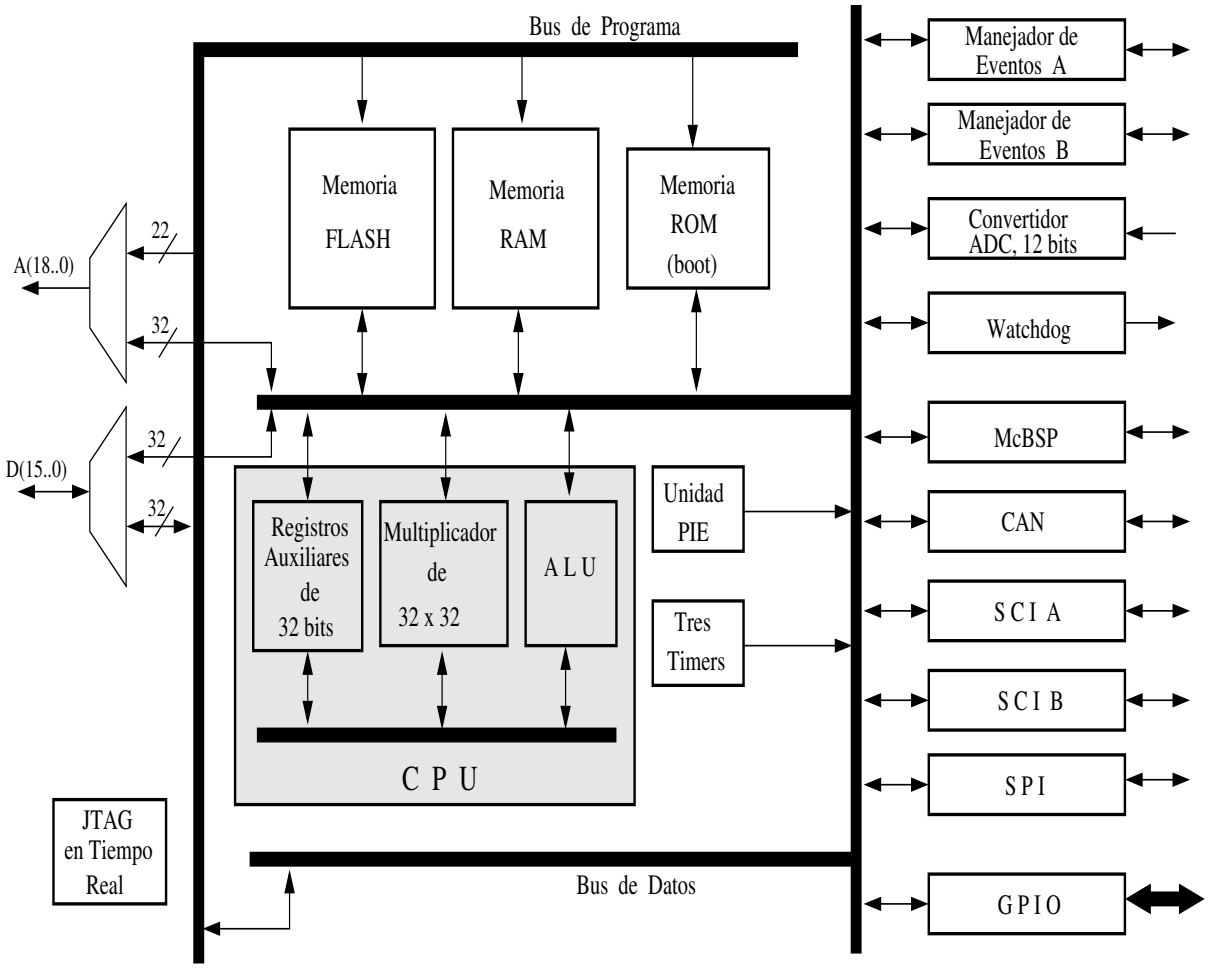


Figura 2.1. Arquitectura general de los DSP C28x

estudiarán en los capítulos siguientes.

La generación TMS320C28x surge como evolución de la familia TMS320C24x de 16 bits, es una versión ampliada, mejorada y compatible con esta familia. Posee un CPU de 32 bits de punto fijo, con una capacidad de hasta los 150 MIPS y voltajes de alimentación de 1.9 V para el CPU y 3.3 V para los periféricos. En versiones avanzadas existen arquitecturas de punto flotante con el estándar 754 de IEEE.

El CPU en sí es un procesador digital de señales de 32 bits para la realización de las operaciones aritméticas y lógicas con **características generales:**

- Arquitectura tipo Harvard modificada.
- Ejecuta instrucciones de 32 bits para mejorar la precisión numérica.
- Ejecuta instrucciones de 16 bits para mejorar la eficiencia en el código.
- Unidad aritmética lógica (ALU) de 32 bits.
- Unidad aritmética de registros auxiliares (ARAU), genera direcciones de memoria dato, realiza aritmética entre apuntadores en paralelo con operaciones de la ALU.
- Registro de corrimiento, ejecuta corrimientos hacia la derecha o izquierda de hasta 16 bits.
- Ejecuta multiplicaciones de 32 x 32 bits con resultado de 64 bits.
- Efectúa una operación multiplicación acumulación (MAC) de 32 x 32 bits en un ciclo de reloj.
- Efectúa dos operaciones MAC de 16 x 16 bits (DMAC) en un ciclo de reloj.
- Emulación de su funcionamiento en tiempo real.
- Protección de código.
- En un ciclo de instrucción puede ejecutar instrucciones que leen, modifican y escriben en memoria.
- Respuesta de interrupciones rápida con salvado automático del contexto.
- Sincronía de eventos con latencia mínima.
- Pipeline de 8 niveles, que permiten un solapamiento máximo de 8 instrucciones en niveles de ejecución diferentes:
  - Búsqueda de instrucción: F1 y F2.
  - Decodificación: D1 y D2.
  - Lectura de operandos: R1 y R2.
  - Ejecución: X.
  - Escritura: W.

Periféricos:
• 16 convertidores A/D a 12 bits.
• Memoria flash interna.
• Interfaz JTAG estándar IEEE 1194.1 1990 para emulación y depuración en tiempo real.
• Canales para interfaz de comunicación serial (SCI).
• Módulos para interfaz de puerto serial UART (SPI).
• Módulos para comunicación por buses I2C.
• Módulo de control de red de área (CAN), que es un protocolo de comunicación industrial para control distribuido en tiempo real.
• Bloque de expansión de periféricos (PIE).
• Módulos de codificación en cuadratura (QEP).

Para versiones avanzadas
• Módulo McBSP (multichannel buffered serial port). Es un puerto serie síncrono multicanal que tiene como características principales:
- Hasta 128 canales
- Comunicación full-duplex
- Relojes y tramas independientes programables para transmisión y recepción con dos registros FIFOs de 32 bits y 16 niveles.
• Seis canales de acceso directo a memoria (DMA).
• Unidad aceleradora de punto flotante a 32 bits.
• Unidad de punto flotante (FPU) con precisión simple IEEE-754.
• Unidad de matemáticas complejas y el algoritmo Viterbi (VCU).

Familias F2823x y F2833x

Las familias de DSP F28232, F28234, F28235, F28332, F28334 y F28335 están basadas en la arquitectura del DSP C28x, con mejoras que agregan más potencialidad en sus periféricos y diferencias en cuanto al mapa de memoria; la familia F2833x incluye una unidad de punto flotante (FPU) mientras que la familia F2823x sigue operando en punto entero y son familias que agregan transferencia por acceso directo a memoria (DMA).

La familia F2833x es un DSP C28x con una unidad FPU (C28x + FPU), está basada en controladores con arquitectura de punto entero a 32 bits, la unidad FPU efectúa operaciones de punto flotante en precisión simple utilizando el estándar 754 IEEE. Esto permite la implementación eficiente de algoritmos de control y PDS a mayor precisión numérica, prescindiendo de un segundo procesador, estas familias son las más avanzadas de los DSP C28x.

Las arquitecturas F2833x, siguen manteniendo la base de las arquitecturas C28x, pero

aumenta en gran medida su potencialidad, ya que se agrega una unidad de punto flotante (FPU) de 32 bits con el estándar IEEE 754, ocho registros de 32 bits para operaciones de FPU, aumenta la cantidad de memoria flash y memoria OTP, agrega puertos seriales multicanal buffereados y seis canales de DMA entre otros. En la tabla 2.1 se resumen y comparan las características más importantes entre algunas familias C28x.

En la actualidad las familias más avanzadas son la C2834x ejecutan 300 MIPS para punto entero y 600 MFLOPS para punto flotante.

Tabla 2.1. Comparación de algunos DSP de la familia C28x

Característica \ **DSP**	**F2808**	**F2812**	**F28027**	**F28069**	**F28235**	**F28335**
CLK (Mhz)	100	150	60	80	150	150
RAM (Kw)	18	18	6	50	36	36
FLASH (Kw)	64	128	36	128	256	256
PWM	16	16	9	16	18	18
HRPWM	4	–	4	8	6	6
QEP	2	2	0	2	2	2
EV	4	6	1	3	6	6
Timers	14	8	9	16	16	16
I2C	1	–	1	1	1	1
SCI UART	2	2	1	2	3	3
SPI	4	1	1	2	1	1
CAN	2	1	–	1	2	2
GPIO	35	56	22	54	88	88
McBSP	–	1	–	1	2	2
DMA	–	–	–	6	6	6
Canales ADC	16	16	16	16	16	16
ADC Tc (ns)	160	80	217	180	80	80
Voltaje de CPU	1.8	1.9	3.3	3.3	1.9	1.9

Resumen

En este capítulo se han presentado las características más sobresalientes de las familias de procesadores C28x, en algunas familias se ha profundizado este aspecto, ya que se pretende que más adelante se realicen ejemplos y prácticas sobre éstas en particular. Se ha introducido un poco a la arquitectura y se irá describiendo y avanzando posteriormente.

Capítulo 3

Arquitectura de la familia F28xx

Las familias de DSP F28x están diseñadas con base en la arquitectura Harvard mejorada, con múltiples buses, estas familias contienen un procesador digital que integra un gran número de periféricos y memoria, es decir, que conjunta la potencialidad de un DSP con las prestaciones de un microcontrolador. La arquitectura del C28x es una arquitectura multibuses, que contiene un bus de direcciones de programa de 22 bits, un bus para lectura de datos y otro para escritura de datos de 32 bits de dirección [18].

En este capítulo se empieza a analizar y estudiar la arquitectura de los DSP utilizando una descripción general a nivel de bloques funcionales y posteriormente se va profundizando en la familia C28x, se enfatiza en la operación convolución de donde se parte para el diseño del núcleo de un DSP. Debido a la cantidad de información en los capítulos siguientes, se continúa con la descripción de otros bloques.

3.1. La convolución y el núcleo de un DSP

La operación convolución entre señales discretas está compuesta de sumas, productos y retardos, y se le puede considerar la operación básica o fundamental del PDS, la cual se encuentra en muchos de sus algoritmos, además los procesadores de señales digitales se han diseñado con el propósito de realizar convoluciones eficientemente en tiempo real [3].
Para un sistema lineal e invariante en el tiempo discreto (SLITD) con entrada $x(n)$, respuesta al impulso $h(n)$, la salida $y(n)$ se puede calcular por la ecuación (3.1)

$$y(n) = \sum_{i=-\infty}^{\infty} h(i)x(n-i) \tag{3.1}$$

si el sistema es causal y $h(n)$ es de longitud finita N, entonces la salida se calcula con

$$y(n) = \sum_{i=0}^{N-1} h(i)x(n-i) = \sum_{i=0}^{N-1} x(i)h(n-i) \tag{3.2}$$

De la ecuación (3.2) se observa que la convolución es conmutativa. Por tanto, para realizar la convolución entre dos secuencias discretas $x(n)$ y $h(n)$ se puede resumir en los siguientes pasos [3]:

1. Seleccionar una secuencia fija y otra móvil. Como ejemplo seleccionamos $x(n)$ fija y $h(n)$ móvil.

2. Se hace girar $h(n)$ sobre el eje $n = 0$ obteniendo $h(-n)$, es decir, efectuar la operación "folding" sobre $h(n)$.

3. Se corre $h(-n)$ k unidades a la derecha, entonces se obtiene $h(-n-k)$, $k = 0, 1, 2, 3, ...N-1$, N longitud de $h(n)$.

4. Se multiplican las muestras que se solapan de $x(n)$ y $h(-n-k)$.

5. Se realiza la suma de los productos del inciso anterior para obtener la salida actual $y(n)$ al tiempo n.

6. El proceso se repite a partir del punto 3.

De la ecuación 3.2 se observa que a nivel de hardware se requieren los elementos o bloques:
- N localidades de memoria ROM o RAM para alojar $h(n)$.
- N localidades de memoria RAM para $x(n)$ y sus $N-1$ retardos.
- Un multiplicador por hardware.
- Un sumador acumulador.
- Periféricos de entrada para $x(n)$ y salida para $y(n)$.
- Buses para datos, direcciones y control.
- Unidad de control para sincronizar y controlar todo el proceso.
- Unidad de direcciones para transferir adecuadamente los operandos.

Todos estos elementos y su forma de interactuar los podemos observar en forma general en la figura 3.1 donde se muestra el núcleo de un DSP, debido a que es la parte donde se ejecuta la operación convolución. Por cada muestra de salida $y(n)$ se efectúan N multiplicaciones de LxL bits, $N-1$ sumas y el resultado se entrega a un periférico de salida. Todas las muestras de $x(n)$ en memoria se deben desplazar una localidad, es decir, realizar un retardo sobre cada muestra, en seguida se lee de un periférico la muestra actual de $x(n)$, el proceso continúa indefinidamente cuando se opera en tiempo real.

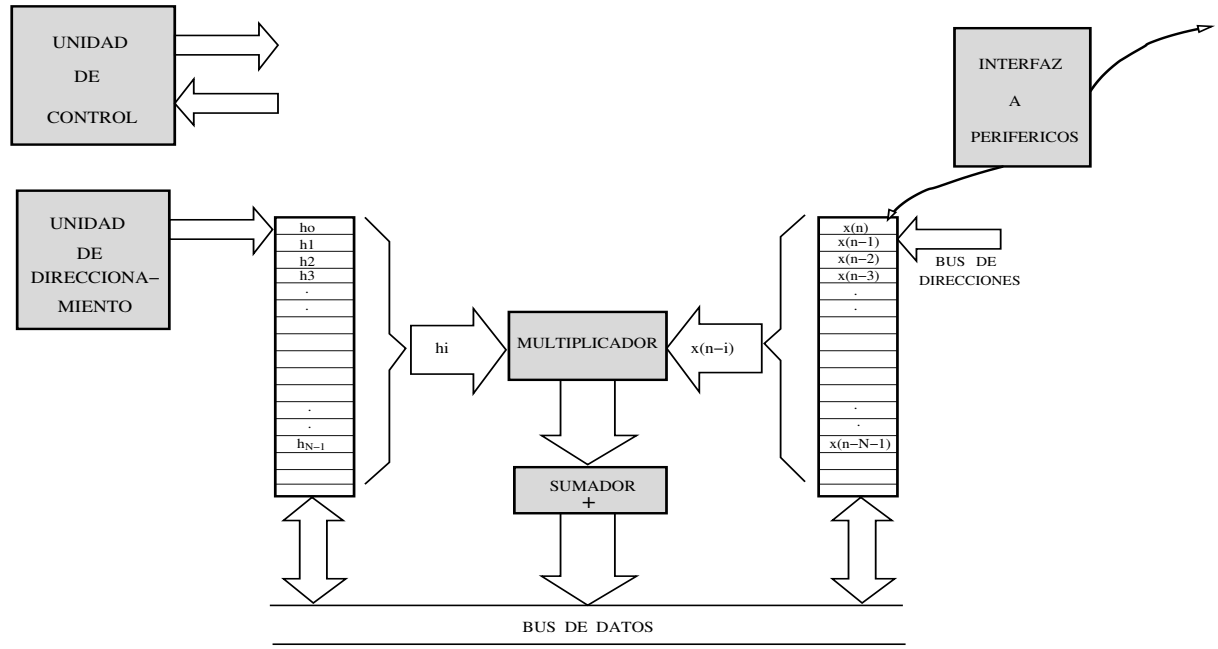


Figura 3.1. Núcleo de un DSP

3.2. Arquitectura general de los DSP C28x

Considerando los elementos del núcleo de un DSP de la figura 3.1, en la figura 3.2 podemos observar la arquitectura ampliada de la familia F28xx que para su análisis y estudio se han estructurado de la siguiente forma:

- Registros
- Memoria y modos de direccionamiento
- Unidad central de proceso
- Unidad de control
- Periféricos

En este capítulo se analizarán las primeras dos partes y en capítulos subsecuentes se irán integrando las demás.

3.3. Buses y registros del CPU

Como muchos dispositivos DSP, se hace uso de múltiples buses para mover datos entre la memoria, los periféricos y el CPU. La arquitectura del bus de memoria contiene:

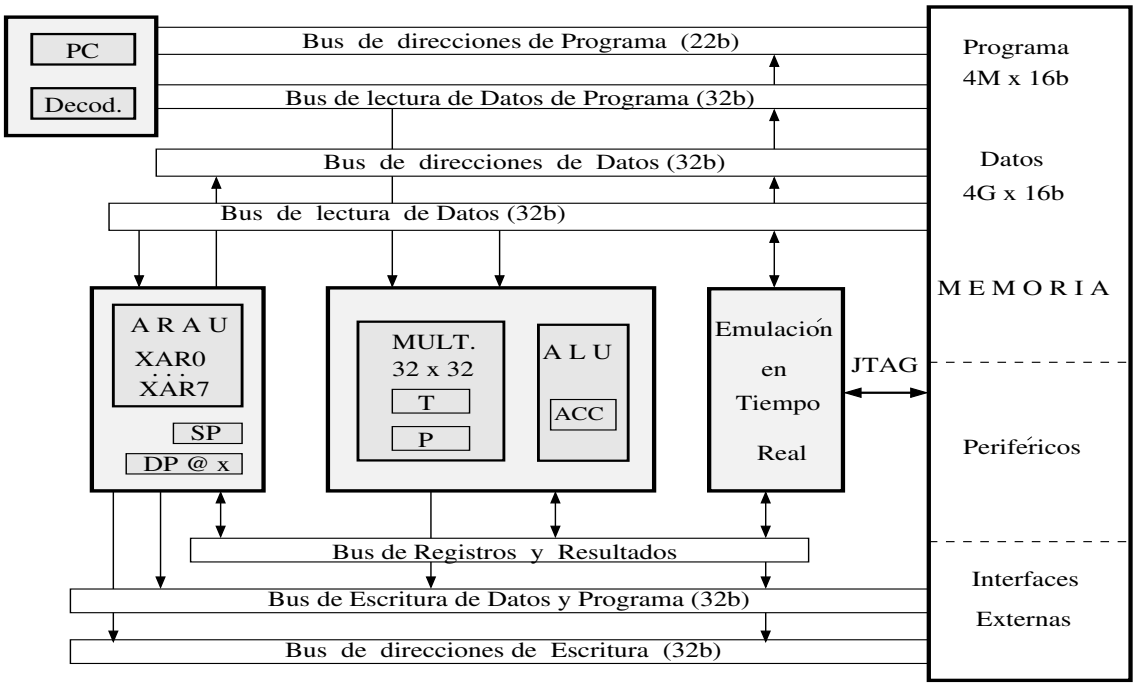


Figura 3.2. Arquitectura general y buses de los DSP C28x

- Un bus de lectura de programa (direcciones de 22 bits y datos de 32 bits)
- Un bus de lectura de datos (direcciones de 32 bits y datos de 32 bits)
- Un bus de escritura de datos (direcciones de 32 bits y datos de 32 bits)

Es decir, que puede direccionar hasta 4 millones de palabras (Mw) de programa y 4 mil millones de palabras (Gw) de datos (w="word"=16 bits).

Los buses de datos de 32 bits de ancho permiten operaciones de 32 bits en un ciclo. Esta arquitectura de buses múltiples es conocida como arquitectura Harvard modificada que permite buscar, leer un valor dato y escribir un valor de dato en un solo ciclo. Todos los periféricos y memorias están conectados al bus de memoria y dará prioridad al acceso directo a memoria.

Estos buses se pueden apreciar en la arquitectura general de la figura 3.2, donde se observa que los buses de direcciones conectan la unidad ARAU con la memoria, los buses de programa conectan el PC y el decodificador con la memoria, los buses de datos conectan la memoria con las unidades de ejecución.

Como toda máquina digital, los DSP C28x utilizan una gran cantidad de registros para su operación, en la tabla 3.1 se resumen los registros del CPU y su valor en el reset, "h" significa valor numérico en hexadecimal.

Tabla 3.1. Registros del CPU del C28x

Nombre	**Descripción**	**Valor en reset (h)**
ACC	Acumulador de 32b	0000 0000
AH	Parte alta del ACC, 16b	0000
AL	Parte baja del ACC, 16b	0000
XAR0	Registro auxiliar 0 de 32b	0000 0000
XAR1	Registro auxiliar 1 de 32b	0000 0000
XAR2	Registro auxiliar 2 de 32b	0000 0000
XAR3	Registro auxiliar 3 de 32b	0000 0000
XAR4	Registro auxiliar 4 de 32b	0000 0000
XAR5	Registro auxiliar 5 de 32b	0000 0000
XAR6	Registro auxiliar 6 de 32b	0000 0000
XAR7	Registro auxiliar 7 de 32b	0000 0000
AR0	Parte baja de XAR0, 16b	0000
AR1	Parte baja de XAR1, 16b	0000
AR2	Parte baja de XAR2, 16b	0000
AR3	Parte baja de XAR3, 16b	0000
AR4	Parte baja de XAR4, 16b	0000
AR5	Parte baja de XAR5, 16b	0000
AR6	Parte baja de XAR6, 16b	0000
AR7	Parte baja de XAR7, 16b	0000
DP	Apuntador de página de 16 bits	0000
IFR	Registro de banderas de interrupción de 16b	0000
IER	Registro habilitador de interrupciones de 16b (INT1 a INT14, DLOGINT, RTOSINT deshabilitados)	0000
DBGIER	Habilitador de depuración de interrupciones de 16b (INT1 a INT14, DLOGINT, RTOSINT deshabilitados)	0000
P	Registro producto de 32 bits	0000 0000
PH	Parte alta de P, 16b	0000
PL	Parte baja de P, 16b	0000
PC	Contador de programa, 22b	3F FFC0
RPC	Retorno del contador de programa 22b	0000 0000
SP	Apuntador de pila de 16b	0400
ST0	Registro de estado 0, 16b	0000
ST1	Registro de estado 1, 16b	080B
XT	Registro multiplicando de 32b	0000 0000
T	Parte alta de XT, 16b	0000
TL	Parte baja de XT, 16b	0000

3.3.1. Descripción de registros

- **Acumulador (ACC)**
  El acumulador es un registro de 32 bits y es el registro principal de trabajo, siempre funciona como un operando fuente de todas las operaciones de la ALU y a la vez es el operando destino de toda operación ALU, también puede aceptar el resultado de 32 bits del multiplicador. Puede accederse por bloques de palabras o bytes:
  - Parte alta (AH): bits 31 al 16
  - Parte baja (AL): bits 15 al 0
  - Byte alto de AH (AH.MSB): bits 31 al 24
  - Byte bajo de AH (AH.LSB): bits 23 al 16
  - Byte alto de AL (AL.MSB): bits 15 al 8
  - Byte bajo de AH (AL.LSB): bits 7 al 0

  Este tipo de particionamiento facilita la manipulación, empaquetado y desempaquetado de bytes.

- **Registro multiplicando (XT)**
  El registro multiplicando es utilizado principalmente para almacenar un operando de 32 bits y para efectuar una multiplicación en la unidad de multiplicación. Además, los cinco bits menos significativos 4 a 0 se utilizan para corrimientos de ACC a la izquierda y del 3 al 0 para corrimientos a la derecha utilizando el registro XT.
  Este registro se puede acceder a su parte alta y baja como:
  T = XT(16..31)
  TL = XT(15..0)

- **Registro producto (P)**
  Es el registro que almacena el resultado de una operación multiplicación. También puede ser cargado desde memoria con un dato de 16 o 32 bits, una constante de 16 bits, o un registro del CPU de 16 o 32 bits. Este registro se puede acceder a su parte alta y baja como:
  PH = P(31..16)
  PL = P(15..0)
  Cuando alguna instrucción accede a P, PH o PL, la operación es afectada por un bloque de corrimiento determinado por los bits de modo producto (PM) del registro de estado ST0. Los modos de corrimiento de la salida del multiplicador son determinados por el valor de PM y se resumen en la tabla 3.2

- **Apuntador de página (DP)**
  En modo de direccionamiento directo, la memoria de datos es manejada como bloques de 64 mil palabras (Kw) llamados páginas. El registro DP mantiene o apunta a la página de datos que se está utilizando, en este modo sólo se pueden acceder a las primeras 65,536 páginas en las 4 Mw bajas del espacio de memoria.

Tabla 3.2. Modo de corrimiento del multiplicador

Instrucción	PM	Corrimiento
SPM +1	000	Un bit a la izquierda
SPM 0	001	Sin corrimiento
SPM -1	010	Un bit derecha
SPM -2	011	Dos bits derecha
SPM -3	100	Tres bits derecha
SPM -4	101	Cuatro bits derecha (bit AMODE = 0)
SPM +4		Cuatro bits izquierda (bit AMODE = 1)
SPM -5	110	Cinco bits derecha
SPM -6	111	Seis bits derecha

- **Apuntador de pila (SP)**
  El apuntador de pila permite utilizar en una pila de memoria datos por software. El registro SP es de 16 bits, por lo que sólo se pueden direccionar 64 Kw del espacio de datos, cuando se utiliza para direccionar datos la parte alta de una dirección es forzada a cero. La operación de la pila es de la siguiente forma:
  - La pila crece de la parte baja de la memoria a la alta.
  - El SP siempre apunta a la próxima localización vacía en la pila.
  - En el reset el SP es inicializado con la dirección 0000 0400h.
  - Cuando se salva algún valor de 32 bits en la pila, la palabra menos significativa se salva primero y en la siguiente localidad la palabra más significativa (formato "little endian").
  - Cuando se lee o escribe un valor de 32 bits por el SP, accede la palabra que apunta el SP y la palabra anterior.
  - Cuando ocurre un sobreflujo del SP más allá de FFFFh o 0000h, el SP opera en forma circular.

- **Registros auxiliares: XAR0 a XAR7 y AR0 a AR7**
  El DSP C28x contiene ocho registros de 32 bits que pueden utilizarse como registros apuntadores a memoria o de propósito general, éstos son llamados registros auxiliares XARi, $i = 0, 1, 2, 3, 4, 5, 6, 7$ y son utilizados con muchas instrucciones en modo de direccionamiento indirecto. La parte baja de estos registros son referidos como AR0-AR7 y pueden utilizarse como registros de propósito general para control de ciclos y comparaciones de 16 bits.

- **Contador de programa (PC)**
  El registro contador de programa es de 22 bits y apunta a la instrucción en memoria de programa a ser buscada. Cuando el pipeline está lleno, el PC apunta a la instrucción

que está siendo procesada y esta instrucción se encuentra en la segunda etapa de decodificación del pipeline.

- **Retorno del contador de programa (RPC)**
  Este registro se utiliza exclusivamente con las instrucciones LCR y LRETR. Cuando se ejecuta una instrucción de llamado a subrutina utilizando la instrucción LCR, la dirección de retorno del PC es salvada en el registro RPC y el valor anterior del RPC es salvado en la pila en dos localidades. Cuando se ejecuta la operación de retorno vía la instrucción LRETR, la dirección de retorno es leída del registro RPC y lo que contenía la pila en dos palabras consecutivas es escrito al registro RPC.

- **Registros de estado ST0 y ST1**
  El C28x contiene dos registros de estado, los cuales guardan banderas y bits de control. Estos registros pueden ser salvados y cargados desde memoria para habilitar configuraciones de operación de la máquina. Los bits del registro ST0 son modificados en la etapa de ejecución de pipeline y los bits del registro ST1 son modificados en la segunda etapa de decodificación del pipeline. Estos registros se utilizan para configurar modos de operación y estados de la máquina, sobre todo el resultado de las operaciones de la ALU.

- **Registro de estado ST0**
  Este registro se utiliza para configurar modos de operación y estados de la máquina, sobre todo el resultado de las operaciones de la ALU:

  - Bits 15..10, OVC/OVCU. Estos bits se comportan diferente para operaciones con signo y sin signo.
    OVC: contador de sobreflujo para operaciones con signo, es de 6 bits signados con un intervalo de -32 a 31.
    - Si OVM = 0, el acumulador alcanza un sobreflujo normal y OVC sigue el sobreflujo.
    - Si OVM = 1, y si ocurre sobreflujo en ACC, el ACC se satura en dirección positiva de 7FFF FFFFh a 8000 0000h y OVC se incrementa. Si el ACC se satura en dirección negativa de 8000 0000h a 7FFF FFFFh, el OVC se decrementa.
    OVCU : contador de operaciones sin signo, se incrementa cuando existe un acarreo y se decrementa cuando existe un préstamo.
  - Bits 9..7, PM: bits de modo producto.
  - Bit 6, V: bandera de sobreflujo.
  - Bit 5, N: bandera de valor negativo.
  - Bit 4, Z: bandera de cero.
  - Bit 3, C: bandera de acarreo.
  - Bit 2, TC: bandera de prueba o control.

- Bit 1, OVM: bit de modo de sobreflujo o saturación.
- Bit 0, SXM: extensión de signo:
  - SXM = 0, aritmética sin extensión de signo de 16 a 32 bits
  - SXM = 1, aritmética con extensión de signo de 16 a 32 bits

- **Registro de estado ST1**
  Todos los bits de este registro son afectados en la etapa D2 de decodificación del pipeline.

  - Bits 15..13, ARP: apuntador de registros auxiliares, apunta a un registro auxiliar XARi. Al reset, por defecto apunta al XAR0.
  - Bit 12, XF: bit de estado que refleja el valor de pin XFS, se pone en uno con la instrucción SETC XF y se limpia con CLRC XF.
  - Bit 11, M0M1MAP: bit de modo de mapeo de los bloques de memoria M0 y M1. En cero, los bloques M0 y M1 se mapean únicamente en el espacio de programa.
  - Bit 10: reservado.
  - Bit 9, OBJMOD: modo de compatibilidad con familias C27x
  - Bit 8, AMODE: en conjunto con el bit de modo de página PAGE0 es utilizado para seleccionar el modo de direccionamiento adecuado.
  - Bit 7, IDLE: bit de estado IDLE, es de solo lectura y se pone en uno cuando alguna instrucción IDLE se está ejecutando. Se limpia con los eventos:
    - Atención de interrupción.
    - Una interrupción no atendida libera al CPU del estado IDLE.
    - En el reset.
  - Bit 6, EALLOW: acceso de emulación, habilita el acceso a emulación y a registros protegidos. Se pone en uno con instrucción EALLOW y se limpia con instrucción EDIS.
  - Bit 5, LOOP: en uno indica que un ciclo LOOP está en proceso.
  - Bit 4, SPA: alineamiento de SP:
    SPA = 0, el SP no se alinea a una dirección par.
    SPA = 1, el SP se alinea a direcciones par.
  - Bit 3, VMAP: bit de mapeo de vectores de interrupción. Determina cuando los vectores de interrupción del CPU (incluyendo el reset) son remapeados a direcciones bajas o altas en memoria de programa. Parte baja 00 0000h a 00 003Fh y parte alta 3F FFC0h a 3F FFFFh.
  - Bit 2, PAGE0: configuración de modo de direccionamiento directo:
    PAGE = 0, utiliza SP para direccionamiento directo.
    PAGE = 1, direccionamiento directo modo paginado.

- Bit 1, DBGM: máscara de habilitación de depuración. En uno, el emulador no puede acceder registros en tiempo real.
- Bit 0, INTM: en cero habilita todas la interrupciones mascarables, en uno las deshabilita. En registro de habilitación de interrupciones (IER) se deben habilitar los bits individuales correspondientes a cada interrupción.

3.4. Unidad central de proceso

La unidad central de proceso (CPU) de los DSP C28x es la encargada de realizar todas las operaciones de procesamiento y está constituida por cuatro bloques principales:

- Registro acumulador (ACC).
- Registros de corrimiento.
- Unidad aritmético lógica (ALU).
- Multiplicador.

Estos bloques están interconectados a la memoria de programa y de datos a través de buses y multiplexores, como se observa en la figura 3.3.

El registro ACC es uno de los principales registros de toda máquina digital, siempre contiene un operando para la realización de operaciones en la unidad ALU, y el resultado de cualquier operación ALU queda en el registro ACC, como se observa en la figura 3.3. ACC se encuentra a la salida de ALU.

3.4.1. Registros de corrimiento

Los registros de corrimiento (SH) son utilizados para realizar operaciones tales como:
• Preescalar operandos de entrada de la memoria o acumulador antes de operar en la unidad ALU.
• Efectuar corrimientos lógicos o aritméticos del acumulador.
• Normalizar el acumulador para implementar operaciones en punto flotante.
• Escalamiento del acumulador antes de almacenarlo en memoria.

Los registros de corrimiento son tres bloques que operan sobre datos que provienen de diferentes fuentes:

- Registro de corrimiento de entrada (SH(i))
  Realiza corrimientos de 16 bits a la izquierda (L) sobre un operando dato o constante de entrada de 16 bits, corrimientos a la derecha (R) o izquierda (L) sobre el ACC. El

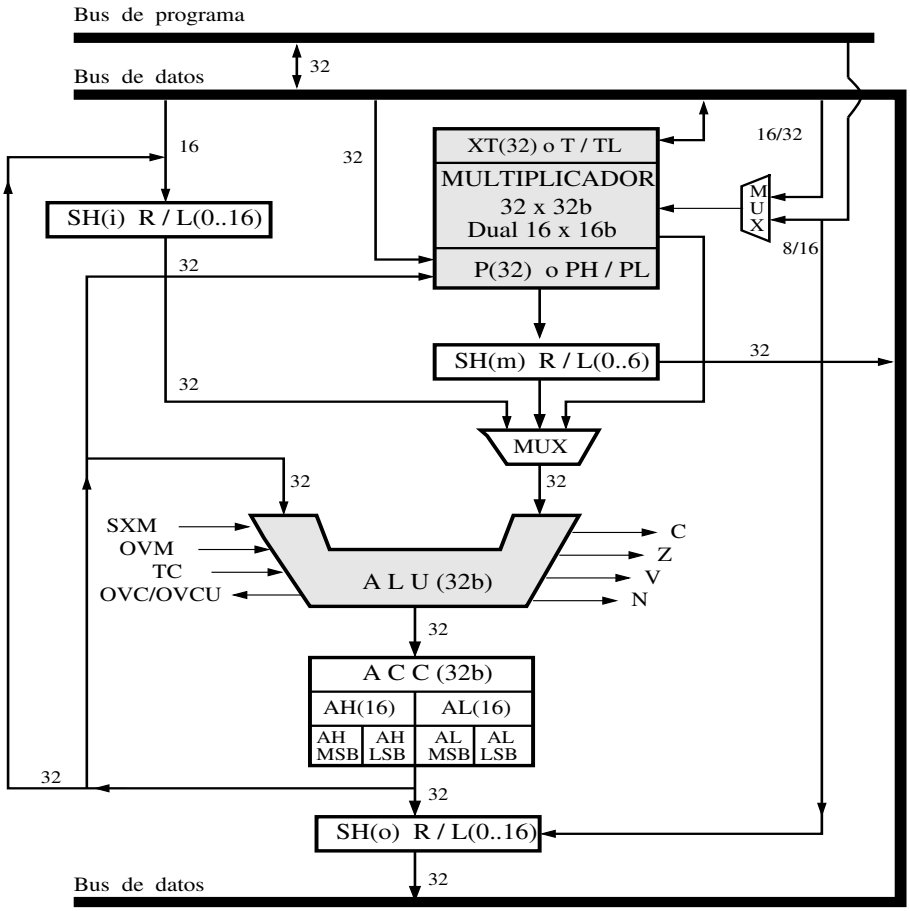


Figura 3.3. Unidad central de proceso de los DSP C28x

corrimiento se indica en el campo de corrimiento de la instrucción, cuando se opera sobre el acumulador también se pueden indicar los corrimientos en los cinco bits menos significativos (4..0) del registro T y en la instrucción se indica que los corrimientos se determinan por T.

- Registro de corrimiento de salida (SH(o))
  Realiza corrimientos de 16 bits R/L sobre el ACC cuando se va a almacenar en la memoria de dato. Este corrimiento se indica en el campo de corrimientos de la instrucción.

- Registro de corrimiento del multiplicador (SH(m))
  Realiza corrimientos de 16 bits R/L sobre la salida del multiplicador, cuando el producto de una operación se va a transferir a la ALU. Este registro se establece de antemano en los bits (9..7) del campo PM del registro de estado ST0.

Cuando se realizan corrimientos a la derecha, los bits LSb de la derecha se pierden y los más significativos se llenan con su bit de signo si se especifica extensión de signo (SXM = 1), de lo contrario se llenan con ceros (SXM = 0). Cuando se realizan corrimientos a la izquierda los bit LSb se van llenando con ceros.

3.4.2. Unidad aritmético lógica (ALU)

La unidad ALU efectúa operaciones aritméticas y lógicas a 32 bits casi siempre en un ciclo de instrucción, el resultado de las operaciones se escribe en los acumuladores ACC. De la figura 3.3 tenemos que una operación de ALU requiere dos operandos y se realiza de la siguiente forma:

$$ACC_{32} = ACC_{32} \quad \text{(operación ALU)} \quad Operando \quad Y_{32} \tag{3.3}$$

donde el ACC siempre contiene un operando y el operando Y puede ser:
• Un dato (de memoria dato) con corrimiento SH(i).
• Una constante (de memoria programa) con corrimiento SH(i).
• Un resultado del producto con corrimiento SH(m).

Como se observa en la figura 3.3, dependiendo de las instrucciones, la unidad ALU manipula las banderas de estado de entrada OVM, SXM y afecta las banderas de estado de salida C, V, Z y TC.

3.4.3. Multiplicador

El multiplicador de las familias C28x efectúa multiplicaciones de 16x16 bits o 32x32 bits en un ciclo de reloj para operaciones de precisión extendida, además en conjunto con

la ALU puede realizar simultáneamente multiplicaciones y acumulaciones (MAC) de 16x16 bits, MAC de 32 x32 bits y MAC dobles (DMAC) de 16x16 bits. En la figura 3.3 se observan la unidad central de proceso de los DSP C28x, que consta del multiplicador, el registro de corrimiento, la unidad ALU y los buses que los conectan con la memoria.

- **Multiplicador de 16x16 bits**
  Realiza multiplicaciones de 16x16 bits con signo o sin signo, el resultado lo deja en 32 bits.
  - Un operando de 16 bits debe estar en el registro multiplicando T.
  - El otro operando se indica en la instrucción de multiplicación y puede venir de:
    - el código de instrucción (una constante)
    - un dato en memoria
    - un registro.
    Por otro lado, la instrucción MAC y algunas versiones de las instrucciones MPY y MPYA cargan el registro T antes de realizar la multiplicación.
  - El resultado se almacena en el registro producto P o en el acumulador ACC, dependiendo de la instrucción.
  - En la multiplicación dual de 16x16 bits, la instrucción DMAC toma dos operandos de 32 bits como entrada y multiplica partes altas entre sí y de forma similar las partes bajas. En el ACC se acumula el resultado de la multiplicación de las partes altas, y en el registro P se acumula el resultado de la multiplicación de las partes bajas.
- **Multiplicación de 32x32 bits**
  El multiplicador acepta dos entradas de 32 bits.
  - El primer operando de entrada es un dato de 32 bits leído de la memoria de programa vía el bus de direcciones del bus programa, utiliza las instrucciones IMPYAL, QMPYAL, IMACL y QMCAL. El dato de 32 bits también puede estar en el registro multiplicando XT.
  - El segundo operando proviene de un dato en la memoria de dato o un registro dependiendo de la instrucción.
  - El resultado de la multiplicación es almacenado en el registro P (parte baja de 64 bits) y en el ACC (parte alta de 64 bits), el programador decide cómo salvar en memoria estas partes.

3.5. Pipeline

El pipeline es una técnica de procesamiento en paralelo que se utiliza ampliamente en los DSP de TI. Esta técnica consiste en traslapar los procesos de cada una de las operaciones de

los buses durante la ejecución de instrucciones, esto puede observarse como una cadena de producción de autos donde existen procesos de ensamblado que se realizan sobre el auto desde que entra hasta que sale ya fabricado. En una arquitectura digital que opere con pipeline, las instrucciones son transferidas a estados sucesivos, donde unidades separadas de hardware realizan diversas operaciones de los buses para completar la ejecución de la instrucción. Las arquitecturas que operan bajo este concepto son conocidas de procesamiento vectorial, y su eficiencia solo es posible si las operaciones a ejecutar son vectorizables, es decir, si es factible arreglarlas en un flujo continuo de datos. En la figura 3.4 se compara la ejecución de una instrucción I1 en una arquitectura secuencial con una arquitectura pipeline, si toda instrucción necesita ocho operaciones a nivel hardware para su ejecución, la arquitectura secuencial requiere ocho ciclos de reloj; mientras que en la arquitectura con pipeline, a partir de la ejecución de la instrucción I1, todas las instrucciones se ejecutan en un ciclo de reloj.

Algunas dificultades que presenta el pipeline son las ramificaciones que puedan existir en un programa, ya que se pierde la secuencia de los procesos sobre una secuencia de instrucciones que van en desarrollo, sin embargo, en algunos DSP existen instrucciones de salto, llamadas a subrutina y retornos de subrutina con retardo para prever que el pipeline siempre esté lleno y se siga conservando el desempeño de la máquina.

El C28x utiliza ocho niveles de pipeline en la ejecución de instrucciones, esto permite maximizar su desempeño y ejecutar cada instrucción en un ciclo de reloj. El proceso de pipeline previene la escritura y lectura en una localización. Como se observa en la figura 3.4, los niveles de pipeline son:

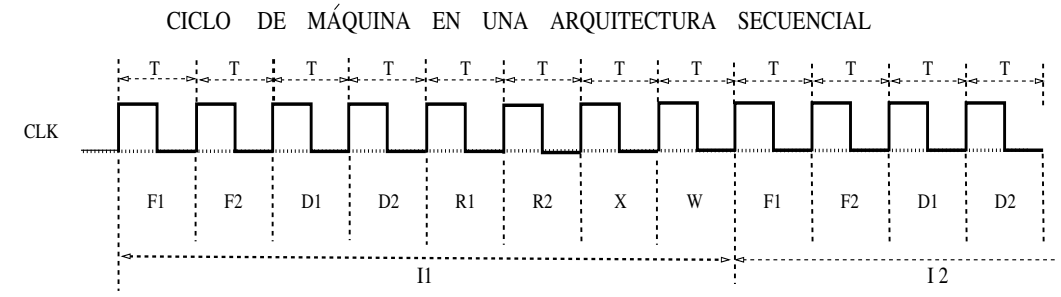
1. **F1: Dirección de instrucción**
   El CPU emite una dirección de 22 bits en el bus de direcciones de programa (PAB).

2. **F2: Contenido de instrucción**
   Lee la instrucción en memoria programa vía el bus de datos de lectura de programa (PRDB) de 32 bits.

3. **D1: Decodificación de instrucción**
   Identifica si la instrucción leída es de 16 o 32 bits y la alinea a direcciones pares o impares para determinar los límites de la siguiente instrucción a ser buscada. También determina si la instrucción buscada es legal.

4. **D2: Resuelve la dirección del operando**
   La instrucción es cargada en el registro de instrucción donde es decodificada completamente. En esta fase determina si es necesario traer operandos y la fuente de los operandos.

5. **R1: Dirección del operando**
   Si el operando es leído de memoria, se emite la dirección apropiada vía el bus de direcciones de datos.



CICLO DE MÁQUINA DE LA ARQUITECTURA PIPELINE DEL C28x

UNIDADES												
F1	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
F2	--	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11
D1	--	--	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
D2	--	--	--	I1	I2	I3	I4	I5	I6	I7	I8	I9
R1	--	--	--	--	I1	I2	I3	I4	I5	I6	I7	I8
R2	--	--	--	--	--	I1	I2	I3	I4	I5	I6	I7
X	--	--	--	--	--	--	I1	I2	I3	I4	I5	I6
W	--	--	--	--	--	--	--	I1	I2	I3	I4	I5

Figura 3.4. Niveles de pipeline de los DSP C28x

6. **R2: Toma el operando**
   Si el dato fue direccionado en R1, en esta etapa se lee el dato vía los buses de datos.

7. **X: Ejecuta la instrucción**
   El CPU ejecuta la instrucción en proceso.

8. **W: Almacena el contenido en memoria**
   Esta fase se presenta cuando se va a escribir un dato a memoria, el CPU maneja la dirección de escritura.

En el proceso de pipeline, el mecanismo de búsqueda de instrucción utiliza tres registros contadores de dirección de programa: contador de programa (PC), el contador de instrucción (IC) y el contador de búsqueda (FC). Cuando el pipeline está lleno, el PC siempre apuntará a la instrucción que está en la segunda parte de decodificación D2, el IC apunta a la próxima instrucción a ser procesada, si el PC apunta a una instrucción de una palabra IC = PC+1, si el PC apunta a una instrucción de dos palabras IC = PC+2. El valor en FC es la dirección de la próxima dirección a ser buscada. El proceso de pipeline en el C28x tiene en consideración dos bloques desacoplados para evitar conflictos en el flujo normal de las instrucciones, el primer bloque va de F1 a D1 y el segundo de D2 a W. Como se verá en la parte de las

interrupciones. Cuando una instrucción en el proceso de pipeline ha alcanzado la etapa D2, ésta completa su ejecución antes de ir a atender la interrupción.

Resumen

En este capítulo se ha descrito las partes fundamentales de la arquitectura de las familias C28x, ya que para utilizar a fondo estos dispositivos y realizar aplicaciones optimizadas, es importante tener un conocimiento amplio y profundo de su arquitectura. En adelante se tomará en consideración todas estas características y conforme se vayan utilizando se agregarán más detalles de la potencialidad de estos DSP.

Capítulo 4

Memoria y modos de direccionamiento

Cuando se diseñan sistemas digitales utilizando dispositivos programables, se debe evaluar de antemano la aplicación para conocer la cantidad de recursos necesarios. Una parte importante del diseño es considerar la cantidad de memoria disponible para que en el transcurso de la operación sea suficiente. Dentro de la misma memoria todavía existen varias posibilidades como: memoria de sólo lectura (ROM), donde normalmente se graba el código del programa del sistema; memoria borrable ROM (EEPROM), que tiene la misma función que la ROM, con la diferencia que puede borrarse eléctricamente, en la actualidad es muy frecuente encontrar memorias del tipo FLASH, que pueden funcionar como memoria ROM o de almacenamiento permanente de datos, esta memoria se puede borrar por instrucciones del mismo dispositivo; por otro lado está la memoria de acceso aleatoria (RAM), que es volátil y sirve para almacenar datos temporales en el proceso; a su vez, en algunos DSP existen bloques de memoria de doble acceso en un ciclo (DARAM) y la memoria RAM de simple acceso se le llama (SARAM).

Conociendo nuestras necesidades y el contenido de memoria de varios dispositivos, podemos tomar la decisión más adecuada en cuanto a recursos de memoria. Por otro lado, es importante conocer las formas de transferencias de datos para ser procesados, conocidos como modos de direccionamiento, que son de mucha utilidad para lograr altos desempeños de nuestra aplicación.

En este capítulo se describen los recursos relacionados con la memoria de los DSP de la familia C28x con el fin de conocerlos y saber donde ubicaremos nuestros programas, constantes, variables etc. También se describirán los modos de direccionamiento, que en combinación con los datos de memoria, transfieren información a las unidades de proceso, como se explicará más adelante.

4.1. Mapa de memoria

El mapa de memoria del C28x está dividido en el espacio para programa y datos, existen diferentes tipos de memoria que pueden ser usados en ambos espacios [18].

En la tabla 4.1 se muestra el mapa de memoria para el DSP F2812 que es muy similar en la mayoría de familias C28x. Estos DSP utilizan 32 bits de dirección datos (puede direccionar 4.29 Gw de dato) y 22 bits para direcciones de programa (4.19 Mw de programa, una palabra = w = 16 bits). Todos los dispositivos C28x contienen los bloques M0 y M1 del tipo SARAM en el chip, cada uno de 1Kw, y pueden utilizarse para espacio de programa y espacio de dato, los bloques M0 y M1 pueden utilizarse para ejecutar código o para datos. Dependiendo del DSP C28x en específico puede variar la disposición de los bloques o tamaños.

Tabla 4.1. Mapa de memoria del DSP TMS320F2812

Dirección (h)	**Bloque**/Tamaño	**Descripción**
00 0000 - 00 03FF	M0 SARAM (1K) bit VMAP=0	SARAM en el chip, dato o programa Vectores de interrupción en RAM
00 0400 - 00 07FF	M1 SARAM (1K)	SARAM en el chip, dato o programa
00 0800 - 00 0CFF	PF 0 (2K)	Mem. dato
00 0D00 - 00 0FFF	PIE 0 (256) bit ENPIE = 1 VMAP =1	Vectores de interrupción mem. dato
00 1000 - 00 5FFF	Reservado	Para XINTF, zonas 0 y 1 (16K)
00 6000 - 00 6FFF	PF 2 (4K)	Mem. dato
00 7000 - 00 7FFF	PF 1 (4K)	Mem. dato
00 8000 - 00 8FFF	L0 SARAM (4K)	Mem. dato y programa
00 9000 - 00 9FFF	L1 SARAM (4K)	Mem. dato y programa
00 A000 - 3D 77FF	Reservado	Para XINTF, zonas 2 y 6 (1M)
3D 7800 - 3D 7BFF	Flash OTP (1K)	Todas las familias, dato o programa
3D 7C00 - 3D 7FFF	Reservado	Para variables
3D 8000 - 3F 7FFF	Flash OTP (128K) 128 bit de “password”	Todas las familias, dato o programa
3F 8000 - 3F 9FFF	HO SARAM (8K)	SARAM, dato o programa
3F A000 - 3F EFFF	Reservado	Para XINTF, zona 7 (16K) MP/MC=1
3F F000 - 3F FFBF	ROM (4K) MP/MC=0	Para arranque, dato o programa
3F FFC0 - 3F FFDF	Vectores BROM (32) MP/MC = 0 bit ENPIE=0	Mem. dato o programa Vectores de interrupción en ROM Si bit VMAP=1

Memoria flash

Todos los dispositivos de la familia F28x contienen al menos 1Kw de memoria flash OTP en

las direcciones 3D 7800h - 3D 7BFFh. Este espacio puede ser mapeado en memoria dato o programa. Dependiendo del dispositivo en específico, se puede tener más memoria flash:
• El F2801 contiene 16Kw segmentados en cuatro sectores de 4Kw.
• El F2802 y el F2806 contienen 32Kw segmentados en cuatro sectores de 8Kw.
• El F2808 contiene 64Kw segmentados en cuatro sectores de 16Kw.
• El F2809 contiene 128Kw segmentados en ocho sectores de 16Kw.

Memoria ROM
Todos los dispositivos contienen los bloques M0, M1 de 1 Kw en memoria SARAM, mapeados en ambos espacios, programa y datos:
• El C2802 contiene 32K x 16 de ROM y el C2801 contiene 16K x 16 de ROM.
• Los DSP F2802, F2801, C2802 y C2801 contienen un bloque SARAM L0 de 4Kw.
• El F2806 contiene 8 Kw en SARAM dividido en los bloque L0 (4Kw) y L1 (4Kw).
• Los DSP F2808 y F2809 contienen 16 Kw en SARAM, dividida en bloques L0 (4Kw), L1 (4Kw) y H0 (8Kw).
Cada uno de los bloques se pueden acceder independientemente y pueden ser mapeados en ambos espacios, programa y dato.

4.1.1. Memoria flash y SARAM

Dependiendo de la familia del DSP, existen varios cambios en cuanto a los bloques de memoria:
• C2801: 16Kw ROM, 6Kw SARAM.
• C2802: 32Kw ROM, 6Kw SARAM.
• F2801: 16Kw flash, 6Kw SARAM.
• F2802: 32Kw flash en cuatro bloques de 8 Kw, 6Kw SARAM.
• F2806: 32Kw flash en cuatro bloques de 8 Kw, 10Kw SARAM.
• F2808: 64Kw flash en cuatro bloques de 16 Kw, 18Kw SARAM.
• F2809: 128Kw Flash en ocho bloques de 16 Kw, 18Kw SARAM.
• 1K x 16 OTP ROM (sólo en dispositivo tipo flash), en la dirección 3D 7800h – 3D 7BFFh.
• Los bloques de memoria flash y OTP están mapeados en ambos espacios, programa y datos.
• Las direcciones 3F 7FF0h - 3F7FF5h son reservadas para datos y variables y no deben de contener código de programa.
• La memoria ROM para inicialización viene programada de fábrica con un código de inicialización del DSP.
En los DSP F2809/F2808/F2806/F2802/F2801, la memoria flash y OTP se puede configurar con estados espera.

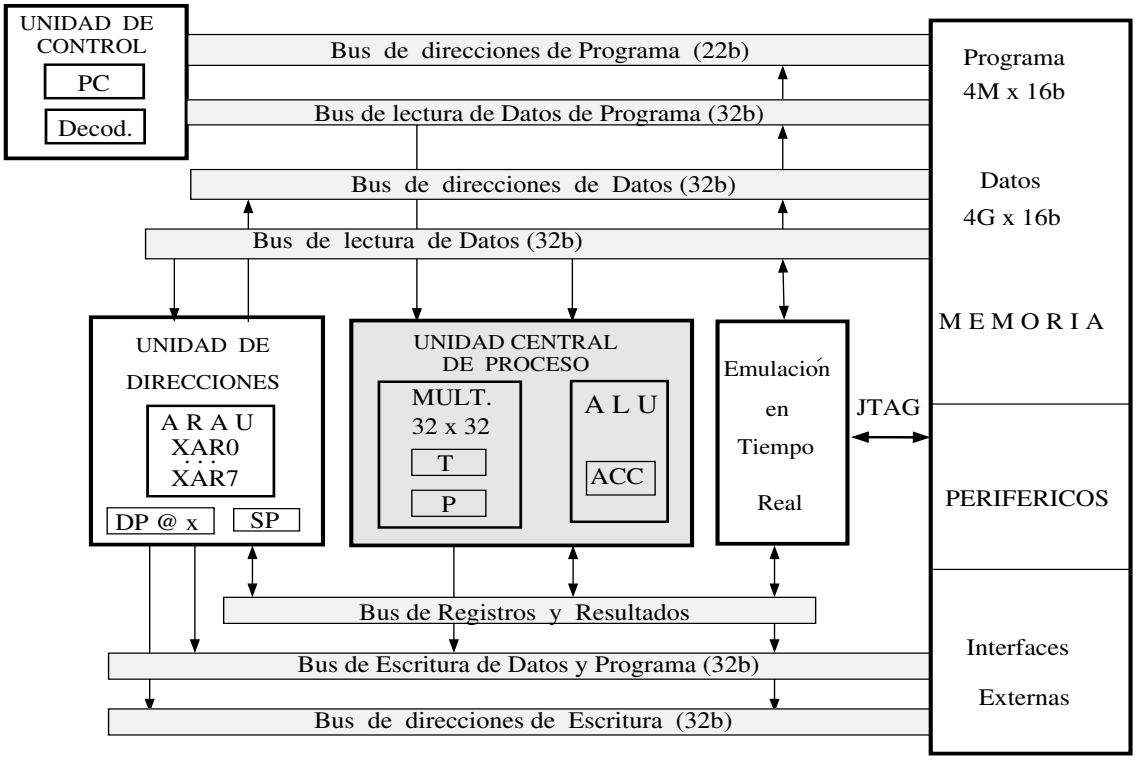


Figura 4.1. Buses y arquitectura general de los DSP C28x

4.1.2. Vectores de interrupción

En el espacio de programa se han reservado 64 localidades para 32 vectores de interrupción del CPU. Estos vectores pueden mapearse en la parte alta o baja del espacio de memoria de programa. Para interrupciones externas existe la expansión de periféricos de interrupción (PIE) que tiene su propio bloque de vectores de interrupción.

4.1.3. Buses internos

La arquitectura del C28x es del tipo Harvard, es decir, una arquitectura de buses separados, como se muestra en la figura 4.1:

- Un bus de direcciones para lectura de programa de 22 líneas.
- Un bus para el código de programa de 32 bits.
- Un bus de direcciones de 32 para lectura de datos.
- Un bus para lectura de datos de 32 bits.

La interfaz a memoria tiene tres buses de dirección y tres buses de datos:

- **PAB**, bus de direcciones de programa de 22 bits, transporta las direcciones de acceso para lectura y escritura en el espacio de memoria.
- **DRAB**, bus de direcciones de lectura de datos, de 32 bits, transporta direcciones para lectura en el espacio de datos.
- **DWAB**, bus de direcciones de escritura de datos, de 32 bits, transporta direcciones para escritura en el espacio de datos.
- **PRDB**, bus de lectura de datos de programa de 32 bits.
- **DRDB**, bus de lectura de datos de 32 bits.
- **DWDB**, bus de escritura en espacio de datos o programa de 32 bits.

4.1.4. Seguridad

Los DSP de las familias C28x soportan un nivel de seguridad para prevenir que le apliquen ingeniería inversa. Las características de seguridad consisten en prevenir contra usuarios no autorizados que traten de examinar la memoria vía el puerto JTAG, ejecutar código desde memoria externa o tratar de levantar el sistema desde otro software para exportar el contenido de la memoria. La seguridad consiste en utilizar un "password" de 128 bits de longitud, que se programa en memoria flash. El módulo de seguridad de código (CSM) es utilizado para proteger los bloques flash/OTP y los bloques L0 y L1.

4.1.5. Sintaxis de instrucciones

Antes de proseguir, es necesario conocer la sintaxis de escritura de las instrucciones y directivas de ensamblador, ya que se utilizarán ampliamente.

Un programa fuente en lenguaje ensamblador consiste en expresiones que pueden contener directivas, instrucciones de ensamblador o macroinstrucciones. En general, en cualquier lenguaje ensamblador existen los siguientes campos:

```
[ETIQUETA][:]      MNEMONICO   OPERANDOS   ;    [COMENTARIOS]
```

Cada campo está separado por uno o más espacios en blanco.

- **Etiqueta:** inicia en la primera columna de la expresión. Es de carácter opcional y cuando se utiliza, normalmente indica una dirección del programa y sirve para transferir el control del programa bajo ciertas decisiones o saltos. También puede ser útil para definir símbolos, constantes y localidades de memoria dato. Los dos puntos no son parte de la etiqueta. Cuando no se utiliza etiqueta, la primera columna antes de la instrucción debe estar en blanco.

- **Mnemónico:** en sí es la orden o comando, y puede contener:
  - Instrucciones de ensamblador.
  - Macroinstrucciones.
  - Directivas de ensamblador.

- **Operandos:** dependiendo de la instrucción pueden ser necesarios varios operandos separados por comas. Los tipos de operandos pueden ser:
  - Una constante que se antecede con un símbolo #.
  - Registros.
  - Símbolos.
  - Etiquetas para saltos o llamados a subrutinas.
  - Condiciones.
  - Nombres de bits de algún registro.

- **Comentarios:** son optativos, sin embargo, si se utilizan en una instrucción, es necesario antecederlos por un punto y coma. Los comentarios no generan código y únicamente son de utilidad para el programador. Si se requiere comentar una línea completa se debe usar un asterisco (*) en la primera columna.

En la tabla 4.2 se resumen una serie de símbolos y abreviaturas que se utilizarán en la programación en lenguaje ensamblador.

Convención de instrucciones:
- Instrucción terminada en B hace referencia a operandos de 8 bits.
- De otra forma, depende del operando y puede ser de 16 o 32 bits.

Tabla 4.2. Convención de símbolos

Símbolos	**Descripción**
XARn	Registros auxiliares XAR0 a XAR7 de 32 bits
ARn	Parte baja de 16 bits de los registros XAR0 a XAR7
ARnH	Parte alta de 16 bits de los registros XAR0 a XAR7
ARPn	Apuntador de registros auxiliares XAR0 a XAR7
AR(ARP)	16 bits bajos del registro auxiliar apuntado por ARP
XAR(ARP)	Registro auxiliar apuntado por ARP
AX	Acumulador parte alta AH y parte baja AL
#	Operando inmediato, antecede a una constante
PM	Modo corrimiento del producto (+4,+1,0,-1,-2,-3,-4,-5,-6)
PC	Contador de programa
:	Concatenación de bits o campos de bits
..	Intervalo de bits
–	Intervalo de memoria
^	Elevar a una potencia
∼	Complemento de bits
loc16	Contenido de una localidad de 16 bits
0:[loc16]	Contenido de una localidad de 16 bits con extensión de ceros
S:[loc16]	Contenido de una localidad de 16 bits con extensión de signo
loc32	Contenido de una localidad de 32 bits
0:[loc32]	Contenido de una localidad de 32 bits con extensión de ceros
S:[loc32]	Contenido de una localidad de 32 bits con extensión de signo
7 bits	Valor de 7 bits inmediatos
0:7bits	Valor de 7 bits inmediatos con extensión de ceros
S:7bits	Valor de 7 bits inmediatos con extensión de signo
8 bits	Valor de 8 bits inmediatos
0:8bits	Valor de 8 bits inmediatos con extensión de ceros
S:8bit	Valor de 8 bits inmediatos con extensión de signo
10 bits	Valor de 10 bits inmediatos
0:10bits	Valor de 10 bits inmediatos con extensión de ceros

Tabla 4.3. Convención de símbolos (continuación)

Símbolos	**Descripción**
16 bits	Valor de 16 bits inmediatos
0:16 bits	Valor de 16 bits inmediatos con extensión de ceros
S:16 bits	Valor de 16 bits inmediatos con extensión de signo
22 bits	Valor de 22 bits inmediatos
0:22 bits	Valor de 22 bits inmediatos con extensión de ceros
LSb	Bit menos significativo
LSB	Byte menos significativo
LSW	Palabra menos significativa
MSb	Bit más significativo
MSB	Byte más significativo
MSW	Palabra menos significativa
OBJ	Estado del bit OBJMODE, válido en algunas instrucciones
{ }	Campo opcional
=	Asignación
==	Equivalente a
!=	Diferente de
B	La instrucción hace referencia a operandos de 8 bits
L	La instrucción hace referencia a operandos de 32 bits
	Sin nada, la instrucción hace referencia a operandos de 16 bits
*	*XARn, acceso en modo indirecto
@	@variable: acceso en modo directo, AMODE = 0
	@Registro: para direccionamiento a registros
@@	@@variable acceso en modo directo, AMODE = 1
xxx Ax,#16b ; word	Operandos, word, byte y long
xxxB Ax,#8b ; byte	xxx instrucción MOV, ADD, SUB,etc
xxxL ACC,#32b ; long	Ax = AH o AL

4.2. Modos de direccionamiento

Los modos de direccionamiento utilizados por estos DSP le dan una gran potencialidad y versatilidad a la utilización de la arquitectura y la ejecución de las operaciones. Los CPU C28x utilizan cuatro modos de direccionamiento básicos [18]:

- Inmediato
- Directo
- Por la pila
- Indirecto
- De registros

La mayoría de instrucciones del C28x utilizan 8 bits para el código de operación donde selecciona el modo de direccionamiento que se utilizará, en una instrucción el campo es referido como:
• loc16: selecciona los modos directo, de stack, indirecto y de registro para datos de 16 bits.
• loc32: selecciona los modos directo, de stack, indirecto y de registro para datos de 32 bits.

4.2.1. Direccionamiento inmediato

Se le llama inmediato porque el operando es una constante especificada en la instrucción, esta constante se codifica en un campo del código de la instrucción. Este modo es muy útil para procesos de inicialización, configuración de registros y operaciones con constantes. La constante a utilizar viene precedida por un símbolo #, cuando la constante es de ocho o menos bits, se dice que es corta y el código de instrucción ocupa sólo una palabra de instrucción. Cuando la constante es de 16 bits, se dice que es larga, y el código de instrucción ocupa dos palabras de código, una para el código y otra para la constante misma, en este caso la instrucción admite corrimientos de 0 a 16 bits a la izquierda sobre la constante.

Ejemplo:

```
ADDB  AL,#N      ; AL = AL + N, N = cte de 8 bits alineada a la derecha
ADD   ACC,#NL,4  ; ACC = ACC + NL << 4
MOV   AR0,#10    ; Carga el registro AR0 con 10
MOVL  XAR6,#x    ; Carga la dirección de x en el registro XAR6
MOVW  DP,#t      ; Carga en registro DP la página de datos t
```

4.2.2. Direccionamiento directo

En este modo la memoria se maneja por páginas apuntadas por el apuntador de página (DP) de 16 bits. La instrucción contiene el offset o desplazamiento de seis o siete bits de la dirección del dato a transferir, el offset concatenado con la página dan la dirección del dato a transferir. De esta forma se puede acceder al espacio de 4M palabras en páginas de 64 palabras (65,536 páginas de 64 palabras cada una), como se observa en la figura 4.2. Este modo es útil para acceder estructuras de datos fijas tales como registros de periféricos y variables globales o estáticas cuando se programa en lenguaje C o C++.

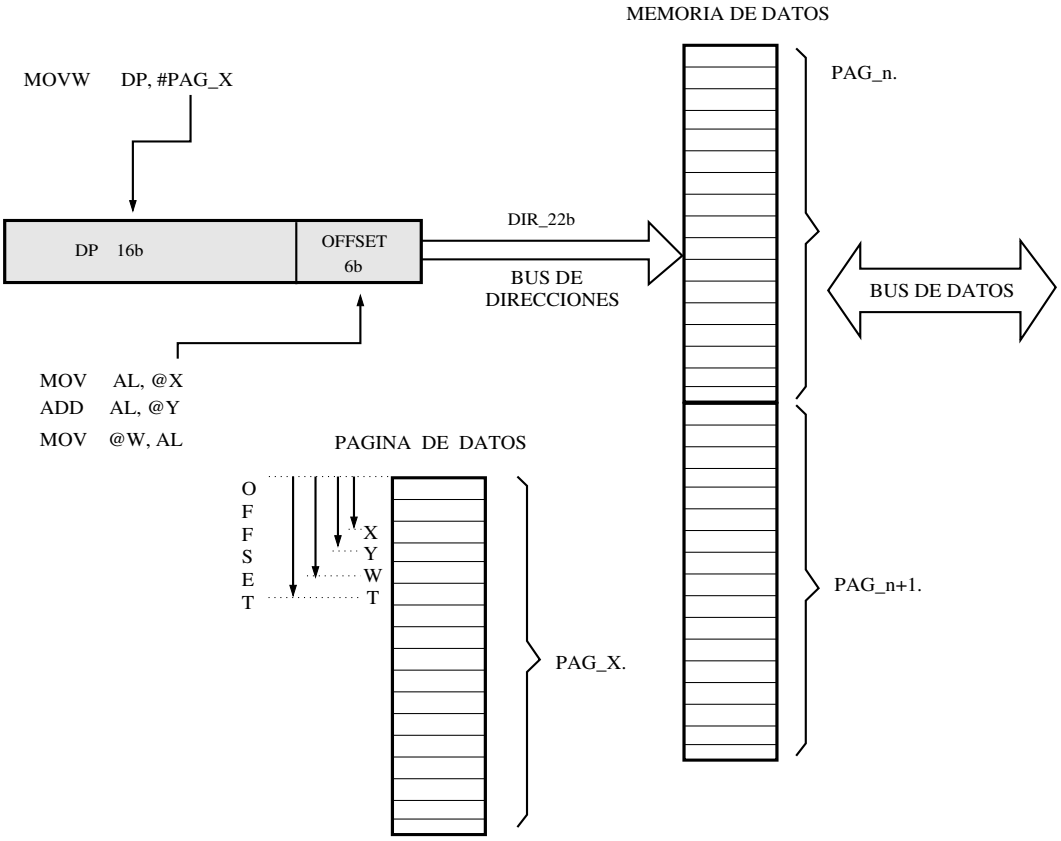


Figura 4.2. Modo de direccionamiento directo

Bit de selección del modo de direccionamiento (AMODE)

Este bit se encuentra en el registro de estado ST1 y se utiliza para seleccionar varios tipos de direccionamiento, con este bit se seleccionan los campos de decodificación de 8 bits para localidades de 16 o 32 bits (loc16/loc32).

- **Si AMODE = 0**, es el modo por defecto del C28x.

Con la instrucción CLRC AMODE, se pone AMODE = 0. La dirección de un dato en modo directo se forma concatenando los 16 bits de DP (DP(15..0)) con los seis bits del offset (la variable con offset de seis le antecede una @).

- **Si AMODE = 1**, permite modos de direccionamiento compatibles con dispositivos de familias anteriores C2xxx.

  Con instrucción SETC AMODE, se pone AMODE = 1. El offset sobre la página es de siete bits y soporta todos los modos de direccionamiento de C2xxx. Una dirección se forma concatenando los 15 bits LSB (DP(15..1)) de DP con los siete bits del offset, el bit LSB de DP se ignora (la variable con offset de siete bits le antecede @@).

Ejemplo:

```
*
    CLRC AMODE      ; AMODE = 0
    MOVW DP,#datA   ; Carga DP con la página que contiene a datA
    MOV  AL,@datA   ; Carga a AL el dato datA
    ADD  AL,@datB   ; AL = AL + datB
    MOV  @datC,AL   ; Almacena  AL en localización de datC
*
    SETC AMODE      ;  AMODE = 1
    MOVW DP,#datA   ;
    MOV  AL,@@datA  ;  AL = dato datA
    SUB  AL,@@datB  ;  AL = AL - datB
    MOV  @@datC,AL  ;  loc. datC = AL
```

Cuando se manejan varios datos en este modo, se debe tener cuidado que todos los datos estén en la misma página, de lo contrario se debe estar cambiando de página.

4.2.3. Modo de direccionamiento por la pila

En este modo se utilizan 16 bits del apuntador de pila (SP) para acceder información de la pila por software, el SP crece de la parte baja a la parte alta de la memoria y el SP siempre apunta a la próxima localidad vacía. La instrucción provee seis bits del campo de offset que son restados del SP para acceder el dato en la pila.

Tabla 4.4. Modificadores de direccionamiento por pila SP

Operando	**Descripción**
* -SP[6b]	Dir. dato = SP(16b) - Offset (constante de 6b)
SP++	si loc16 SP = SP + 1
	si loc32 SP = SP + 2
SP–	si loc16 SP = SP - 1
	si loc32 SP = SP - 2

Ejemplos:

```
* AMODE = 0,           Operando *-SP[6bits] , dir = SP - 6b
    ADD  AL,*-SP[5]     ; AL = AL + dato en loc SP - 5
    MOV  *-SP[8],AL     ; Almacena AL en loc. SP - 8
    ADDL ACC,*-SP[12]   ; ACC = ACC + dato en loc. SP - 12
    MOVL *-SP[34],ACC   ; Almacena ACC en loc SP - 34

* PUSH: AMODE no afecta, Si(loc16), SP = SP + 1; si(loc32), SP = SP + 2
    MOV  *SP++,AL       ; Pone AL (16b) en la parte alta del SP, SP = SP + 1
    MOVL *SP++,P        ; Pone P (32b) en la parte alta del SP, SP = SP+2

* POP: AMODE no afecta , Si(loc16), SP = SP - 1; si(loc32), SP = SP - 2
    ADD  AL,*--SP       ; El contenido de TOS lo pone en AL, SP = SP-1
    MOVL ACC,*--SP      ; El contenido de TOS lo pone en ACC, SP = SP-2
                        ; TOS: parte alta de la pila
```

4.2.4. Modo de direccionamiento indirecto

En este modo la dirección fuente o destino de un operando está contenida en otro registro, las familias de DSP de TI han utilizado lo que llaman *registros auxiliares* ARi (AR0 - AR7 de 16 bits, acceden hasta 64K palabras), y para el caso concreto de las familias C28x XARi

(XAR0 - XAR7 de 32 bits, acceden hasta 4G palabras). Este modo permite acceder dos datos en un ciclo de instrucción.

La unidad aritmética de registros auxiliares ARAU efectúa aritmética no signada sobre los registros XARi, con los registros auxiliares pueden efectuarse las operaciones:

- Cargar una dirección en los registros XARi.
- Cargar los XARi desde memoria por el bus de datos.
- Modificarse en el mismo ciclo de instrucción a través de unidades ARAU.
- Modificarse por instrucción MAR (modifica registro auxiliar).
- Utilizarse como contador de ciclo utilizando instrucción BANZ.

En la figura 4.3 se ilustra la conexión de los buses de datos con los registros XARi para direccionar la memoria dato, por otro lado, se observa cómo se genera una dirección en el modo directo, a través del registro apuntador de página (DP) y un offset que viene en la instrucción, vía el bus de programa. En la tabla 4.5 se describen los modificadores de los registros ARi para este modo, y en la figura 4.4 se ilustra la forma de acceder los datos en memoria dato utilizando este modo.

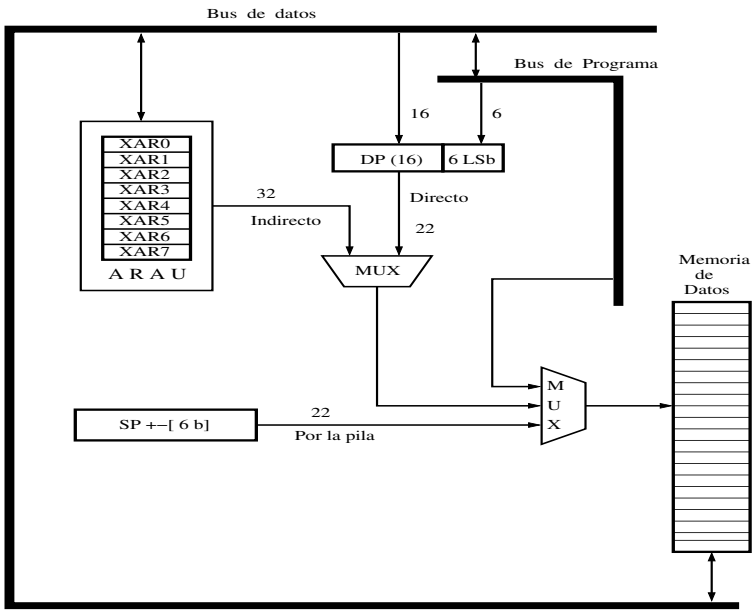


Figura 4.3. Unidad de direccionamiento, ARAU y registros XARi

En modo de direccionamiento indirecto, el valor de los registros AR0 y AR1 puede ser utilizado como un desplazamiento de 16 bits sobre los registros XARi utilizados. Por otro lado, el registro AR0 se carga con el valor de N/2 para direccionamiento en acarreo inverso para el cálculo de la transformada rápida de Fourier (FFT) radix dos [3], [10].

Tabla 4.5. Modificadores de direccionamiento indirecto

Operando	**Dirección**	**Descripción**
*XARi	dir=XARi	XARi contiene una dirección de memoria dato
*XARi- -	dir=XARi Si loc16 XARi=XARi - 1, si loc32 XARi=XARi - 2	Post-decrementa XARi después de acceder el dato
*XARi++	dir=XARi si loc16 XARi=XARi + 1, si loc32 XARi=XARi + 2	Post-incrementa XARi después de acceder el dato
*- -XARi	si loc16, dir=XARi - 1 si loc32 XARi=XARi - 2	Pre-decrementa XARi antes de acceder el dato
*++XARi	si loc16, dir=XARi + 1 si loc32 XARi=XARi + 2	Pre-incrementa XARi antes de acceder el dato
*XARi0- -	dir=XARi XARi=XARi - AR0	Después de acceder el dato, al XARi se le resta AR0
*XARi0++	dir=XARi XARi=XARi + AR0	Después de acceder el dato, al XARi se le suma AR0
*+XARi[AR0]	dir=XARi + AR0	Suma no signada de AR0
*-XARi[AR0]	dir=XARi - AR0	Resta no signada de AR0
*+XARi[3 bits]	dir= XARi + constante de 3 bits	Suma a XARi una constante de 3 bits Se trata como no signado
*XARiBRO++	dir=ARi ARi=BR(ARi+AR0)	Después de acceder el dato, al ARi se le suma AR0 en acarreo Inverso (BR) para la FFT radix 2
*XARiBRO- -	dir=ARi ARi=BR(ARi-AR0)	Después de acceder el dato, a ARi se le resta AR0 en en acarreo inverso
*XARi %- -	dir=XARi XARi=circ(XARi - 1)	Post-decrementa XAR6 o XAR1H en direccionamiento circular
*ARi-0 %	dir=ARi ARi=circ(ARi - AR0)	Post-decrementa ARi en AR0 en direccionamiento circular
*ARi %++	dir=ARi ARi=circ(ARi + 1)	Post-incrementa ARi en 1 si loc16, en 2 si loc32 en direccionamiento circular

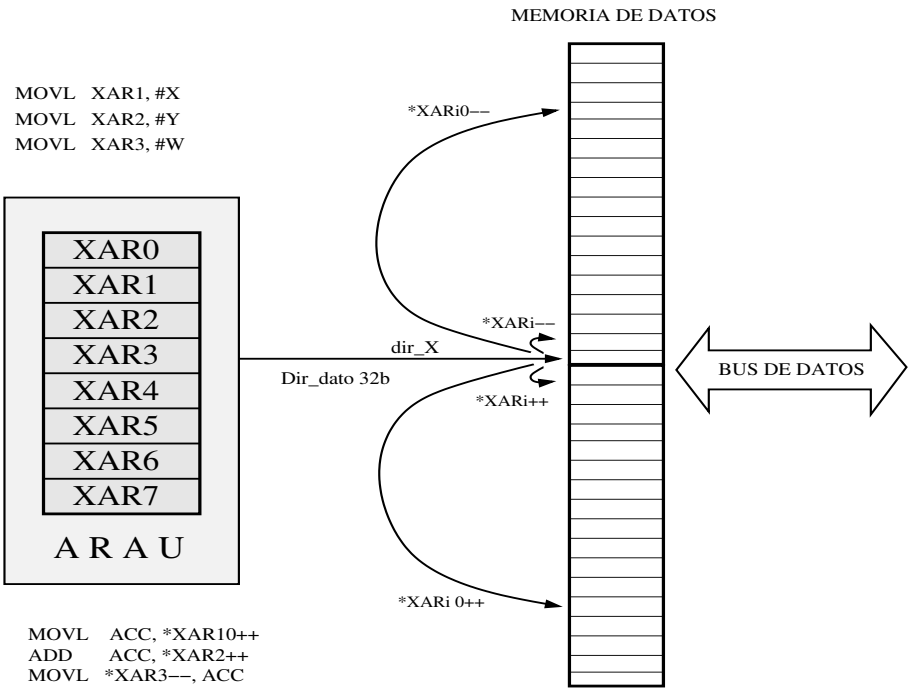


Figura 4.4. Modo de direccionamiento indirecto

Ejemplos de direccionamiento indirecto:

```
 *  AMODE no afecta, Si(loc16),  XARi = XARi + 1; si(loc32), XARi = XARi + 2
     MOVL  XAR2,#Vec1       ; XAR2 = dir. de Vec1
     MOVL  XAR3,#Vec2       ; XAR3 = dir. de Vec2
     MOV   AR0,#N-1         ; XAR0 = constante N-1, para contador de Ciclo
* Copia N datos del vector Vec1 al vector Vec2
Ciclo:
     MOVL ACC,*XAR2++       ; ACC = dat. en dir. XAR2, XAR2=XAR2+2
     MOVL *XAR3++,ACC       ; guarda ACC en dir. XAR3, XAR3=XAR3+2
     BANZ Ciclo,AR0 --      ; realiza el Ciclo hasta que AR= 0,
                            ; post decrementa AR0

*  AMODE no afecta,  Modificador con offset  +XARn[AR0 o AR1 ],
*  implica que dir.32bit = XARi + AR0 o AR1
     MOVL  XAR2,#Vec        ; XAR2 apunta a dir.Vec
     MOVB  AR0,#20          ; AR0 = 20 , AR0H = 0
     MOVB  AR1,#40          ; AR1 = 40 , AR1H = 0
     MOVL  ACC,*+XAR2[AR0]  ; ACC = dato en dir. XAR2+AR0
     MOVL  P,*+XAR2[AR1]    ; P =  dato en dir. XAR2+AR1
```

```
     MOVL  *+XAR2[AR1],ACC ; salva ACC en dir. XAR2+AR1
     MOVL  *+XAR2[AR0],P   ; salva P en dir. XAR2+AR0

*  AMODE  no afecta, Modificador con offset  *+XARn[3bits],
*  implica que dir.32bit = XARi + 3bits sin signo

     MOVL  XAR2,#Vec       ; XAR2 apunta a dir.Vec
     MOVL  ACC,*+XAR2[1]   ; ACC = dato en dir. XAR2+1
     MOVL  P,*+XAR2[4]     ; P =  dato en dir. XAR2+4
     MOVL  *+XAR2[4],ACC   ; salva ACC en dir. XAR2+4
     MOVL  *+XAR2[1],P     ; salva P en dir. XAR2+1
```

En modo de direccionamiento indirecto en el C28x, el registro auxiliar XARi con su modificador es un operando de la instrucción, y en la arquitectura C2xxx el registro ARP de tres bits es utilizado para seleccionar el registro auxiliar.

```
*  Ejemplo en el DSP C28X
     ADD AL,*XAR4++    ; ACC = ACC + loc16 apuntada por XAR4
                       ; XAR4 = XAR4 + 1
*  Para el C2XLP
     ADD AL,*++        ; ACC = ACC + loc16 apuntada por XARi en uso
                       ; XARi = XARi + 1
```

Nota del autor

El autor quiere hacer una sugerencia respecto a los post desplazamientos positivos o negativos con AR0 de un XARi, la única forma que los ha hecho funcionar en lenguaje ensamblador es utilizando la nomenclatura al estilo de las familias TMS320C2xx o TMS320C5x, es decir, utilizar un XARi en uso, como se muestra en el siguiente ejemplo.

```
*  Ejemplo
        MOV  AR0,#DELTA     ; AR0 contien el desplazamiento
        NOP   *,ARP2        ; selecciona XAR2 como registro auxiliar en uso
        MOVL XAR2,#dir_dato ;

        MOV  AL,*0++        ; AL = dato direccionado por XAR2
                            ; XAR2 = XAR2 + AR0
```

4.2.5. Direccionamiento circular

Este modo se puede considerar como un modificador o un caso especial del modo de direccionamiento indirecto. Todo buffer circular de tamaño R debe empezar en límites o fronteras de "n" bits, dependiendo del valor de bit AMODE, este modo tiene sus variaciones. Como se observa en la figura 4.5 un bloque de memoria o buffer se visualiza como un conjunto de localidades arregladas en un círculo.

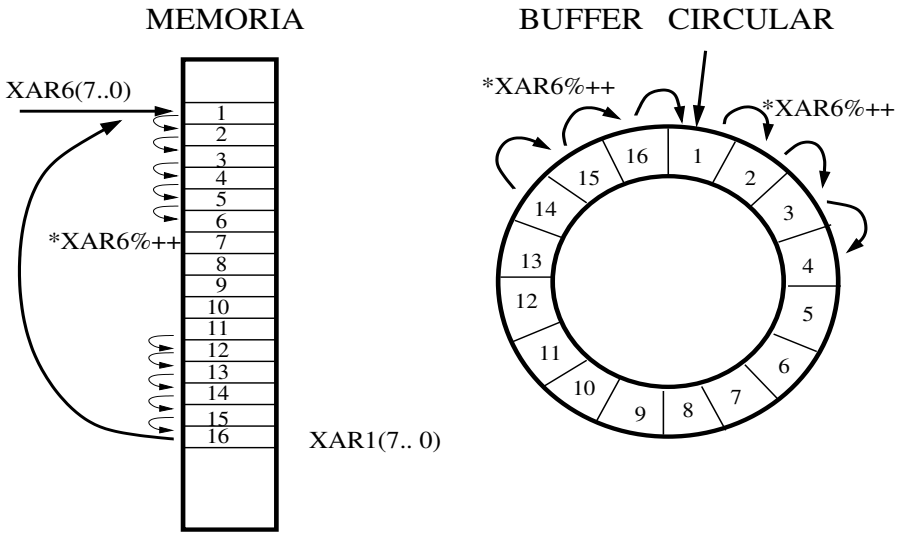


Figura 4.5. Bloque de memoria vista como un buffer circular

Buffer circular con AMODE = 0

El tamaño del buffer circular es determinado por los 8 LSb de AR1(7..0) + 1. El registro XAR6 con los bits 7..0 en ceros apunta a la dirección inicial del buffer, durante la operación XAR6 contiene una dirección dentro del buffer circular. Si la instrucción que opera en buffer circular accede un dato de 32b, debe asegurarse que XAR6 y AR1 sean pares antes de utilizar este modo. El algoritmo que realiza el hardware del DSP cuando una instrucción utiliza modificadores *AR6[AR1 %++], *AR6 %++ o *AR6 %– es el siguiente:

```
XAR6 = dir. 32b
   si( XAR6(7..0) == XAR1(7..0) )
    {
     XAR6(7..0)  = 0x00       ; XAR6 regresa a posición inicial
     XAR6(15..8) = sin cambio
    }
    else
```

```
    {
    si(dato 16b), XAR6(15..0) =+ 1 ; incrementa XAR6 dentro del buffer
    si(dato 32b), XAR6(15..0) =+ 2 ; hasta que XAR6(7..0) = XAR1(7..0)
    }
    XAR6(31..16) = Sin cambio
    ARP = 6
```

Buffer circular con AMODE = 1

Algoritmo con modificador *XAR6[AR1%++], XAR6 apunta a la dirección inicio del buffer, XAR1(31..16) + 1 contiene la longitud del buffer y XAR1(15..0) el índice o desplazamiento dentro del buffer:

```
XAR6 = dir. 32b + AR1
 si( XAR1(15..0) == XAR1(31..16) )
   {
    XAR1(15..0)  = 0x0000     ;  XAR1 regresa a posición inicial
        }
   else
   {
   si(dato 16b), XAR1(15..0) =+ 1 ; incrementa XAR1 dentro del buffer
   si(dato 32b), XAR1(15..0) =+ 2 ; hasta que XAR1(15..0) = XAR1(31..16)
   }
   XAR1(31..16) = Sin cambio
   ARP = 6
```

4.2.6. Direccionamiento de registros

Este modo consiste en la transferencia, intercambio y operaciones con los registros principales, tales como el acumulador ACC, el producto P, el temporal XT, XARi, AH, AL, PH, PL, T, TL DP, SP y los registros ARi, que en algunos casos se accede a su parte alta (H) o baja (L) o los 32 bits, cuando se copian 16 bits a la parte alta o baja, la otra parte no se afecta. En la tabla 4.6 se especifican las abreviaturas de los registros. En todos los casos el estado del bit AMODE no afecta.

Tabla 4.6. Operandos de direccionamiento a registros

Operando	Descripción
@ACC	Accede al contenido del ACC (32b)
@P	Accede al contenido del registro P (32b)
@XT	Accede al contenido del registro XT (32b)
@XARi	Accede al contenido del registro XARi (32b)
@AL	Accede al contenido del AL (16b)
@AH	Accede al contenido del AH (16b)
@PL	Accede al contenido del PL (16b)
@PH	Accede al contenido del PH (16b)
@TL	Accede al contenido del TL (16b)
@SP	Accede al contenido del SP (16b)
@ARi	Accede al contenido del ARi (16b)
@AH	Accede al contenido del AH (16b)

Algunos ejemplos de direccionamiento entre registros:

```
*  A 32 bits
     MOVL  XAR2,@ACC    ; carga XAR2 con ACC
     MOVL  @ACC,XT      ; carga ACC con XT
     ADDL  ACC,@ACC     ; ACC = ACC + ACC
     MOVL  XAR1,@P      ; carga XAR1 con P
     MOVL  @P,XT        ; P = XT
     ADDL  ACC,@P       ; ACC = ACC + P
     MOVL  XAR7,@XT     ; XAR7 = XT
     MOVL  P,@XT        ; P = XT
     ADDL  ACC,@XT      ; ACC = ACC + XT
     MOVL  XAR1,@XAR3   ; XAR1 = XAR3
     MOVL  P,@XAR4      ; P = XAR4
```

```
     ADDL  ACC,@XAR5    ; ACC = XAR5

*  Utilizando sólo parte baja o alta a 16 bits
     MOV   PH,@AL       ; PH = AL
     ADD   AH,@AL       ; AH = AH + AL
     MOV   T,@AL        ; T  = AL

     MOV   PH,@AH       ; PH = AH
     ADD   AL,@AH       ; AL = AL + AH
     MOV   T,@AH        ; T  = AH
     MOV   PH,@PL       ; PH = PL
     ADD   AL,@PL       ; AL = AL + PL
     MOV   T,@PL        ; T  = PL
     MOV   PL,@PH       ; PL = PH
     ADD   AL,@PH       ; AL = AL + PH
     MOV   T,@PH        ; T  = PH

     MOV   PL,@T        ; PL  = T
     ADD   AL,@T        ; AL  = AL + T
     MOVZ  AR1,@T       ; AR1 = T  , XAR1H = 0

     MOVZ  AR5,@SP      ; AR5 = SP , XAR5H = 0
     MOV   AL,@SP       ; AL  = SP
     MOV   @SP,AH       ; SP  = AH
     MOVZ  AR1,@AR6     ; AR1 = AR6, XAR1H = 0
     MOV   AL,@AR7      ; AL  = AR7
     MOV   @AR3,AH      ; AR3 = AH
```

En este modo el uso del símbolo @ es opcional.

4.2.7. Modo de direccionamiento al espacio de dato, de programa y espacio I/O

Este modo permite la transferencia de datos entre diferentes espacios del mapa de memoria y puertos I/O, pueden presentarse varias posibilidades, como se muestra a continuación.

Del espacio de memoria dato a dato

```
MOV loc16,*(0..16)  ; dat(loc16)   = dat(0..16)
MOV *(0..16),loc16  ; dat(0..16) = dat(loc16)
```

Del espacio de puertos I/O a memoria y viceversa

```
OUT  *(PA),loc16      ; Escribe a puerto IO(0..PA) dato en loc16
IN   loc16,*(PA)      ; Lee un dato en puerto IO(0..PA)
                      ; y lo copia en dat(loc16)
```

Del espacio de memoria de programa

```
XPREAD loc16,*(pma)   ; (loc16) = Prog[0x3F:pma]
XMAC   P,loc16,*(pma) ; ACC = ACC + P << PM
                      ; P = dat(loc16) * prog(0x3F:pma)
XMACD  P,loc16,*(pma) ; ACC = ACC + P << PM
                      ; P = dat(loc16) * prog(0x3F:pma)
                      ; dat(loc16+1) = dat(loc16)
MAC P,loc16,0:pma     ; ACC = ACC + P << PM
                      ; P = dat(loc16) * prog(0:pma)
```

Direccionamiento al espacio de programa

```
XPREAD  loc16,*AL     ; dat(loc16)    = prog(0x3F:AL)
XPWRITE *AL,loc16     ; prog(0x3F:AL) = dat (loc16)

MAC  P,loc16,*XAR1++  ;  ACC = ACC + P << PM
                      ;  P = dat[loc16] * Prog[*XAR1++]
DMAC ACC:P,loc32,*XAR2++
                      ; ACC=ACC +(dat(loc32).MSW * prog(*XAR2++).MSW)>>PM
                      ; P =  P + (dat(loc32).LSW * prog(*XAR7).LSW) >> PM
QMACL P,loc32,*XAR3++ ; ACC = ACC + P >> PM
                      ; P = (dat(loc32) * prog(*XAR3++)) >> 32
IMACL P,loc32,*XAR4++ ; ACC = ACC + P
                      ; P = (dat(loc32) * prog(*XAR4++)) << PM
PREAD  loc16,*XAR5    ; (loc16) = prog(*XAR7)
PWRITE *XAR7,loc16    ; prog(*XAR7) = dat(loc16)
```

4.2.8. Modo de direccionamiento byte

Se utiliza normalmente para direccionar datos de un byte de una localidad de memoria, o una parte de longitud un byte del acumulador y viceversa. En el siguiente ejemplo el significado de las abreviaturas son:

- AH: acumulador parte alta.
- AL: acumulador parte baja.
- LSB: byte menos significativo.
- MSB: byte mas significativo.
- loc16 = *+XARi[offset], offset impar toma la parte LSB del dato de 16 bits, offset par toma la parte MSB del dato de 16 bits.

```
MOVB  AX.LSB,loc16   ; AX.LSB = un byte de loc16, AX.MSB no cambia
MOVB  AX.MSB,loc16   ; AX.MSB = un byte de loc16, AX.LSB no cambia
MOVB  loc16,AX.LSB   ; un byte de loc16 = AX.LSB, loc16.MSB no cambia
MOVB  loc16,AX.MSB   ; un byte de loc16 = AX.MSB, loc16.LSB no cambia
```

Este modo de direccionamiento puede ser útil para intercambio de bytes, el cambio de ordenamiento de formatos "little endian" a "big endian" y viceversa, encriptado y desencriptado de datos, extracción de bytes, manejo de la memoria como bytes, etc.

Resumen

En este capítulo se han descrito el mapa de memoria de varias familias C28x, la sintaxis de instrucciones de ensamblador, los bloques principales de memoria, los modos de direccionamiento para transferir datos y algunos ejemplos con bloques de programas que realizan operaciones simples. Es importante señalar que el conocimiento de estos conceptos aunados a la teoría son fundamentales para entender los demás capítulos de este trabajo y realizar aplicaciones con estas familias.

Capítulo 5

Unidad de control

El sistema de control del C28x está conformado fundamentalmente por la lógica del generador de direcciones en memoria de programa, registros de control y estado ST0 y ST1, el contador de programa (PC), el apuntador de pila (SP), la señal de reset externo, las interrupciones, los registros de estado y el contador de repetición (RPTC) e instrucciones que cambian el flujo del programa, el código de seguridad y el watchdog [18], [20].

En este capítulo se tratarán la mayoría de estas partes proponiendo ejemplos que incluyan lo que se ha visto anteriormente. Aunque las interrupciones son parte del sistema de control se abordarán por separado en el siguiente capítulo debido a su relevancia y extensión.

5.1. Registros de control

Los registros de control determinan cómo va operar una máquina digital, por tanto, algunos de éstos se deben configurar desde un inicio. Los registros principales de control del C28x son: el PC, el SP, RPTC, de estado ST0 y ST1, que se describen a continuación.

5.1.1. Contador de programa (PC)

El contador de programa PC es un registro de 22 bits y permite direccionar hasta 4M palabras de memoria programa externas o internas en la búsqueda de las instrucciones, un operando inmediato, o una constante de 16 bits en memoria de programa. Al direccionar la memoria de programa, la dirección que está en el PC es puesta en el bus PAB.

5.1.2. Apuntador de pila (SP)

Siempre mantiene la dirección superior al último dato accedido en la pila. El apuntador de pila permite utilizar una pila en la memoria de datos por software, este registro es de 16 bits, por lo que sólo se pueden direccionar 64Kw del espacio de datos, cuando se utiliza para

direccionar datos la parte alta de una dirección es forzada a cero.
La operación de la pila es de la siguiente forma:

- La pila crece de la parte baja de la memoria a la parte alta.
- El SP siempre apunta la próxima localidad vacía en la pila.
- En el reset el SP es inicializado con la dirección 0000 0400h.
- Cuando se salva algún valor de 32 bits en la pila, la palabra menos significativa se salva primero y en la siguiente localidad la palabra más significativa (formato "little endian").
- Cuando ocurre un sobreflujo del SP, es decir, que tiene una dirección más allá de FFFFh o 0000h, el SP opera en forma circular.

La pila
El "stack" o la pila, es utilizado cuando se hace un llamado a subrutina, se atiende una interrupción y también puede ser accedido a través de las instrucciones PUSH y POP. Las instrucciones POP y PUSH permiten utilizar el stack para el almacenamiento temporal de la memoria de datos.

Los registros ST0 y ST1 tienen asociado un nivel de profundidad en la pila para salvarse automáticamente cuando ocurre una interrupción. En el retorno de la rutina de atención de interrupción, con la instrucción IRET le devuelve a los registros de estado sus valores originales anteriores a la interrupción.

5.2. Instrucciones que cambian el flujo del programa

En general, se puede decir que las instrucciones de un programa en el C28x se ejecutan en forma secuencial, es decir, que el contador de programa (PC) se va incrementando consecutivamente y el pipeline siempre está lleno sin ninguna pérdida de sincronía, sin embargo, cualquier aplicación o programa necesita cambios del flujo del programa, toma de decisiones y ejecución de otros bloques de programa, por tanto, existen instrucciones que alteran el flujo del programa tales como: saltos, llamadas a subrutina, atención de rutinas de interrupción, realización de ciclos y retornos de subrutinas. Estas instrucciones afectan el contador de programa PC y el pipeline.

5.2.1. Saltos incondicionales

Un salto incondicional siempre se ejecuta, transfiere el control del programa a una nueva localización en memoria programa durante su ejecución, el PC es cargado con la dirección de salto e inicia la ejecución de una nueva sección de código. Cuando la instrucción de salto alcanza el estado de ejecución de pipeline, las dos instrucciones siguientes ya han sido buscadas. Estos saltos se observan en la figura 5.1.a.

5.2.2. Saltos condicionales

El C28x permite efectuar saltos condicionales con las instrucciones indicadas en la tabla 5.1. El salto se efectúa si la condición se cumple, de lo contrario el programa continúa en la siguiente instrucción. Las condiciones que se pueden evaluar se enumeran en la tabla 5.2. Los saltos condicionales requieren uno o más ciclos que los saltos incondicionales, su operación se muestra en la figura 5.1.b.

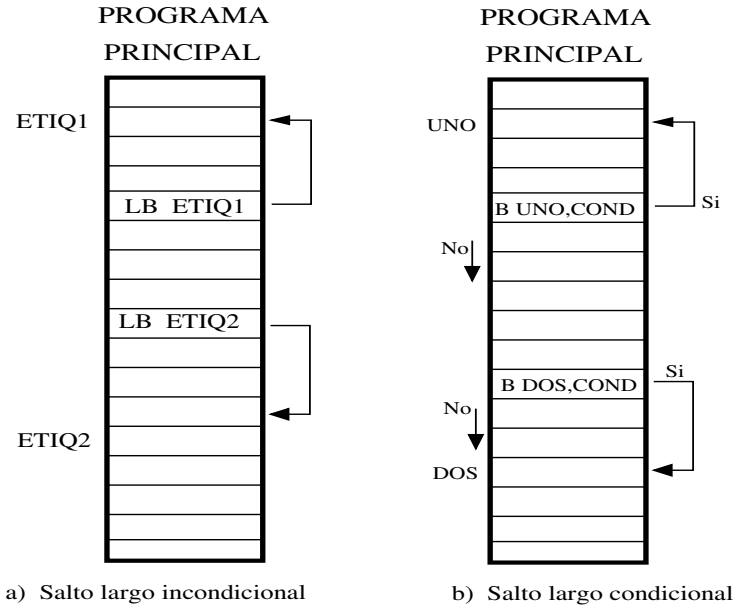


Figura 5.1. Saltos incondicional y condicionales

Las siguientes instrucciones ejecutan saltos condicionales: B, BANZ, BAR, BF, SB, SBF, XBANZ; llamada a subrutina condicional XCALL y retorno condicional de subrutina XRETC.

5.2.3. Saltos dinámicos

Este tipo de saltos permite variar la dirección de salto en forma dinámica, es decir, realizar saltos computados o calculados. Este proceso se realiza a través del registro XAR7 donde de antemano se ha cargado la dirección del salto. En la figura 5.2 se muestra como se realiza este proceso.

Tabla 5.1. Saltos y llamada a subrutina

Instrucción	**Descripción**
SB 8 bits,COND	Salto corto
SBF 8 bits,EQ|NEQ|TC|NTC	Salto corto rápido (± 8 bits)
B 16 bits,COND	Salto relativo rápido
BF 16 bits,COND	Salto rápido
LB 22 bits	Salto absoluto
LB *XAR7	Salto dinámico
BANZ 16 bits, ARi- -	Salta si ARi != 0, ARi = ARi -1
BAR 16 bits, ARi, ARj, EQ|NEQ	Salta con comparación de ARi con ARj
BF 16 bits,COND	Salto rápido
FFC XAR7,22 bits	Llamada rápida a subrutina
LC 22 bits	Llamada incondicional a subrutina
LC *XAR7	Llamada dinámica dirección apuntada por AR7
LCR 22 bits	Llamada incondicional a subrutina usando RPC
LCR *XARi	Llamada dinámica usando RPC
LOOPZ loc16, #16 bits	Ciclo infinito mientras (loc16) & 16 bits = 0
LOOPNZ loc16,#16 bits	Ciclo infinito mientras (loc16) & 16 bits != 0
LRET	Retorno largo
LRETE	Retorno largo habilitando interrupciones
LRETR	Retorno largo utilizando RPC
	RPC: registro apuntador de retorno del PC
IRET	Retorno de interrupción

Tabla 5.2. Condiciones generales

Condición	**Descripción**	**Operando**
ACC=0	ACC igual cero	EQ
ACC≠0	ACC no igual cero	NEQ
ACC <0	ACC menor que cero	LT
ACC ≤ 0	ACC menor o igual que cero	LEQ
ACC >0	ACC mayor que cero	GT
ACC ≥ 0	ACC mayor o igual que cero	GEQ
C=1 y Z=0	ACC muy grande	HI
C=1	ACC muy grande o el mismo	C
C=0	ACC pequeño	LO,NC
C=0 o Z=1	ACC pequeño o igual	C,Z
OV=1	Existe sobreflujo en ACC	OV
OV=0	No existe sobreflujo en ACC	NOV
C=1	Existe acarreo en ACC	C
C=0	No existe acarreo en ACC	NC
TC=1	Bandera de control activa	TC
TC=0	Bandera de control inactiva	NTC
/BIO=1	Señal externa /BIO=1	NBIO
Ninguna	Operación incondicional	UNC

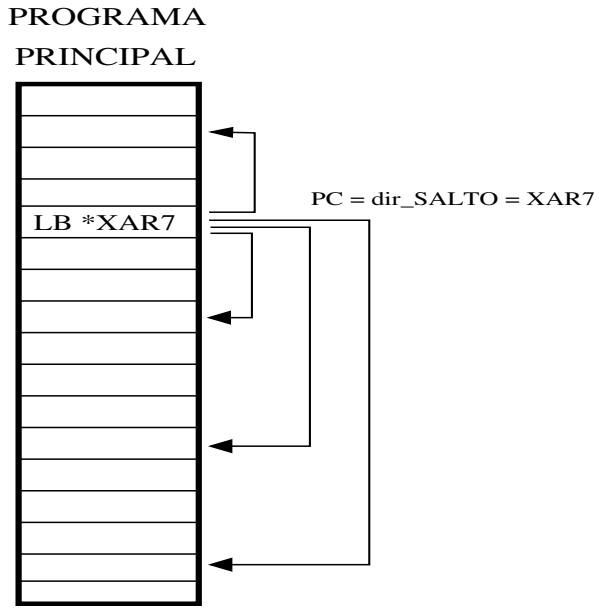


Figura 5.2. Saltos dinámicos

5.2.4. Instrucciones de llamada a subrutina

Una subrutina es un subprograma o una función que cuando es invocada se transfiere el control temporalmente a otra localidad de memoria programa. En el programa principal, la dirección (PC+2) de la instrucción siguiente es salvada en la pila, esta dirección es utilizada para el retorno de la ejecución de la subrutina. El proceso de una llamada incondicional a subrutina se realiza como se muestra en la figura 5.3.

5.2.5. Llamadas incondicionales a subrutina

Este tipo de llamadas siempre se ejecuta. El PC es cargado con una dirección de programa especificada y la ejecución de la subrutina empieza en esa dirección dirSUB. Para finalizar la subrutina debe existir una instrucción de retorno, esa instrucción recupera de la pila la dirección de la instrucción siguiente al llamado de subrutina, y se continúa con la ejecución del programa. Este proceso se observa en la figura 5.3.

```
LCR  dirSUB ; Carga el PC con la dirección dir_SUB
            ; Guarda PC+2 en la pila para el retorno de subrutina
LCR  *XARi  ; Llamada dinámica a subrutina
```

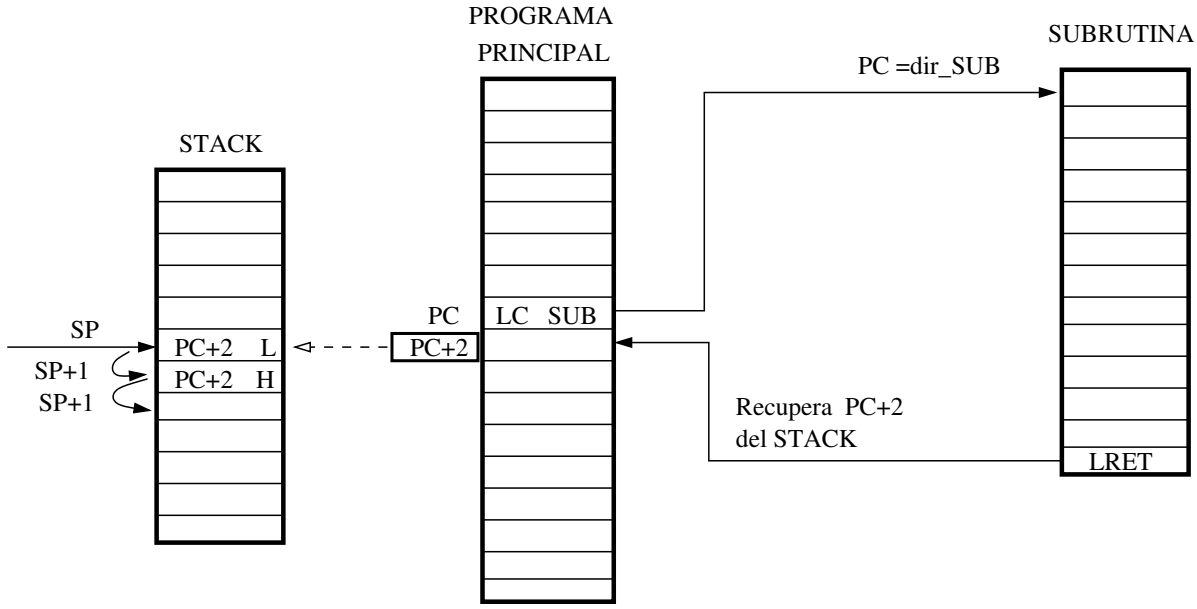


Figura 5.3. Llamada incondicional a subrutina

5.2.6. Llamadas dinámicas a subrutina

En la figura 5.4 se muestra una llamada a subrutina en forma dinámica, esto es equivalente a una llamada computada a subrutina, es decir, que se puede cambiar el acceso a otra subrutina cuando se ejecute de nuevo la llamada, esto se logra ubicando de antemano el registro XAR7 con la dirección de la subrutina que se quiera llamar.

5.2.7. Retorno de subrutinas

El retorno de una subrutina es el mecanismo para la finalización correcta de una subrutina y la continuación del programa que la invocó. Este procedimiento es válido para subrutinas normales y subrutinas de interrupción. En el retorno se debe pasar la dirección alta de la pila (TOS) al PC, es decir, la dirección de la siguiente instrucción al llamado de subrutina. Se utiliza la instrucción LRETR sin operandos. La dirección de retorno de subrutina puede ser leída de la pila o el registro RPC.

5.2.8. Instrucciones de repetición

Es una instrucción que opera como un ciclo que ejecuta una instrucción. Con la instrucción RPT #N se puede ejecutar N+1 veces la siguiente instrucción, donde la constante N es cargada en el contador de repeticiones RPTC (la constante N puede ser de 8 o 16 bits,

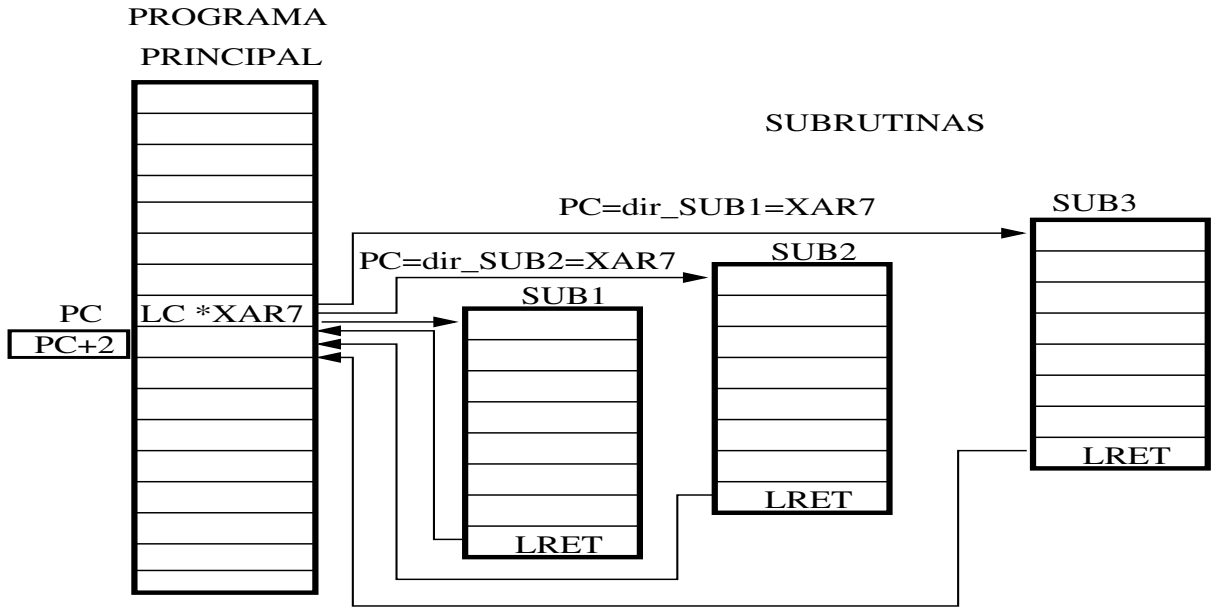


Figura 5.4. Llamada dinámica a subrutina

la instrucción RPT también puede operar en modo directo e indirecto). En la figura 5.5.a se ilustra su diagrama de flujo.

- Cuando se inicia la ejecución de la instrucción RPT, el registro contador de repetición RPTC es cargado con el valor de N, RPTC es decrementado cada vez que se repite la instrucción del ciclo.
- Esta instrucción es no interrumpible.
- Puede anidarse dentro de ciclos BANZ.

Contador de repetición (RPTC)

Es un registro de 16 bits y es cargado con el valor N cuando se utiliza la instrucción RPT, N puede ser un número desde 0 a 65,535 usando RPT #N. Esta instrucción es comúnmente utilizada con multiplicación/acumulación, movimiento de bloques, transferencias de I/O y tablas de lectura/escritura. Cuando la instrucción de repetición RPT es decodificada, todas las interrupciones mascarables son deshabilitadas. El registro RPTC también puede ser cargado con un valor de 16 bits en memoria dato a través de la instrucción RPT en modo directo o indirecto. Hay que hacer notar que no todas las instrucciones se pueden utilizar con RPT, en las instrucciones que no es válido se ignora el ciclo de repetición y la instrucción siguiente a RPT se ejecuta una sola vez.

Por ejemplo, si se quiere ejecutar la siguiente sumatoria

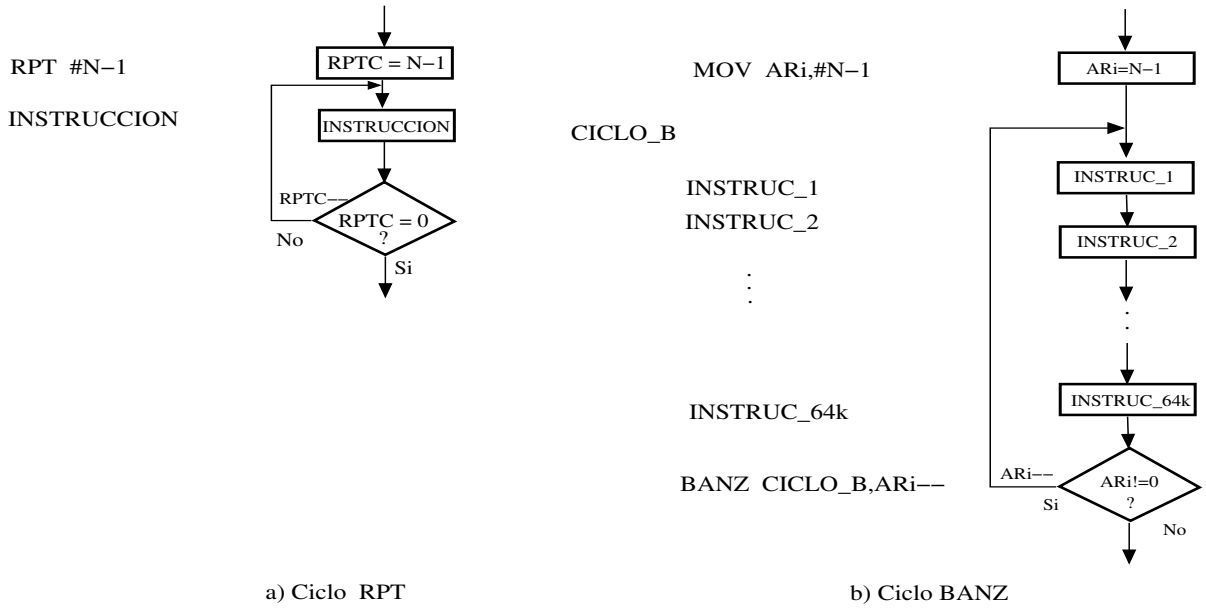
$$suma = \sum_{n=0}^{N-1} x_n \tag{5.1}$$

se puede realizar con el código

```
MOVL   XAR1,#x   ; XAR1 apunta a dir. inicio de datos x
MOVW   DP,#suma  ; Página de datos DP en variable suma
ZAPA             ; ACC = 0, P = 0
RPT #9           ; repite 10 veces la siguiente instrucción
ADD ACC,*XAR1++  ; ACC = ACC + dato apuntado por XAR1
                 ; XAR1 = XAR1 +1
MOVL @suma,ACC   ; Salva ACC en loc. suma
```

Ciclo de varias instrucciones

Otro tipo de salto condicional que se utiliza para realizar ciclos es con la instrucción BANZ, ésta evalúa si un registro auxiliar ARi es diferente de cero para efectuar el salto del PC a alguna dirección en memoria de programa. El programa fluye como se muestra en la figura 5.5.b, el código siguiente ejemplifica su forma de utilizarlo.



a) Ciclo RPT b) Ciclo BANZ

Figura 5.5. Ciclos RPT y BANZ

```
         MOV   ARi,#N-1       ; ARi = N-1, contador del ciclo
CICLO     ---
          ---
          ---
         BANZ  CICLO,ARi--    ; Carga el PC con la dirección de CICLO
                              ; si  ARi !=0
                              ; ARi = ARi - 1
                              ; de lo contrario PC=PC+2
```

Almacenamiento condicional

Permite realizar almacenamiento o movimientos de datos o registros si la condición de la instrucción se cumple, de lo contrario no realiza el movimiento.

```
* Ejemplos
     MOV  loc16,AL,COND      ; Si condición es cierta:
                             ; (loc16) = AL
     MOV  loc16,#8bit,COND   ; Si condición es cierta:
                             ; (loc16) = Cte. de 8 bits
     MOV  loc32,ACC,COND     ; Si condición es cierta:
                             ; (loc32) = ACC
```

5.3. Registros de estado

Estos registros contienen algunos estados, ciertas condiciones y modos de operación del DSP. Los registros de estado pueden ser almacenados dentro de la memoria dato y cargados de memoria dato, permitiendo así que el estado de la máquina sea salvado y restaurado para interrupciones y subrutinas. Cada registro contiene conjuntos de bits o bits individuales con diferente función.

Los DSP C28x contienen dos registros de estado ST0 y ST1, y éstos se utilizan para configurar modos de operación y estados de la máquina, sobre todo el resultado de las operaciones de la ALU. Todos los bits de este registro son afectados en la etapa dos de decodificación del pipeline. Estos registros se describen las tablas 5.3 y 5.4.

Los tres bits del campo PM se pueden configurar con la instrucción SPM n, donde n indica la cantidad de corrimientos sobre el registro producto, y n = +1,0,-1,-2,-3,±4,-5,-6, un corrimiento con signo positivo (+) es a la izquierda y con signo negativo (-) a la derecha. El caso n=-4 es cuando el bit AMODE = 0 y n= +4 para AMODE = 1. En la instrucción donde se requiera el corrimiento sobre el registro P, se indica con el operando: P $\ll$ PM.

Tabla 5.3. Registro de estado STO

Bits	Nombre	Función
15..10	OVC/OVCU	
	OVC	Contador de sobreflujo
	Si OVM = 0	ACC alcanza un sobreflujo normal OVC sigue el sobreflujo
	Si OVM = 1	ACC se satura a 7FFF FFFFh o 8000 0000h OVC se incrementa o decrementa en uno en sentido opuesto al valor de ACC
	OVCU	Contador de operaciones sin signo, se incrementa cuando existe un acarreo, se decrementa cuando existe un préstamo
9..7	PM	Bits de modo corrimiento del producto, corrimientos +1,0,-1,-2,-3,±4,-5,-6
6	V	Bandera de sobreflujo
5	N	Bandera de valor negativo
4	Z	Bandera de cero
3	C	Bit de acarreo
2	TC	Bandera de prueba o control
1	OVM	Bit de modo de sobreflujo o saturación
0	SXM	Extensión de signo 16 a 32 bits SXM = 0, aritmética sin extensión de signo SXM = 1, aritmética con extensión de signo

Tabla 5.4. Registro de estado ST1

Bits	**Nombre**	**Descripción**
15..13	ARP	Apuntador de registros auxiliares
12	XF	Refleja el valor de pin XFS
		- se pone a uno con instrucción SETC XF
		- se limpia con instrucción CLRC XF
11	M0M1MAP	Modo de mapeo de los bloques de memoria M0 y M1
		1 : operación normal, 0: para pruebas del fabricante
10	reservado	
9	OBJMOD	Modo de compatibilidad con familias C27x
8	AMODE	Bit de modo de página PAG0,
		selecciona el modo de direccionamiento
7	IDLE	En "1", existe un proceso IDLE
6	EALLOW	Habilita el acceso a emulación y a registros protegidos,
	en 1	con instrucción EALLOW se habilita,
	en 0	con instrucción EDIS se deshabilita
5	LOOP	Un ciclo LOOP está en proceso
4	SPA	Alineamiento de SP
	SPA = 0	El SP no se alinea a una dirección par
	SPA = 1	El SP se alinea a direcciones par
3	VMAP	Bit de mapeo de vectores de interrupción
		en memoria programa
	VMAP =0	En parte baja 00 000016 .. 00 003Fh
	VMAP =1	En parte alta 3F FFC016 .. 3F FFF FFFFh
2	PAGE0	Configuración de modo de direccionamiento directo
	PAGE = 0	Utiliza SP para direccionamiento directo
	PAGE = 1	Direccionamiento directo modo paginado
1	DBGM	Máscara de habilitación de depuración,
		el emulador no puede acceder registros en tiempo real
0	INTM	En cero habilita las interrupciones mascarables

5.4. Módulo de código de seguridad (CSM)

Una característica importante de la familia C28x es que previene el acceso y visibilidad de la memoria por personas no autorizadas, por tanto, evita la violación del código o la ingeniería en inversa. Generalmente, CSM se utiliza cuando se tiene una aplicación final muy bien probada y depurada, y se va a escribir en memoria flash, una vez que el "password" es escrito, el código queda asegurado poniendo a uno el bit 15 FORCESEC del registro de control y estado CSMSCR, dirección 0x00AEF.

La operación con un código de seguridad consiste de un password de 128 bits, éste es almacenado en 8 localidades consecutivas de 16 bits en memoria flash (0x3F 7FF8 a 0x3F 7FFF), adicionalmente las localidades 0x3F 7F80 a 0x3F 7FF5 deben escribirse con cero y no se deben utilizar para programa o dato. Para la operación sin código de seguridad, se escribe en los 128 bits sólo unos.

5.5. Instrucciones

En esta sección se resumen algunas instrucciones básicas en su forma más simple, ya que cada una de las instrucciones se pueden utilizar en diferente forma y modos de direccionamiento, para mayor detalle consultar el manual de usuario [18].

Tabla 5.5. Instrucciones básicas de ALU

Instrucción	**Ejecución**	**Descripción**
MOV AX,loc16	AX = loc16	Carga AX con dato en loc16
MOV loc16,AX	loc16 = AX	Almacena AX en loc16
MOV ACC,#16b,shift	AX = 16b≪shift	Carga ACC con #16b con corrimiento
ADD AX,loc16	AX = AX+loc16	Suma AX con dato en loc16
SUB AX,loc16	AX = AX-loc16	Resta AX con dato en loc16
AND AX,loc16	AX = AX AND loc16	AND AX con dato en loc16
OR AX,loc16	AX = AX OR loc16	OR AX con dato en loc16
XOR AX,loc16	AX = AX XOR loc16	XOR AX con dato en loc16
NOT AX	AX = NOT AX	Complemento de AX
NEG AX	AX = - AX	AX negativo
LSL ACC,shift	ACC = ACC≪shift	Corrimiento a la izquierda de ACC
LSR ACC,shift	ACC = ACC≫shift	Corrimiento a la derecha de ACC
LSL ACC,T	ACC = ACC≪ T	Corrimiento a la izquierda de ACC con valor de T(4..0)
LSR ACC,T	ACC = ACC≫T	Corrimiento a la derecha de ACC con valor de T(4..0)
LSL AX,shift	AX = AX≪shift	Corrimiento a la izquierda de AX
LSR AX,shift	AX = AX≫shift	Corrimiento lógico a la derecha de AX
ASR AX,shift	AX = AX≫shift	Corrimiento aritmético a la derecha de AX
LSL AX,T	AX = AX≪T	Corrimiento a la izquierda de AX con T(4..0)
LSR AX,T	AX = AX≫T	Corrimiento lógico a la derecha de AX con T(4..0)

Tabla 5.6. Instrucciones de multiplicación

Instrucción	**Ejecución**	**Descripción**
MOV T,@loc16	T = loc16	Carga T con dato en loc16
MPY ACC,T,@loc16	ACC = T*loc16	ACC = T por dato en loc16
MPY P,T,@loc16	P = T*loc16	P = T por dato en loc16
MPYB ACC,T,#8bu	ACC = T*8bu	ACC = T por constante de 8 bits sin signo
MPYB P,T,#8bu	P = T*8bu	P = T por constante de 8 bits sin signo
MOV ACC,P	ACC = P≪PM	Mueve P a ACC con corrimiento PM
MOVP T,@loc16	ACC = P≪PM T = loc16	Carga ACC con P con corrimiento PM carga T con dato en loc16
MOVA T,@loc16	ACC + = P≪PM T = loc16	ACC = ACC + P≪PM carga T con dato en loc16
MOVS T,@loc16	ACC - = P≪PM T = loc16	ACC = ACC - P ≪PM carga T con dato en loc16
MPYA P,T,#16b	ACC + = P≪PM P = T*#16b	ACC = ACC + P≪PM P = T*Constante 16b
MPYA P,T,@loc16	ACC + = P≪PM P = T*loc16	ACC = ACC + P≪PM P= T * dato en loc16
MPYS P,T,@loc16	ACC - = P≪PM P = T*loc16	ACC = ACC - P≪PM P= T*dato en loc16
MAC P,loc16,0:pma	ACC = ACC + P ≪ PM T = loc16 P = sig(T) * sig(Prog(0x00:pma))	Multiplicación acumulación signada a 16b
MAC P,loc16,*XAR7++	ACC = ACC + P ≪ PM T = loc16 P = sig(T) * sig(Prog(*XAR7++))	Multiplicación acumulación signada a 16b
QMPYAL ACC,XT,@loc32	ACC = (XT)*(loc32)	Multiplicación signada a 32b
IMPYAL P,XT,@loc32	P = u(XT)*u(loc32)	Multiplicación no signada a 32b
QMACL P,@loc32,*XAR7	ACC = ACC + P P = loc32*(loc32)	Multiplicación acumulación signada a 32b
IMPYAL P,@loc32,*XAR7	ACC = ACC + P P = u(loc32)*u(loc32)	Multiplicación acumulación no signada a 32b

Tabla 5.7. Instrucciones varias

Instrucción	**Ejecución**	**Descripción**
MOV loc16,AX,COND	loc16 = AX si COND	Almacenamiento condicional
MOVB loc16,#8b,COND	loc16 = #8b si COND	Almacenamiento condicional
MOV loc16,ACC,COND	loc16 = ACC si COND	Almacenamiento condicional
TBIT loc16,#bit n	ST0(TC) = loc16(bit n)	TC = bit n del loc16
TCLR loc16,#bit n	TC = loc16(bit n)	
	loc16(nbit) = 0	Limpia el bit n de loc16
TSET loc16,#bit n	TC = loc16(bit n)	
	loc16(bit n) = 1	Pone el bit n a uno de loc16
CMPB AX,#8b	prueba: AX - #8b sin signo	Afecta banderas C,N,Z
CMP AX,loc16b	prueba: AX - loc16	Afecta banderas C,N,Z
CMP loc16,#16b	prueba: loc16 - #16b signada	Afecta banderas C,N,Z
CMPL ACC,@P	prueba: ACC - P≪PM	Afecta banderas C,N,Z
INC loc16	loc16 = loc16 + 1	Incrementa en uno loc16
DEC loc16	loc16 = loc16 - 1	Decrementa en uno loc16
AND loc16,AX(#16b)	loc16 = loc16 AND AX(#16b)	AND con loc16
OR loc16,AX(#16b)	loc16 = loc16 OR AX(#16b)	OR con loc16
XOR loc16,AX(#16b)	loc16 = loc16 XOR AX(#16b)	XOR con loc16

Tabla 5.8. Instrucciones para máximos y mínimos

Instrucción	**Descripción**
MAX ACC,loc16	ACC = máximo(ACC y loc16)
MIN ACC,loc16	ACC = mínimo(ACC y loc16)
MAXL ACC,loc32	ACC = máximo(ACC y loc32)
MINL ACC,loc32	ACC = mínimo(ACC y loc32)
MAXCUL P,loc32	P = máximo(P y loc32)
MINCUL P,loc32	P = mínimo(P y loc32)

5.6. Ejemplos en lenguaje ensamblador

El propósito de estos ejemplos es introducir al lector en la realización e implementación de programas en lenguaje ensamblador utilizando los modos de direccionamiento del DSP, las instrucciones y estructuras de programación, con el fin de empezar a realizar pequeñas aplicaciones. Los primeros ejemplos parece muy elementales, sin embargo, tienen como objetivo distinguir claramente los modos de direccionamiento y practicarlos en conjunto.

5.6.1. Suma varias constantes en modo inmediato

En este ejemplo se suman cinco constantes en modo inmediato y el resultado se salva en modo directo. Las constantes se codifican en la instrucción, por lo tanto, este modo es muy inflexible, debido a que no podemos cambiar los valores de las constantes. Como puede observarse en este primer ejemplo, existe un bloque de desactivación del WatchDog para evitar que esté interrumpiendo al CPU y tener que estar atendiéndolo. En los ejemplos subsecuentes se prescindirá de este bloque.

```
*
*    Suma varias constantes en modo inmediato
*
        .global   _c_int00 ; Símbolo global para inicio de código
        .data              ; Sección de datos
WDCR    .set   07029h      ; Dirección registro de control WatchDog
CTE_WD  .set   0068h       ; Constante para desactivar el  WatchDog
D1      .set  1            ; Se define  D1 = constante = 1 entero
D2      .set  2            ; Se define  D2 = constante = 2
D3      .set  3            ; Se define  D3 = constante = 3
D4      .set  4            ; Se define  D4 = constante = 4
D5      .set  5            ; Se define  D5 = constante = 5
total   .word 0            ; Aparta una localidad para variable total
                           ; y la inicializa con 0

* SECCION DE CODIGO
        .text              ; Sección de código
_c_int00                   ; Inicio de código

* Deshabilitación del WatchDog
         EALLOW            ; Habilita escritura a registros protegidos
         MOVL XAR1, #WDCR  ; Registro XAR1 apunta dir, WDCR
         MOV  *XAR1,#0068h ; Desactiva WatchDog, escribe en WDCR
         EDIS              ; Deshabilita escritura a registros protegidos
*
```

```
         MOVW  DP, #total  ; Apuntador de página de datos en página de
                           ; variable total
         MOV   ACC, #D1    ; Mueve D1 a acumulador
         ADD   ACC, #D2    ; Suma D2 a acumulador
         ADD   ACC, #D3    ; Suma D3 a acumulador
         ADD   ACC, #D4    ; Suma D4 a acumulador
         ADD   ACC, #D5    ; Suma D5 a acumulador
         MOV   @total, AL  ; Salva  la suma AL en localidad total
REGRESA  NOP               ; Ciclo infinito para fin de programa
         LB    REGRESA
        .end               ; Fin de ensamblado
```

5.6.2. Suma varios datos en modo directo

Este modo es más flexible que el modo inmediato, ya que se escriben datos a la memoria de dato, por lo que podemos estar cambiando el valor de las variable, sin embargo, si se requiere sumar muchos números, este modo deja de ser eficiente, ya que se requiere de una palabra de instrucción por cada suma.

```
*
*    Suma varios datos en modo directo
*
        .global   _c_int00 ; Símbolo global para inicio de código
        .data              ; Sección de datos
x1      .word 1            ; Aparta una localidad para variable x1
                           ; y la inicializa con el valor 1
x2      .word 2            ; Aparta una localidad para variable x1
                           ; y la inicializa con el valor 2
x3      .word 3
x4      .word 4
x5      .word 5
total   .word 0            ; Aparta una localidad para variable total
                           ; y la inicializa con 0
                .text      ; Sección de código
_c_int00
         MOVW  DP, #total  ; Apuntador de página de datos en página de total
         MOV   ACC, @x1    ; Mueve dato x1 a acumulador
         ADD   ACC, @x2    ; Suma x2 a acumulador
         ADD   ACC, @x3    ; Suma x3 a acumulador
         ADD   ACC, @x4    ; Suma x4 a acumulador
```

```
        ADD   ACC, @x5     ; Suma x5 a acumulador
        MOV   @total, ACC  ; Salva  la suma AL en localidad total
REGRESA NOP                ; Ciclo infinito para fin de programa
        LB    REGRESA
        .end                ; Fin de ensamblado
```

5.6.3. Suma varios datos en modo indirecto

En este ejemplo se observa la eficiencia del modo indirecto, ya que otro registro apunta a la dirección del dato a transferir. Los datos se pueden escribir como componentes de un vector.

```
*
*    Suma varios datos en modo indirecto
*
        .global _c_int00
        .data
N       .set    5
D       .word   1,2,3,4,5 ; Aparta cinco localidades para el vector
                          ; de datos D y le escribe los valores 1,2,3,4,5
total   .word   0
        .text
_c_int00
        MOVL XAR1,#D       ; Registro XAR1 apunta al arreglo D
        MOV  ACC, *XAR1++  ; Carga ACC con el dato apuntado por XAR1
                           ; Postincrementa XAR1: XAR1 = XAR1 + 1
        ADD ACC,*XAR1++    ; ACC = ACC + dato apuntado por XAR1
                           ; XAR1 = XAR1 + 1
        ADD ACC,*XAR1++    ; ACC = ACC + dato apuntado por XAR1
                           ; XAR1 = XAR1 + 1
        ADD ACC,*XAR1++    ; ACC = ACC + dato apuntado por XAR1
                           ; XAR1 = XAR1 + 1
        ADD ACC,*XAR1++    ; ACC = ACC + dato apuntado por XAR1
                           ; XAR1 = XAR1 + 1, apunta a loc. total
        MOV *XAR1,AL       ; Guarda la parte baja del acumulador en total
REGRE   NOP
        LB    REGRE
        .end
```

De los ejemplos anteriores, hemos visto que para sumar N datos se requiere $N-1$ instrucciones suma, si N es muy grande, estos programas resultan ineficientes, por tanto, una forma de mejorar estos programas es a través de un ciclo que repita la instrucción suma. Esto se logra con el modo de direccionamiento indirecto modificando el programa anterior y utilizando la instrucción de repetición RPT.

```
*
*    Suma varios datos en modo indirecto con instrucción RPT
*
        .global _c_int00
        .data
N       .set    5
D       .word   1,2,3,4,5 ; Aparta cinco localidades para el vector de
                          ; datos D y le escribe los valores  1,2,3,4,5
total   .word   0
        .text
_c_int00
         MOVL XAR1,#D     ; Registro XAR1 apunta al arreglo D
         MOV  ACC, #0     ; Se limpia acumulador
         RPT #N-1         ; Repite N  veces #N la siguiente instrucción
      || ADD ACC,*XAR1++  ; ACC = ACC + dato apuntado por XAR1
                          ; XAR1 = XAR1 + 1
        MOVW DP,#total    ; Ubica al DP donde se encuentra la página total
        MOV @total,AL     ; Guarda la parte baja del acumulador en total
REGRESA NOP
        LB   REGRESA
        .end
```

Promedio de una secuencia de datos

Para obtener el promedio de un conjunto o una secuencia de datos es necesario efectuar la suma de los datos y dividirla entre la cantidad de datos, matemáticamente se muestra en la ecuación (5.2).

$$x_{med} = \frac{1}{N}\sum_{i=0}^{N-1} x_i \tag{5.2}$$

En los ejemplos anteriores hemos realizado la suma de datos, entonces sería suficiente multiplicar la suma por el inverso de N, es decir, agregar el siguiente código:

```
      MOV   T,@total    ; Carga en T la suma total
      MPY   ACC,T,@Ninv ; ACC = total*Ninv
```

Al final, salvar el resultado en ACC y ajustarlo al formato adecuado.

5.6.4. Decimación de una secuencia

La transformada rápida de Fourier (FFT) radix 2 es un algoritmo de amplia aplicación en el procesamiento digital de señales, existiendo las versiones decimada en el tiempo y en la frecuencia. El concepto de decimación consiste en aplicar un reordenamiento a una secuencia de datos $N = 2^l$, donde este proceso se puede realizar de varias formas [3]. Los DSP utilizan el direccionamiento en acarreo inverso para realizar la decimación de una secuencia, este proceso se muestra en el siguiente ejemplo, utilizando $N = 16$, el lector puede extender el valor de N.

```
*
*        Decimación de una secuencia de datos x(n)
*        para realizar la FFT radix 2
*        BR : significa bit reverse o acarreo inverso
*
        .global _c_int00
        .data
xn      .word  0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 ; x(n) ordenada
xnd     .word  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0       ; xd(n) decimada
N       .set   16
        .text
_c_int00
        MOV  XAR1,#xn    ; AR1 = dir_inicio de xn
        MOV  AR0,#N/2    ; AR0 = N/2 para la decimación
        MOV  XAR2,#xnd   ; AR2 = dir inicio de xnd
        MOV  AR3,#N-1    ; AR3 contador de ciclo BANZ

Ciclo   NOP  *,ARP1      ; Selecciona a AR1 como registro en uso
        MOV  AL,*BR0+    ; AL = dato xn , AR1 = AR1 + BR(AR0)
        MOV  *XAR2+,AL   ; Mueve xn a xnd
        BANZ Ciclo,AR3-- ; Retorna a Ciclo si AR3 != 0, AR3 = AR3-1
NOP
        LB   FIN_R
        .end
```

5.6.5. Modo de direccionamiento de buffer circular

En este ejemplo se aplica de una forma simple el concepto de buffer circular y su modo de direccionamiento. Se tiene un vector **x** de 10 números y se requiere sumar sus componentes 15 veces, en el ejemplo el registro XAR6 opera en buffer circular y recorre 15 veces el bloque de memoria donde se encuentra el vector **x**.

```
*
*    Modo de direccionamiento de buffer circular
*
        .global _c_int00 ;
        .global _main
        .data
x       .word  1,2,3,4,5,6,7,8,9,10
y       .word  0
NB      .set   10
N2      .set   150
suma    .word  0
        .text
_main:
_c_int00
        SETC SXM           ; Modo extensión de signo
        CLRC AMODE         ; Bit AMODE = 0
        MOV AR1,#NB-1      ; AR1 = longitud del buffer - 1
        MOVL XAR6,#x       ; XAR6 apunta a dir. inicio de x
        ZAPA               ; Cero a ACC y P
        RPT #N2-1          ; Repite siguiente instrucción 150 veces
     || ADD ACC,*XAR6%++   ; ACC = ACC + dato apuntado por XAR6
                           ; XAR6 = XAR6 + 1 en forma circular
        MOVW DP,#suma      ; DP = página de suma
        MOV @suma, AL      ; Guarda resultado en loc. suma
FIN_S   NOP
        LB  FIN_S
        .end
```

5.6.6. Producto punto entre dos vectores

Debe hacerse notar que el producto punto en sí es una operación de productos y sumas como se muestra en la ecuación (5.3), por lo que es la base para la realización de convoluciones y de los filtros digitales, salvo los corrimientos de algunas secuencia, de ahí la importancia de realizarla en forma eficiente. De acuerdo a las diversas instrucciones que contienen los DSP

de las familias C28x, el producto punto se puede efectuar de varias formas, sin embargo, con la instrucción MAC es la forma más eficiente.

$$total = \sum_{i=0}^{N-1} x_i h_i \tag{5.3}$$

A continuación se presentan varias versiones de la realización del producto punto entre dos vectores x y h y al final se escribe un programa más óptimo utilizando la instrucción MAC dentro de un ciclo RPT.

Producto punto entre dos vectores utilizando instrucciones carga, multiplicación y suma:

```
        .global  _c_int00            ; símbolo global para inicio del
                                     ; programa
        .data                        ; sección de datos
x       .word   1,2,3,4,5,6,7,8,9,10 ; datos del vector x
h       .word   1,2,1,2,1,2,1,2,1,2  ; datos del vector h
total   .word   0                    ; resultado
N       .set    10                   ; define N = 10 = constante
WDCR    .set    07029h           ; dir. de registro de Control WatchDog

        .text                    ; sección de código
_c_int00                         ; inicia el código
         SETC   SXM              ; modo extensión de signo
         SPM    #0               ; sin corrimiento en el multiplicador
         MOVW   DP,#total        ; carga a DP página de total
         MOVL   XAR1,#x          ; XAR1 apunta a dir_r
         MOVL   XAR7,#h          ; XAR7 apunta a dir_h
         MOV    AR0,#N-1         ; AR0 = N, para contador de ciclo
         ZAPA                    ; ACC = 0, P =0
CICLO_P1
         MOV    T,*XAR1++        ; T = dato apuntado por XAR1
                                 ; XAR1 = XAR1 + 1
         MPY    P,T,*XAR2++      ; P = T * dato  apuntado por XAR2
                                 ; XAR2 = XAR2 + 1
         ADDL   ACC,P            ; ACC = ACC + P << PM
         BANZ   CICLO_P1,*AR0--  ; Regresa a CICLO_P1 si AR0 != 0
                                 ; AR0 = XAR0 - 1
         MOVL    @total,ACC      ; total = AL

FIN_R       NOP                  ; ciclo infinito
```

```
            LB    FIN_R

            .end
```

Carga del registro T y acumula, luego multiplica. Fuera del ciclo debe recuperar el último producto.

```
         ZAPA                    ; ACC = 0, P =0
CICLO_P1
         MOVA    T,*XAR1++       ; T = dato apuntado por XAR1
                                 ; XAR1 = XAR1 + 1
                                 ; ACC = ACC + P << PM
         MPY    P,T,*XAR2++      ; P = T * dato  apuntado por XAR2
                                 ; XAR2 = XAR2 + 1
         BANZ   CICLO_P1,*AR0--  ; Regresa a CICLO_P1 si AR0 != 0
                                 ; AR0 = XAR0 - 1
         ADDL   ACC,P            ; ACC = ACC + P << PM
```

Carga registro T, luego multiplica y acumula. Fuera del ciclo debe recuperar el último producto.

```
         ZAPA                    ; ACC = 0, P =0
CICLO_P1
         MOV    T,*XAR1++        ; T = dato apuntado por XAR1
                                 ; XAR1 = XAR1 + 1
         MPYA    P,T,*XAR2++     ; ACC = ACC + P << PM
                                 ; P = T * dato  apuntado por XAR2
                                 ; XAR2 = XAR2 + 1
         ADDL   ACC,P            ; ACC = ACC + P << PM
         BANZ   CICLO_P1,*AR0--  ; Regresa a CICLO_P1 si AR0 != 0
                                 ; AR0 = XAR0 - 1
              ADDL   ACC,P       ; ACC = ACC + P << PM
```

Programa completo con instrucción multiplicación acumulación dentro de un ciclo RPT.

```
*
*    Producto punto entre dos vectores con instrucción MAC
*
        .global  _c_int00            ; símbolo global para inicio del
                                     ; programa
        .data                        ; sección de datos
x       .word   1,2,3,4,5,6,7,8,9,10 ; datos del vector x
h       .word   1,2,1,2,1,2,1,2,1,2  ; datos del vector h
total   .word   0                    ; resultado
N       .set    10                   ; define N = 10 = constante
WDCR    .set    07029h               ; dir. de registro de Control WatchDog
        .text                        ; sección de código
_c_int00                             ; inicia el código
         CLRC   XF                   ; borra bandera XF
         SETC   SXM                  ; modo extensión de signo
         SPM    #0                   ; sin corrimiento en el multiplicador
         MOVW   DP,#total            ; carga a DP página de total
         MOVL    XAR1,#x             ; XAR1 apunta a dir_r
         MOVL    XAR7,#h             ; XAR7 apunta a dir_h
         ZAPA                        ; ACC = 0, P =0
            RPT     #N-1             ; repite MAC N-1+1 = N veces
         || MAC   P,*XAR1++,*XAR7++  ;  ACC = ACC + P
                                     ; T = dato apuntado por XAR1
                                     ; P = T * (dato apuntado por XAR7)
                                     ; XAR1 = XAR1 + 1
                                     ; XAR7 = XAR7 + 1
            ADDL    ACC,P << PM      ; ACC = ACC + P, recupera último producto
            MOVL    @total,ACC       ; total = AL
FIN_R       NOP                      ; ciclo infinito
            LB    FIN_R
            .end
```

5.6.7. Correlación y autocorrelación de señales discretas

La correlación es una operación matemática similar a la convolución, se utiliza para determinar la similitud entre dos señales, calcular retardos entre señales, verificar y calcular la periodicidad de un señal, calcular energías, etc. La correlación entre señales se encuentra en aplicaciones como radar, sonar, síntesis de voz, reconocimiento de voz, procesamiento de imágenes, geología y muchas áreas de la ingeniería [3]. La correlación entre dos señales $x(n)$ e $y(n)$ se define como $r_{xy}(n)$

$$r_{xy}(l) = \sum_{n=\infty}^{\infty} x(n)y(n-l) \qquad ; \qquad l = 0, \pm 1, \pm 2, \cdots \tag{5.4}$$

si $x(n)$=$y(n)$, se tiene la autocorrelación $r_{xx}(l)$ definida

$$r_{xx}(l) = \sum_{n=\infty}^{\infty} x(n)x(n-l) \qquad ; \qquad l = 0, \pm 1, \pm 2, \cdots \tag{5.5}$$

l es el desplazamiento entre señales cuando se calcula la sumatoria. La primera señal del subíndice se considera fija y la segunda móvil o desplazada en l. En el siguiente ejemplo se calcula $r_{xh}(l)$ para $l \geq 0$ si las dos señales $x(n)$ y $h(n)$ son de igual longitud N. En la implementación de $r_{xh}(l)$, en aritmética de punto entero siempre se debe ser cuidadoso en el manejo de los datos, debido a que las longitudes pueden ser grandes y la suma de productos puede desbordar los registros. En el ejemplo se supone que los datos de entrada están en un $Qi = 10$, que es pequeño en cuanto a precisión y la salida se ajusta $Qi = 5$. Este mismo ejemplo se puede aprovechar para implementar para la autocorrelación $r_{xx}(l)$, si se escribe en las localidades de $h(n)$ la señal $x(n)$, una programación más óptima de $r_{xx}(l)$ implicaría cambiar el programa.

```
*
*      Correlación rxh(l) entre dos señal discretas:
*      x(n) y h(n) en Qix=10
*      Se considera sólo l > 0
        .global _c_int00
        .data
N       .set    500
cont    .word   0       ; Contador de desplazamientos l
x       .space  N*16    ; Reserva N localidades  para x(n)
basx    .word   0
h       .space  N*16    ; h(n), Qih=10
bash    .word   0
rxh     .space  N*16    ; rxx(l), Qiy=5.
        .text
_c_int00
```

```
        SETC SXM
        SPM  +1         ; Qiy=Qih+Qix=Q20+Q1 =Q21
        MOVW DP,#cont   ; DP apunta a página de cont
        MOV  AR4,#N-1   ; Control de ciclo CORRELA
        MOVL XAR5,#rxh  ; Salida de datos
        MOVL XAR2,#h    ; XAR2 con Dir. de h
        MOV  AL,#N-1    ;
        MOV  @cont,AL   ;

CORRELA:
        MOVL XAR7,#x
        ZAPA
        RPT @cont               ; Repite la siguiente instrucción
                                ; cont + 1 vez
     ||    MAC P,*XAR2++,*XAR7++ ; ACC = ACC + P << 1  (Q21)
                                ; P = x(n)*h(n-l), XAR2++ y XAR7++
        ADDL ACC,P <<PM         ; Recupera último producto
        MOV *XAR5++,AH          ; Almacena en rxh

        DEC @cont               ; cont = cont -1
        MOVL XAR2,#h            ; XAR2 con Dir. inicio de h
        MOV  ACC,@XAR2
        ADD  ACC,#N
        SUB  ACC,@cont          ; Dir. XAR2 = Dir. incio + l
        MOV  XAR2,@ACC          ; Ajusta retardo ‘‘l’’ de h(l)

        BANZ CORRELA,AR4--

FIN     NOP
        LB  FIN_S
       .end
```

El lector puede verificar que en este ejemplo lo que realmente se desplaza l posiciones es el apuntador XAR2, y si observamos detalladamente la secuencia móvil realmente es $x(n)$.

5.6.8. Operación a 32 bits

Cuando se requiere mayor precisión numérica, una solución en esta arquitectura puede ser aprovechando las operaciones a 32 bits. Para esto se tienen varios casos:
- Que las entradas estén a 16 bits y las salidas a 32 bits.

- Entradas y salidas a 32 bits.
- Las entradas pueden estar a 16 y 32 bits y las salidas a 32 bits.

Entradas a 16 bits y salidas a 32 bits

Se puede considerar que es el modo de operación por defecto de los DSP C28x, donde los operandos de entrada se manejan a 16 bits y los resultados quedan en el registro producto P o en el acumulador ACC. Para obtener la salida a 32 bits, se debe salvar P o ACC completos, es decir, un movimiento de palabra larga que normalmente se le agrega una L al final en la instrucción MOV, es decir, MOVL.

Entradas y salidas a 32 bits

Los operandos son leídos y operados a 32 bits. Las cargas se hacen de dos localidades consecutivas de memoria hacia ACC, hacia P, entre registros, de memoria a memoria o de constantes a registros con instrucciones:

```
MOVL  ACC,@dat32    ;  Carga dat32 en ACC
MOVL  P,@dat32      ;  Carga dat32 en P
MOVL  XT,*XARi++    ;  Carga dato apuntado por XARi en XT
MOVL  XARi,@dat32   ;  Carga XARi con dat32
MOVL  XARi,#cte22   ;  Carga XARi con una constante de 22 bits
MOVL  ACC,P<<PM     ;  Carga ACC con P con corrimiento PM
MOVL  P,ACC         ;  Carga P con ACC
```

Operaciones a 32 bits

Como los registros ACC, P y XP de las unidades de proceso son de 32 bits, entonces es factible efectuar operaciones aritméticas a 32 bits entre estos registros; además, se pueden transferir datos a 32 bits para operar. Algunos ejemplos de instrucciones:

```
ADD   ACC,#cte16<<#0-15  ; ACC = ACC + cte*2^(shift 0-15)
ADD   ACC,@dat16<<#0-15  ; ACC = ACC + dat*2^(shift 0-15)
ADD   ACC,@dat16<< T     ; ACC = ACC + dat*2^(T(3..0))
ADDL  ACC,@loc32         ; ACC = ACC + loc32
ADDCL ACC,@loc32         ; ACC = ACC + loc32 + acarreo C
ADDL  ACC,@dat<<#0-15    ; ACC = ACC + dat*2^(shift 0-15)
ADDL  ACC,P<<PM          ; ACC = ACC + P^(PM)
ADDL  @loc32,ACC         ; loc32 = loc32 + ACC

SUB   ACC,#cte16<<#0-15  ; ACC = ACC - cte*2^(shift 0-15)
SUB   ACC,@dat16<<#0-16  ; ACC = ACC - dat*2^(shift 0-15)
```

```
SUB   ACC,@dat16<<T      ; ACC = ACC - dat*2^(T(3..0))
SUBL  ACC,@loc32         ; ACC = ACC - loc32
SUBBL ACC,@loc32         ; ACC = ACC - loc32 - ~C
SUBL  ACC,P<<PM          ; ACC = ACC - P^(PM)
SUBL  @loc32,ACC         ; loc32 = loc32 - ACC
SUBRL @loc32,ACC         ; loc32 = ACC - loc32

IMPYL   ACC,XT,@loc32    ; ACC= Bits bajos(sig.XT + sig.loc32)
IMPYL   P,XT,@loc32      ; P= Bits bajos(sig.XT + sig.loc32)<<PM
IMPYAL  P,XT,@loc32      ; ACC = ACC + unsig(P)
                         ; P= Bits bajos(sig.XT + sig.loc32)<<PM
IMPYAL  P,@loc32,*XAR7   ; ACC=ACC + unsig(P)
                         ; P= Bits bajos(sig.loc32 + sig.loc32)<<PM
IMPYSL  P,XT,@loc32      ; ACC = ACC - unsig(P)
                         ; P= Bits bajos(sig.XT + sig.loc32)<<PM

QMPYAL  ACC,XT,@loc32    ; ACC=ACC + P<<PM, P=XT*loc32>>32
QMACL   P,@loc32,*XAR7   ; ACC=ACC + P<<PM,  P=loc32*loc32>>32
QMPYL   ACC,XT,@loc32    ; ACC=XT*loc32>>32
QMPYL   P,XT,@loc32      ; P=XT*loc32>>32
QMPYSL  ACC,XT,@loc32    ; ACC=ACC - P<<PM, P=XT*loc32>>32
```

Los resultados se salvan a 32 bits de registros a memoria con instrucciones

```
MOVL  @dat32,ACC         ; Salva ACC en dat32
MOVL  @dat32,P           ; Salva P en dat32
MOVL  @dat32,XT          ; Salva XT en dat32
MOVL  @loc32,XARi        ; Salva XARi en loc32
MOVL  *XAR1++,XAR6       ; Salva XAR6 en loc. puntada por XAR1
```

Las entradas pueden estar a 16 y 32 bits, y las salidas a 32 bits

Los operandos de 16 bits deben convertirse a 32 bits a través de corrimientos aunque no sean de la misma precisión de 32, las operaciones se realizan a 32 bits y el resultado se guarda a 32 bits utilizando las instrucciones del caso anterior.

5.6.9. Aplicaciones propuestas

Para el lector interesado en incrementar sus conocimientos y destrezas en el manejo del DSP, se proponen algunas aplicaciones de regular complicación, para realizarlos en alguna de las tarjetas de los DSP Piccolo F28027 o F28069.

1. Rectificador de media onda: dada una entrada $x(n)$ senoidal de N puntos, descargarla a memoria y entregar como salida los valores positivos y los negativos los hace cero.

2. Para la señal del inciso anterior realizar la rectificación de onda completa:

3. Recorta una señal a un valor de umbral de la amplitud $\pm H$. Proponer el umbral de tal forma que algunas partes de la señal rebasen dicho umbral.

4. Calcular el máximo, el mínimo y las posiciones respectivas de una señal aleatoria $x(n)$ de N puntos propuesta.

5. Dada una señal aleatoria $x(n)$ de N puntos, calcular la autocorrelación normalizada.

6. Dada una señal periódica $x(n)$ de N puntos con ruido agregado, utilizando las propiedades de la autocorrelación, determinar el período de la señal.

7. Para los siguientes polinomios:
   $f_1(x) = x^5 - 10.5x^4 + 34.06x^3 - 35.1x^2 + 11.74x - 1.2$
   $f_2(x) = x^5 - 12.5x^4 + 37.99x^3 - 15.875x^2 - 0.38x + 0.16$
   $f_2(x) = x^5 - 15.5x^4 + 89.34x^3 - 228.58x^2 + 225.98x - 20.4$
   $f_4(x) = x^5 + 1.1x^4 - 73x^3 - 347.9x^2 - 348x + 270$
   $f_5(x) = x^5 + 0.9x^4 - 59.7x^3 + 132.1x^2 + 110.7x - 81$
   $f_6(x) = x^5 - 3.73x^4 - 8.415x^3 + 31.156x^2 - 12.0903x + 1.2448$
   $f_7(x) = x^5 - 11.86x^4 + 14.604x^3 + 7.4704x^2 - 15.401x + 4.5926$

   Evaluar cada uno de los siguientes incisos en aritmética de $L = 16$ y $L = 32$ bits verificando los errores de precisión:
   - Las raíces.
   - El área del polinomio en el intervalo de sus raíces.
   - Graficar la función $f(x)$ en el intervalo de sus raíces.

8. Completar el ejemplo de la correlación $r_{xh}(l)$ para valores de $l < 0$, y luego calcular $r_{xx}(l)$ y $r_{hh}(l)$

Resumen

Hasta este capítulo se han expuesto los fundamentos de hardware y software suficientes para que el lector pueda inicializarse en la programación y uso de los DSPs. Se profundizó en las estructuras de programación que son herramientas muy poderosas y sobre todo para hacer grandes aplicaciones al incluir el uso de interrupciones. Asimismo, en los siguientes capítulos se abundará sobre periféricos y aplicación en el manejo y configuración de interrupciones.

Capítulo 6

Implementación de filtros digitales

En la actualidad, los filtros digitales están presentes en la solución a problemas de medición, control, filtrado, análisis de señales, análisis espectral, comunicaciones, en general, en cualquier aplicación relacionada con procesamiento digital de señales (PDS), esto hace que un filtro digital (FD) sea un elemento fundamental en muchas aplicaciones.

En este capítulo se presentan algunas implementaciones de filtros digitales lineales e invariantes en el tiempo de respuesta finita al impulso (FIR), de respuesta infinita al impulso (IIR) y algunas aplicaciones utilizando la potencialidad de las instrucciones de la familia de DSP C28x.

6.1. Filtros de respuesta finita al impulso

Los filtros digitales de respuesta finita al impulso (FIR) son utilizados ampliamente en el PDS por su estabilidad y sus características de fase lineal que son de importancia en algunas aplicaciones de voz y audio. Debido a que la salida de este tipo de filtro sólo depende de la muestra actual de entrada y de *N-1* retardos de la entrada, también son conocidos como no recursivos, donde N es la longitud del filtro. Por las características de estos filtros, su implementación es precisamente la operación convolución entre la respuesta al impulso del filtro y una ventana de tiempo de la señal de entrada de longitud N, por lo que su implementación en un procesador de señales digitales (DSP) es muy óptima.

6.1.1. Estructuras de los filtros FIR

La estructura de un filtro digital es la forma de seleccionar cómo se van a efectuar las operaciones matemáticas, por tanto, la selección de una estructura es de especial interés cuando los filtros se van a implementar en arquitecturas DSP, ya que se debe considerar la cantidad de memoria, el número de operaciones matemáticas, el orden de las operaciones y

los efectos de la longitud finita de palabra de la arquitectura [10], [7], [2].

Un sistema lineal e invariante en el tiempo discreto (SLITD) puede ser descrito por una ecuación en diferencias

$$y(n) = \sum_{m=0}^{q} b_m x(n-m) - \sum_{k=1}^{p} a_k y(n-k) \tag{6.1}$$

Al aplicar la transformada Z, se obtiene la función de transferencia

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_q z^{-q}}{1 + a_1 z^{-1} + \cdots + a_p z^{-p}} \tag{6.2}$$

La respuesta al impulso unitario del sistema, $h(n)$ es la transformada inversa Z de la función de transferencia $H(z)$.

El *problema básico en el diseño de filtros digitales* es el cálculo de los coeficientes que aproximen las características ideales de una respuesta en frecuencia de un sistema SLITD $H_I(\omega)$ con la respuesta en frecuencia de un sistema $H(\omega)$. Ésta se obtiene al evaluar la ecuación (6.2) en $z = e^{j\omega}$ obteniendo la ecuación (6.3)

$$H(\omega) = \frac{Y(\omega)}{X(\omega)} = \frac{B(\omega)}{A(\omega)} = \frac{\sum_{k=0}^{q} b_k e^{-j\omega k}}{1 + \sum_{k=1}^{p} a_k e^{-j\omega k}} \tag{6.3}$$

Un filtro FIR sólo tiene coeficientes b_k=h_k, es decir, que tiene la característica de ser un sistema no recursivo, por lo que teóricamente siempre es estable. Su respuesta en frecuencia es

$$H(\omega) = \frac{Y(\omega)}{X(\omega)} = \sum_{k=0}^{q} h_k e^{-j\omega k} \tag{6.4}$$

Una desventaja de estos filtros es la necesidad de utilizar un mayor orden para lograr pendientes de corte pronunciadas, esto implica un mayor tiempo de procesamiento y mayor retardo en la respuesta. La respuesta al impulso de estos filtros normalmente son funciones similares a funciones $sen(x)/x$ con longitud finita, como se verá más adelante.

De la ecuación (6.2), un filtro FIR tiene la forma

$$H(z) = h_0 + h_1 z^{-1} + \cdots + h_{N-1} z^{-N+1} = \sum_{i=0}^{N-1} h(i) z^{-i} \tag{6.5}$$

con respuesta al impulso

$$h(n) = \begin{cases} h_i & 0 \leq i \leq N-1 \\ 0 & \text{otro } i \end{cases} \tag{6.6}$$

y ecuación en diferencias

$$y(n) = h_0 x(n) + h_1 x(1) + \cdots + h_{N-1} x(n - N + 1) = \sum_{i=0}^{N-1} h(i) x(n - i) \tag{6.7}$$

la cual es una convolución lineal de la entrada $x(n)$ con la respuesta al impulso $h(n)$ del filtro FIR, N es la longitud del filtro y es igual al número de coeficientes del filtro.

Se puede observar que la salida $y(n)$ del sistema FIR es una suma ponderada de los coeficientes $h(i)$ por la entrada actual $x(n)$ y las muestras retardadas; además, un filtro FIR se caracteriza por ser siempre estable y de fase lineal. La sumatoria que permite efectuar un filtro FIR es la operación de convolución de los coeficientes por una ventana temporal de una señal, como se observa en la figura 6.1.

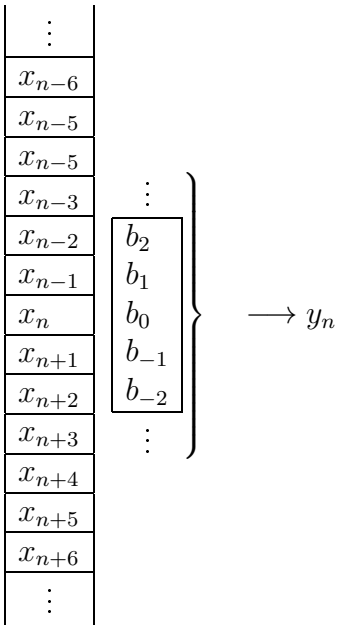


Figura 6.1. Convolución de una señal con los coeficientes del filtro

La estructura de un FD se entiende como la forma de realizar la implementación de la ecuación en diferencias del filtro (6.7) o la función de transferencia (6.5) y cómo se efectúan las operaciones matemáticas. Dependiendo de la implementación de las ecuaciones, se pueden tener varias estructuras o formas. La mayoría de métodos para generar estructuras se basan en algoritmos específicos, por tanto, se puede decir que existen muchas estructuras equivalentes para una misma función de transferencia [9].

En este trabajo, por su enfoque de aplicaciones en hardware, se utilizará únicamente la estructura directa, otras estructuras se pueden consultar en [10], [12], [2].

Forma directa

Es la estructura más sencilla y como su nombre lo indica, la ecuación (6.7) se implementa directamente utilizando los bloques básicos de un sistema discreto. Para cualquier longitud N se presenta un diagrama de bloques en la figura 6.2.

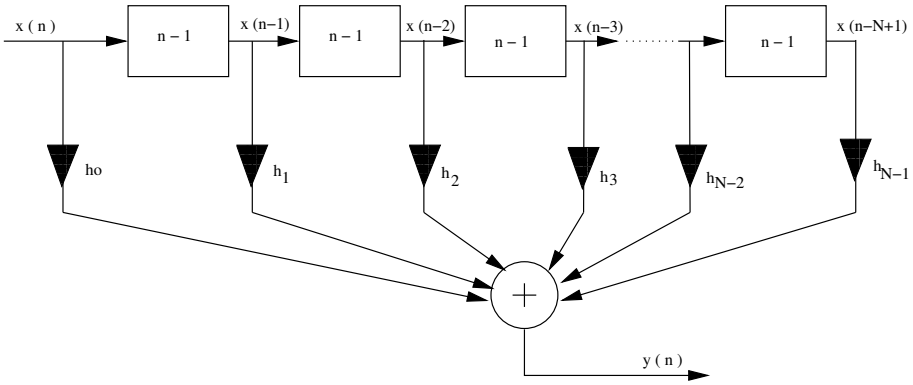


Figura 6.2. Forma directa de un filtro FIR

6.1.2. Implementación de líneas de retardo o buffer lineal

En esencia un filtro FIR es una suma de productos de los coeficientes $h(i)$ por las muestras retrasadas $x(n-i)$ de la señal de entrada. Para la implementación de estos filtros eficientemente es necesario tener la posibilidad de generar una línea de retardos de la señal $x(n)$, como se muestra en la figura 6.3.

La forma más simple de implementar una línea de retardo en un microprocesador es mediante un buffer lineal, donde un filtro de *N-1* retardos opera sobre las N muestras más recientes. Cada vez que se efectúa la sumatoria del filtro FIR, se adquiere un nuevo dato, se agrega a la parte superior del buffer y el dato inferior es descartado. Es decir, que una vez calculada la salida, un dato es movido a la localización siguiente para realizar el retardo.

En un procesador convencional una línea de retardo involucraría el acceso a dos localidades de memorias continuas y un almacenamiento temporal del dato (por lo menos tres instrucciones por cada dato desplazado).

En el siguiente código se utiliza la instrucción DMOV para la realización de líneas de retardo en filtros FIR. En la figura 6.3 se muestra el bloque de datos del buffer lineal y dónde debe de apuntar al inicio el registro XAR1.

```
*
*    Implementación de una línea de retardo
*    Con instrucción DMOV del DSP C28x
*
       MOVL XAR1,#xN1  ; XAR1 apunta al último dato del buffer
       RPT #N-1        ;
   ||  DMOV *--XAR1    ; Copia el dato apuntado por XAR1 a la
                       ; siguiente localidad y XAR1 = XAR1-1
```

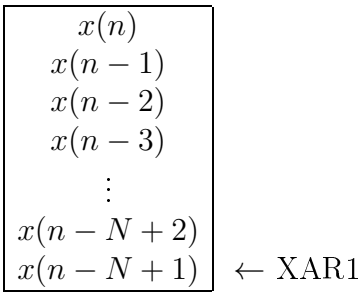


Figura 6.3. Buffer lineal

6.1.3. Implementación de filtros FIR

Para implementar la ecuación (6.7), se deben utilizar dos apuntadores, uno que inicie en la localidad $x(n)$ y el otro en la localidad $h(i)$, e ir multiplicando los valores respectivos, acumular productos, salvar el resultado $y(n)$ y posteriormente efectuar retardos sobre la señal de entrada $x(n)$. En la figura 6.4 se muestran los buffers en memoria necesarios para ubicar los valores de $h(i)$ y $x(n)$, en los siguientes bloques de código se muestran varias posibilidades para implementar un filtro FIR.

Filtro FIR con instrucción MAC y DMOV

```
SETC   SXM       ; Modo extensión de signo
MOVW   DP,#xn    ; Carga en DP página de xn
MOVL   XAR1,#x   ; XAR1 apunta a inicio de x(n) en Q12
MOVL   XAR2,#y   ; XAR2 apunta a inicio de y(n) en Q12
SPM    0         ; Salida del producto P sin corrimiento
MOV    AR4,#N-1  ; Carga N en AR4, N=tamaño de señal x(n)
```

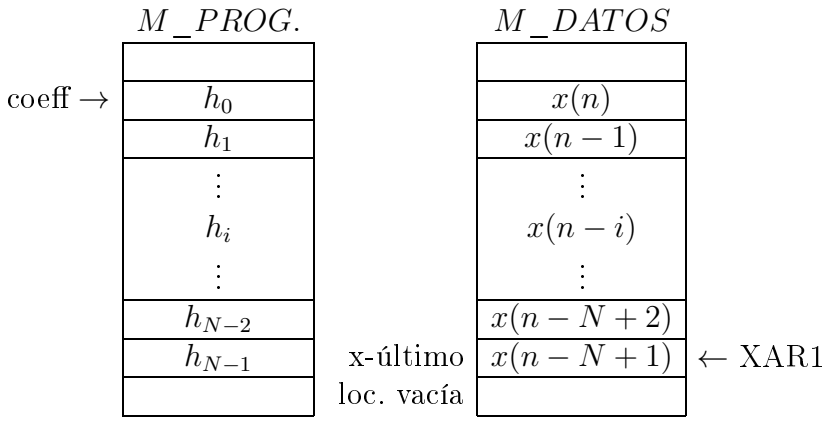


Figura 6.4. Buffers para $h(i)$ y $x(n-i)$

```
CICLO_M
         MOV AL,*XAR1++    ; Muestrea el dato x(n)
         MOV @xn,AL        ; Escribe x(n) al inicio del buffer
         MOVL  XAR7,#xn    ; XAR7 apunta al inicio del buffer
         MOVL  XAR3,#h     ; XAR3 apunta a inicio de coeficientes h(n)
         ZAPA              ; ACC = 0  y P = 0
         RPT  #Nf-1
     ||  MAC   P,*XAR3++,*XAR7++ ; Multiplica y acumula h(n)*x(n-i)
         ADDL  ACC,P<<PM
         LSL   ACC,#4      ; Ajusta ACC a Q28
         MOV   *XAR2++,AH  ; Escribe AL a salida y(n) en Q12
         MOVL  XAR3,#xnf   ; XAR3 apunta a loc. fin  del buffer
         RPT  #Nf-1        ; Nf tamaño del buffer
     ||  DMOV *--XAR3      ; Mueve los datos del buffer
         BANZ  CICLO_M,AR4-- ; Regresa a CICLO_M  si AR4!=0, AR4 = AR4-1
```

Filtro FIR con instrucción XMACD

```
         SETC   SXM
         SETC   OBJMODE
         SPM    0
         MOV    AR4,#N-1
CICLO_M
         MOV AL,*XAR1++       ; Muestrea el dato x(n)
         MOV @xn,AL           ; Escribe x(n) al inicio del buffer
         ZAPA                 ; ACC = 0  y P = 0
```

```
        MOVL  XAR3, #xnf    ; xnf última localidad del buffer
        NOP   *,ARP3        ; Selecciona XAR3 en uso al estilo c2xx
        RPT  #N
     ||  XMACD  P,*--,*(h)  ; MAC con DMOV, coeficientes h(n)
                            ; en memoria programa loc. 0x3F0000
        ADDL  ACC,P
        LSL   ACC,#4        ; Ajusta ACC a Q28
        MOV   *XAR2++,AH    ; Salva AH en Q12
        BANZ  CICLO_M,AR4-- ; Regresa a  CICLO_M si AR4!=0 y AR4 = AR4-1
```

Filtro FIR con instrucción MAC en buffer circular

```
        SPM    0            ; Salida del producto P sin corrimiento
        MOV    AL,*XAR2++   ; Muestrea el dato x(n)
        MOV    @xn,AL       ; Escribe muestra x(n) al inicio del buffer
        MOV    AR4,#N-1     ; N, tamaño de señal x(n)
        MOVL   XAR6,#xn     ; Ubica XAR6 al inicio del buffer de x(n)
CICLO_C
        MOVL  XAR7,#h       ; XAR7 apunta al inicio de coeficientes h(n)
        ZAPA
        RPT  #N
     ||  MAC  P,*AR6%++,*XAR7++ ; MAC en buffer circular
        ADDL  ACC,P << PM
        LSL   ACC,#4        ; Ajusta ACC a Q28
        MOV   *XAR3++,AH    ; Salva y(n) en Q12
        MOV   AL,*XAR2++    ; Muestrea el dato x(n)
        MOV   *XAR6,AL      ; Copia dato x(n) en el buffer
        BANZ  CICLO_C,AR4-- ; Regresa a  CICLO_C si AR4!=0 y AR4 = AR4-1
```

Filtro FIR promediador

Si se considera una señal con ruido agregado y con una relación señal a ruido (SNR) mediana, entonces una manera simple de eliminar el ruido o suavizar la señal, sería ir calculando el promedio de las muestras sobre una ventana de tiempo y recorrer la ventana. Es decir, que la $h(n)$ propuesta es $1/N$ en toda la ventana y cumple con la condición de simetría de los filtros FIR con diseño de fase lineal.

Este tipo de filtro también es conocido como *Moving average (MA)*, ya que realiza el promedio sobre la muestra actual de entrada $x(n)$ y *N-1* muestras pasadas. Si se visualiza una señal a baja frecuencia con ruido agregado (figura 6.5) y aplicando promedios a una

señal de entrada $x(n)$ en una ventana de tiempo de longitud N, se puede obtener la salida

$$y(n) = \sum_{i=0}^{N-1} \frac{1}{N} x(n-i) \tag{6.8}$$

si $h(n) = 1/N$, se tiene un filtro que realiza promedios sobre N muestras, aplicando la transformada Z

$$Y(z) = \frac{1}{N} \sum_{i=0}^{N-1} X(z) z^{-i} \tag{6.9}$$

Entre los filtros digitales, los filtros MA producen el menor ruido para bordes muy agudos, la cantidad de reducción de ruido es igual a la raíz cuadrada del número de puntos promediados, por ejemplo si $N = 100$, un filtro FIR MA reduce el ruido por un factor de 10. La función de transferencia es

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{N} \frac{1 - z^{-N}}{1 - z^{-1}} \tag{6.10}$$

y si $z = e^{j\omega}$, su respuesta en frecuencia es

$$H(\omega) = \frac{\operatorname{sen}(\pi\omega N/2)}{N \operatorname{sen}(\pi\omega/2)} e^{-j(N-1)\omega/2} \tag{6.11}$$

Debido a la respuesta en frecuencia, este tipo de filtros no puede separar eficientemente una frecuencia de otra [14].

Para los filtros FIR y de la figura 6.5, se observa que las primeras *N-1* primeras salidas corresponden al retardo del filtro o el comportamiento transitorio, y para tiempos $n >$*N-1* el buffer de entrada está completamente lleno y el filtro opera sobre una ventana completa de datos.

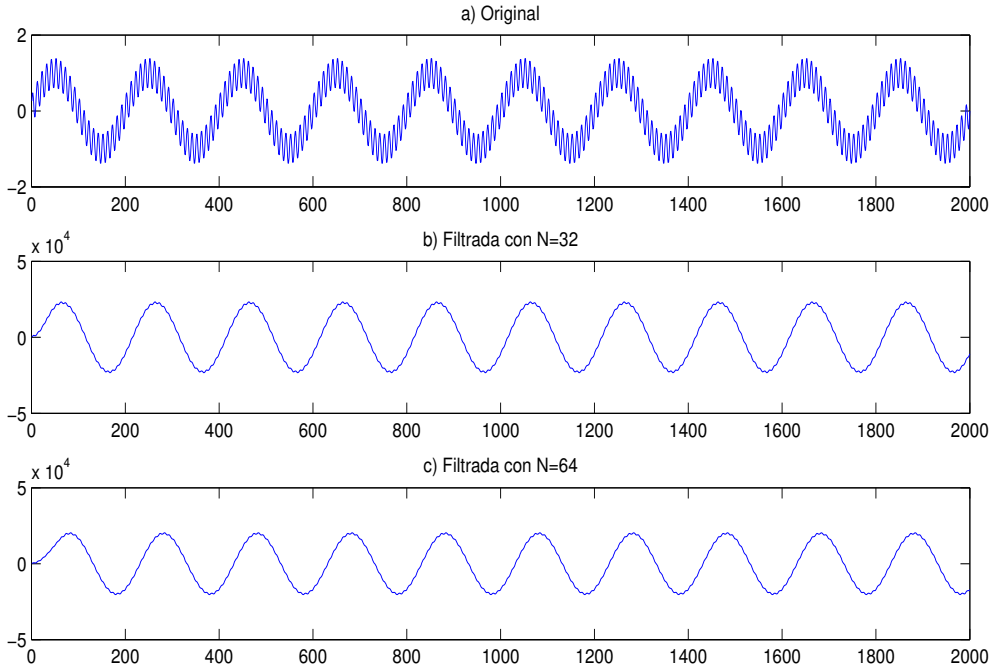


Figura 6.5. Señal filtrada con filtro FIR promediador

```
*
*    Filtro FIR promediador
*
       .global _c_int00
       .data
N      .set   800        ; longitud de datos de x(n)
xn     .word  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
xn1    .word  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
xnf    .word  0,0,0
x      .space  N*16      ; Espacio para x(n)
y      .space  N*16      ; Espacio para y(n)
WDCR    .set   07029h    ; Dirección registro de control WatchDog
CTE_WD  .set   0068h     ; Constante para desactivar el  WatchDog
Nf      .set   32        ; Longitud de datos a promediar
DR      .set   5         ; corrimiento de ACC
       .text
_c_int00
```

```
* Deshabilitación del WatchDog
         EALLOW             ; Habilita escritura a registros protegidos
         MOVL XAR1, #WDCR   ; Registro XAR1 apunta a dir. WDCR
         MOV  *XAR1,#0068h  ; Desactiva WatchDog, escribe en WDCR
         EDIS               ; Deshabilita escritura a registros protegidos

         SETC SXM           ; Modo extensión de signo
         SETC OVM
         SPM  #0
         MOVW   DP,#xn      ; Página de datos del bufer
         MOVL   XAR1,#x     ; XAR1 apunta a inicio de x(n)
         MOVL   XAR2,#y     ; XAR2 apunta a inicio de y(n)
         MOV    AR4,#N-1
CICLO_M
         MOV AL,*XAR1++     ; Muestrea el dato x(n)
         MOV @xn,AL         ; Escribe x(n) al inicio del buffer
         MOVL  XAR3,#xn     ; XAR3 apunta al inicio del buffer
         ZAPA               ; ACC = 0  y P = 0
         RPT  #Nf-1
     ||   ADD   ACC,*XAR3++ ; Suma los datos del buffer

         MOV    T,#DR       ; Corrimiento a la derecha de ACC
         ASRL   ACC,T       ; Divide datos entre 32
         MOV   *XAR2++,AL   ; Escribe AL a salida y(n)
         MOVL  XAR3,#xnf    ; XAR3 apunta al inicio del buffer
         RPT  #Nf-1
     ||   DMOV *--XAR3      ; Mueve los datos del buffer
         BANZ  CICLO_M,AR4-- ; Regresa a  CICLO_M si AR4!=0 y AR4 = AR4-1
FIN_M   NOP
        LB   FIN_M
```

En la figura 6.5 se muestra el resultado de aplicar el filtro promediador a una señal senoidal con ruido utilizando $N = 32$ y $N = 64$. La estructura e idea de esta implementación se puede utilizar para filtros FIR de longitud N, sólo que ahora tenemos promedios ponderados por la función al impulso $h(n)$ del filtro, por tanto tenemos que utilizar la instrucción MAC.

6.2. Filtros de respuesta infinita al impulso

Los filtros de respuesta infinita al impulso (IIR) también son llamados recursivos o autorregresivos de movimiento promedio (ARMA), ya que constan de una parte que efectúa la suma ponderada de la entrada $x(n)$ y retardos de $x(n-i)$, y otra que efectúa una suma ponderada de las salidas $y(n-i)$ retrasadas. Recordando que todo filtro analógico (FA) genera una respuesta infinita al impulso, parece obvio que un sistema discreto con respuesta infinita al impulso pueda emular las características de un filtro analógico. La respuesta infinita de un sistema discreto IIR se debe a su carácter recursivo que retroalimenta la salida retrasada. Una característica de un filtro IIR que lo diferencia de un filtro FIR es que realimenta la señal de salida y que puede llegar a ser inestable. En un filtro IIR los valores de los coeficientes a_m son diferentes de cero y también pueden existir los coeficientes b_m.

6.2.1. Estructuras de filtros digitales IIR

Recordando que la salida de un sistema lineal y discreto puede escribirse como la convolución de la entrada con su respuesta al impulso, y si en un filtro IIR su respuesta al impulso es infinita, entonces

$$y(n) = x(n) * h(n) = \sum_{i=0}^{\infty} h(i)x(n-i) \tag{6.12}$$

se observa que esta operación para un sistema IIR computacionalmente no es realizable ya que se necesitan infinitos coeficientes, sumas y productos. Por tanto, para que un sistema discreto de componentes finitas pueda generar una salida infinita con una entrada impulso, el sistema debe ser recursivo, entonces para poder realizar un filtro o sistema IIR es necesario una ecuación en diferencias que retroalimente la salida retardada, es decir

$$y(n) = \sum_{i=0}^{q} b(i)x(n-i) - \sum_{i=1}^{p} a(i)y(n-i) \tag{6.13}$$

De forma similar a los filtros digitales FIR, se tienen diferentes estructuras de implementación de filtros IIR, considerando la ecuación en diferencias que se utiliza para implementar un filtro IIR y aplicando la Transformada Z (TZ), se obtiene la función de transferencia $H(z)$ del filtro

$$H(z) = \frac{Y(z)}{X(z)} = \frac{B(z)}{A(z)} = \frac{\sum_{i=0}^{q} b(i)z^{-i}}{\sum_{i=0}^{p} a(i)z^{-i}} = \frac{b_0 + b_{-1}z^{-1} + \cdots + b_q z^{-q}}{a_0 + a_1 z^{-1} + ... + a_p z^{-p}}, \qquad a_0 = 1 \tag{6.14}$$

donde p es el número de polos, q es el número de ceros de $H(z)$, a_i y b_i son los coeficientes del filtro.

Cualquier tipo de estructura que se seleccione debe ser equivalente al mismo sistema. Dependiendo de como se traten las dos ecuaciones anteriores, se pueden obtener cuatro formas o estructuras de implementación más utilizadas:

- Forma directa I
- Forma directa II o canónica
- Forma cascada
- Forma paralela

6.2.2. Filtro IIR forma directa

En esta forma, la ecuación (6.13) es implementada directamente en un diagrama de bloques básicos sin efectuar ningún cambio. Existen dos partes en este filtro denominadas movimiento promedio y parte autorregresiva (ARMA). A la vez esta implementación conduce a dos formas: directa I y directa II.

Forma directa I

Si la ecuación en diferencias (6.13) se desarrolla directamente, entonces se tiene:

$$y(n) = b_0x(n) + b_1x(1) + \cdots + b_qx(n-q) - a_1y(n-1) - a_2x(n-2) - \cdots - a_py(n-p) \quad (6.15)$$

Se observa que existen dos líneas de retardo, una para la señal de entrada $x(n)$ y la otra para la señal de salida $y(n)$, es decir, que se tendrán $p+q$ bloques de retardo, como se aprecia en la figura 6.6.

La implementación de un filtro IIR se puede ver como la convolución de los coeficientes b_m con la señal de entrada menos la convolución de la señal de salida retardada con los coeficientes a_m, esto se aprecia en la figura 6.7.

Forma directa II o canónica

Manipulando la ecuación (6.14), las líneas de retardo se pueden convertir en una sola, lo que conduce a la forma directa II de un filtro IIR o estructura canónica. Esta forma se deduce al descomponer la ecuación (6.14) en dos factores

$$H(z) = \frac{Y(z)}{X(z)} = \frac{B(z)}{1+A(z)} = B(z)\Big(1+A(z)\Big)^{-1} = B(z)C(z) \quad (6.16)$$

$$\frac{Y(z)}{X(z)} = B(z)C(z)$$

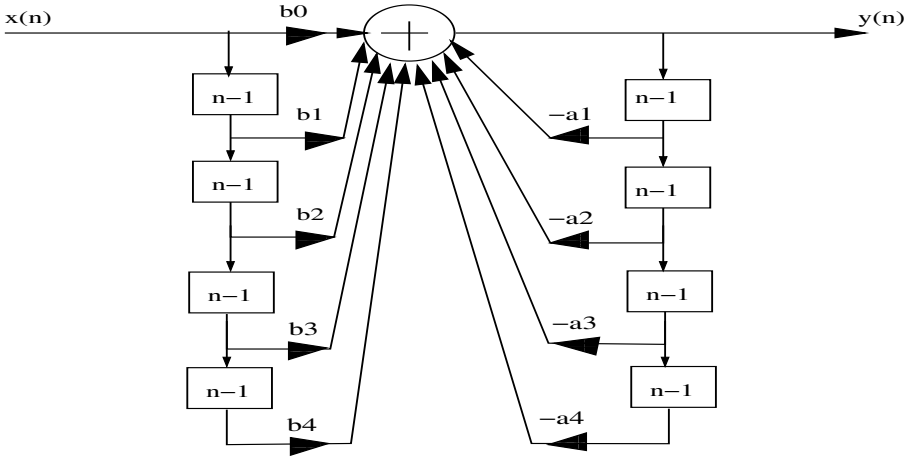


Figura 6.6. IIR forma directa I

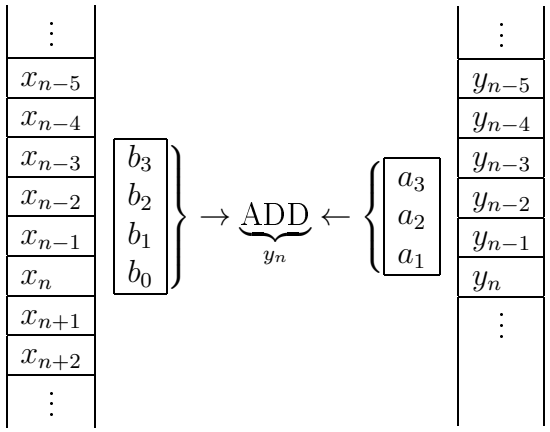


Figura 6.7. Líneas de retardo de un filtro IIR.

para un punto intermedio $l(n)$

$$\Rightarrow Y(z) = B(z)C(z)X(z) = B(z)L(z) \tag{6.17}$$

$$L(z) = C(z)X(z)$$

$$L(z) = X(z) - a_1 L(z)z^{-1} - a_2 L(z)z^{-2} + \cdots \tag{6.18}$$

entonces, por transformada Z inversa

$$l(n) = x(n) - a_1 l(n-1) - a_2 l(n-2) + \cdots \tag{6.19}$$

para la salida

$$Y(z) = b_0 L(z) + b_1 L(z) z^{-1} + b_2 l(z) z^{-2} + \cdots \tag{6.20}$$

$$y(n) = b_0 l(n) + b_1 L(n-1) + b_2 l(n-2) + \cdots \tag{6.21}$$

Esta estructura se expresa gráficamente en la figura 6.8 y se puede calcular con el siguiente programa, donde si el usuario quiere hacer uso del código debe de configurar PM y los formatos Qi de las señales, variables temporales y las constantes, de forma similar cuando se van a salvar los resultados.

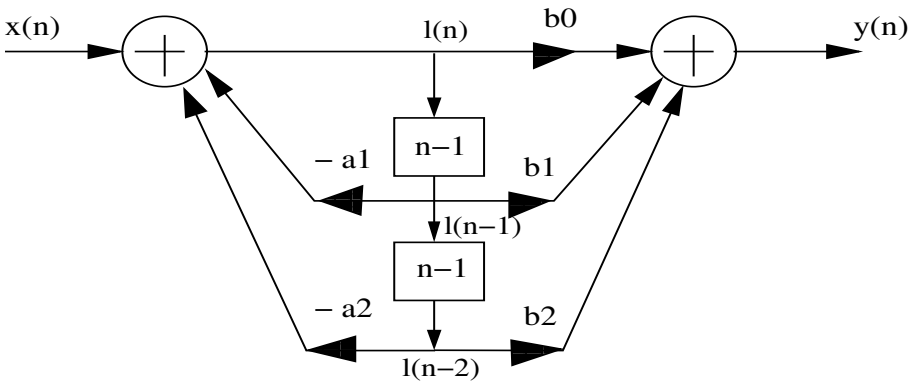


Figura 6.8. IIR forma directa II o canónica

```
*   Ciclo para el filtro IIR de segundo orden forma directa II
*   considerando que  a1 y a2 son  negativos
            MOV   AH,@xn        ; AH = x(n)
            MPY   P,@ln1,#a1    ; P  = a1*l(n-1)
            MPYA  P,@ln2,#a2    ; ACC = x(n) + a1*l(n-1), P = a2*l(n-2)
            ADDL  ACC,P
            MOV   @ln,ACC       ; Salva l(n)= x(n)+a1*l(n-1)+a2*l(n-2)

            ZAPA                ; ACC = 0,  P = 0
            MPY   P,@ln,#b0     ; P = b0*l(n)
            MPYA  P,@ln1,#b1    ; ACC = ACC +  b0*l(n), P = b1*l(n-1)
            MPYA  P,@ln2,#b2    ; ACC = b0*l(n) + b1*l(n-1) , P = b2*l(n-2)
            ADDL  ACC,P         ; ACC = b0*l(n) + b1*l(n-1) + b2*l(n-2)
            MOV   @yn,ACC       ; Salva y(n)=b0*l(n)+b1*l(n-1)+b2*l(n-2)
*   Retardos
            DMOV  @ln1          ; ln2 = l(n-1)
            DMOV  @ln           ; ln1 = l(n)
```

Forma canónica transpuesta

Si se manipulan las operaciones en la ecuación en diferencia, se puede obtener otra forma canónica conocida como *forma directa transpuesta*, que se ilustra en la figura 6.9 y también tiene un mínimo de retardos. Para la figura 6.9 se tiene el siguiente código, donde el valor de $y(n)$ para iniciar los cálculos es el valor anterior, por tanto el valor de $y(n)$ actual se calcula al último.

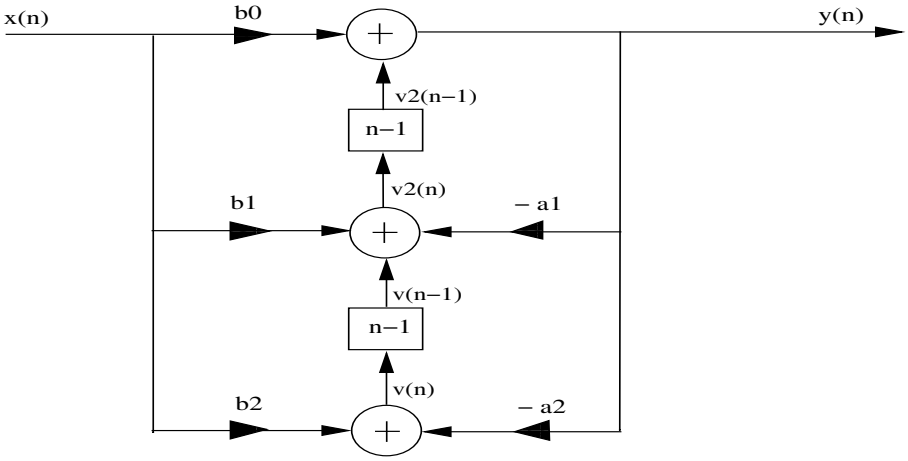


Figura 6.9. Forma directa transpuesta

```
*   Ciclo para el filtro IIR de segundo orden forma transpuesta
*   considerando que  a1 y a2 son  negativos
            ZAPA
            MPY   P,@xn,#b2
            MPYA  P,@yn,#a2
            ADDL  ACC,P         ; ACC = b2*x(n)-a2*y(n)
            MOV   @vn,ACC       ; Salva v(n)= b2*x(n)-a2*y(n)
            MOV   AH,@vn1
            MPY   P,@xn,#b1
            MPYA  P,@yn,#a1
            ADDL  ACC,P
            MOV   @vn2,ACC      ; Salva v2(n)= vn1+b1*x(n)-a2*y(n)
            MOV   AH,@vn3
            MPY   P,@xn,#b0
            ADDL  ACC,P
            MOV   @yn,ACC       ; Salva y(n)= b0*x(n)+vn3
*   Retardos
```

```
DMOV  @vn          ; vn1 = v(n-1)
DMOV  @vn2         ; vn3 = v2(n-1)
```

A esta forma se le llama canónica, dado que la implementación de la función de transferencia se realiza con la menor cantidad de bloques de retardo. Una estructura de un filtro digital se dice que es canónica si el número de retardos de su representación en diagramas de bloques es igual al orden de la ecuación en diferencias [2], [6], [10].

6.3. Osciladores digitales

Los osciladores son sistemas con características inestables, que son la base de otros sistemas más complicados como generadores senoidales, osciladores controlados por voltaje (VCO), moduladores, mallas de fase amarrada (PLL), etc. En general existen tres métodos para generar una función o señal senoidal: por medio de una tabla de valores senoidales, por el desarrollo de la función seno por una serie de Taylor y un sistema SLITD inestable. Si utilizamos el último método, un oscilador senoidal puede diseñarse como un filtro paso banda de alta calidad que solo deja pasar una frecuencia. Si se diseña en el plano z, un oscilador consta de dos polos conjugados complejos p_1 y p_2 ubicados sobre el círculo unitario a una frecuencia de oscilación ω_0. Para un sistema discreto SLITD de segundo orden con la función de transferencia de un oscilador se puede escribir como

$$H(z) = \frac{1}{(1 - p_1 z^{-1})(1 - p_2 z^{-1})} \tag{6.22}$$

donde $p1 = p2^*$ son polos conjugados y $p1 = e^{j\omega_0}$ como se muestra en la figura 6.10.a, donde ω_0 es la frecuencia de oscilación normalizada de cero a π, y se puede calcular como $\omega_0 = 2\pi f_{osc}/F_s$, F_{osc} es la frecuencia de oscilación analógica en Hertz y f_s la frecuencia de muestreo que cumple con el teorema de Nyquist. Efectuando operaciones

$$H(z) = \frac{b_0}{(1 - a_1 z^{-1} + a_2 z^{-2})} \tag{6.23}$$

con los valores de las constantes: $a_1 = -2\cos(\omega_0)$ y $a_2 = 1$.

Si se aplican las fórmulas de tablas y propiedades de TZ se puede obtener la respuesta al impulso del sistema como

$$h(n) = \frac{b_0}{\text{sen}(\omega_0)}\text{sen}((n+1)\omega_0)U(n) \tag{6.24}$$

haciendo la constante $b_0 = \text{sen}(\omega_0)$ en la ecuación se obtiene la respuesta

$$h(n) = \text{sen}((n+1)\omega_0)U(n) \tag{6.25}$$

para que el sistema oscile y se tenga una salida $y(n)$ igual a la respuesta al impulso $h(n)$, la entrada debe ser un impulso $\delta(n)$ con ecuación en diferencias

$$y(n) = b_0 x(n) + a_1 y(n-1) - a_2 y(n-2) \tag{6.26}$$

cuyo diagrama de bloques se muestra en la figura 6.10.b, con condiciones iniciales $y(-2) = 0$, $y(-1) = -A\text{sen}(\omega)$ y A la amplitud de la señal senoidal generada.

Por otro lado, un oscilador discreto también puede diseñarse considerando que su respuesta al impulso es una señal cosenoidal con la TZ para una entrada impulso $\delta(n)$. Para obtener la salida como una ecuación en diferencias se escribe $H(z)$ como el cociente de $Y(z)$ con $X(z)$, se calcula la transformada Z inversa y se despeja $y(n)$.

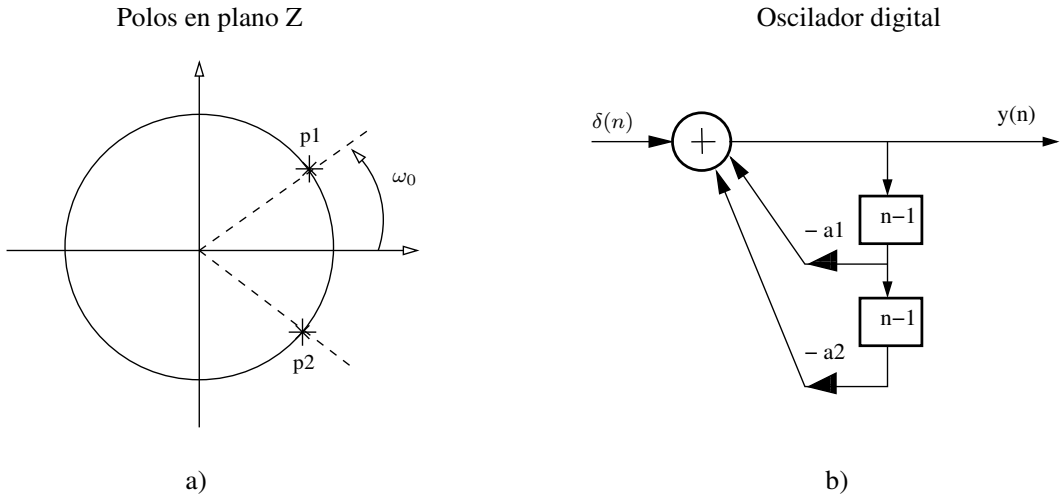


Figura 6.10. Oscilador con sistema IIR inestable

```
*
*    Oscilador digital
*
*    Ecuación en diferencias
*    y(n)  = bo*d(n) + a1*y(n-1) - a2*y(n-2)
*
*    wo = 0.034159  frecuencia de oscilación
*    sin(wo)   = 0.034152357  en Q12
*    cos(wo)   = 0.999416638
*    2*cos(wo) = 1.998833276  en q12
       .global _c_int00
       .data             ; Sección de datos
b0     .word     139     ; Constantes En Q12  139.88
```

```
a1      .word     8187     ; a1  = 2* cos(wo)
a2      .word     0xF000   ; -a2 = 1 en Q12
yn1     .word     0        ; y(n-1)
yn2     .word     0        ; y(n-2)
ynx     .word     0        ; Para mover dato yn2
N       .set     1000      ; Número de muestras
y       .space   N*16      ; reserva 1000 localidades para y
WDCR    .set   07029h      ; Dirección registro de control WatchDog
CTE_WD  .set   0068h       ; Constante para desactivar el  WatchDog
        .text              ; Sección de código
_c_int00
* Deshabilitación del WatchDog
        EALLOW             ; Habilita escritura a registros protegidos
        MOVL XAR1, #WDCR   ; Registro XAR1 apunta dir, WDCR
        MOV  *XAR1,#0068h  ; Desactiva WatchDog, escribe en WDCR
        EDIS               ; Deshabilita escritura a registros protegidos
*
        SETC SXM           ; Modo extensión de signo
        SETC OVM           ; Modo saturado
        SPM  #0            ; Sin corrimientos en el multiplicador
        MOVW DP,#yn1       ; Selecciona página de yn1
        MOVL XAR1,#y       ; XAR1 apunta a dirección de y
        MOV  AL,@b0        ;
        MOV  @yn2,AL       ; Pone en el buffer y(0)
        MOV  *XAR1++,AL    ; Salva y(0)
        MOV  T,@yn2        ; T    = y(0)
        MPY  ACC,T,@a1     ; y(1) =  b0*a1
        LSL   ACC,#4       ; Ajusta ACC a Q28
        MOV  @yn1,AH       ; Salva AH en Q12
        MOV  *XAR1++,AH    ; Salva y(1) en Q12
        MOV  AR4,#N-1      ; contador de ciclo OSCILA
OSCILA
        MOV  T,@a1
        MPY  ACC,T,@yn1    ; ACC = a1*y(n-1)
        MOV  T,@a2
        MPY  P,T,@yn2      ; P   = a2*y(n-2)
        ADDL  ACC,P        ; ACC = a1*y(n-1) + a2*y(n-2)
        LSL   ACC,#4       ; Ajuste a Q28
        MOV  *XAR1++,AH    ; Salva y(n)  a salida
   ; Retarda datos en el bufer y(n-1) e  y(n-2)
        DMOV   @yn2
```

```
        DMOV   @yn1
        MOV  @yn1,AH        ; Salva salida y(n) al buffer
        BANZ  OSCILA,AR4--  ; Regresa a OSCIALA si AR4 != 0
FIN_O   NOP                 ; Ciclo infinito
        NOP
        LB  FIN_O
```

6.4. Generación de señales DTMF

La codificación o modulación DTMF (Dual tone modulation frequency) es utilizada ampliamente en los sistemas telefónicos, cuando un usuario oprime una secuencia de números para comunicarse, por cada tecla oprimida se genera una señal de doble tono durante un tiempo determinado, es decir, se utilizan dos osciladores. Las señales DTMF generadas viajan por la red telefónica y a través de la decodificación en subestaciones permite enrutar la llamada hasta el usuario final. Un generador de doble tono se puede generar con dos osciladores sinusoidales en sus respectivas frecuencias estandarizadas como se muestra en la tabla 6.1, donde a cada dígito del teclado telefónico se le asocian dos frecuencias, la del renglón y columna.

Tabla 6.1. Teclado telefónico y frecuencias DTMF asociadas

	1209	**1336**	**1477**	**1633**
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Los dos tonos correspondientes a cada tecla oprimida se pueden generar con dos osciladores individuales, como el visto anteriormente; sin embargo, existe la posibilidad de implementar los dos tonos acoplados en un oscilador de un doble tono simultáneo, como se muestra en seguida.

Partiendo del análisis en el tiempo discreto, si se suman dos señales senoidales causales como

$$y(n) = \text{sen}(\omega_1 n) + \text{sen}(\omega_2 n) \tag{6.27}$$

donde las frecuencias de oscilación discretas bajas y altas corresponden a ω_1 y ω_2, y $\omega_i = 2\pi fosc/Fs$, $fosc$ es la frecuencia de oscilación analógica y Fs la frecuencia de muestreo. Si $y(n)$ es igual a la respuesta al impulso $h(n)$ del sistema, y aplicando la transformada Z donde $H(z) = Y(z)/X(z)$ tenemos

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\text{sen}(\omega_1)z^{-1}}{1 - 2\cos(\omega_1)z^{-1} + z^{-2}} + \frac{\text{sen}(\omega_2)z^{-1}}{1 - 2\cos(\omega_2)z^{-1} + z^{-2}} \tag{6.28}$$

después de un manejo algebraico se llega a la ecuación

$$H(z) = \frac{Y(z)}{X(z)} = \frac{Az^{-1} - Bz^{-2} + Az^{-3}}{1 - Cz^{-1} + Dz^{-2} - Cz^{-3} + z^{-4}} \tag{6.29}$$

donde:
$A = \text{sen}(\omega_1) + \text{sen}(\omega_2)$
$B = 2[\cos(\omega_1)\text{sen}(\omega_2) + \cos(\omega_2)\text{sen}(\omega_1)]$
$C = 2[\cos(\omega_1) + \cos(\omega_2)]$
$D = 2 + 4[\cos(\omega_1)\cos(\omega_2)]$

aplicando transformada Z inversa a la ecuación (6.29), se obtiene la ecuación en diferencias para un sistema IIR de cuarto orden

$$y(n) = Ax(n-1) - Bx(n-2) + Ax(n-3) + Cy(n-1) - Dy(n-2) + Cy(n-3) - y(n-4) \tag{6.30}$$

considerando la entrada $x(n) = \delta(n)$ y condiciones iniciales $y(-4) = y(-3) = y(-2) = y(-1) = 0$, entonces se tienen las salidas:

$y(0) = 0$; $\quad y(1) = A$; $\quad y(2) = -B + Cy(1)$
$y(3) = A + Cy(2) - Dy(1)$; $\quad y(4) = Cy(3) - Dy(2) + Cy(1) - y(0)$

es decir, que a partir de $n = 4$ se puede calcular la salida $y(n)$ en forma recursiva como

$$y(n) = Cy(n-1) - Dy(n-2) + Cy(n-3) - y(n-4) \quad ; \quad \text{para} \quad n \geq 4 \tag{6.31}$$

la ecuación (6.31) se programaría para obtener una salida $y(n)$ de doble tono a partir de $n \geq 4$ y los valores para $0 \geq n \geq 3$ se pueden calcular fuera de línea. Para su implementación en un DSP de punto entero es necesario tener presente la dinámica de las variables, de los coeficientes de la ecuación (6.30) se observa que los máximos valores son $A \leq 2$, $B \leq 4$, $A \leq 4$ y $A \leq 6$, esto implica que la representación de los coeficientes debe ser de un $Qi \geq 12$. El diagrama de bloque de la implementación de una señal DTMF se muestra en la figura 6.11 y el código para la tecla cinco con $f_1 = 770$ Hz y $f_2 = 13360$ Hz se muestra a continuación.

En la implementación de sistemas tipo IIR en aritmética de punto entero a 16 bits, lo más efectivo es utilizar estructuras de segundo orden ya sea en cascada o en paralelo, ya que debido a los efectos de precisión numérica, para sistemas de orden mayor de dos se van acumulando errores numéricos que pueden mover los polos del sistema y hacerlo inestable. Sin embargo, en el siguiente código se muestra la implementación del diseño anterior de un generador DTMF para la tecla 5 a 32 bits, lo cual nos permite implementar un sistema IIR de cuarto orden en forma directa.

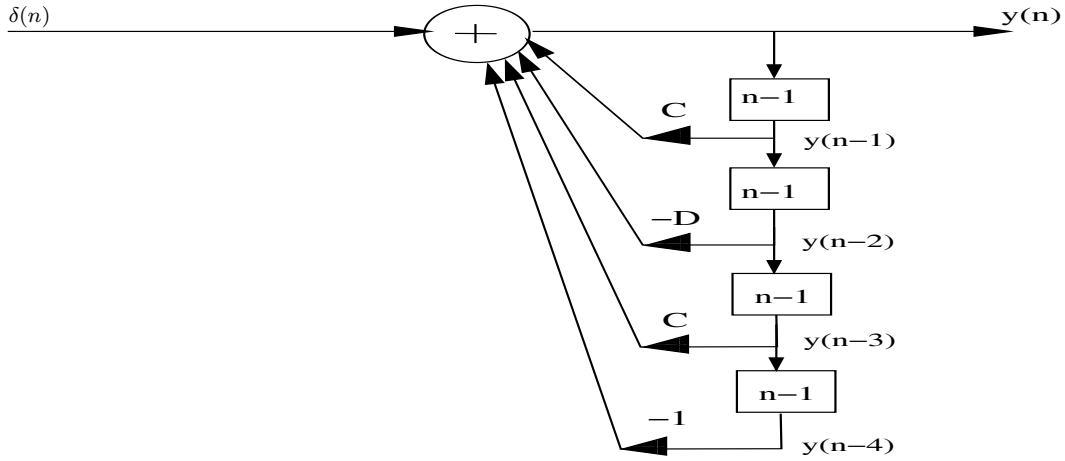


Figura 6.11. Sistema IIR para generar una señal DTMF

```
*
*    OSCILADOR DE DOBLE TONO DTMF  PARA TECLA 5
*    EN EL  DSP C28x
*    Constantes en Q28, variables y(n-i) en Q28
*
*
*    Ecuaciones
*    y(0) = 0
*    y(1) = A                          = 0.1322  = y(n-3)
*    y(2) = -B + C*y(1)                = 0.2637  = y(n-2)
*    y(3) = A + C*y(2) - D*y(1)        = 0.3938  = y(n-1)
*    y(4) = C*y(3) - D*y(2) + Cy(1) - y(0)
*    Para n >= 4
*    y(n) = C*y(n-1) - D*y(n-2) + C*y(n-3) - y(n-4)
*     Si N = 1000
*     w1 = 2*pi*7.7/N   = 0.048380527
*     w2 = 2*pi*13.36/N = 0.083943356
*     A = 0.132206462   =  541.51 en Q12
*     B = 0.263876129   = 1080.83 en Q12
*     C = 3.990617431   = 16345.56
*     D = 5.981251342   = 24499.20
*
       .global _c_int00
       .data
CD1    .long    1071223209d
```

```
D       .long    -1605579931
C       .long    1071223209d
E1      .long    0F0000000h,0
*
yn      .long     0          ; Buffer temporal de y(n) en Q28
yn1     .long     105712363  ; 0.39453125       y(n-1)
yn2     .long     70788921   ; 0.263671875      y(n-2)
yn3     .long     35488901   ; 0.132206462 = A  y(n-3)
yn4     .long     0          ; 0
yn5     .long     0
N        .set     800        ; Número de muestras
ND       .set     10         ; Para retardos
y        .space   N*16       ; Reserva 1000 localidades para y(n)
N4       .set     4          ; Número de retardos
WDCR     .set   07029h       ; Dirección registro de control WatchDog
CTE_WD   .set   0068h        ; Constante para desactivar el  WatchDog

         .text
_c_int00
* Deshabilitación del WatchDog
          EALLOW             ; Habilita escritura a registros protegidos
          MOVL XAR1, #WDCR   ; Registro XAR1 apunta a dir. WDCR
          MOV  *XAR1,#0068h  ; Desactiva WatchDog, escribe en WDCR
          EDIS               ; Deshabilita escritura a registros protegidos

        SETC SXM             ; Modo extensión de signo
        SETC OVM
        SPM  #0
        MOVW DP,#yn
        MOVL XAR2,#y         ;  Salida y(n)
        MOVL  ACC,@yn4
        MOVL   *XAR2++,ACC   ; Salva valores iniciales de y(n) en Q12
        MOVL   ACC,@yn3
        MOVL   *XAR2++,ACC   ; Salva valores iniciales de y(n)
        MOVL   ACC,@yn2
        MOVL   *XAR2++,ACC   ; Salva valores iniciales de y(n)
        MOVL   ACC,@yn1
        MOVL  *XAR2++,ACC    ; Salva valores iniciales de y(n)

        MOV  AR4,#N-1        ; contador de ciclo OSCILA
OSCILA
```

```
       MOVL   XAR6,#yn1     ; Inicio del buffer
       MOVL   XAR7,#CD1     ; Inicio de constantes ec. dif.
       ZAPA
       RPT    #N4-1
       ||     QMACL P,*XAR6++,*XAR7++ ; Acumula y Multiplica a 32b
       ADDL   ACC,P          ;  y(n) en Q56
       LSL    ACC,#N4        ;  Ajuste a Q60
       MOVL   @yn,ACC        ;  Salva y(n) al buffer en Q28
       MOVL   *XAR2++,ACC    ;  Salva y(n)  a salida en Q28
 ; Retarda datos en el buffer
       MOVL   XAR1,#yn5
       RPT    #ND-1
     || DMOV   *--XAR1       ; Retardo en el buffer

       MOVL   XAR1,#yn5
       RPT    #ND-1
     || DMOV   *--XAR1       ; Retardo en el buffer
       BANZ   OSCILA,AR4--

FIN_O  NOP
       B  FIN_O,UNC
```

El código anterior se puede hacer más eficiente utilizando buffer circular, tal como se expresa en seguida para el ciclo OSCILAC:

```
       MOVL   XAR6,#yn1     ; Inicio del buffer circular
       MOV    AR1,#2*N4     ; Buffer circular N4+1 = 5

       MOV  AR4,#N-1        ; contador de ciclo OSCILAC
OSCILAC
       MOVL   XAR7,#CD1     ; Inicio de constantes ec. dif.
       ZAPA
       RPT    #N4-1
       ||     QMACL P,*XAR6%++,*XAR7++
       ADDL   ACC,P          ;  y(n) en Q56
       LSL    ACC,#N4        ;  Ajuste a Q60
       MOVL    *XAR6,ACC     ;  Salva y(n) al buffer CIRCULAR en Q28
       MOVL   *XAR2++,ACC    ;  Salva y(n) a salida en Q28

       BANZ   OSCILAC,AR4--
```

```
FIN_O  NOP
       B  FIN_O,UNC
```

En la figura 6.12 se observa la señal DTMF para la tecla cinco y su espectro, obtenidos con el código anterior en el DSP F28027.

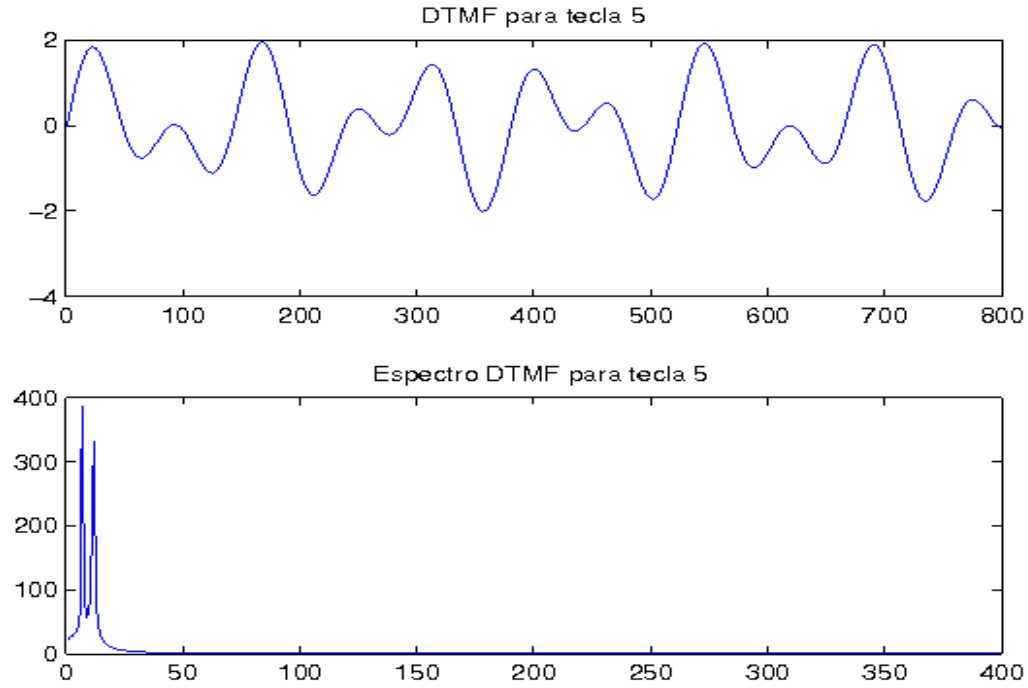


Figura 6.12. Salida del generador DTMF

6.5. Aplicaciones propuestas

En esta sección se proponen proyectos más complicados considerando que el lector ha avanzado en el estudio e implementación de aplicaciones en los DSP de las familias C28x. Se proponen realizar estas aplicaciones en la tarjeta de los DSP Piccolo F28069 por su capacidad de memoria.

1. Dada una señal de voz $v(n)$ de N puntos, con un programa corriendo en el DSP, agregarle una interferencia de un tono sinusoidal de 100 Hz con una relación señal a ruido SNR menor de 5 db. Diseñar y programar un sistema digital que cancele con la mejor calidad posible los tonos agregados, utilizando:
   - Filtros FIR.
   - Filtros IIR.
   Los filtros se pueden realizar en cualquier estructura y programarse a 16 y 32 bits.

2. Diseñar, realizar y programar un sistema tipo oscilador controlado por voltaje (VCO) que al menos genere 20 señales senoidales de frecuencia diferente. El diseñador debe proponer la forma y el orden de la generación de los tonos.

Resumen

En este capítulo se ha mostrado una breve introducción a los filtros digitales FIR e IIR, sus estructuras y algunos ejemplos de implementación del DSP C28x utilizando las potencialidades de su hardware e instrucciones. La implementación de los filtros digitales se puede considerar de mucha importancia cuando se realiza una aplicación de procesamiento digital de señales. Los ejemplos y resultados mostrados pueden ser un buen inicio para los interesados en empezar a explotar las potencialidades de los DSP C28x.

Capítulo 7

Interrupciones

Las interrupciones son eventos síncronos o asíncronos que afectan el flujo de un programa de una máquina digital, por lo que su manejo y configuración está ligado al control de la máquina. Este procedimiento constituye una de las formas más efectivas de transferencia de información hacia el CPU. La familia de DSP C28x puede manejar una gran cantidad de interrupciones debido a los periféricos que contiene. Las interrupciones constituyen un recurso para suspender la actividad del DSP con el fin de responder a algún requerimiento, esto evita la necesidad de estar probando por software un evento o dispositivo. Las interrupciones típicas son generadas por dispositivos que requieren transferir información al DSP como en el caso de convertidores A/D, algún puerto serial u otro periférico.

En este capítulo, se exponen los conceptos, la configuración y procedimientos del manejo de interrupciones. Además, se explica la unidad o módulo PIE, que controla, expande y arbitra hasta 96 interrupciones de periféricos.

7.1. Interrupciones del C28x

En el DSP C28x sus fuentes de interrupción se presentan por software (INTR o TRAP) o por hardware (pines externos, periféricos externos o internos). Estos DSP incluyen un módulo de expansión de interrupciones de periféricos (PIE) que permite multiplexar interrupciones de periféricos en una interrupción al CPU [18], [20].

A nivel del CPU, las interrupciones se pueden agrupar en:

- **Interrupciones mascarables**, éstas pueden ser bloqueadas o desbloqueadas por software.

- **Interrupciones no mascarables**, éstas no pueden ser bloqueadas, cuando se presentan el C28x las atiende de inmediato saltando a la subrutina correspondiente. Todas las interrupciones inicializadas por software entran en esta categoría.

7.1.1. Operación y manejo de interrupciones

Para que el programador del DSP utilice las interrupciones debe conocer su funcionamiento, registros de configuración y las etapas que se llevan a cabo. Como se observa en la figura 7.1, el proceso general de atención y ejecución de interrupciones sucede en las siguientes etapas:

- **Requerimiento o recepción de interrupción**
  Cuando una interrupción ocurre, ésta es almacenada en el registro de banderas de interrupción de 16 bits (IFR). Cada interrupción es identificada en un bit del registro IFR hasta que es reconocida y automáticamente borrada por el reconocimiento de interrupciones (/IACK), en el reset (/RS), se escribe un "uno" en el bit correspondiente en el registro IFR. La interrupción de reset no es almacenada en el registro IFR, inmediatamente después de un reset, se borran todas las interrupciones pendientes en el registro IFR.

- **Reconocimiento o aprobación**
  El C28x debe aprobar el requerimiento de la interrupción, si ésta es mascarable debe cumplir ciertos requerimientos, si no es mascarable, la aprobación es inmediata.
  En la preparación para la rutina de servicio de interrupción y salvar registros se realizan las tareas:
  - Completa la ejecución de la instrucción que viene fluyendo en el pipeline y que ha alcanzado la fase de decodificación D2.
  - Salva automáticamente en la pila 14 registros principales: ST0, T, AL, AH, PL, PH, AR0, AR1, ST1, DP, IER, DBGSTAT, PCL y PCH.
  - Busca el vector de interrupción correspondiente y lo carga en el PC. Para dispositivos con módulo PIE, el vector buscado depende de la asignación de PIE y los registros de banderas.

- **Atención o ejecución de la subrutina de servicio de interrupción (ISR)**
  El C28x salta a la subrutina correspondiente ISR para su atención. Al final de cualquier subrutina ISR debe existir La instrucción de retorno de subrutina de interrupción IRET, para regresar al programa principal, en el retorno se recuperan de la pila los registros salvados y el valor PC+1 para continuar el programa.

Cada vector de interrupción consta de dos localidades para ubicar un salto de 22 bits a la dirección de la subrutina de atención de interrupciones. Los vectores son almacenados en localidades consecutivas de 32 bits.

Como se observa en la tabla 7.1, los vectores de interrupción se pueden mapear en dos direcciones diferentes dependiendo del bit VMAP del registro de estado ST1.

- Si VMAP = 0, los vectores inician en la localidad 00 0000h.
- Si VMAP = 1, inician en 3F FFC0h.

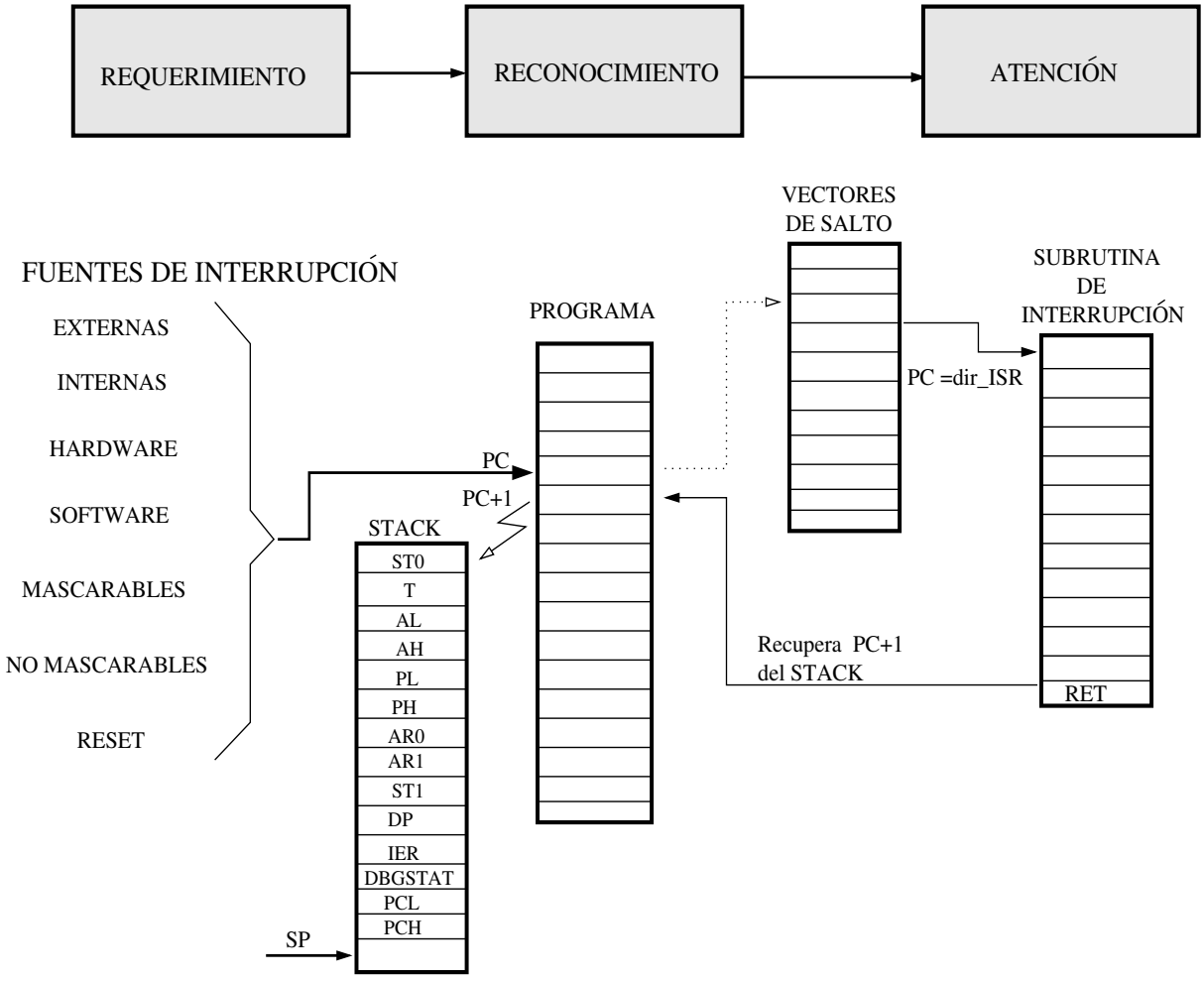


Figura 7.1. Fuentes de interrupciones de los DSP C28x

Tabla 7.1. Vectores de interrupción, direcciones y prioridades

Vector	**Dir. (h) VMAP=0**	**Dir(h) VMAP =1**	**Prioridad**	**Descripción**
RESET	00 0000	3F FFC0	1 (alta)	Reset
INT1	00 0002	3F FFC2	5	Mascarable 1
INT2	00 0004	3F FFC4	6	Mascarable 2
INT3	00 0006	3F FFC6	7	Mascarable 3
INT4	00 0008	3F FFC8	8	Mascarable 4
INT5	00 000A	3F FFCA	9	Mascarable 5
INT6	00 000C	3F FFCC	10	Mascarable 6
INT7	00 000E	3F FFCE	11	Mascarable 7
INT8	00 0010	3F FFD0	12	Mascarable 8
INT9	00 0012	3F FFD2	13	Mascarable 9
INT10	00 0014	3F FFD4	14	Mascarable 10
INT11	00 0016	3F FFD6	15	Mascarable 11
INT12	00 0018	3F FFD8	16	Mascarable 12
INT13	00 001A	3F FFDA	17	Mascarable 13
INT14	00 001C	3F FFDC	18	Mascarable 14
DLOGINT	00 001E	3F FFDE	19 (baja)	Mascarable, (data log)
RTOSINT	00 0020	3F FFE0	4	Mascarable, para sistemas de tiempo real
Reservado	00 0022	3F FFE2	2	Reservado
NMI	00 0024	3F FFE4	3	No mascarable
ILEGAL	00 0026	3F FFE6	-	Instrucción trap ilegal
USER1	00 0028	3F FFE8	-	De usuario por software
USER2	00 002A	3F FFEA	-	De usuario por software
USER3	00 002C	3F FFEC	-	De usuario por software
USER4	00 002E	3F FFEE	-	De usuario por software
USER5	00 0030	3F FFF0	-	De usuario por software
USER6	00 0032	3F FFF2	-	De usuario por software
USER7	00 0034	3F FFF4	-	De usuario por software
USER8	00 0036	3F FFF6	-	De usuario por software
USER9	00 0038	3F FFF8	-	De usuario por software
USER10	00 003A	3F FFFA	-	De usuario por software
USER11	00 003C	3F FFFC	-	De usuario por software
USER12	00 003F	3F FFFE	-	De usuario por software

Cada vector de interrupción ocupa dos localidades y se puede direccionar con 22 bits en el mapa de memoria de programa. Las interrupciones /INT1 a /INT14 son de propósito general, DLOGINT y RTOSINT son para propósitos de emulación.

Las 16 interrupciones mascarables son soportadas por tres registros:

IFR: Registro de banderas de interrupciones, en este registro existe un bit por cada interrupción y es puesto a uno por el requerimiento de una interrupción pendiente. Con la instrucción OR IFR se pueden poner a uno todos sus bits, y con AND IFR se pueden limpiar las interrupciones pendientes. Cuando una interrupción es atendida, se limpia el bit correspondiente.

IER: Registro de habilitación de interrupciones mascarables, se utiliza para enmascarar (bloquear o desbloquear) las interrupciones internas y externas. Un "uno" en cualquiera de los bits 0..15 del registro IER habilita la interrupción correspondiente siempre que el bit de estado INTM esté en cero. Los bits de los registros IFR e IER se muestran en la figura 7.2.

DBGIER: Registro de habilitación de interrupciones de depuración, permite habilitar o deshabilitar interrupciones mascarables.

En el registro ST1 existe el bit INTM que permite habilitar todas la interrupciones mascarables con INTM = 0 y deshabilitarlas con INTM = 1.

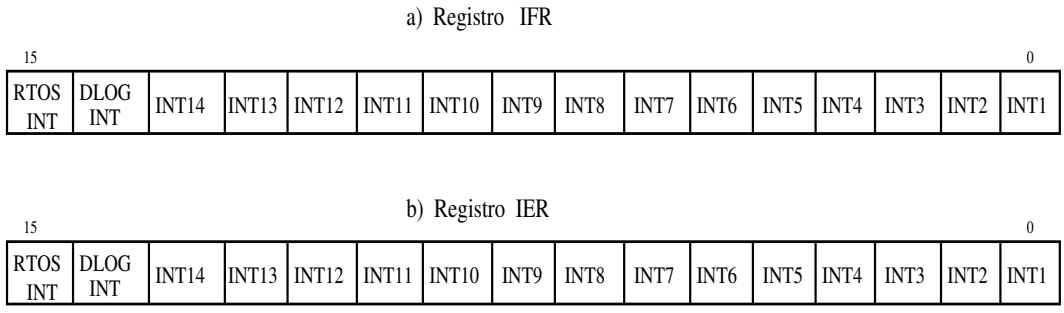


Figura 7.2. Registros IFR e IER

7.1.2. Interrupciones mascarables

A las interrupciones mascarables de la tabla 7.1 les corresponde un bit en los registros IFR, IER y DBGIER, en el orden del bit LSb al MSb, de INT1 a INT14, y luego DLOGINT y RTOSINT.

Si una interrupción ocurre durante un ciclo múltiple de instrucciones, la interrupción no puede ser procesada hasta que la instrucción es completada, este mecanismo de protección también es aplicado a instrucciones de ciclo múltiple a través de la señal de READY. Tampoco se permite que se procesen interrupciones cuando una instrucción está siendo repetida por medio de instrucción RPT, la interrupción es identificada en el registro IFR hasta que el ciclo de repetición se termine, después la interrupción es procesada. Las interrupciones no pueden ser procesadas cuando se está borrando el bit INTM, sino hasta la próxima instrucción en la secuencia del programa.

Como se verá en este capítulo, debido a que el CPU no puede centralizar muchas interrupciones, existe la unidad PIE, que controla, expande y arbitra las interrupciones de muchos periféricos. PIE puede soportar hasta 96 periféricos que interrumpen al CPU. Estos periféricos están agrupados en 12 bloques de ocho cada uno, cada bloque multiplexa el conjunto de interrupciones y alimenta una línea de interrupción del CPU, de INT1 a INT12. Cada uno de los 96 periféricos tiene su propio vector de interrupciones asociado a un bit de habilitación y un bit de reconocimiento.

7.1.3. Proceso de atención de interrupciones mascarables

Cuando las interrupciones están correctamente configuradas, el programa principal aceptará una interrupción en la ejecución de cualquier instrucción, este proceso sigue los pasos:

1. Se envía un requerimiento de interrupción al CPU a través de los eventos:
   - Cualquiera de los pines /XINT1 o /XINT2 en bajo, un periférico o un requerimiento de PIE.
   - La lógica de emulación envía señales al CPU por DLOGINT o RTOSINT.
   - Alguna de las interrupciones /INT1 a /INT14, DLOGINT y RTOSINT son inicializadas por la instrucción OR IFR.

2. Se fija el bit correspondiente de IFR cuando el CPU detecta una señal de interrupción.

3. El CPU valida la interrupción que cumple las condiciones:
   - El correspondiente bit en IER está en uno.
   - La habilitación global INTM = 0.

4. Una vez validada la interrupción, el correspondiente bit en IFR es limpiado.

5. Vacía el pipeline. El CPU completa cualquier instrucción que ha alcanzado la fase D2 del pipeline, las que no alcanzan esta fase se limpian del pipeline.

6. Se incrementa el PC en uno o dos, dependiendo del tamaño de código de la siguiente instrucción. El PC incrementado es guardado en la pila y es la dirección de retorno de atención de interrupción.

7. Búsqueda del vector de interrupción, el PC se carga con la dirección correspondiente del vector de interrupción. Los vectores de interrupción son realmente las direcciones asignadas para el salto a subrutina de atención de la interrupción, estas direcciones se pueden visualizar en la tabla 7.1.

8. El registro SP debe estar adecuadamente ubicado para preparar el almacenado del contexto.

9. Se salvan automáticamente 14 registros en la pila por pares, dando lugar a que el acceso sea de 32 bits y el incremento del SP de dos. Los registros se almacenan en el orden: ST0 y T, AL y AH, PL y PH, AR0 y AR1, ST1 y DP, IER y DBGSTAT, y PCL con PCH.

10. Deshabilita la interrupción a atender, es decir, el bit correspondiente en IER. Se debe tener presente que el IER original ya fue salvado en la pila, esto evita que ingrese otra interrupción durante la atención de interrupción en curso.

11. Fija los bits INTM y DBGM, limpia bits LOOP, EALLOW y IDLESTAT del registro de estado ST1. El CPU previene cualquier evento que afecte la atención de interrupción.

12. El PC se carga con la dirección de subrutina de atención de interrupción (ISR).

13. Ejecución de la rutina ISR. Los registros del contexto salvados se restablecen en el retorno de ISR.

14. Si la interrupción no es validada por el CPU, la interrupción es ignorada y el programa continúa su ejecución.

7.1.4. Interrupciones no mascarables

Estas interrupciones no están incluidas en los registros IFR e IER, y por su misma naturaleza no pueden ser bloqueadas con el bit INTM en el registro de estado ST1. Estas interrupciones son:
• Interrupciones de software.
• Interrupción no mascarable /NMI por hardware.
• Trap ilegal.
• Reset /RS por hardware.

7.1.5. Interrupciones por software

La característica de estas interrupciones es que el programador puede invocar por software una rutina de atención de interrupción mascarable cuando no se estén usando por

hardware, lo importante de esta forma de interrupción es que el usuario puede definir sus propias rutinas de atención de interrupción.

Estas interrupciones se pueden invocar con la instrucción INTR int, donde "int" puede ser cualquiera de las interrupciones mascarables /INT1 a /INT14, DLOGINT, RTOSINT, NMI y EMUINT. Cuando se invocan de esta forma, la bandera correspondiente en IFR no se activa y la interrupción se ejecuta de forma similar a la interrupción no mascarable.

/NMI

Esta interrupción se activa por hardware a través del pin NMI cuando éste se pone en bajo, también se puede activar por software con la instrucción INTR NMI.

Interrupciones de TRAP

La instrucción TRAP #k permite efectuar hasta 32 interrupciones por software con las mismas características de una interrupción por hardware. El usuario puede utilizar cualquiera de las 32 interrupciones invocándolas por los números 0 a 31 de la forma:

```
TRAP #Num
```

Es decir, que el usuario puede invocar cualquier interrupción de la tabla 7.1, donde TRAP 0 corresponde a RESET, y así sucesivamente hasta TRAP 31, que corresponde a USER12. Los bits de los registros IFR e IER no son afectados por esta interrupción. Los pasos para ejecutar una interrupción de TRAP son similares a los explicados en las interrupciones mascarables:

1. Búsqueda de la instrucción TRAP, el vector de interrupción se especifica como operando de esta instrucción.
2. Vaciado de pipeline, el CPU ejecuta la instrucción que alcanzó el nivel D2 de pipeline.
3. Incremento del PC en uno y almacenado en la pila para el retorno de ISR.
4. Búsqueda del vector de interrupción en la tabla de vectores de interrupción. La dirección de salto de la ISR es cargada en el PC.
5. Incremento del SP en uno.
6. Salvado automático del contexto. En el orden de registros ST0 y T, AL y AH, PL y PH, AR0 y AR1, ST1 y DP, IER y DBGSTAT, y PCL con PCH.
7. Fija los bits INTM y DBGM, limpia bits LOOP, EALLOW y IDLESTAT del registro de estado ST1. El CPU previene cualquier evento que afecte la atención de interrupción.
8. El PC se carga con la dirección de subrutina de atención de interrupción (ISR).

9. Ejecución de la rutina ISR.

10. El programa continúa su ejecución. Los registros del contexto salvados se restablecen en el retorno de ISR.

7.2. Módulo de expansión de interrupción de periféricos (PIE)

El módulo PIE permite multiplexar hasta 96 interrupciones en un conjunto de 12 entradas de interrupciones (INT1 a INT12), de estas interrupciones la mayoría son utilizadas por periféricos. Las 96 interrupciones son agrupadas en 12 bloques de ocho interrupciones cada una, los cuales alimentan las líneas de entrada INT1 a INT12 del CPU. Cada una de las 96 interrupciones tiene asignado su propio vector de interrupciones almacenados en memoria RAM; la búsqueda del vector consume ocho ciclos y salva los registros importantes del CPU. Cada interrupción puede ser habilitada o deshabilitada dentro del bloque PIE [20].

Debido a que el CPU no soporta muchas interrupciones, el módulo PIE es utilizado para controlar y arbitrar el requerimiento de muchas más fuentes de interrupciones. Como se observa en la figura 7.3, se ingresa un bloque "x" de ocho interrupciones que son reconocidas por el registro de banderas PIEIFRx. En el registro PIEIERx se puede enmascarar, luego a través de un multiplexor (MUX) se deja pasar una interrupción para ingresar por los pines /INT1 a /INT12 e interrumpir al CPU, donde:
x: representa el número de grupo de, x=1,2,3,...,12 (asignados a INT1 a INT12)
y: el número de interrupción dentro del grupo, y=1,2,3,...,8
PIEIERx.y : habilita interrupción x.y.
PIEIFRx.y : reconoce interrupción x.y.
PIEACKx : bits de reconocimiento de grupo x. Determina si el CPU está listo para atender la interrupción.

En la figura 7.3 se muestran todas las fuentes posibles de interrupción al CPU de los DSP C28x, como se observa, existe un bloque intermedio PIE que se encarga de aceptar hasta 96 interrupciones, y a través de un proceso de selección las introduce en los puntos INT1 al INT12 del DSP.

En la figura 7.4, se hace más explícito la forma de cómo la interfaz PIE, mapea las interrupciones. Es decir, que las interrupciones son divididas en 12 grupos de ocho cada uno: del grupo PIE1 al PIE12, estas ocho interrupciones de cada grupo son multiplexadas para interrumpir al CPU. Como se observa en la figura 7.3, el grupo PIE1 es encauzado dentro de la interrupción INT1 del CPU hasta el grupo PIE12 con INT12 del CPU. Las interrupciones restantes de CPU no son multiplexadas.

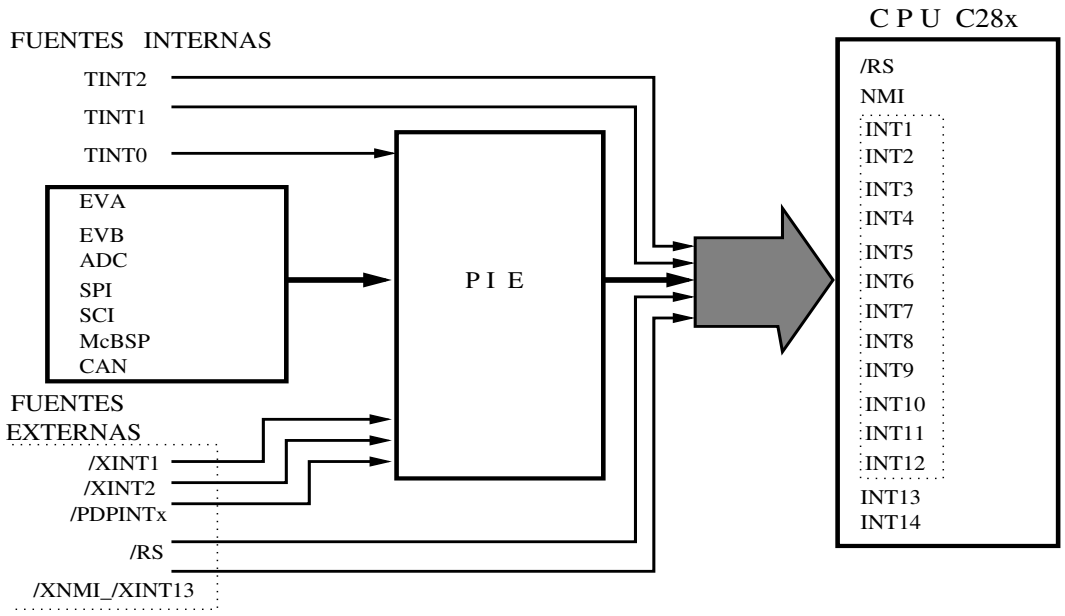


Figura 7.3. Fuentes de interrupciones de los DSP C28x

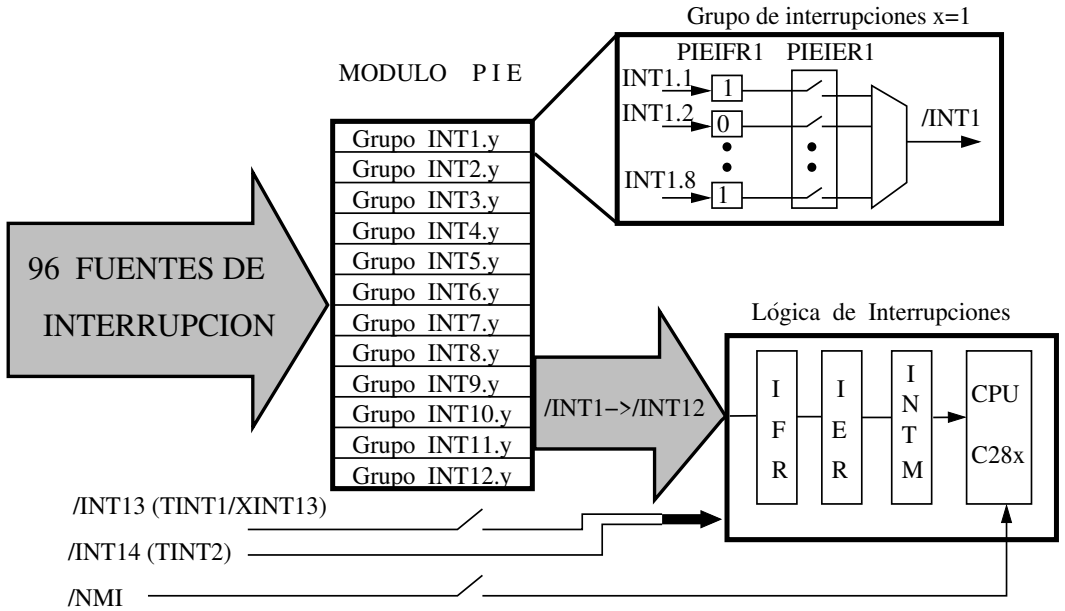


Figura 7.4. Selección de interrupciones por el módulo PIE

Para la selección de las fuentes de interrupción, cada grupo PIE tiene asociado un bit de bandera de interrupción PIEIFRx.y y un bit de habilitación PIEIERx.y. PIE verifica si el bit PIEACKx se puso en uno para determinar si el CPU está listo para la interrupción del grupo "x", si está en cero, entonces PIE envía un requerimiento de interrupción al CPU.

Asociado a PIE existen los registros: control PIECTRL, uno de reconocimiento PIEACK, 12 registros de banderas de interrupción PIEIFRx y 12 registros de habilitación PIEIERx (x:1, 2, ... 12), de las direcciones 0000h–0CE0h a 0000h–0CF9h. Cada vez que se atiende una interrupción se debe habilitar de nuevo el bit "x" en los registros PIEACKx, de lo contrario no se vuelve a reconocer.

En la tabla 7.2 se tienen las 96 fuentes de interrupción asociadas a los 12 grupos de interrupción de PIE para la familia F281x.

Tabla 7.2. Asignación de interrupciones de PIE, familia F281x

	INTx.8	**INTx.7**	**INTx.6**	**INTx.5**	**INTx.4**	**INTx.3**	**INTx.2**	**INTx.1**
INT1	Wake	TINT0	ADC	X2	X1		PDPB	PDP
INT2	–	T1OF	T1UF	T1C	T1P	COMP3	COMP2	COMP1
INT3	–	CAP3	CAP2	CAP1	T2OF	T2UF	T2C	T2P
INT4	–	T3OF	T3UF	T3C	T3P	CMP6I	CMP5	CMP4
INT5	–	CAPT6	CAPT5	CAPT4	T4OFT	T4UFI	T4C	T4P
INT6	–	–	MX	MR	–	–	SPITXA	SPIRXA
INT7	–	–	–	–	–	–	–	–
INT8	–	–	–	–	–	–	–	–
INT9	–	–	ECAN1	ECAN0	SCITXB	SCIRXB	SCITXA	SCIRXA
INT10	–	–	–	–	–	–	–	–
INT11	–	–	–	–	–	–	–	–
INT12	–	–	–	–	–	–	–	–

En la tabla 7.3 se tienen las 96 fuentes de interrupción asociadas a los 12 grupos de interrupción de PIE para la familia Piccolo.
Wake: corresponde al watchdog.
TZ: son señales de la unidad ePWM.
CLA: Acelerador de operaciones matemáticas.
Dependiendo de la familia, la tabla 7.3 puede variar significativamente, sin embargo, se conservan las fuentes de interrupción desde las primeras familias C28x. En la tabla 7.4 se muestra el remapeo de los vectores del módulo PIE.

Tabla 7.3. Asignación de interrupciones de PIE de DSP Piccolo

	INTx.8	**INTx.7**	**INTx.6**	**INTx.5**	**INTx.4**	**INTx.3**	**INTx.2**	**INTx.1**
1	Wake	TINT0	ADC INT9	XINT2	XINT1	–	ADC INT1	ADC INT2
2	EPWM8 TZINT	EPWM7 TZINT	EPWM6 TZINT	EPWM5 TZINT	EPWM4 TZINT	EPWM3 TZINT	EPWM2 TZINT	EPWM1 TZINT
3	EPWM8 INT	EPWM7 INT	EPWM6 INT	EPWM5 INT	EPWM4 INT	EPWM3 INT	EPWM2 INT	EPWM1 INT
4	HR CAP2	HR CAP1	–	–	–	ECAP3	ECAP2	ECAP1
5	–	–	–	HR CAP4	HR CAP1	–	EQEP2	EQEP1
6	–	–	MXINTA McBSP	MRINTA McBSP	SPI-B TXINT	SPI-B RXINT	SPI-A TXINT	SPI-A RXINT
7	–	–	DINT6 DMA	DINT5 DMA	DINT4 DMA	DINT3 DMA	DINT2 DMA	DINT1 DMA
8	–	–	–	–	–	–	I2C INT2A	I2C INT1A
9	–	–	ECAN1	ECAN0	SCI TXB	SCI RXB	SCI TXA	SCI RXA
10	ADC INT8	ADC INT7	ADC INT6	ADC INT5	ADC INT4	ADC INT3	ADC INT2	ADC INT1
11	CLA INT8	CLA INT7	CLA INT6	CLA INT5	CLA INT4	CLA INT3	CLA INT2	CLA INT1
12	LUF	LVF	–	–	–	–	–	XINT3

Tabla 7.4. Remapeo de vectores de interrupciones para el módulo PIE, bit ENPIE = 1

Vector	**Dirección (h)**	**Descripción**
No usado	00 0D00	
INT1	00 0D02	INT1 remapeado (abajo 0D40h)
...	...	..
...	...	..
INT12	00 0D18	INT12 remapeado (abajo 0DF0h)
INT13	00 0D1A	Vector XINT1
INT14	00 0D1C	Vector de Temporizador 2
Datalog	00 0D1D	Vector de Data log
...	...	..
De usuario	00 0D3E	De usuario para TRAP
INT1.1	00 0D40	Vector PIEINT1.1
INT1.2	00 0D42	Vector PIEINT1.2
...	...	..
INT1.7	00 0D4C	Vector PIEINT1.7
INT1.8	00 0D4E	Vector PIEINT1.8
INT2.1	00 0D50	Vector PIEINT2.1
INT2.2	00 0D52	Vector PIEINT2.2
...	...	..
INT2.7	00 0D5C	Vector PIEINT2.7
INT2.8	00 0D5E	Vector PIEINT2.8
...	...	..
...	...	..
INT12.1	00 0DF0	Vector PIEINT12.1
INT12.2	00 0DF2	Vector PIEINT12.2
...	...	..
INT12.7	00 0DFC	Vector PIEINT12.7
INT12.8	00 0DFE	Vector PIEINT12.8

Proceso de ejecución de una interrupción mascarable con unidad PIE

Cuando una aplicación utiliza interrupciones, éstas se deben configurar de antemano. El proceso de configuración de interrupciones es una de las primeras rutinas que debe realizar un programa para garantizar su utilización correcta. Este proceso debe seguir los siguientes pasos para interrupciones mascarables:

- Deshabilitar toda interrupción mascarable globalmente (INTM = 1), esto se realiza con la instrucción

```
SETC INTM
```

- Ubicar en apuntador de pila en una localidad de memoria RAM

```
MOV   SP,#DIR_SP   ; localiza el SP en DIR_SP
```

- Limpiar cualquier interrupción pendiente con

```
AND IFR,#0FFFFh
```

- Habilitar interrupciones mascarables de interés poniendo en uno los bits correspondientes en el registro IER (de /INT1 a /INT12). Se debe tomar en consideración la interrupción de grupo x de PIE y habilitar el respectivo bit de INTx. Para el grupo x=1:

```
OR  IER,#0x0001    ; Habilita INT1.x grupo x=1 de PIE
```

- Habilitar interrupciones mascarables del o los periféricos de interés.

  Ejemplo. Habilitar la interrupción de TIMER0 del CPU en el registro PIEIERx:

```
; PIECTRL   0x0000-0CE0 16b PIE, Registro de control
; PIEACK    0x0000-0CE1 16b PIE, Registro de reconocimiento
; PIEIER1   0x0000-0CE2 16b PIE, Registro de habilitación grupo INT1
; PIEIFR1   0x0000-0CE3 16b PIE, Registro de banderas grupo INT1
; En PIE el vector de INT1.7 corresponde al timer0 en 0x0D4C
; Activar las interrupciones de INT1.7 con el bit 6
; (grupo uno, int 7 de PIE)
```

```
EALLOW           ; Habilita escritura a registros protegidos
MOVL  XAR0,#DIR_PIEIER1 ; XAR0 = DIR_PIEIER1
MOV   AL,*XAR0          ; AL = contenido de PIEIER1
OR    AL,#0x0040        ; Pone bit 6 de AL a uno
MOV   *XAR0,AL          ; Reescribe en PIEIER1
EDIS             ; Deshabilita escritura a registros protegidos
```

- Habilitar las direcciones de los vectores de interrupción PIE

```
EALLOW
MOVL   XAR0,#DIR_PIECTRL
MOV    AL,*XAR0
OR     AL,#0x0001  ; pone el bit 0 de PIERCRL en uno (ENPIE=1)
MOV    *XAR0,AL
EDIS
```

- Ligar la subrutina con el vector de interrupción PIE

```
EALLOW
MOVL  XAR2,#DIR_SUB_PER  ; Dirección de vector de interrupciones
                         ; del periférico
MOV   ACC, #_SUB_INT_PER ; Subrutina de atención de
                         ; interrupción del periférico
MOV    AH,#03Fh          ; Agrega parte alta de DIR_SUB_PER
                         ; (depende del mapa de memoria utilizado)
MOVL  *XAR2,ACC          ; Escribe en el vector de interrupciones
                         ; la dirección de salto a subrutina ISR
EDIS
```

Antes de la instrucción de retorno (IRET), la subrutina de interrupción del periférico _SUB_INT_PER debe habilitar de nuevo la interrupción del periférico, en particular, para que vuelva a permitir atender la interrupción. Esto se realiza en el registro PIEACK en el bit de grupo correspondiente

```
; Habilitar grupo  INT1.x en PIEACK dir:  0x0000-0CE1
  EALLOW
  MOVL   XAR0,#DIR_PIEACK    ;  0x0000-0CE1
  MOV    AL, #0x0001
```

```
MOV    *AR0, AL
EDIS
```

- Configurar los periféricos del C28x que utilizarán interrupciones.
- Habilitar toda interrupción mascarable globalmente (INTM = 0), esto se realiza con la instrucción

```
CLRC INTM
```

- El programa principal debe quedarse ejecutando algún ciclo de interés o un ciclo infinito en espera de alguna interrupción. Cuando se presente la interrupción, el CPU debe atenderla si ésta ha sido habilitada.

Para la escritura en registros protegidos se habilitan con EALLOW y se deshabilitan con EDIS; en el procedimiento anterior, se puede agrupar toda la escritura a estos registros con un sólo EALLOW y EDIS.

7.2.1. El watchdog o perro guardián

Es un módulo del DSP que permite estar monitoreando el funcionamiento del sistema, corregir algún problema, en cuanto a su uso o algún problema de bloqueo. Normalmente, los sistemas de este tipo consisten de un temporizador que opera en modo descendente y cuando llega a cero reinicializa al CPU, por tanto debe diseñarse por software un bloque de código que esté vigilando que el contador de watchdog no llege a cero, es decir, esté monitoreando constantemente el funcionamiento del sistema. En el caso de la familia C28x, el contador de watchdog trabaja en modo ascendente y cuando llega a su cuenta máxima reinicializa al DSP. El watchdog puede utilizarse también para reducir los consumos de energía cuando se detecte algún estado de inactividad en el sistema.

El módulo watchdog de la familia de DSP Piccolo puede configurarse para que opere en dos modos, reset (WDRST) e interrupciones (WDINT).
- Modo reset:
La señal WDRST ocasiona que el pin XSR se ponga en bajo cuando el contador de watchdog alcanza el máximo en 512 ciclos de OSCCLK.
- Modo interrupción:
El watchdog es configurado para que emita la interrupción WAKEINT manejada por el módulo PIE.

Configuración de operación en modos de baja potencia

En modo STANDBY todos los relojes de periféricos permanecen apagados, el único periférico que sigue funcionando es el watchdog. La señal WDINT alimenta modos de baja potencia (LPM) y puede utilizarse para levantar al DSP del modo STANDBY.
En modo IDLE, la señal de interrupción WDINT puede generar una interrupción de CPU que lo libere del modo IDLE.

En este trabajo se ha desactivado al watchdog en la mayoría de los caso con el código:

```
DIR_WDCR  .set   07029h      ; Dir. del registro de control de WatchDog
C_WDCR    .set   0068h       ; Máscara  para deshabilitar WatchDog
  EALLOW                     ; Habilita escritura a registros protegidos
          MOVL  XAR1, #DIR_WDCR
          MOV  *AR1, #C_WDCR ; desactiva WatchDog
  EDIS
```

De lo contrario el usuario necesita configurar el watchdog para que opere en sus modos correspondientes y realice un control más estricto sobre el funcionamiento del DSP, siendo necesario realizar la escritura a los registros del watchdog.

Resumen

En este capítulo se trató el proceso de interrupción del CPU por medio de varias fuentes, ya que el proceso de interrupciones es una técnica muy efectiva para la realización de aplicaciones en tiempo real, transferencia de información entre periféricos y dispositivos. Para explotar la potencialidad de cualquier máquina digital siempre se hace uso del manejo de las interrupciones, por tanto, el diseñador debe ser muy cuidadoso en cuanto a su configuración y manejo. Como se vió en este tema, eventualmente se puede perder el contador de programa y puede llevar a terminar el programa de forma inadecuada. Las familias actuales de DSP C28x han introducido el manejo de gran cantidad de fuentes de interrupción a través del módulo PIE, lo que hace que esta familia sea muy versátil en el manejo de periféricos vía esta unidad. En el manejo de periféricos será de mucho interés el manejo de interrupciones para la comunicación con un sistema central, que en nuestro caso es el DSP C28x.

Capítulo 8

Periféricos

Las familias C28x son dispositivos que realizan funciones de DSP y de un microcontrolador, debido a estas características contienen una gran cantidad de periféricos que le permiten conectarse con el mundo externo, pero a la vez realizan funciones específicas. Dependiendo de la familia en concreto los periféricos pueden variar, sin embargo, existe una gama de periféricos básicos que se presentan en casi todas estas familias.

En este capítulo presentamos los periféricos generales tomando como base la familia F28xxx, además, se hace mención a la familia Piccolo F28027 y F28069. Se realiza una descripción muy general, ya que por cada periférico existe su propio manual. En los capítulos siguientes se abordarán otros periféricos.

8.1. Periféricos de las familias C28xxx

Estas familias contienen una diversidad de periféricos que se pueden encontrar en los manuales [19], [36], etc. Aunque en general no todas las familias contienen todos los periféricos, a continuación se enuncian la mayoría de éstos:

- Entradas y salidas (I/O) digitales de propósito general (GPIO).
- Temporizadores del CPU de 32 bits.
- Dos módulos manejadores de eventos (EVA y EVB).
- Un módulo de conversión análogo - digital (ADC).
- Módulo de interfaz de comunicación serial (SCI-A, SCI-B).
- Módulo de interfaz de puerto serial (SPI).
- Controlador de red de área (eCAN) mejorado.

- Módulo I2C.
- Módulo de puerto serial multicanal con buffer (McBSP).
- Transferencia por DMA

8.2. Entradas y salidas de propósito general

El DSP C28x maneja puertos de entrada/salida de propósito general (GPIO), que le permite seleccionar y configurar sus puertos a través de pines externos GPIO, estos pines están agrupados en puertos y dependiendo del DSP puede tener varios puertos; por otro lado, cada uno de los pines GPIO se puede utilizar para diferentes funciones. Los puertos GPIO contienen un conjunto de registros que son utilizados para seleccionar operaciones compartidas en los pines de los dispositivos de familias C281x y más recientes. Estos pines pueden seleccionarse individualmente para operar como entradas/salidas digitales de propósito general o señales I/O de periféricos. También existe un registro calificador de señal de entrada para remover ruido indeseado.

8.2.1. Registros de configuración de GPIO

Permiten la configuración de los puertos y pines de GPIO. Estos registros son protegidos para evitar sobreescritura o datos espurios. Los pines correspondientes a GPIO pueden funcionar como entrada o salida (I/O) y se configuran por los registros GPxDIR. En las familias C281x existen varios conjuntos de estos registros agrupados en los puertos GPIO: A (70C0h), B (70C4h), D (70CCh), E (70D0h), F(70D4h) y G (70D8h) y registros de datos I/O, SET, CLEAR y cambio de estado GPIO. En la tabla 8.1 se describen sólo los registros que corresponden al puerto A. Los registros de datos son no protegidos y permiten accesos normales del usuario. Cada puerto GPIO es controlado por registros que permiten determinar la dirección I/O, fijarlos a uno, limpiarlos, cambiar su estado o enviar un dato.

8.2.2. Puertos GPIO del DSP Piccolo TMS320F28069

Como se ha mencionado, dependiendo de la familia específica, los periféricos, el mapa de memoria y su operación pueden variar significativamente. En la familia Piccolo existen dos puertos GPIO A y B de 32 bits y un puerto de 16 entradas analógicas AIO.
Los registros GPIO del DSP Piccolo se localizan en las direcciones 00 6F80h a 00 6FFFh y consta de dos grupos: GPIOA del 0 a 31 y GPIOB de 32 a 38 (en algunas versiones hasta 47 o 58); cada pin GPIO puede seleccionar hasta cuatro posibles entradas o salidas de periféricos. Como en las familias F281x, los registros de control GPIO son protegidos y los de datos de acceso libre por el usuario. En la figura 8.1 se muestra la forma de seleccionar los pines GPIO y su definición como entrada o salida.

Tabla 8.1. Registros de control y datos del puerto GPIO A familia C281x

Nombre	Dirección (h)	Descripción
GPAMUX	00 70C0	Control de multiplexión
GPADIR	00 70C1	Control de dirección I/O
GPAQUAL	00 70C2	Control de calificación de entrada
GPADAT	00 70E0	de datos
GPASET	00 70E1	de set
GPACLEAR	00 70E2	de clear
GPATOGGLE	00 70E3	de Toggle

Configuración y manejo del módulo GPIO para la familia F2806x

De acuerdo con las necesidades del diseño se debe establecer cómo van a operar los pines asociados a GPIO. En la tabla 8.2, se describen los registros de la familia Piccolo F2806x. En los registros de control o configuración de la tabla 8.2, cada bit "n" corresponde a un bit GPIOn.

- Con el registro GPACTRL se determina por grupos de ocho GPIO a qué velocidad del reloj SYSCLKOUT se muestrearán, es decir que estos pines se pueden estar verificando muy rápido o muy lento.

- Selección del calificador de entrada. El calificador de entrada es especificado en los registros: GPACTRL, GPBCTRL, GPAQSEL1, GPAQSEL2, GPBQSEL1 y GPBQSEL2. La calificación puede ser por muestra de reloj, cada tres o seis muestras o de manera asíncrona.

- Habilitar o deshabilitar las resistencias internas de "pull-up", esto se realiza a través de los registros GPAPUD (GPIO31-0) y GPBPUD (GPIO32-38). Para los pines que manejan funciones de PWM están deshabilitadas por defecto y para los pines con capacidades GPIO, están habilitados.

- Selección de la función del pin. Se configura con los registros GPxMUXn o AIOMUXn, de tal forma que el pin opere como GPIO o cualquiera otro de tres periféricos. En el reset los pines son configurados por defecto como GPIO.
  Si GPxMUX.bit = 0, el pin es configurado como I/O
  Si GPxMUX.bit = 1, el pin es configurado con la funcionalidad de un periférico asociado.
  En el caso de las familias F2802x y F2806x, cada pin GPIO puede multiplexar cuatro funciones una de GPIO y tres de periféricos, por lo tanto, para la selección se utilizan dos bits por GPIO, la combinación 00 corresponde a la configuración I/O.

Tabla 8.2. Registros de control y datos de GPIO F2806x

Nombre	**Dirección (h)**	**Descripción**
GPACTRLA	00 6F80	Control A (GPIO0 a 31)
GPAQSEL1	00 6F82	Calificador A1 (GPIO0 a 15)
GPAQSEL2	00 6F84	Calificador A2 (GPIO16 a 31)
GPAMUX1	00 6F86	Multiplexión A1 (GPIO0 a 15)
GPAMUX2	00 6F88	Multiplexión A2 (GPIO16 a 31)
GPADIR	00 6F8A	Dirección I/O (GPIO0 a 31)
GPAPUD	00 6F8C	Deshabilitación de resistencia "pull up" puerto A (GPIO0 a 31)
GPBCTRLB	00 6F90	Control B (GPIO32 a 58)
GPBQSEL1	00 6F92	Calificador B1 (GPIO32 a 58)
GPBQSEL2	00 6F94	Calificador B2
GPBMUX1	00 6F96	Registro MUXB 1 (GPIO32 a 44)
GPBDIR	00 6F9A	Dirección I/O (GPIO32 a 58)
GPBPUD	00 6F9C	Deshabilitación de resistencia "pull up" puerto B (GPI32 a 44)
AIOMUX1	00 6FB6	Mux Analógico 1 I/O (AIO0 a AIO15)
AIODIR	00 6FBA	Reg. de direcciones analógicas I/O
GPADAT	00 6FC0	Datos A (GPIO0 a 31)
GPASET	00 6FC2	Set A (GPIO0 a 31)
GPACLEAR	00 6FC4	Clear A (GPIO0 a 31)
GPATOGGLE	00 6FC6	Toggle A (GPIO0 a 31)
GPBDAT	00 6FC8	Datos B (GPIO32 a 38)
GPBSET	00 6FCA	Set B (GPIO32 a 38)
GPBCLEAR	00 6FCC	Clear B (GPIO32 a 38)
GPBTOGGLE	00 6FCE	Toggle B (GPIO32 a 38)
GPIOXINT1SEL	0x6FE0	Selección de interrupción de entrada XINT1(GPIO0 a 31)
GPIOXINT2SEL	0x6FE1	Selección de interrupción de entrada XINT2(GPIO0 a 31)
GPIOXINT3SEL	0x6FE2	Selección de interrupción de entrada XINT3(GPIO0 a 31)
GPIOLPMSEL	0x6FE8	Selección LPM GPIO
AIODAT	0x6FD8	Dato I/O analógico (AIO0 a AIO15)
AIOSET	0x6FDA	Set I/O analógico (AIO0 a AIO15)
AIOCLEAR	0x6FDC	Clear I/O analógico (AIO0 a AIO15)
AIOTOGGLE	0x6FDE	Toggle I/O analógico

- Seleccionar la dirección del pin para propósitos generales de I/O. Determina si el pin opera como entrada o salida y se utilizan los registros GPADIR (GPIO31-0), GPBDIR o AIODIR. Por defecto, estos pines se configuran como entrada. Un bit por cada GPIO en estos registros.
  Si GPxDIR.bit = 0, el pin se configura como entrada
  Si GPxDIR.bit = 1, el pin se configura como salida
  x = A o B

- Seleccionar los modos de baja potencia y fuente de inicialización. Se especifican los modos de HALT y STANDBY con los registros GPIOLPMSEL.

- Seleccionar las fuentes externas de interrupción XINT1 a XINT3 a través de los registros GPIOXINTnSEL, con $n = 1, 2, 3$, realmente sólo se utilizan los cinco bits LSb, dándonos 32 posibles combinaciones, donde cada una de ellas corresponde a los GPIO del 0 al 31. La polaridad de la interrupción puede ser configurada en el registro XINTnCR.

Registros de datos GPADAT (GPIO31-0) y GPABDAT (GPIO38-32)

Cada puerto I/O tiene un registro de datos de lectura o escritura. Si se lee, refleja el estado del puerto después del calificador, al escribirlo fija los valores correspondientes en el puerto de salida. Cada bit del registro corresponde a un pin GPIO. El registro GPxDAT refleja el estado de los pines.

Registros GPxSET y AIOSET

Son de sólo escritura, si se le escribe un uno, lo fija en el bit GPIO correspondiente. Escribiéndole un cero, lo ignora. Estos registros se utilizan para manejar los pines específicos de GPIO sin afectar otros pines, si se leen regresan un cero.

Registros GPxCLEAR y AIOCLEAR

Son de sólo escritura y limpia el bit correspondiente al escribirle un uno. Escribiéndole un cero, lo ignora. Si se leen regresan un cero.

Registros GPxTOGGLE y AIOTOGGLE

Son de sólo escritura y escribiéndole un uno en el bit correspondiente cambia su estado. Escribiéndole un cero, lo ignora.

Registro GPAMUX1 del DSP Piccolo

De los periféricos que se verán en este capítulo y posteriores, se pueden seleccionar los pines GPIO como salida o entrada. En el caso particular del DSP Piccolo, con los registros GPAMUX1 y GPAMUX2 se pueden multiplexar los periféricos para utilizarlos en los pines que les corresponde, por cada pareja de bits en estos registros se pueden tener hasta cuatro posibilidades de periféricos en los pines GPIO, como se muestra a continuación. En forma gráfica se puede observar en la figura 8.1.

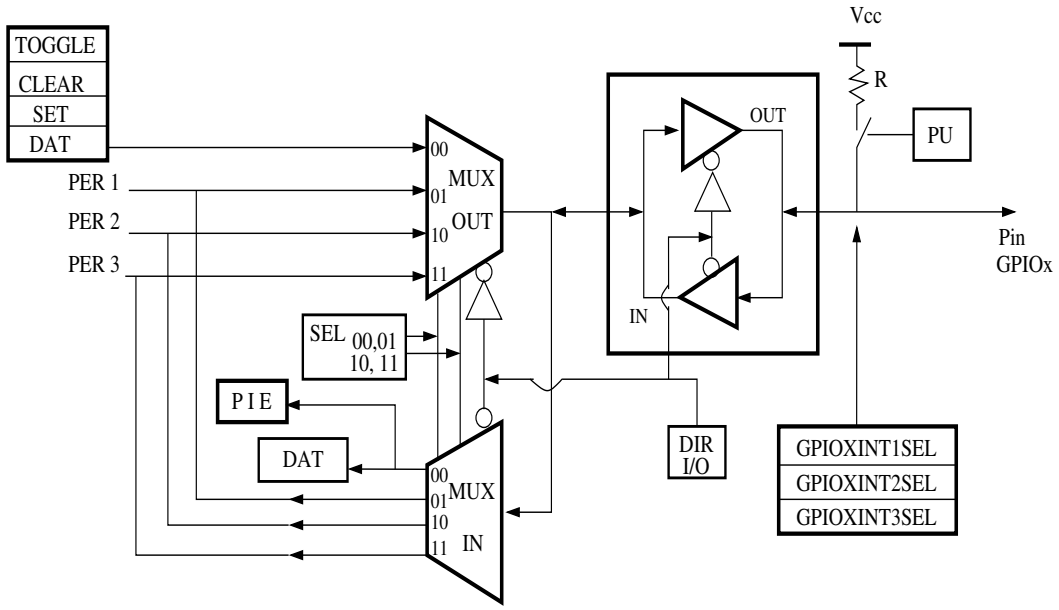


Figura 8.1. Selección de funciones para pines GPIO

Registro GPAMUX1 de DSP Piccolo F28069

Bits \ valor	00	01	10	11
1-0	GPIO0	EPWM1A(O)	Reservado	Reservado
3-2	GPIO1	EPWM1B(O)	Reservado	COMP1OUT(O)
5-4	GPIO2	EPWM2A(O)	Reservado	Reservado
7-6	GPIO3	EPWM2B(O)	SPISOMIA(I/O)	COMP2OUT(O)
9-8	GPIO4	EPWM3A(O)	Reservado	Reservado
11-10	GPIO5	EPWM3B(O)	SPISIMOA(I/O)	ECAP1(I/O)
13-12	GPIO6	EPWM4A(O)	EPWMSYNCI(I)	EPWMSYNCO(O)
15-14	GPIO7	EPWM4B(O)	SCIRXDA(I)	ECAP2(I/O)
17-16	GPIO8	EPWM5A(O)	Reservado	ADCSOCAO(O)
19-18	GPIO9	EPWM5B(O)	SCITXDB(O)	ECAP3(I/O)
21-20	GPIO10	EPWM6A(O)	Reservado	ADCSOCBO(O)

23-22	GPIO11	EPWM6B(O)	SCIRXDB(I)	ECAP1(I/O)
25-24	GPIO12	/TZ1(I)	SCITXDA(O)	SPISIMOB(I/O)
27-26	GPIO13	/TZ2(I)	Reservado	SPISOMIB(I/O)
29-28	GPIO14	/TZ3(I)	SCITXDB(O)	SPICLKB(I/O)
31-30	GPIO15	ECAP2(I/O)	SCIRXDB(I)	SPISTEB(I/O)

Registro GPAMUX2 del DSP Piccolo

Bits \ valor	00	01	10	11
1-0	GPIO16	SPISIMOA(I/O)	Reservado	TZ2(I)
3-2	GPIO17	SPISOMIA(I/O)	Reservado	TZ3(I)
5-4	GPIO18	SPICLKA(I/O)	SCITXDB(O)	XCLKOUT(O)
7-6	GPIO19/XCLKIN	SPISTEA(I/O)	SCIRXDB(I)	ECAP1(I/O)
9-8	GPIO20	EQEP1A(I)	MDXA(O)	COMP1OUT(O)
11-10	GPIO21	EQEP1B(I)	MDRA(I)	COMP2OUT(O)
13-12	GPIO22	EQEP1S(I/O)	MCLKXA(I/O)	SCITXDB(O)
15-14	GPIO23	EQEP1I(I/O)	MFSXA(I/O)	SCIRXDB(I)
17-16	GPIO24	ECAP1(I/O)	EQEP2A(I)	SPISIMOB(I/O)
19-18	GPIO25	ECAP2(I/O)	EQEP2B(I)	SPISOMIB(I/O)
21-20	GPIO26	ECAP3 (I/O)	EQEP2I(I/O)	SPICLKB(I/O)
23-22	GPIO27	HRCAP2(I)	EQEP2S(I/O)	SPISTEB (I/O)
25-24	GPIO28	SCIRXDA(I)	SDAA(I/OD)	/TZ2(I)
27-26	GPIO29	SCITXDA(O)	SCLA(I/OD)	TZ3 (I)
29-28	GPIO30	CANRXA(I)	EQEP2I(I/O)	EPWM7A(O)
31-30	GPIO31	CANTXA(O)	EQEP2S(I/O)	EPWM8A (O)

Registro GPBMUX1 del DSP Piccolo

Bits \ valor	00	01	10	11
1-0	GPIO32	SDAA(I/OD)	EPWMSYNCI(I)	ADCSOCAO(O)
3-2	GPIO33	SCLA(I/OD)	EPWMSYNCO(O)	ADCSOCBO(O)
5-4	GPIO34	COMP2OUT(O)	Reservado	COMP3OUT(O)
7-6	GPIO35(TDI)	Reservado	Reservado	Reservado
9-8	GPIO36(TMS)	Reservado	Reservado	Reservado
11-10	GPIO37(TDO)	Reservado	Reservado	Reservado
13-12	GPIO38/XCLKIN	Reservado	Reservado	Reservado
15-14	GPIO39	Reservado	Reservado	Reservado
17-16	GPIO40	EPWM7A(O)	SCITXDB(O)	Reservado
19-18	GPIO41	EPWM7B(O)	SCIRXDB(I)	Reservado

21-20	GPIO42	EPWM8A(O)	/TZ1(I)	COMP1OUT(O)
23-22	GPIO43	EPWM8B(O)	/TZ2(I)	COMP2OUT(O)
25-24	GPIO44	MFSRA(I/O)	SCIRXDB(I)	EPWM7B(O)
27-26	Reservado	Reservado	Reservado	Reservado
29-28	Reservado	Reservado	Reservado	Reservado
31-30	Reservado	Reservado	Reservado	Reservado

Registro GPBMUX2 del DSP Piccolo

Bits \ valor	00	01	10	11
1-0	Reservado	Reservado	Reservado	Reservado
3-2	Reservado	Reservado	Reservado	Reservado
5-4	GPIO50	EQEP1A(I)	MDXA(O)	/TZ1(I)
7-6	GPIO51	EQEP1B(I)	MDRA(I)	/TZ2(I)
9-8	GPIO52	EQEP1S(I/O)	MCLKXA(I/O)	/TZ3(I)
11-10	GPIO53	EQEP1I(I/O)	MFSXA(I/O)	Reservado
13-12	GPIO54	SPISIMOA(I/O)	EQEP2A(I)	HRCAP1(I)
15-14	GPIO55	SPISOMIA(I/O)	EQEP2B(I)	HRCAP2(I)
17-16	GPIO56	SPICLKA(I/O)	EQEP2I(I/O)	HRCAP3(I)
19-18	GPIO57	SPISTEA(I/O)	EQEP2S(I/O)	HRCAP4(I)
21-20	GPIO58	MCLKRA(I/O)	SCITXDB(O)	EPWM7A(O)
23-22	Reservado	Reservado	Reservado	Reservado
25-24	Reservado	Reservado	Reservado	Reservado
27-26	Reservado	Reservado	Reservado	Reservado
29-28	Reservado	Reservado	Reservado	Reservado
31-30	Reservado	Reservado	Reservado	Reservado

Algunos GPIO no están disponibles en todas la versiones Piccolo o encapsulado. Para la descripción bit a bit de los registros consultar [19], [36], [37].

Ejemplo del uso de los pines GPIO en la tarjeta Piccolo Launch Pad TMS320F28027

El código muestra un conteo ascendente en los cuatro LEDS que están conectados a los pines GPIO0-3.

```
*  Tarjeta Launch Pad TMS320F28027
;  Manejo de los leds de salida en GPI0 al GPIO3
;  Los tiempos se establecen por una subrutina de retardo
;  Switch en GPIO12
;  Oct.  de 2012
               .global    _c_int00
ND             .set 60000   ; Constante de retardo
;  DIRECCIONES DE REGISTROS
DIR_WDCR       .set 07029h  ; Dirección del registro de control WatchDog
DIR_SP         .set 00400h  ; Parte alta de Stack
DIR_GPAMUX1    .set 0x06F86 ; GPIO A MUX 1     (GPIO0-15)
DIR_GPAMUX2    .set 0x06F88 ; GPIO A MUX 2     (GPIO16-31)
DIR_GPADIR     .set 0x06F8A ; GPIO A Dirección (GPIO0-31)
DIR_GPBMUX1    .set 0x06F96 ; GPIO B MUX 1     (GPIO32-GPIO34)
DIR_GPBDIR     .set 0x06F9A ; GPIO B Dirección (GPIO32-GPIO34)
DIR_GPADAT     .set 0x06FC0 ; GPIO B Data   (GPIO0-31)
DIR_GPASET     .set 0x06FC2 ; GPIO B Set    (GPIO0-31)
DIR_GPACLEAR   .set 0x06FC4 ; GPIO B Clear  (GPIO0-31)
DIR_GPATOGGLE  .set 0x06FC6 ; GPIO B Toggle (GPIO0-31)
DIR_GPBDAT     .set 0x06FC8 ; GPIO B Data   (GPIO32-GPIO34)
DIR_GPBSET     .set 0x06FCA ; GPIO B Set    (GPIO32-GPIO34)
DIR_GPBCLEAR   .set 0x06FCC ; GPIO B Clear  (GPIO32-GPIO34)
DIR_GPBTOGGLE  .set 0x06FCE ; GPIO B Toggle (GPIO32-GPIO34)
C_WDCR         .set   0068h ; máscara de WD

               .text
_c_int00
      SETC  INTM             ; Deshabilita INTM

      EALLOW                 ; Habilita escritura a registros protegidos
      MOVL XAR1, #DIR_WDCR
      MOV  *AR1, #C_WDCR     ; Desactiva WatchDog
      EDIS
      MOV  SP,#DIR_SP        ; Localiza el Stack
; Configura  GPAMUX1 para poner el pines como GPIO => MUX 00
      EALLOW
      MOVL  XAR0,#DIR_GPAMUX1
```

```
      MOV   ACC, #0x0000
      MOVL  *XAR0, ACC
; Salidas, LEDS en GPI0 al GPIO3,  DIR = 1
; Switch en GPIO12    como entrada DIR = 0
      MOVL  XAR0,#DIR_GPADIR
      MOV   ACC, #0x100F
      MOVL  *XAR0, ACC
      EDIS

          ZAPA
CICLO_G   MOVL  XAR0,#DIR_GPADAT
          NEG   AL          ; Leds en lógica invertida
          MOV   *XAR0, AL
          NEG   AL
          LC    TARDA       ; ---> Subrutina de retardo
          ADD   AL,#1
          B     CICLO_G,UNC

FIN_R   NOP
        SB   FIN_R,UNC
******  Rutina de retardo
TARDA    MOV  AR1,#ND
TARDA_1  NOP
         RPT #ND
        || NOP
         NOP
         BANZ TARDA_1,AR1--
         NOP
         LRET
        .end
```

8.3. Sistema de reloj

Los DSP Piccolo F28069 contienen cuatro fuentes de reloj, como se observa en la figura 8.2, y a través de los módulos PLL y sus registros pueden cambiar la velocidad del reloj de CPU. Aunque no todas estas fuentes tienen la misma función, si pueden alimentar al CPU, estas fuentes de reloj son [18], [19], [21], [36], [37]:

- OSC1: oscilador interno 1, provee el reloj para el "watchdog", el CPU y el temporizador 2 de CPU.

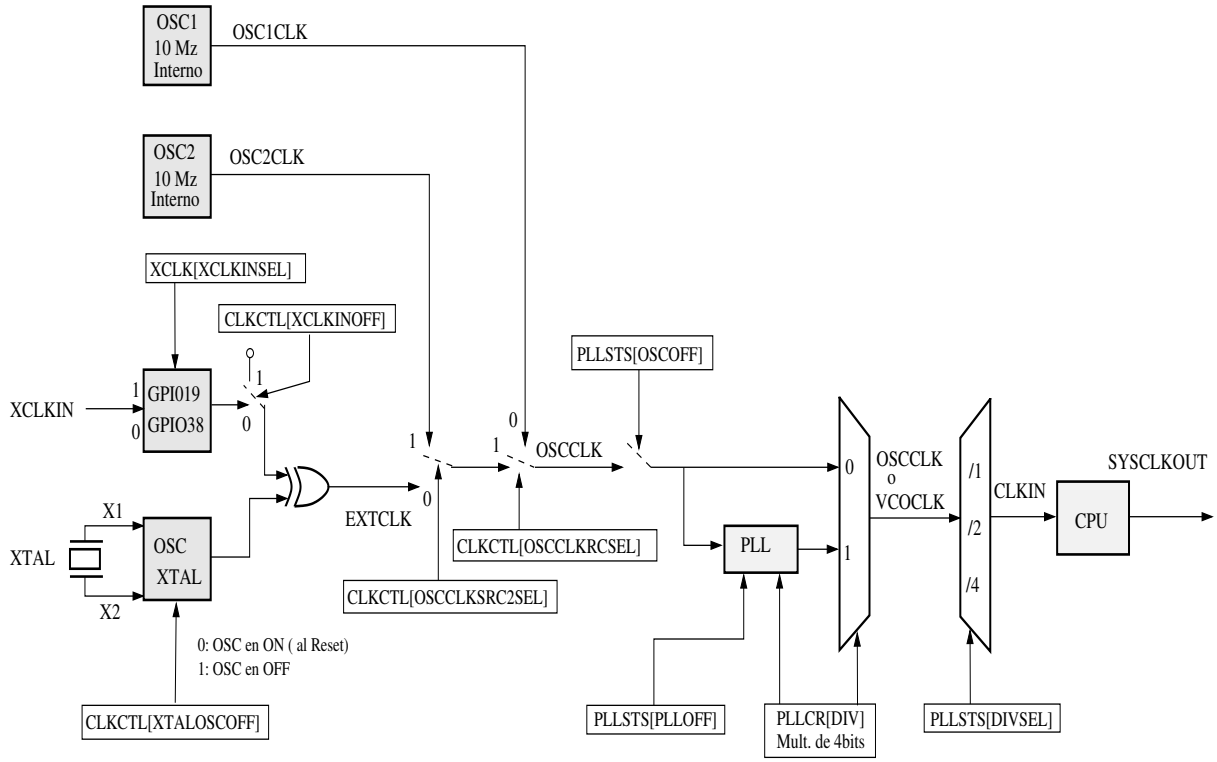


Figura 8.2. Sistema de reloj de los DSP Piccolo F28069

- OSC2: oscilador interno 2, es independiente del OSC1 y también provee el reloj para el "watchdog", el CPU y el temporizador 2 de CPU.
- Cristal externo XTAL, se conecta a los pines de entrada X1 y X2.
- Reloj externo, se conecta la entrada XCLKIN a través de los pines GPIO19 o GPIO38 seleccionados por el bit XCLKINSEL del registro XCLK.

Como se observa en la figura 8.2, la selección, configuración y escalamiento de reloj se realiza a través de los bits respectivos de los registros de la tabla 8.3. Cuando el DSP se inicializa o resetea, por defecto el "watchdog" y el CPU inician con la señal de reloj OSC1CLK, es decir, OSC1 a 10 Mhz.

8.3.1. Reloj de periféricos

La mayoría de dispositivos o periféricos que realizan funciones secuenciadas o sincronizadas, necesitan de un reloj. En la familia C28x existen tres registros PCLKCR0/1/2/3, que permiten habilitar o deshabilitar el reloj de cada uno de los periféricos internos, esto se realiza con el fin de evitar consumos de energía o simplemente que el periférico esté funcionando. En

los registros mencionados existe un bit de habilitación por cada periférico [19], [21], [36], [37], por tanto, cuando utilicemos algún periférico debemos verificar que su reloj de alimentación esté habilitado, escribiendo un "uno" en el bit que le corresponde, ya que por defecto, al reset la mayoría de los periféricos están deshabilitados.

Los registros del sistema de reloj del DSP Piccolo F28069 se encuentran mapeados en las direcciones de la tabla 8.3 que pueden variar para cada dispositivo.

Tabla 8.3. Registros de reloj de periféricos, Piccolo F28069

Registro	**Dirección (h)**	**Descripción**
BORCFG	00 0985	De configuración BOR
XCLK	00 7010	Control de XCLKOUT
PLLSTS	00 7011	De estado PLL
CLKCTL	00 7012	Control de reloj
PLLLOCKPRD	00 7013	Periodo de PLL
INTOSC1TRIM	00 7014	Oscilador recortador interno 1
INTOSC2TRIM	00 7016	Oscilador recortador interno 2
PCLKCR2	00 7019	Control de reloj de periféricos 2
LOSPCP	00 701B	Preescalador de reloj de periféricos
PCLKCR0	00 701C	Control de reloj de periféricos 0
PCLKCR1	00 701D	Control de reloj de periféricos 1
LPMCR0	00 701E	Control de modo de baja potencia 0
PCLKCR3	00 7020	Control de reloj de periféricos 3
PLLCR	00 7021	Control de PLL
SCSR	00 7022	Control y estado
WDCNTR	00 7023	Contador de Watchdog
WDKEY	00 7025	Reset de Watchdog Reset
WDCR	00 7029	Control de Watchdog
PLL2CTL	00 7030	Configuración de PLL2
PLL2MULT	00 7032	Multiplicador de PLL2
PLL2STS	00 7034	Estado de PLL2
SYSCLK2CNTR	00 7036	Contador de reloj SYSCLK2
EPWMCFG	00 703A	Configuración de ePWM, DMA y CLA

En la figura 8.3 se muestra la distribución de los bits de habilitación de reloj para cada periférico de la familia Piccolo.

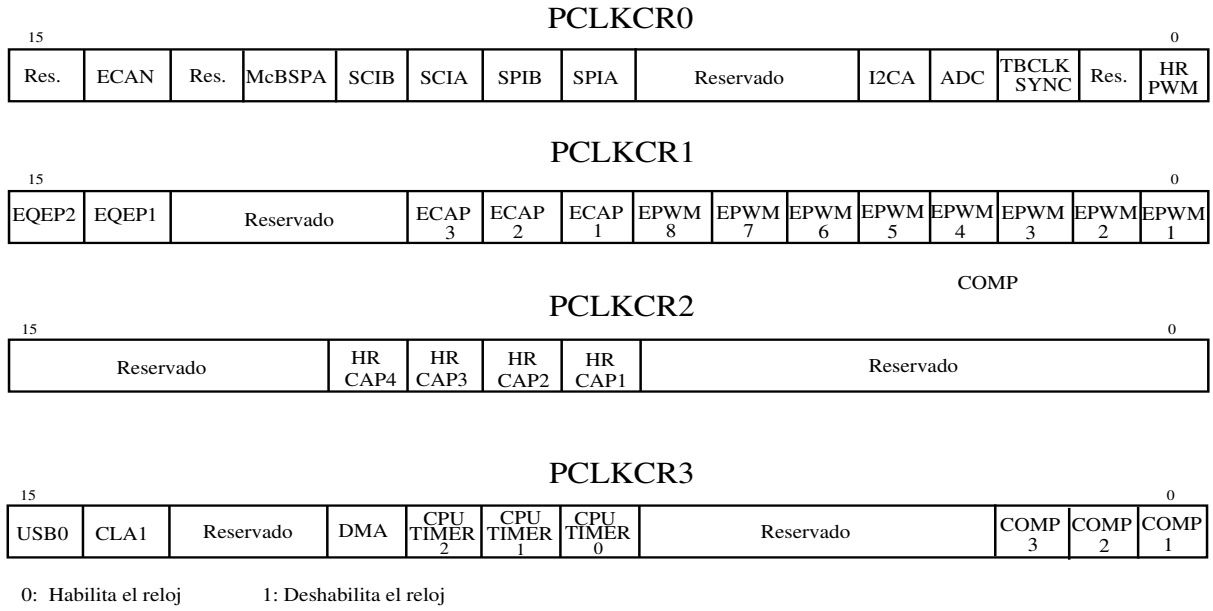


Figura 8.3. Registros PCLKCR0/1/2/3

8.4. Temporizadores del CPU de 32 bits

Las familias C281x y F281x contienen tres temporizadores de CPU (CPU-TIMER0/1/2) su resolución depende del reloj del CPU del procesador. El temporizador TIMER0 se maneja por interrupciones del módulo PIE, el TIMER2 está reservado para utilizarse con herramientas de DSP/BIOS y se maneja exclusivamente por la interrupción INT14, y el TIMER3 comparte la interrupción INT13 con una interrupción externa [19], [36], [37].

Para su operación consta de varios registros que pueden ser modificados por el usuario, dos registros de escalamiento PRD (32b) y TDDR (16b), registros de hardware TIM (32b) y PSC (16b). Los registros TIM y PRD son fijados a su máximo valor 0FFFFh en el reset, el módulo es alimentado por el reloj SYSCLKOUT que se configura por los registros del PLL. Los registros TIM:PSC del temporizador operan como contadores en modo descendente y cuando el registro TIMH:TIM llega a cero, se genera una interrupción del temporizador (TINT), el registro TIMH:TIM es reinicializado con el valor almacenado en el registro PRDH:PRD en el próximo ciclo. Esta característica es útil para operaciones de control y sincronización de muestreo o escritura de periféricos. El diagrama de bloques de operación del temporizador se presenta en la figura 8.4.

La razón de tiempo de interrupción del temporizador está dada por la ecuación (8.1):

$$f_{TINT} = \frac{1}{t_c(TDDRH:TDDR+1)(PRDH:PRD+1)} \tag{8.1}$$

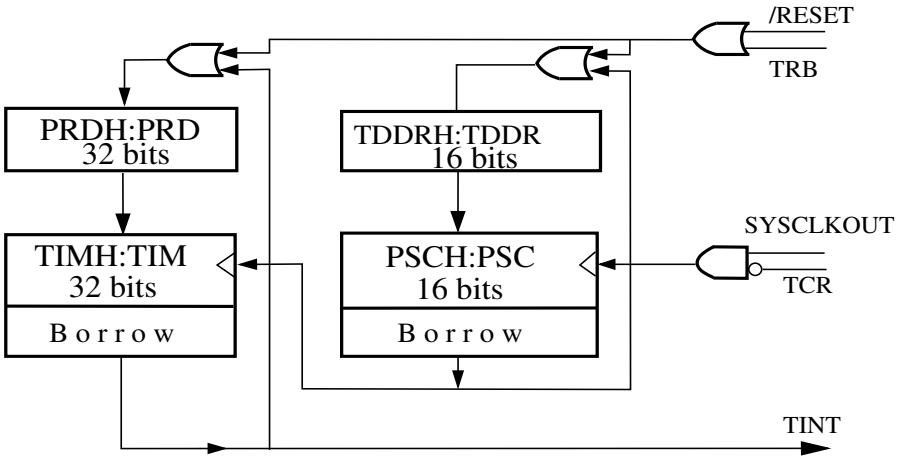


Figura 8.4. Temporizador de CPU

donde:

f_{TINT}, es la razón de interrupción del temporizador
t_c, es el período de reloj SYSCLKOUT
TDDRH:TDDR, es el escalamiento de 16 bits
PRDH:PRD, es el escalamiento de 32 bits,

Por tanto, los temporizadores TIMER0/1/2 de CPU pueden operar a 48 bits.

La operación de estos temporizadores se explica enseguida y va de acuerdo al hardware de la figura 8.4 y se controlan a través de los registros TIMERxTCR cuyos bits se muestran en la tabla 8.4.

- El registro contador de 32 bits TIMH:TIM es cargado con el valor del registro período PRDH:PRD.
- El registro contador PSCH:PSC se decrementa a razón del reloj SYSCLKOUT, y cuando llega a cero decrementa al registro TIMH:TIM. Los registros PSCH:PSC son reinicializados con TDDRH:TDDR cuando llegan a cero.
- Cuando el registro contador TIMH:TIM alcanza el valor de cero, el temporizador emite una señal de interrupción. El registro TIMH:TIM es reinicializado con los registros PRDH:PRD.
- Los registros asociados a los temporizadores deben cargarse de antemano con los valores de diseño, sus nombres y localidades están descritos en la tabla 8.5, todos estos registros son de 16 bits.

La secuencia para la configuración del temporizador es como sigue:

- Deshabilitar interrupciones poniendo en uno el bit de estado INTM (habilitación de interrupciones mascarables).
- Escribir los valores necesarios a los registros PRDH y PRD.
- Escribir los campos TDDR en los bits (7..0) del registro TPR y TDDRH en los bits (7..0) del registro TPRH.
- Habilitar la interrupción del temporizador en el registro PIEIER (Registro de Interrupciones Mascarables).
- Limpiar cualquier interrupción pendiente en el registro PIEIFRx (registro de banderas de interrupción).
- Habilitar interrupciones limpiando el bit de estado INTM.

Los registros TIMERxTPR están formados de los campos PSC(15..8) y TDDR(7..0) y los registros TIMERxTPRH están formados de los campos PSCH(15..8) y TDDRH(7..0).

Tabla 8.4. Registros de control de temporizadores de CPU TIMERxTCR

Bit	**Descripción**
15	TIF: Bandera de interrupción del temporizador 0 : el conteo no ha llegado a cero 1 : el conteo ha llegado a cero
14	TIE: Habilitación de interrupción de temporizador 0 : deshabilita interrupción de temporizador 1 : habilita interrupción de temporizador
13..12	Reservado
11..10	FREE y SOFT: determinan el estado del temporizador cuando éste es detenido. FREE = 1, el TIMERx sigue contando libremente. FREE = 0 y SOFT = 0, es un paro fuerte, el TIMERx se detiene de inmediato en el próximo descuento de TIMH:TIM. Si FREE = 0 y SOFT = 1, es un paro suave el TIMERx se detiene cuando TIMH:TIM llega a cero es decir, cumple con un período de temporizador
9..6	reservado
5	TRB, en 1, permite cargar de nuevo el período TDDRH:TDDR a TIMH:TIM y TDDRH:TDDR a PSCH:PSC En 0, es ignorado
4	TSS permite detener la operación del temporizador en 0, el temporizador está trabajando en 1, el temporizador se detiene
3..0	reservado

Tabla 8.5. Registros de control y configuración de temporizadores del CPU, DSP Piccolo

Nombre	**Dirección (h)**	**Descripción**
TIMER0TIM	00 0C00	Contador de temporizador 0
TIMER0TIMH	00 0C01	Contador H de temporizador 0
TIMER0PRD	00 0C02	Período de temporizador 0
TIMER0PRDH	00 0C03	Período H de temporizador 0
TIMER0TCR	00 0C04	Control de temporizador 0
Reservado	00 0C05	
TIMER0TPR	00 0C06	Preescalador de temporizador 0
TIMER0TPRH	00 0C07	Preescalador H de temporizador 0
TIMER1TIM	00 0C08	Contador de temporizador 1
TIMER1TIMH	00 0C09	Contador H de temporizador 1
TIMER1PRD	00 0C0A	Período de temporizador 1
TIMER1PRDH	00 0C0B	Período H de temporizador 1
TIMER1TCR	00 0C0C	Control de temporizador 1
Reservado	00 0C0D	
TIMER1TPR	00 0C0E	Preescalador de temporizador 1
TIMER1TPRH	00 0C0F	Preescalador H de temporizador 1
TIMER2TIM	00 0C10	Contador de temporizador 2
TIMER2TIMH	00 0C11	Contador H de temporizador 2
TIMER2PRD	00 0C12	Período de temporizador 2
TIMER2PRDH	00 0C13	Período H de temporizador 2
TIMER2TCR	00 0C14	Control de temporizador 2
Reservado	00 0C15	
TIMER2TPR	00 0C16	Preescalador de temporizador 2
TIMER2TPRH	00 0C17	Preescalador H de temporizador 2

Ejemplo de manejo de temporizadores de CPU

```
*  Utilizando interrupciones de TIMER0 se prende y apaga el LED
*  de usuario de la tarjeta Piccolo del F28069
              .global   _c_int00 ;
; Direcciones de registros
DIR_WDCR      .set 07029h  ; Registro WatchDog
DIR_SP        .set 00400h  ; Parte alta de stack
DIR_PIECTRL   .set 0x00CE0 ; PIE, registro de control
DIR_PIEACK    .set 0x00CE1 ; PIE, registro de reconocimiento
DIR_PIEIER1   .set 0x00CE2 ; PIE, registro habilitador de grupo INT1
DIR_PIEIFR1   .set 0x00CE3 ; PIE, registro de banderas de grupo INT1
DIR_GPBMUX1   .set 0x06F96 ; GPIO B MUX 1  (GPIO32-GPIO34)
DIR_GPBDIR    .set 0x06F9A ; GPIO B Dirección (GPIO32-GPIO34)
DIR_GPBDAT    .set 0x06FC8 ; GPIO B Data   (GPIO32-GPIO34)
DIR_GPBSET    .set 0x06FCA ; GPIO B Set    (GPIO32-GPIO34)
DIR_GPBCLEAR  .set 0x06FCC ; GPIO B Clear  (GPIO32-GPIO34)
DIR_GPBTOGGLE .set 0x06FCE ; GPIO B Toggle (GPIO32-GPIO34)
DIR_PIEACK    .set 0x0CE1  ; Reconocimiento de PIE
DIR_PCLKCR3   .set 0x07020 ; De reloj
DIR_ITIM0     .set 0x0D4C  ; Vector de interrupción de TIMER0
DIR_TIMPRDL   .set 0x0C02  ; PRDL de TIMER0
DIR_TIMPRDH   .set 0x0C03  ; PRDH de TIMER0
DIR_TIM0TCR   .set 0x0C04  ; Control de TIMER0
; Máscaras  y valores
C_WDCR        .set 0068h   ; Máscara de WD
PER_H         .set 001FFh  ; Para PRD H
PER_L         .set 0F111h  ; Para PRD L
;
              .text
_c_int00
      SETC    INTM ; deshabilita toda interrupción mascarable
      MOV   SP,#DIR_SP     ; Localiza el Stack

    EALLOW                 ; Habilita escritura a registros protegidos
      MOVL  XAR1, #DIR_WDCR
      MOV   *AR1, #C_WDCR ; desactiva WatchDog
; Configuracion del PIE, ==> habilitar PIE
; En pie el vector de INT1.7 corresponde al timer0 en 0x0D4C
    EALLOW
      MOVL  XAR0,#DIR_PIECTRL
```

```
      MOV   AL,*AR0
      OR    AL,#0x0001    ; 1 al bit PIE
      MOV   *AR0,AL
; Activar las interrupciones de INT1.7 con el bit 6 (Grupo uno int 7 de PIE)
      MOVL  XAR0,#DIR_PIEIER1
      MOV   AL,*XAR0
      OR    AL,#0x0040
      MOV   *AR0,AL
; Ligar la subrutina ISR con el vector de interrupcion PIE
      MOVL  XAR2,#DIR_ITIM0
      MOV   ACC, #_TIMEX
      MOV   AH,#03Fh      ; Cuando ISR está e direcciones 0x03F xxxx
      MOVL  *XAR2,ACC
    EDIS
; Habilita INT1.X grupo x=1 de PIE
      OR   IER,#0x0001

; Se verifica que el byte de control de reloj de periféricos
; tenga habilitado el reloj del timer0 (octavo bit)  PCLKCR3
    EALLOW;               ; Habilita escritura de registros protegidos
      MOVL  XAR0,#DIR_PCLKCR3
      MOV   AL, *XAR0     ; AL bajo contiene el PCLKCR3
      OR    AL, #0x0100   ; Máscara para asegurar el bit ocho en alto
;   AND   AL, #0xFEFF     ; Esta instrucción deshabilitaría el timer0
;   MOV   *AR0, AL
    EDIS                  ; Deshabilita escritura de registros protegidos

; Cargar el PRD con el período deseado
    EALLOW
      MOVL  XAR1,#DIR_TIMPRDH
      MOV   AL,#PER_H     ; Carga el ACC con el valor de período deseado
      MOV   *AR1,AL
      MOVL  XAR1,#DIR_TIMPRDL
      MOV   AL,#PER_L     ; Carga el ACC con el valor de período PRD bajo
      MOV  *AR1, AL       ; Pasa el valor al PRD bajo
; Activar TIE en TIMER0TCR 0x0C04
      MOVL  XAR0,#DIR_TIM0TCR
      MOV   AL, *XAR0
      OR    AL, #0x4000
      MOV   *XAR0, AL
    EDIS
```

```
; Configurar los GPIO, el led esta en GPIO34 que corresponde al puerto B
; en los MUX 0=GPIO, en DIR 1=''output'' 0=''input'', se ocupan registros
; para clear,  set y toggle
; se configura el GPBMUX1 para poner pines como GPIO (0x 0000)
    EALLOW
      MOVL  XAR0,#DIR_GPBMUX1
      MOV   ACC, #0x0000
      MOVL  *XAR0, ACC
; Pines GPIO 32-34 como salidas se ponen a ``1''
      MOVL  XAR0,#DIR_GPBDIR
      MOV   ACC, #0x0007
      MOVL *XAR0, ACC
    EDIS

; Habilita globalmente interrupciones mascarables
       CLRC INTM

ESPERA NOP                ; Ciclo de espera infinito
       B  ESPERA,UNC
; Subrutina de interrupción de temporizador TIMER0
_TIMEX MOVL XAR0,#DIR_GPBTOGGLE
       MOV  ACC, #0x0007  ; TOGGLE al LED
       MOVL *XAR0,ACC
; Reconocer la interrupción para permitir futuras interrupciones
; de INT1.x PIEACK
    EALLOW
      MOVL XAR0,#DIR_PIEACK
      MOV  AL,#0x0001
      MOV  *AR0,AL
    EDIS
; Reset al TIF de  TIMER0TCR para futuras interrupciones del timer0
     MOVL XAR0,#0x0C04
     MOV  AL,*AR0
     OR   AL,#0x8000
     MOV  *AR0,AL
     IRET                 ; Regreso de interrupción
     .end
```

Ejemplo de manejo de leds en la tarjeta Launch pad Piccolo F28027

```
;   Uso del TIMER0 con  Launch Pad F28027
;   Enciende LEDS secuencialmente
;   base de tiempo TIMER0
;
                .global     _c_int00

;*  DIRECCIONES DE REGISTROS
DIR_WDCR        .set 07029h  ; Dir. del registro de control de WatchDog
DIR_SP          .set 00600h  ; Parte alta de stack
DIR_PIECTRL     .set 0x00CE0 ; Dir. registro de control de PIE
DIR_PIEACK      .set 0x00CE1 ; Dir. registro de reconocimiento de PIE
DIR_PIEIER1     .set 0x00CE2 ; Dir. registro de grupo IER de PIE
DIR_PIEIFR1     .set 0x00CE3 ; Dir. registro de grupo IFR de PIE
DIR_GPAMUX1     .set 0x06F86 ; GPIO A MUX 1     (GPIO0-15)
DIR_GPAMUX2     .set 0x06F88 ; GPIO A MUX 2     (GPIO16-31)
DIR_GPADIR      .set 0x06F8A ; GPIO A Dirección (GPIO0-31)
DIR_GPADAT      .set 0x06FC0 ; GPIO B Dato      (GPIO0-31)
DIR_GPASET      .set 0x06FC2 ; GPIO B Set       (GPIO0-31)
DIR_GPACLEAR    .set 0x06FC4 ; GPIO B Clear     (GPIO0-31)
DIR_GPATOGGLE   .set 0x06FC6 ; GPIO B Toggle    (GPIO0-31)
DIR_GPBMUX1     .set 0x06F96 ; GPIO B MUX 1     (GPIO32-GPIO34)
DIR_GPBDIR      .set 0x06F9A ; GPIO B Direción  (GPIO32-GPIO34)
DIR_GPBDAT      .set 0x06FC8 ; GPIO B Dato      (GPIO32-GPIO34)
DIR_GPBSET      .set 0x06FCA ; GPIO B Set       (GPIO32-GPIO34)
DIR_GPBCLEAR    .set 0x06FCC ; GPIO B Clear     (GPIO32-GPIO34)
DIR_GPBTOGGLE   .set 0x06FCE ; GPIO B Toggle    (GPIO32-GPIO34)
DIR_PIEACK      .set 0x0CE1
DIR_PCLKCR3     .set 0x07020
DIR_ITIM0       .set 0x0D4C
DIR_TIMPRDL     .set 0x0C02
DIR_TIMPRDH     .set 0x0C03
DIR_TIM0TCR     .set 0x0C04
C_WDCR          .set 0068h   ; Máscara de WD
PER_H           .set 000A4h  ; Para PRD H E04h
PER_L           .set  0E1C0h ; Para PRD L E1C0h
cont            .word 0      ; Guarda contador
                .text
_c_int00
        SETC    INTM         ; Deshabilita INTM
```

```
        MOV   SP,#DIR_SP     ; Localiza el stack
      EALLOW                 ; Habilita escritura a registros protegidos
        MOVL  XAR1, #DIR_WDCR
        MOV   *AR1, #C_WDCR  ; Desactiva WatchDog
; Habilitar PIE
        MOVL  XAR0,#DIR_PIECTRL
        MOV   AL,*AR0
        OR    AL,#0x0001
        MOV   *AR0,AL
; Activar las interrupciones de INT1.7 con el bit 6 (Grupo uno int 7 de PIE)
        MOVL  XAR0,#DIR_PIEIER1
        MOV   AL,*XAR0
        OR    AL,#0x0040
        MOV   *AR0,AL
; Ligar la subrutina con el vector de interrupción PIE
        MOVL  XAR2,#DIR_ITIM0
        MOV   ACC, #_TIMEX
        MOVL  *XAR2,ACC
; Carga el periodo de TIMER0
        MOVL  XAR1,#DIR_TIMPRDH
        MOV   AL, #PER_H     ; Carga el ACC con el valor de período deseado
        MOV   *AR1,AL
     MOVL  XAR1,#DIR_TIMPRDL
        MOV   AL,#PER_L      ; Carga el ACC con el valor de período PRD bajo
        MOV   *AR1, AL       ; Pasa el valor al PRD bajo
; Activar TIE en TIMER0TCR 0x0C04, habilita interrup. de TIMER0
        MOVL  XAR0,#DIR_TIM0TCR
        MOV   AL, *XAR0
        OR    AL, #0x4000
        MOV   *XAR0, AL
; Configura el GPAMUX1, pone pines como GPIO
        MOVL  XAR0,#DIR_GPAMUX1
        MOV   ACC, #0x0000
        MOVL  *XAR0, ACC
; Salidas, LEDS en GPI0 al GPIO3, DIR = 1
; Switch en GPI012 como entrada   DIR = 0
        MOVL  XAR0,#DIR_GPADIR
        MOV   ACC, #0x100F
        MOVL  *XAR0, ACC
; PoneGPIO 32-34 como salidas
        MOVL  XAR0,#DIR_GPADIR
```

```
        MOV   ACC, #0x0007
        MOVL  *XAR0, ACC
; Se verifica que el byte de control de reloj de periféricos
; tenga habilitado el reloj del timer0 (octavo bit) en
; PCLKCR3, dirección 0x00007020
        MOVL  XAR0,#DIR_PCLKCR3
        MOV   AL, *XAR0      ; AL bajo contiene el PCLKCR3
        OR    AL, #0x0100    ; Máscara para asegurar el bit ocho en alto
; AND AL, #0xFEFF esta instrucción deshabilitaría el timer0
;       MOV   *AR0, AL
; Reset al TIF para futuras interrupciones del timer0
; TIMER0TCR 0x0C04
        MOVL XAR0,#0x0C04
        MOV  AL, *AR0
        OR   AL, #0x8000
        MOV  *AR0, AL
        EDIS                 ; Deshabilita escritura de registros protegidos
; Habilita INT1.X grupo x=1 de PIE
        OR  IER,#0x0001
; Habilita INT globales
        CLRC INTM
        ZAPA
        MOV  @cont,AL
FIN_R NOP
        SB   FIN_R,UNC
;  Subrutina de interrupción de TIMER0
_TIMEX MOV  AL,@cont
        NEG  AL              ; Leds en lógica invertida
        MOVL XAR0,#DIR_GPADAT
        MOV  *AR0,AL
        NEG  AL
        ADD  AL,#1
        MOV  @cont,AL
; Reconocer la interrupción para permitir futuras interrupciones
; de INT1.x PIEACK 0x0000-0CE1
       EALLOW
        MOVL XAR0,#DIR_PIEACK
        MOV  AL, #0x0001
        MOV  *XAR0, AL
; Reset al TIF y TIE para futuras interrupciones del timer0
        MOVL XAR0,#DIR_TIM0TCR
```

```
 MOV  AL, *XAR0
 OR   AL, #0xC000
 MOV  *XAR0, AL
EDIS
 IRET
 .end
```

8.5. Convertidor análogo digital (ADC)

Las arquitecturas de DSP C28x contienen un módulo de conversión analógico digital (ADC) de 16 canales, que se pueden configurar para trabajar con los manejadores de eventos A y B. Aunque existen múltiples entradas y dos secuenciadores, cuenta con dos bloques de muestreo y retención, y un sólo convertidor ADC.
El módulo puede operar en dos modos [22]:
• Como un módulo de 16 canales simultáneos en cascada, figura 8.5.
• Como dos módulos de ocho canales secuenciados, figura 8.6.

Características y funciones

- El sistema ADC consiste de 16 canales de entradas analógicas con conversión de 12 bits.
- El intervalo de voltajes de entrada es de 0 a 3 volts.
- Utiliza dos bloques de ocho entradas analógicas cada una. Cada bloque contiene su propio circuito muestreador retenedor.
- La señal de inicio de conversión (SOC), puede ser manejada externamente, por software o un manejador de eventos EVA y EVB.
- 16 registros individuales para almacenar el resultado de cada canal.
- Conversión rápida, de 12.5 Millones de muestras por segundo (MSPS), con un reloj del ADC de 25 Mhz en la familia F2812.
- Control por interrupciones en el fin de secuencia (EOS).
- Los dos módulos de ocho canales tienen la capacidad de autosecuenciar una serie de conversiones, cada módulo puede seleccionar cualquiera de los ocho canales a través del multiplexor analógico (MUX).

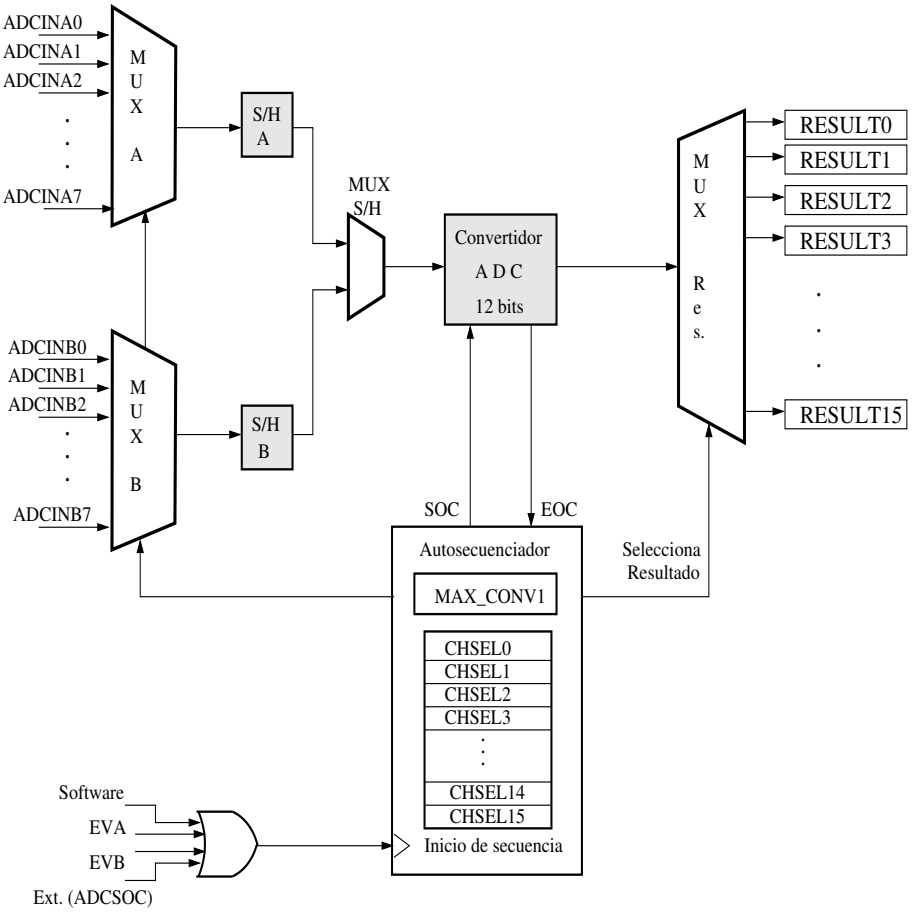


Figura 8.5. Convertidor ADC operando en modo cascada

- En modo cascada, puede autosecuenciar los 16 canales en forma simple, una vez que la conversión es completa, el valor digital del canal convertido se puede leer en el registro ADCRESULTn seleccionado, $n = 0, \cdots, 15$. En este modo se puede convertir el mismo canal múltiples veces, permitiendo al usuario ejecutar algoritmos de sobremuestreo.

- El valor digital de entrada analógica está dado por:

$$\text{Valor digital} = 4095 \frac{\text{Voltaje analógico de entrada} - V_{VREFLO}}{Vcc} \tag{8.2}$$

donde V_{VREFLO} es un voltaje de referencia bajo, y el voltaje analógico de entrada está entre $0 < Vi(t) < Vcc$, Vcc puede variar dependiendo de la familia, usualmente es 3.3 V.

En la tabla 8.6 se describen los registros asociados con el módulo convertidor ADC del DSP C28x. Estos registros están mapeados en el espacio de periféricos y pueden ser accedidos sólo como operandos de 16 bits.

Dependiendo de la familia C28x, la localización, nombre y funcionalidad de los registros del módulo ADC pueden cambiar. Por ejemplo, en el DSP TMS320F28069, los registros del módulo convertidor ADC se encuentran en las localidades 00 7100 h a 00 717F h.

Tabla 8.6. Registros del módulo convertidor ADC familia C281x

Registro	**Dirección (h)**	**Descripción**
ADCTRL1	00 7100	Control ADC 1
ADCTRL2	00 7101	Control ADC 2
ADCMAXCONV	00 7102	Conversión máxima
ADCCHSELSEQ1	00 7103	Control de selección de secuencia 1
ADCCHSELSEQ2	00 7104	Control de selección de secuencia 2
ADCCHSELSEQ3	00 7105	Control de selección de secuencia 3
ADCCHSELSEQ4	00 7106	Control de selección de secuencia 4
ADCASEQSR	00 7107	Estado de autosecuencia
ADCRESULT0	00 7108	Resultado de conversión 0
ADCRESULT1	00 7109	Resultado de conversión 1
ADCRESULT2	00 710A	Resultado de conversión 2
...	...	...
...	...	...
ADCRESULT14	00 7116	Resultado de conversión 14
ADCRESULT15	00 7117	Resultado de conversión 15
ADCTRL3	00 7118	Control ADC 3
ADCTST	00 7119	Bandera de estados

8.5.1. Principio de operación del secuenciador de autoconversión

El convertidor ADC consta de dos secuenciadores de ocho estados independientes (SEQ1 y SEQ2) o un secuenciador de 16 estados. Un estado significa el número de conversiones que pueden ser ejecutadas en la secuencia. Por cada conversión se puede seleccionar cualquiera de los 16 canales de entrada a través del multiplexor analógico, después de que se realiza la conversión, el valor del canal seleccionado es almacenado en el registro resultado ADCRESULTn, en cada registro resultado $n = 0, \cdots, 15$ se encontrará un valor convertido [17].

También es posible realizar un sobremuestreo en un mismo canal. El registro CONVxx (xx:0 a 15) contiene el número de canal muestreado y retenido, y el resultado de la conversión es escrito en el registro resultado ADCRESULTn.

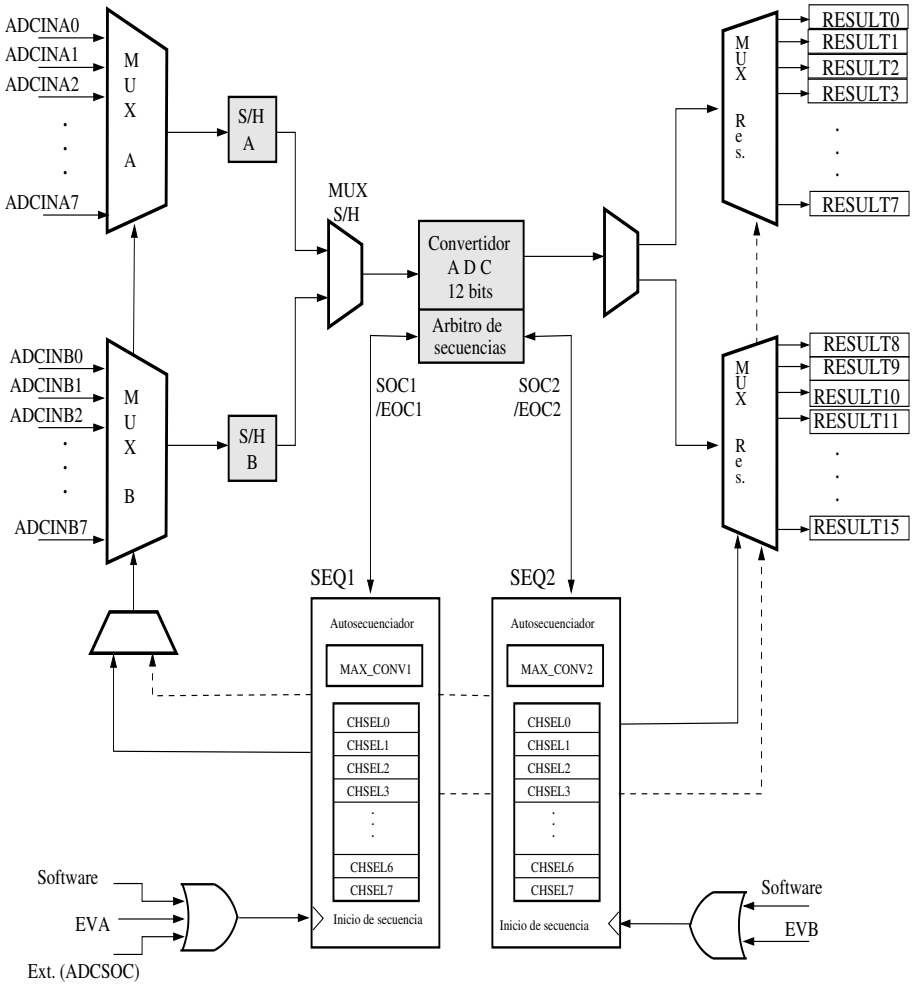


Figura 8.6. Módulo convertidor ADC operando en modo secuenciador dual

8.5.2. Inicialización de registros

Para la utilización del convertidor ADC, es necesario configurarlo e iniciarlo a través de sus registros. El procedimiento general se enumera a continuación:

1. Configurar los registros del ADC definiendo el modo de operación.
   - Reinicializar el módulo ADC. Bit reset (14) del registro ADCTRL1.
   - Modo cascada =1, o secuencial= 0, bit SEQ_CASC(4) del registro ADCTRL1.
   - Modo de muestreo simultáneo. En bit SMODE_SEL del registro ADCTRL3.
   - Fuente de señal SOC, registro ADCTRL2.
   - Divisor de reloj, ADCCLKPS (3..0) del registro ADCTRL3.

- Cantidad máxima de conversiones, registro ADCMAXCONV

2. La conversión inicia cuando se presenta una señal de inicio de conversión SOC.

3. Se lee el valor MAX CONVn del registro ADCMAXCONV y se carga en SEQ CNTRn en el registro ADCASEQSR.

4. La conversión inicia, los bits SEQ CNTR3 ... CNTR0 son decrementados en uno por cada conversión.

5. Cuando la conversión actual es completa, el resultado de cada canal es escrito en su respectivo registro ADCRESULTn.

6. Al completar todas las conversiones, SEQ CNTR3 ... CNTR0 = 0, se emite una interrupción INT SEQn. De lo contrario se regresa al paso de inicio de conversión.

La inicialización de la conversión SOC se puede realizar por varias fuentes: una fuente externa, por software, por eventos de PWM, interrupciones del temporizador de CPU o una interrupción ADCINT1/2.

Preescalamiento de reloj

El reloj de periféricos HSPCLK es dividido por los bits ADCCLKP(3..0) del registro ADCTRL3, y en cascada todavía se puede dividir por 2 si el bit CPS del registro ADCTRL1(7) está en uno. El reloj ADCCLK que alimenta al generador de pulso de inicio de conversión SOC se calcula:

$$ADCDLK = \frac{HSPCLK}{ADCCLKP(3..0) + 1 + CPS} \tag{8.3}$$

El número de conversiones en una secuencia es controlado por el registro ADCMAXCONV, el cual es cargado automáticamente en el contador de estados de secuencia (SEQ CNTR3 – 0), este valor puede ser de 0 a 15, donde el número máximo de conversiones es (MAX CONVn + 1).

La conversión inicia de forma similar a la mayoría de convertidores ADC, cuando recibe una señal de inicio SOC, esta señal carga los bits SEQ CNTRn y se consideran los canales que fueron especificados en el registro ADCCHSELSEQn.

Una vez que SEQ CNTRn llega a cero, pueden pasar dos cosas dependiendo del estado del bit de corrida libre (CONT RUN) en el registro ADCTRL1:
• Si el bit CONT RUN = 1, la secuencia de conversión se inicializa toda de nuevo. En este caso se debe asegurar haber leído los registros resultado para evitar sobreescritura.
• Si el bit CONT RUN = 0, la secuencia está en el último estado y SEQ CNTr está en cero.

Disparadores de SOC

Cada secuencia tiene su propio disparador de entrada que puede ser habilitado o no. El evento SOC, puede ser levantado de diferentes formas:
• Por software.
• Por el manejador de eventos EVA o EVB.
• Por el pin externo SOC.

Una vez que se disparó el SOC la secuencia de conversión no puede detenerse, y debe esperarse hasta el fin de conversión de la secuencia (EOS) o inicializar otra secuencia a través del reset.

Preescalador del reloj ADC

El reloj de periféricos HSPCLK es dividido por el valor de los bits ADCCLKPS(3:0) del registro ADCTRL3. Además se puede agregar un divisor por dos si el bit CPS = 1 del registro ADCTRL1.

8.6. Convertidor ADC de la familia Piccolo

La operación de los convertidores ADC de la familia Piccolo es un poco diferente a las familias anteriores C28x. Este convertidor no está basado en secuencias sino en pulsos de inicio de conversión (SOC). Su principio de operación está centrado alrededor de la configuración individual de los canales de conversión llamados SOC.

La diferencia básica respecto de las familias anteriores es que se pueden configurar 16 fuentes SOC por disparo, una ventana y múltiples fuentes del disparo del canal:
- Inicialización por software (S/W).
- Generadores de ePWM del 1 al 8, SOCA y SOCB.
- Interrupción XINT2 vía GPIO.
- Temporizadores 0,1,2 del CPU.

Adicionalmente retroalimenta las salidas ADCINT1/2 para utilizarlas como interrupciones. Por otro lado, puede generar hasta nueve interrupciones para utilizarlas por el módulo PIE.

Estas diferencias pueden notarse respecto a otras familias C28x, como se observa en la figura 8.7.

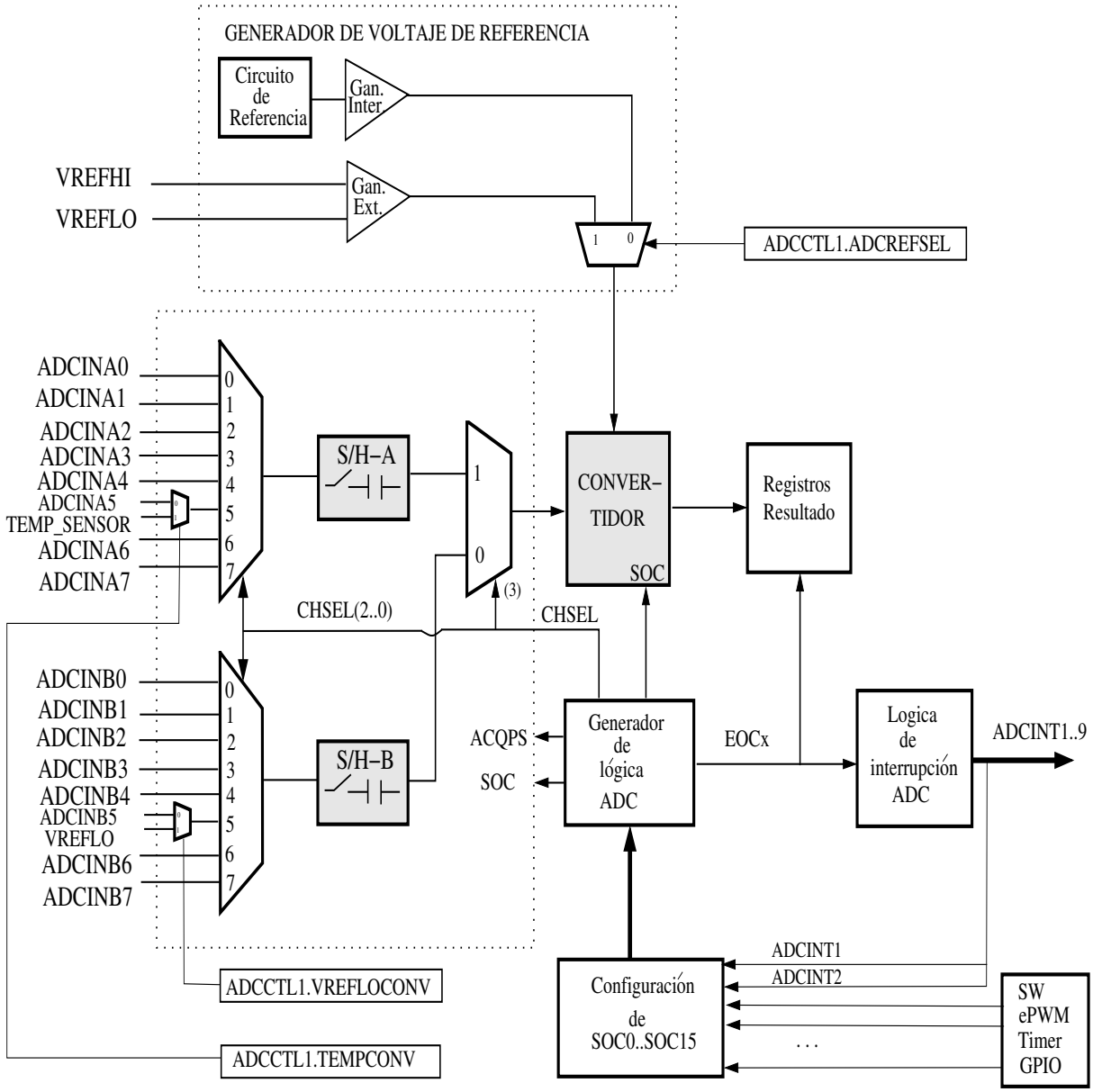


Figura 8.7. Convertidor ADC del DSP Piccolo

8.6.1. Principio de operación

El concepto SOC es un tipo de configuración que permite a cada canal operar en modo de conversión simple. Existen tres tipos de configuración:
- Un pulso fuente inicia la conversión del canal.
- Canal a convertir.
- Adquisición por tamaño de ventana.

Cada SOC es configurado independientemente y puede tener cualquier combinación de disparo, canal y tamaño de ventana disponible. Cada SOC puede configurarse para el mismo disparo, canal y/o una ventana de adquisición, lo que permite una gran flexibilidad de configuración de intervalos de conversión para muestras individuales de diferentes canales con diferentes disparos, para un sobremuestreo en un canal con un simple disparo, y para la creación de una serie propia de conversiones de todos los diferentes canales con un simple disparo.

La fuente de SOCx es configurada por una combinación del campo TRIGSEL de los registros ADCSOCxCTL y los bits apropiados en registros ADCINTSOCSEL1 o ADCINTSOCSEL2. Por software también se puede forzar un evento SOC con el registro ADSOCFRC1. El canal y el tamaño de la ventana de muestreo para SOCx es configurado con los campos CHSEL y ACQPS de los registros ADCSOCxCTL.

8.6.2. Configuración

Para la configuración del convertidor de la familia Piccolo se requieren los siguientes pasos:
• Habilitar el reloj vía el registro PCLKCRO.
• En los registros ADCSOCxCTL ($x = 0..15$), configurar los campos ACQPS, CHSEL y TRIGSEL.
- El campo ACQPS (5..0) de 6b, determina el tamaño de ventana de muestreo y retención (S/H), el valor mínimo es 7 y se debe escribir el valor menos uno, el valor máximo es de 64.
- CHSEL (9..6): Selecciona el canal a convertir.
- TRIGSEL (15..11): Selecciona la fuente de disparo SOC para el canal.
• Definir los modos de muestreo:
- Secuencial:
Los cuatro bits CHSEL del registro ADCSOCxCTL definen el canal a convertir.
- Simultáneo:
Los tres bits LSb de CHSEL de ADCSOCxCTL determinan qué par de canales son convertidos.

8.6.3. Conversión simple por disparo de SOC

Este modo permite ejecutar una conversión simple en el próximo disparo SOC en un esquema circular, el modo sólo es válido para canales presentes en el círculo. De lo contrario, las prioridades se establecen en el campo SOCPRIORITY del registro ADCSOCPRIORITYCTL.

Existe un apuntador de prioridad circular RRPOINTER que apunta a algún SOC, en reset se pone con un valor de 32 y establece la prioridad máxima para SOC0. Los canales se van convirtiendo en el orden de 0 a 15 en forma circular conforme se presente su respectiva señal SOC. Si se presenta un requerimiento SOC de menor prioridad del apuntador RRPOINTER, entonces se atiende en la siguiente vuelta [17].

8.6.4. Modo de muestreo simultáneo

El módulo convertidor ADC contiene dos circuitos S/H que le permiten a dos canales muestrearse simultáneamente, este modo se configura con el registro ADCSAMPLEMODE junto con el bit SIMULEN0 de la siguiente forma:

- Cada SOC disparará un par de conversiones.
- El par de canales a convertir, corresponderán uno al bloque A y otro al B seleccionados en el campo CHSEL del registro SOCx, con valores válidos en este modo de 0..7.
- Ambos canales seleccionados se muestrearán simultáneamente.
- El canal A se convertirá primero y el B en seguida.
- El pulso par de EOCx será generado con base en la finalización de la conversión del canal A y el pulso impar para el fin de B.
- El resultado de la conversión del canal A es almacenado en el registro par ADCRESULTx y el B en el impar de ADCRESULTx.

8.6.5. Operación de interrupción y EOC

Cuando se configuran las 16 SOC independientemente, se generan 16 pulsos de fin de conversión (EOC). En modo de muestreo secuencial se asocia una señal EOCx a cada SOCx. En modo simultáneo una señal EOCx par seguida de una EOCx impar es asociada con dos señales consecutivas par e impar EOCx.

El módulo ADC maneja nueve interrupciones vía el módulo PIE. Cada una de las nueve interrupciones se pueden manejar por cualquiera de las 16 señales EOCx, éstas se configuran en el registro INTSELxNy. Además, las señales ADCINT1 y ADCINT2 se pueden configurar para generar disparos SOCx, esto es útil en la creación de un flujo continuo de conversiones. La forma de asociar las señales EOCx con las interrupciones ADINTx se muestra en la figura 8.8

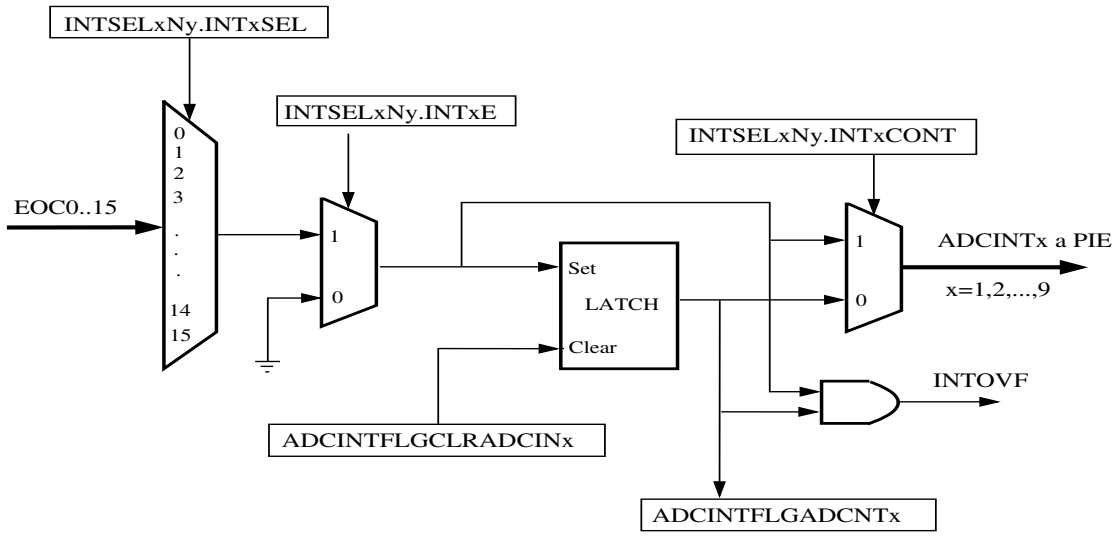


Figura 8.8. Interrupciones de ADC

8.6.6. Secuencia de encendido de ADC

En el reset el módulo ADC permanece apagado, por tanto, para su encendido se siguen los pasos:

- Poner en uno el bit ADCENCLK (b3) del registro de control PCLKCR0 del DSP Piccolo. Este bit en uno habilita el reloj que alimenta al módulo ADC [21].
- Para el uso de una referencia externa, habilitar el bit ADCREFSEL en el registro ADCCTL1.
- Encender las referencias, de rejilla y circuitos analógicos, bits ADCPWDN, ADCBGPWD, ADCREFPWD del registro ADCCTL1. Antes de la primera conversión se requiere un retardo de un milisegundo.
- Habilitar el ADC poniendo el bit ADCENABLE del registro ADCCTL1.

8.6.7. Voltajes convertidos con referencias interna y externa

Con referencia interna, VREFLO es conectado a tierra.

$$\text{Valor digital} = \begin{cases} 0 & Vin \leq 0 \text{ V} \\ 4096\frac{(Vin - VREFLO)}{3.3V} & 0 < Vin < 3.3 \text{ V} \\ 4095 & Vin \geq 3.3 \text{ V} \end{cases} \tag{8.4}$$

Con referencias externa VREFLO (baja) y VREFHI (alta)

$$\text{Valor digital} = \begin{cases} 0 & Vin \leq VREFLO \\ 4096\frac{(Vin-VREFLO)}{VREFHI-VREFLO} & VREFLO < Vin < VREFHI \\ 4095 & Vin \geq VREFHI \end{cases} \tag{8.5}$$

Todos los valores fraccionarios son truncados.

Registros

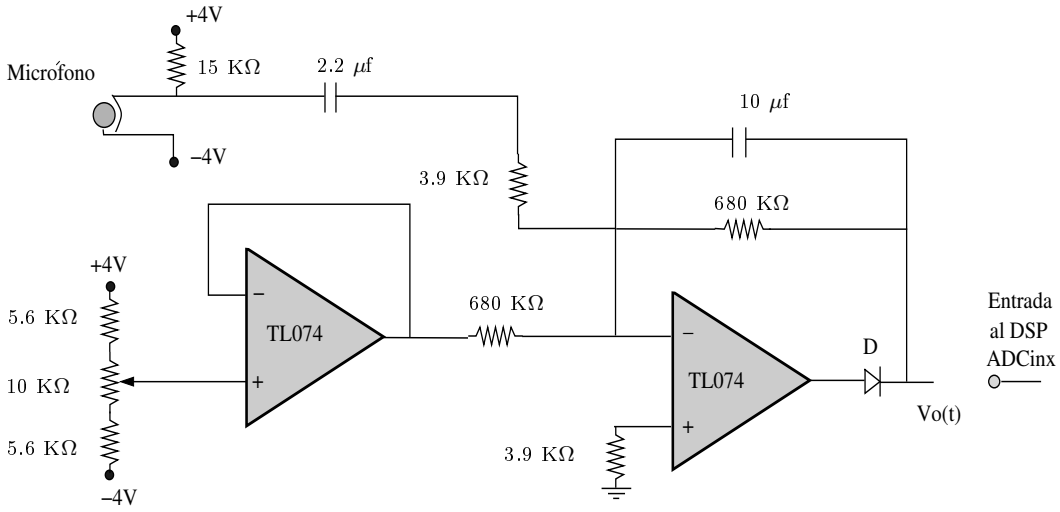
Los registros del convertidor de configuración y estado ADC de la familia Piccolo se encuentran localizados en el mapa de memoria en el bloque dos de periféricos, dirección 00 7000h, la mayoría son del tipo protegidos, a excepción de los registros de resultados ADCRESULTx que se encuentran en el bloque cero de periféricos, direcciones 00 0800h (00 00E00h). Los nombres y funciones de estos registros se muestran en la tabla 8.7, presentando su dirección como un offset sobre la dirección base, la descripción bit a bit de estos registros se pueden encontrar en el manual [17]. Los registros marcados en la tabla 8.7 con (*), son del tipo protegidos.

8.6.8. Circuitos electrónicos externos para el ADC y DAC

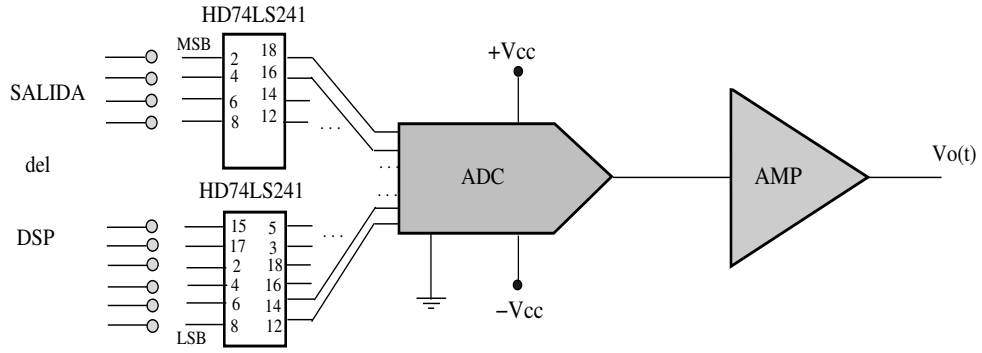
Para poder introducir señales analógicas del tipo acústicas a los convertidores analógicos digitales del DSP, es necesario que estén en el intervalo de 0 a 3 volts, por tanto, se les debe agregar un voltaje de corriente directa de 1.5 volts, además, si las señales son sensadas a través de micrófonos que nos entregan señales muy pequeñas del orden de microvolts, entonces es necesario amplificarlas. En el circuito propuesto de la figura 8.9.a se muestra un diseño para el acondicionamiento y amplificación de señales en la entrada, y en 8.9.b, la salida a través de un circuito DAC [13], [38]. El circuito de entrada se puede aplicar a cualquiera de los 16 canales ADC del DSP. En la entrada del micrófono se tiene implementado un filtro paso banda con f_L=20 Hz y f_H=20 KHz para eliminar el alias de señales acústicas.

Tabla 8.7. Registros del módulo convertidor ADC familia Piccolo, dirección base 00 7000h

Registro	**Dirección (h)**	**Descripción**
ADCCTL1	00	Control 1(*)
ADCCTL2	01	Control 2(*)
ADCINTFLG	04	Banderas de interrupción ADCINT9..1
ADCINTFLGCLR	05	Limpia banderas de interrupción
ADCINTOVF	06	Sobreflujo de interrupción
ADCINTOVFCLR	07	Limpia banderas de sobreflujo de interrupción
INTSEL1N2	08	Habilita y selecciona interrupción 1 y 2(*)
INTSEL3N4	09	Selecciona interrupción 3 y 4(*)
INTSEL5N6	0A	Selecciona interrupción 5 y 6(*)
INTSEL7N8	0B	Selecciona interrupción 7 y 8(*)
INTSEL9N10	0C	Selecciona interrupción 9(*)
SOCPRICTL	10	Control de prioridad de SOC(*)
ADCSAMPLEMODE	12	Modo de muestreo(*)
ADCINTSOCSEL1	14	Selección de interrupción SOC 1(*) para 8 canales: 7..0
ADCINTSOCSEL2	15	Selección de interrupción SOC 2(*) para 8 canales: 15..8
ADCSOCFLG1	18	Banderas de SOC 1 para 16 canales
ADCSOCFRC1	1A	Fuerza el inicio de SOC para 16 canales
ADCSOCOVF1	1C	Sobreflujo de SOC 1 para 16 canales
ADCSOCOVFCLR1	1E	Limpia sobreflujo de SOC 1 para 16 canales
ADCSOC0CTL	20	Control de SOC0 a SOC15(*)
...	...	
ADCSOC15CTL	2F	
ADCREFTRIM	40	Referencia de recorte(*)
ADCOFFTRIM	41	Offset de recorte(*)
COMPHYSTCTL	4C	Control de comparación de histéresis(*)
ADCREV	4F	De revisión (reservado)
ADCRESULT0	00	Resultado 0
...	...	...
ADCRESULT15	0F	Resultado 15



a) Circuito acondicionador y amplificador



b) Salida del DSP y convertidor DAC

Figura 8.9. Circuitos para conectar el ADC y el DAC al DSP

Ejemplo de adquisición y salida de un sólo canal

En el siguiente programa se da una ejemplo para que el lector interesado construya el circuito 8.9 y pruebe la adquisición y salida de un canal simple.

```
;  Adquisición del canal 1 del ADC del Piccolo F28027
;  Para el SOC utilizar interrupción de TIMER0

                .global     _c_int00 ;
;  DIRECCIONES DE REGISTROS
DIR_WDCR        .set 07029h  ; Dir. de registro de control de WatchDog
DIR_SP          .set 00600h  ; parte alta de stack
DIR_PIECTRL     .set 0x00CE0 ; Registro de control de PIE
DIR_PIEACK      .set 0x00CE1 ; Registro de reconocimiento de PIE
DIR_PIEIER1     .set 0x00CE2 ; Registro habilitador de interrupciones de PIE
                             ; grupo  INT1
DIR_PIEIFR1     .set 0x00CE3 ; Registro de banderas de interrupción de PIE
                             ; grupo  INT1
DIR_PIEIER10    .set 0x0CF4  ; Registro habilitador de interrupciones de PIE
                             ; grupo  INT10   (INTADC1-8)
DIR_PIEIFR10    .set 0x0CF5  ; Registro de banderas de interrupción de PIE
                             ; grupo  INT10

DIR_GPAMUX1     .set 0x06F86 ; GPIO A MUX 1  (GPIO0-15)
DIR_GPAMUX2     .set 0x06F88 ; GPIO A MUX 2  (GPIO16-31)
DIR_GPADIR      .set 0x06F8A ; GPIO A Dirección (GPIO0-31)
DIR_GPADAT      .set 0x06FC0 ; GPIO B Dato   (GPIO0-31)
DIR_GPATOG      .set 0x06FC6 ; GPIO A TOGGLE (GPIO0-31)

DIR_GPBMUX1     .set 0x06F96 ; GPIO B MUX 1  (GPIO32-GPIO34)
DIR_GPBDIR      .set 0x06F9A ; GPIO B Dirección  (GPIO32-GPIO34)
DIR_GPBDAT      .set 0x06FC8 ; GPIO B Dato   (GPIO32-GPIO34)
DIR_GPBSET      .set 0x06FCA ; GPIO B Set    (GPIO32-GPIO34)
DIR_GPBCLEAR    .set 0x06FCC ; GPIO B Clear  (GPIO32-GPIO34)

DIR_PCLKCR3     .set 0x07020
DIR_ITIM0       .set 0x0D4C
DIR_TIMPRDL     .set 0x0C02
DIR_TIMPRDH     .set 0x0C03
DIR_TIM0TCR     .set 0x0C04

C_WDCR          .set   0068h ; Máscara de WD
```

```
PER_H           .set   0000h ; Para PRD H
PER_L           .set   625   ; Para PRD L

DIR_PCLKCR0     .set  0701Ch ; Dir. registro de control de CLK
                             ; de periféricos.
DIR_ADCCTL1     .set  07100h ;

DIR_ADCSOCFOR   .set  0711Ah ;
DIR_ADCSOC0CTL  .set  07120h ;
DIR_ADCMODE     .set  07112h ; Modo secuencial=0, simultáneo 1
DIR_INTSEL1N2   .set  07108h ; Selección de INT ADC1 y ADC2
DIR_ADCRESULT0  .set  00B00h ;
DIR_V_ADCINT10  .set  00DD0h ; Vector de interrupción 10.1

                .text
_c_int00
        SETC  INTM            ; Deshabilita toda interrupción mascarable
        MOV   SP,#DIR_SP      ; Localiza el stack
        AND   IFR,#0FFFFh     ; Limpia cualquier interrupción pendiente
     EALLOW                   ; Deshabilita escritura a registros protegidos
        MOVL  XAR1, #DIR_WDCR
        MOV   *AR1, #C_WDCR   ; Desactiva el WatchDog
********************************* PIE
; Habilitar PIE
        MOVL  XAR0,#DIR_PIECTRL
        MOV   AL,*AR0
        OR    AL,#0x0001
        MOV   *AR0,AL
; Activar las interrupciones de INT10.1 con el bit 1 (Grupo uno int 10 de PIE)
        MOVL  XAR0,#DIR_PIEIER10
        MOV   AL,*XAR0
        OR    AL,#0x0001
        MOV   *XAR0,AL
; Ligar la subrutina con el vector de interrupción PIE
        MOVL  XAR2,#DIR_V_ADCINT10
        MOV   ACC, #_MI_ADCINT
*       MOV   AH,#03Fh
        MOVL  *XAR2,ACC
*********************************  GPIO
; Configura el GPAMUX1 para poner el pines como GPIO
        MOVL  XAR0,#DIR_GPAMUX1
```

```
        MOV   ACC, #0x0000
        MOVL  *XAR0, ACC
; Salidas, LEDS en GPIO al GPIO7,  DIR = 1
; Switch en GPIO12    como entrada DIR = 0
        MOVL  XAR0,#DIR_GPADIR
        MOV   AL, #0x00FF
        MOV   *XAR0,AL
*********************************  TIMER 0
; Carga el período de TIMER0
        MOVL  XAR1,#DIR_TIMPRDH
        MOV   AL, #PER_H        ; Carga el ACC con el valor de período
        MOV   *XAR1,AL          ;
        MOVL  XAR1,#DIR_TIMPRDL ;
        MOV   AL,#PER_L         ; Carga el ACC con el valor de período PRD
        MOV   *XAR1, AL         ; Pasa el valor el PRD bajo
; Activar TIE en TIMER0TCR 0x0C04, habilita interrup. de TIMER0
        MOVL  XAR0,#DIR_TIM0TCR
        MOV   AL,*XAR0
        OR    AL,#0x4000
        MOV   *XAR0, AL
*********************************  ADC
; Habilitar el reloj del ADC, vía el registro PCLKCR0 bit 3
        MOVL  XAR1,#DIR_PCLKCR0
        MOV   AL,*XAR1
        OR    AL, #0008h
        MOV   *XAR1,AL
        NOP                   ; Retardos
        NOP
        NOP
; Registro ADCCTL1 bit14=adcenable bit7=powerdown (activo bajo)
        MOVL  XAR1,#DIR_ADCCTL1
        MOV   AL,*XAR1
        ;OR    AL,#0C0E0h
        OR    AL,#40E0h
        MOV   *XAR1,AL
; Configuración del ADC Piccolo
        MOVL  XAR0,#DIR_ADCMODE
        MOV   AL,#00            ; Modo secuencial
        MOV   *XAR0, AL
; Escribir al registro  ADCSOC0CTL, los campos  ACQPS, CHSEL y TRIGSEL
; - El campo ACQPS (5..0) de 6b,  ventana de muestreo y retención (S/H)
```

```
; - CHSEL (9..6): Selecciona el canal a convertir.
; - TRIGSEL (15..11): Selecciona la fuente de disparo SOC para el canal
        MOVL  XAR1,#DIR_ADCSOC0CTL
        MOV   AL,#0806h          ; TRIGSEL=TIMER0, CHSEL=Ch1, ACQPS=6+1
        MOV   *XAR1,AL
; Asocia EOC1 a ADCINT1
        MOVL  XAR1,#DIR_INTSEL1N2
        MOV   AL,#0060h          ; Habilita canal 1 (EOC0) para ADCINT1
        MOV   *XAR1,AL
     EDIS
****************************************
; Habilita INT1.X grupo x=10 de PIE  en IER
        OR    IER,#0x0200        ; HABILITA INT10
; Habilita INT globales
        CLRC  INTM               ; listo para recibir interrupciones
*********************  CICLO INFINITO
FIN_R   NOP
        B     FIN_R,UNC

*******************  SUBRUTINA DE INTERRUPCION
_MI_ADCINT
        MOVL  XAR0,#DIR_ADCRESULT0 ; Lee dato convertido
        MOV   AL,*XAR0
        SFR   ACC,#4  ; Corre a la derecha 4 bits el ACC
;       AND   AL,#00FFh          ; Salida de sólo 8 bits
        MOVL  XAR0,#DIR_GPADAT
        MOV   *AR0,AL            ; Escribe dato al puerto GPIOA

;        MOVL  XAR0,#DIR_ADCSOCFOR
;        MOV   AL,#0001h
;        MOV   *AR0,AL
; Reconocer la interrupción para permitir futuras interrupciones
; de INT10.1 PIEACK 0x0000-0CE1
      EALLOW
         MOVL XAR0,#DIR_PIEACK   ;  0x0000-0CE10x0CE1
         MOV  AL, #0x0200
         MOV  *XAR0, AL
     EDIS
        IRET                     ; Retorno de interrupción.
 .end
```

8.7. Aplicaciones propuestas

En esta sección se proponen proyectos que deben operar en tiempo real con el fin de probar las aplicaciones que funcionen con algunos de los periféricos descritos. La situación adicional es que se debe agregar el hardware externo necesario, sin embargo, esta complejidad lleva al usuario al inicio de aplicaciones más avanzadas.

1. Diseñar y realizar un sistema de adquisición de audio en tiempo real que utilice al menos cuatros canales del convertidor ADC del DSP y que se pueda seleccionar cualquiera de ellos como salida a través de una bocina.

2. Diseñar y realizar un sistema que genere señales de salida de doble tono (DTMF), esta señal se seleccionará a través de un teclado que emule un teclado telefónico.

3. Diseñar y realizar filtros paso banda y supresor de banda centrados en 200 Hz, con la mejor calidad posible, utilizando:
   - Filtros FIR.
   - Filtros IIR.
   Probar los filtros con cualquier estructura y programarlos a 16 y 32 bits.

8.8. Módulos manejadores de eventos

El módulo manejador de eventos (EV) provee un amplio intervalo de funciones y características, que son útiles en aplicaciones de movimiento y control de motores. EV incluye temporizadores de propósito general, unidad de modulación por ancho de pulso (PWM), unidades de captura, codificación de pulso en cuadratura (QEP). El módulo EV está compuesto de dos módulos A (EVA) y B (EVB) con periféricos idénticos, cada uno es capaz de controlar tres puentes híbridos H. Cada módulo EV también contiene dos PWMs [23], [27], [36].

Los manejadores de eventos EVA y EVB son similares en cuanto a funcionalidad, el mapeo de bits y registros, lo que cambia son sus direcciones y nombres, es decir, que es suficiente entender sólo uno de ellos para el manejo de ambos. Para su funcionamiento, estos módulos utilizan otros periféricos como los temporizadores de propósito general, la unidad completa de comparación PWM, unidad de captura y el circuito codificador QEP. Debido a las similitudes, en la tabla 8.8 únicamente se enumeran los registros para EVA que empiezan en la dirección 00 7400h y de forma similar para EVB empiezan en 00 7500h.

Tabla 8.8. Registros de control y configuración de manejador de eventos EVA del C281x

Nombre	Dirección (h)	Descripción
		Registros de temporizador
GPTCONA	00 7400	Control de temporizador GP para EVA
T1CNT	00 7401	Contador de temporizador GP 1
T1CMPR	00 7402	Comparador de temporizador GP 1
T1PR	00 7403	Período de temporizador GP 1
T1CON	00 7404	Control de temporizador GP 1
T2CNT	00 7405	Contador de temporizador GP 2
T2CMPR	00 7406	Comparador de temporizador GP 2
T2PR	00 7407	Período de temporizador GP 2
T2CON	00 7408	Control de temporizador GP 2
EXTCONA	00 7409	De extensión de control GP A
		Registros de comparación
COMCONA	00 7411	Control de comparación A
ACTRA	00 7413	Control de acción de comparación A
DBTCONA	00 7415	Control de temporizador "dead-band" A
CMPR1	00 7417	Comparador 1
CMPR2	00 7418	Comparador 2
CMPR3	00 7419	Comparador 3
		Registros de captura
CAPCONA	00 7420	Control del captura A
CAPFIFOA	00 7422	Estado del capturador FIFO A
CAP1FIFO	00 7423	Dos niveles de profundidad del FIFO en stack 1
CAP2FIFO	00 7424	Dos niveles de profundidad del FIFO en stack 2
CAP3FIFO	00 7425	Dos niveles de profundidad del FIFO en stack 3
CAP1FBOT	00 7427	Parte baja del stack de FIFO 1
CAP2FBOT	00 7428	Parte baja del stack de FIFO 2
CAP3FBOT	00 7429	Parte baja del stack de FIFO 3
		Registros de interrupción
EVAIMRA	00 742C	Máscara de interrupción A
EVAIMRB	00 742D	Máscara de interrupción B
EVAIMRC	00 742E	Máscara de interrupción C
EVAIFRA	00 742F	Banderas de interrupción A
EVAIFRB	00 7430	Banderas de interrupción B
EVAIFRC	00 7431	Banderas de interrupción C

8.8.1. Temporizadores de propósito general para el módulo EV

El C28x tiene al menos dos temporizadores de propósito general (GP) en cada módulo EV, éstos pueden ser utilizados como bases de tiempo independientes en aplicaciones tales como [23], [27], [28], [29]:

- Generadores de período de muestreo en sistemas de control.
- Proveer una base de tiempo para la operación del circuito codificador de pulso en cuadratura (QEP).
- Proveer una base de tiempo para la operación de las unidades de comparación y los circuitos generadores de PWM.

Contienen dos temporizadores de propósito general (GP), el temporizador GP x (x = 1 o 2 para EVA, x = 3 o 4 para EVB) que incluye:

- Un temporizador de 16 bits, contador up/down (U/D), TxCNT para lectura y escritura. Este registro mantiene el conteo actual y es incrementado o decrementado dependiendo de la dirección de conteo.
- Un registro comparador de 16 bits, TxCMPR (doble buffereado con registros sombra) para lectura y escritura. Este registro mantiene el valor que se está comparando con el valor del temporizador.
- Un registro período, TxPR (doble buffereado con registros sombra) para lectura y escritura. En este registro se escribe la razón en que el temporizador se resetea o cambia de dirección de conteo.
- Un registro de control, TxCON para lectura y escritura. Controla el modo de operación del temporizador x. Se puede habilitar o deshabilitar y establecer su modo de operación.
- El registro de control GPTCONA/B especifica la acción que debe realizar el temporizador en los diferentes eventos. Con este registro se conoce la acción que está siguiendo el temporizador en la dirección de conteo.
- Selección de reloj, interno o externo.
- Un preescalador programable para entrada de reloj externa o interna.
- Una lógica de control de interrupciones para cuatro interrupciones mascarables: sobreflujo, "underflow", comparación de temporizador y período.
- Pin de dirección de entrada seleccionable TDIRx para conteo U/D cuando se selecciona el modo U/D.

- Pines de salida de comparación de temporizador TxCMP (x=1,2,3,4).
- Los registros de período y de comparación son buffereados dobles en registros sombra para permitir al usuario actualizarlos.
- Los temporizadores GP pueden operar independientemente o sincronizarse a algún evento, el registro de comparación asociado con cada temporizador GP puede ser usado para comparar la función y forma de onda PWM generada.

Cada módulo o subsistema EV tiene sus propias banderas de interrupción, y su funcionamiento es muy diferente al sistema principal de interrupciones. Cada EV tiene su mecanismo de interrupción que incluye sus registros de máscara de interrupciones y de banderas. Después de que se presenta una interrupción de EV, ésta fluye a través del manejador de interrupciones PIE. Los temporizadores contienen 16 banderas de interrupción en los registros EVBIFRA y EVBIFRB. Estas interrupciones son:

- "Underflow", TxUFINT, el contador alcanza 0000h.
- Sobreflujo, TxOFINT, el contador alcanza FFFFh.
- Comparación, TxCINT, el contador alcanza el valor del registro de comparación.
- Período, TxPINT, el contador alcanza el valor del registro período.

Para x=1,2,3,4.

Operación de los temporizadores GP de EV

El registro de período TxPR mantiene el período de conteo especificado. TxPR es cargado automáticamente del registro buffer de período cuando ocurre un cero en el conteo, definido como TxCNT = 0, por tanto, el período de tiempo puede cambiarse dinámicamente.

La fuente de reloj para los temporizadores GP pueden ser: del reloj interno del CPU, un reloj externo en el pin TCLKINA/B o una salida de QEP (la frecuencia máxima de QEP es CLKIN/4). Una vez seleccionado el reloj se puede escalar a través de un contador preescalador.

Cada temporizador de eventos tiene cuatro modos de operación:

- Paro y mantenimiento: el temporizador se detiene y conserva su estado actual. El contador de tiempo, el comparador de salida y el contador preescalador se mantienen sin cambios.
- Continuo hacia arriba.
- Continuo hacia arriba/abajo.
- Continuo direccional hacia arriba/abajo.

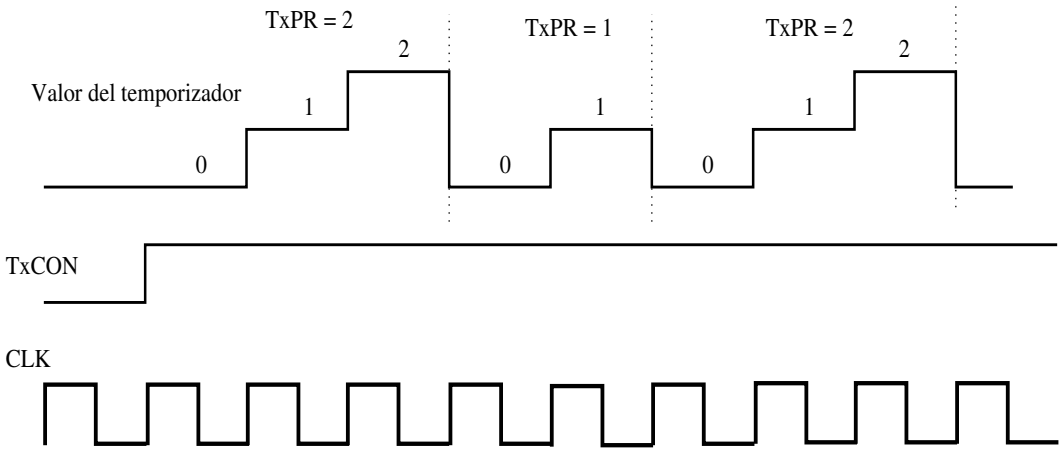


Figura 8.10. Temporizador en modo conteo hacia arriba

Operación continua hacia arriba

El temporizador incrementa su cuenta hasta alcanzar el valor del registro de período TxPR. La bandera de interrupción del temporizador se pone en uno y emite una interrupción. Luego el temporizador se resetea a cero e inicia la cuenta de nuevo hasta alcanzar el valor en el registro TxPR, como se observa en la figura 8.10. Este modo es útil en la generación de señales PWM asimétricas que se aplican ampliamente en control de motores y potencia eléctrica.

Operación continua arriba/abajo

El temporizador cambia el sentido de conteo arriba/abajo cuando alcanza el valor del período TxPR o FFFFh. De otra forma cambia de dirección de abajo hacia arriba cuando el temporizador alcanza el cero, como se observa en la figura 8.11. Este modo no es afectado por el pin TDIRA/B y es útil para la generación de señales PWM simétricas.

Operación en modo continuo direccional arriba/abajo

El temporizador cuenta hacia arriba o abajo dependiendo de la entrada de escalamiento de reloj TDIRA/B. La operación en este modo se muestra en la figura 8.12.

- Si TDIRA/B está en cero, el temporizador cuenta hacia abajo hasta alcanzar el valor de cero, entonces el temporizador es cargado de nuevo con el valor del registro período y empieza el conteo hacia abajo de nuevo.
- Si TDIRA/B está en uno, el temporizador cuenta hacia arriba hasta que alcanza el valor del registro período, se resetea a cero e inicia el conteo hacia arriba.

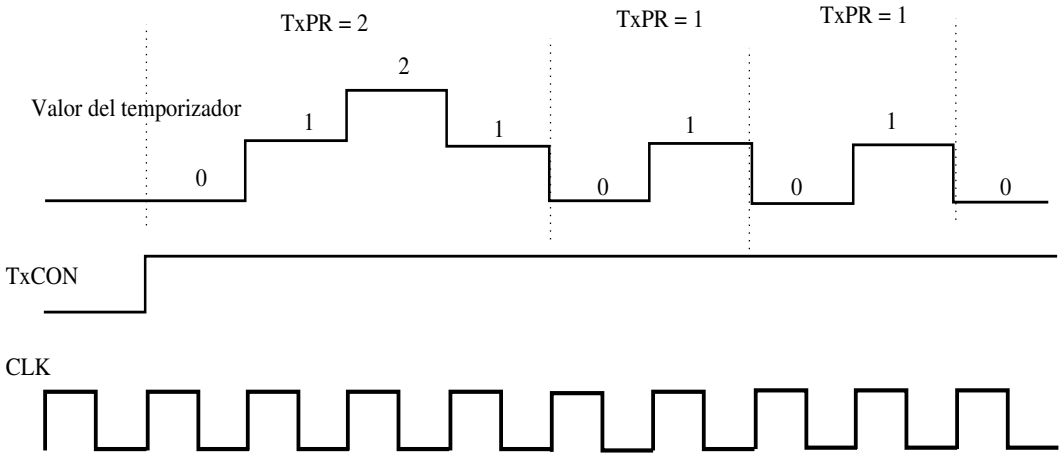


Figura 8.11. Temporizador en modo conteo arriba/abajo

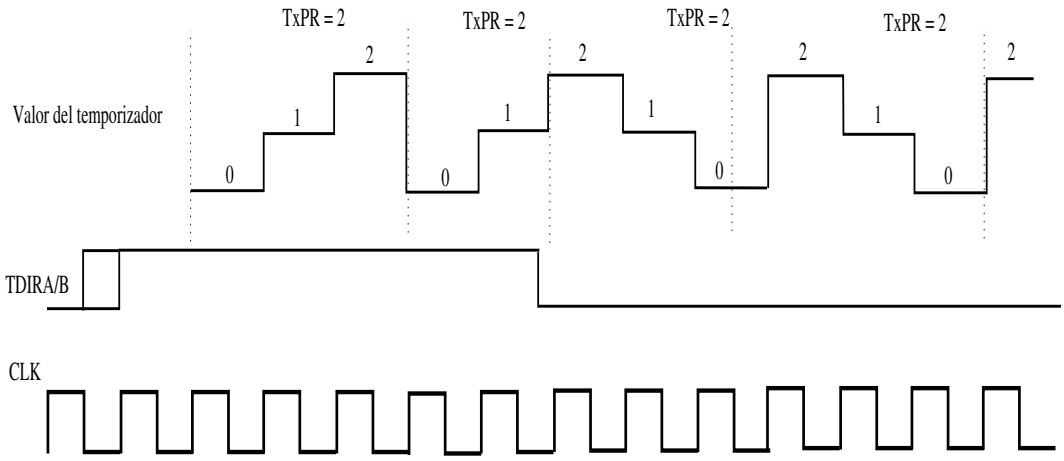


Figura 8.12. Temporizador en modo conteo direccional arriba/abajo

Interrupciones de los temporizadores GP de EV

Los temporizadores de eventos tienen sus propias interrupciones y son enmascaradas individualmente.

- TxPINT: interrupción de período, su bandera es puesta a uno cuando el contador del temporizador alcanza el valor del registro período.
- TxUFINT: interrupción de cero, la bandera se pone a uno cuando el contador llega a cero.
- TxOFINT: interrupción de sobreflujo, su bandera se fija cuando el contador alcanza FFFFh.
- TxCINT: interrupción de comparación, su bandera se fija cuando el contador es igual al registro de comparación.

Unidad de comparación

Existen tres unidades completas de comparación en cada manejador de eventos, estas unidades de comparación utilizan el temporizador TIMER1 de GP como base de tiempo, un generador de seis salidas por comparador y un circuito generador de PWM programable, el estado de cada una de las seis salidas es configurada independientemente. Los registros de comparación de la unidad de comparación son doblemente buffereados y permiten cambios programables de los pulsos PWM [23], [27].

8.9. Señales moduladas por ancho de pulso

La modulación por ancho de pulso (PWM) es una forma de representar una señal analógica con una aproximación digital. Las señales moduladas PWM son señales cuadradas periódicas que se les puede variar el ciclo de trabajo de acuerdo con las necesidades de aplicación, por tanto, se puede decir que PWM también es una forma de representar señales analógicas por medio de pulsos digitales, donde el ancho de pulso es proporcional a la amplitud instantánea de la señal, y la energía de la señal PWM generada es aproximadamente igual a la energía de la señal original. El período de PWM siempre es fijo, su inverso es llamado frecuencia de PWM y su amplitud es constante. El ancho del pulso PWM se establece de acuerdo con la amplitud de otra secuencia de valores deseados de la señal moduladora, es decir, la amplitud de la señal analógica determina el ancho del pulso, el cual puede variar desde 0 % al 100 %, con un período de tiempo constante cuya resolución depende del generador o dispositivo. Este proceso se puede observar en la figura 8.13.

La familia básica C28x puede generar hasta ocho señales de salida PWM simultáneamente por cada manejador de eventos:
• Tres pares independientes (seis PWM) por cada unidad de comparación.

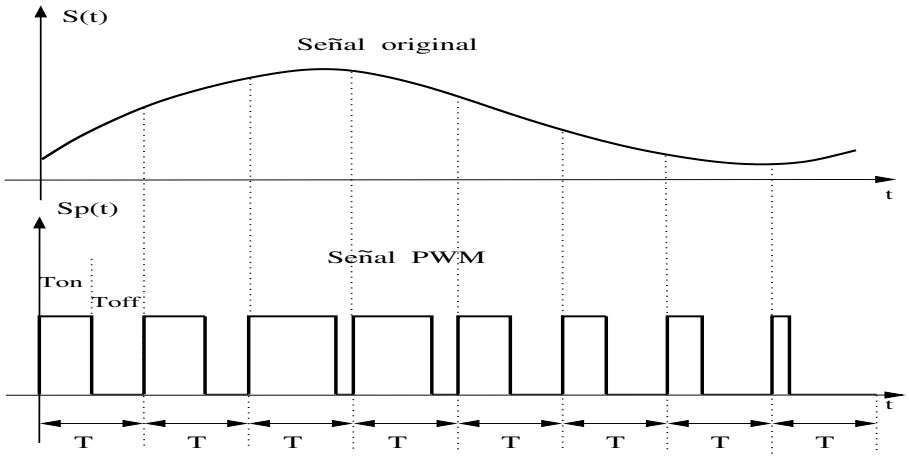


Figura 8.13. Generación de una señal PWM

• Dos PWM independientes por comparación del temporizador GP.

Sin embargo, la familia F28335 trae 18 generadores de PWM con seis de alta resolución (HRPWM), la familia Piccolo F28069 trae 16 generadores PWM con ocho HRPWM.

Características de PWM

Características del módulo PWM
• Registros de 16 bits.
• Intervalos amplios de programación de bandas muertas para un par de salidas PWM.
• Cambio dinámico de la frecuencia portadora de PWM, como sea necesario.
• Cambio dinámico del ancho de pulso dentro y después de cada período PWM, como sea necesario.
• Mascarable externamente y protección de interrupciones.
• Circuito generador de patrón de pulsos, para señales PWM programables simétricas y asimétricas.

8.9.1. Generador de señales PWM

En sistemas de control de motores, las señales PWM son utilizadas para el control de los tiempos de "switcheo" ON/OFF para los dispositivos de potencia que proveen la corriente a los embobinados del motor.

En la generación de una señal PWM con temporizadores GP, es necesario:
• Un temporizador que repita el conteo de período de PWM.

• Un registro de comparación utilizado para mantener el valor modulado.
• El valor del registro de comparación es comparado constantemente con el valor del temporizador. Cuando ambos valores son iguales, en la salida asociada se da una transición (bajo a alto, alto a bajo).
• De esta forma se genera un pulso de duración ON u OFF proporcional al registro de comparación.
• Este proceso se repite por cada período para diferentes valores del registro de comparación.

Configuración de la operación de PWM:
• Cargar en el registro TxPR el período deseado de PWM. El valor de este registro sirve como un divisor de la fuente de reloj del temporizador Tx, x = 1,2,3,4.
• Cargar el registro TxCON con el modo específico de conteo y la fuente de reloj e iniciar la operación.
• Cargar el registro TxCMPR con el valor correspondiente con el ciclo de servicio del PWM.

Los circuitos generadores de PWM también se pueden usar para controlar motores de DC y de pasos.

8.9.2. Generador simétrico y asimétrico de PWM

Con los módulos EV se pueden generar señales PWM simétricas y asimétricas. Existen tres unidades de comparación que se pueden utilizar en conjunto para generar tres fases simétricas en el espacio de vectores de salida de PWM. Una señal PWM se puede generar con una señal moduladora tipo diente de sierra, por lo que la señal PWM será asimétrica, esta señal se puede lograr considerando temporizadores que cuentan en modo ascendente hasta un valor determinado. Por otro lado, para generar una señal PWM simétrica se puede utilizar una señal de comparación tipo triangular para generar con un temporizador con conteo hacia arriba y abajo.

Configuración de registros para generar PWM

Las tres clases de generadores de PWM con las unidades de comparación asociadas, requieren configurar los registros del manejador de eventos. Esto incluye los siguientes pasos:
• Configurar los registros ACTx.
• Configurar los registros DBTCONx, si se utiliza la banda muerta [1].
• Inicializar los registros COMNx.
• Configurar los registros COMCONx.
• Configurar los registros T1CON de EVA o T3CON de EVB para empezar la operación.
• Reescribir CMPRx con los nuevos valores determinados.

[1] Este concepto se explicará adelante en este mismo capítulo

Generación de señal PWM asimétrica

Una señal de PWM asimétrica es generada cuando se utiliza un temporizador en modo subida, como se observa en la figura 8.14.a. La señal de salida cambia de acuerdo con la secuencia:
• Está en cero antes de que empiece la operación de conteo.
• Permanece sin cambios hasta que el temporizador alcance el valor de comparación.
• Cambia de valor (ON a OFF) cuando el temporizador alcanza el valor de comparación.
• Permanece sin cambios hasta el fin del período de PWM.
• Se resetea a cero al fin del período si el nuevo valor de comparación no es cero.
• La salida es uno en todo el período, si el valor de comparación es cero al inicio del período.

Permite variaciones del ciclo de servicio de 0 % al 100 %. En la señal PWM asimétrica, los cambios de valor en el registro de comparación sólo afectan un lado del pulso de PWM.

Generación de señal PWM simétrica

Una señal de PWM *centrada o simétrica* es caracterizada por la modulación por pulsos centrada respecto de cada período de PWM, como se observa en la figura 8.14.b. La ventaja de las señales PWM simétricas respecto de las asimétricas es que tiene dos zonas inactivas de igual duración, al inicio y al final de cada período PWM. Se ha comprobado que las señales PWM simétricas causan menos armónicos que las asimétricas en la corriente de un motor de DC.

Operación:
• Un temporizador GP es configurado en modo continuo hacia arriba, este registro es cargado de antemano con el valor deseado del período de portadora de PWM.
• El registro COMNONx es configurado para habilitar la operación de comparación, fija el pin de salida de PWM y habilita la salida.
• Si la banda muerta es utilizada, el valor de banda muerta debe ser escrito en el registro DBTCONx.

Una señal PWM que puede estar asociada a la salida con una unidad de comparación, mientras que la otra es mantenida en OFF u ON, al inicio, en medio o fin del período PWM. También se puede generar con temporizadores que operan en modo continuo arriba/abajo, usando este modo, la salida de PWM es determinada de la siguiente forma:
• Está en cero antes de que empiece la operación de conteo.
• Permanece sin cambios hasta que el temporizador alcanza el valor de comparación por primera vez y cambia a ON.
• En el descenso (conteo hacia abajo del temporizador), vuelve a cambiar a estado OFF cuando el temporizador llega de nuevo al valor de comparación.
• Permanece sin cambios hasta el fin de período.
• Se resetea a cero en el fin de período si no se produce una segunda comparación y el nuevo

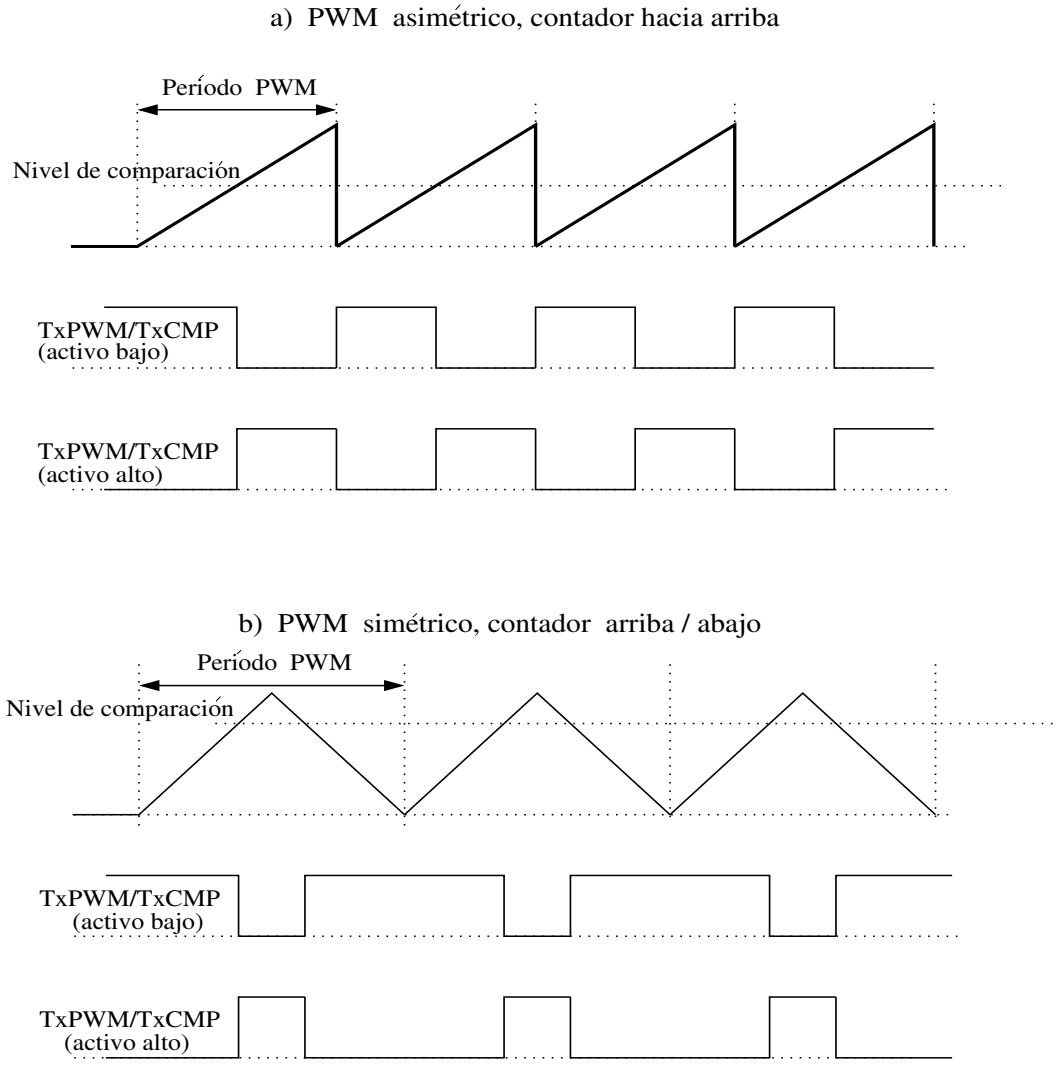


Figura 8.14. Generación de señales de PWM asimétrica y simétrica

valor de comparación para el siguiente período no es cero.

Existen dos comparaciones en la generación de un período de PWM simétrico, una durante la subida y otro en la bajada. Una nueva comparación se realiza después que el período es alcanzado, porque éste hace posible avanzar o retardar el segundo flanco del pulso PWM.

Los valores para el registro período TxPR en la generación de PWM se calculan:
Para PWM simétrico:

$$TxPR = \frac{\text{período PWM}}{\text{período de reloj}} \tag{8.6}$$

Para PWM asimétrico:

$$TxPR = \frac{\text{período PWM}}{\text{período de reloj}} - 1 \tag{8.7}$$

8.9.3. Unidad de captura

Los capturadores se utilizan para sincronizar eventos que inicializan un convertidor ADC, miden la duración de un ancho de pulso, estiman velocidades de dispositivos externos, mediciones de intervalos de cruces por cero de una señal y mediciones de períodos de una señal, etc. [19].

La unidad de captura consta de seis capturadores, CAP1, CAP2 y CAP3 para el módulo EVA y CAP4, CAP5 y CAP6 para el módulo EVB. La unidad de captura provee la función de detección de diferentes eventos o transiciones, cuando se detecta una transición seleccionada sobre un pin de entrada de captura CAPx ($x = 1, 2, 3$ para EVA y $x = 4, 5, 6$ para EVB), el valor de un contador de temporizador GP es capturado y almacenado en dos niveles de profundidad de pila, primero en entrar primero en salir (FIFO) [28]. Para que la transición de un evento sea capturado, la entrada debe mantenerse en el nivel actual al menos dos ciclos de reloj del CPU. El temporizador GP correspondiente es capturado en un registro FIFO y se emite una interrupción, por software puede leerse el registro FIFO y utilizarse en algún algoritmo.

La unidad de captura consta de tres circuitos que incluyen las siguientes características:
• Registros de control de captura de 16 bits CAPCONx (R/W).
• Registros de estado de captura de 16 bits CAPFIFOx.
• Selección de temporizador GP 1/2 (para EVA) o 3/4 (para EVB) como base de tiempo.
• Tres localidades de 16 bits en una pila FIFO de dos niveles de profundidad, una para cada unidad de captura.
• Seis pines de captura de entradas (CAP1/2/3 para EVA, CAP4/5/6 para EVB). Los pines CAP1/2 y CAP4/5 también pueden utilizarse como entradas al circuito QEP.
• Transiciones de uso específico: en flanco de subida, bajada o ambos.
• Tres banderas para interrupciones mascarables, una por cada unidad de captura.
• Los pines de captura también pueden utilizarse como pines de interrupciones de propósito general.

Una señal de PWM también puede ser generada utilizando la unidad de comparación CMPR. La unidad de comparación CMPRx realiza las mismas funciones que los temporizadores de GP, aunque la señal de salida PWM asociada con los comparadores permite la generación de hasta seis señales PWM por evento.

Cada módulo EV (A/B) incluye:
• Seis registros comparadores de 16 bits: CMPR1, CMPR2 y CMPR3 para EVA y CMPR4, CMPR5 y CMPR6 para EVB.
• Un registro comparador de control de 16 bits: COMCONA para EVA y COMCONB para EVB.
• Un registro de control acción de 16 bits: ACTRA para EVA y ACTRB para EVB.
• 12 pines de salida PWM: 1,2,3,4,5 y 6 para EVA y 7,8,9,10,11 y 12 para EVB.
• Lógica de control de interrupciones.

Los capturadores EVA 3 y EVB 6 pueden configurarse para disparar el convertidor ADC que está sincronizado con algún evento.

Configuración de los módulos EV

• Inicializar el registro CAPFIFOx y limpiar el bit de estado apropiado.
• Definir el modo de operación del temporizador GP a utilizar.
• Definir el registro comparador de temporizador GP asociado o registro de período de temporizador GP si se necesita.
• Definir los registros CAPCONA o CAPCONB.

Modos de operación

Los modos de operación de la unidad de comparación se determinan a través los bits COMCONx:
• Habilitación de la operación de comparación.
• Habilitación de la comparación de salida.
• Condiciones en las que el registro de comparación es actualizado con el valor del registro sombra.
• Los vectores de espacio de PWM son habilitados.

Operación

Utilizando el módulo EVA y el temporizador GP1 que es continuamente comparado con el registro de comparación, cuando ambos son iguales, aparece una transición en las dos salidas de la unidad de comparación de acuerdo con los bits en el registro de acción de control (ACTRA). Entonces se activa la bandera de la interrupción asociada con la unidad de comparación. La salida de la unidad de comparación en modo comparación está sujeta a

modificarse por la lógica de salida, las unidades de banda muerta y la lógica de los vectores de espacio de PWM.

8.9.4. Generador programable de banda muerta "dead band"

En algunas aplicaciones de potencia eléctrica como inversores, motores y puentes de potencia H, existen dos transistores de potencia conectados en serie, uno superior y otro inferior, los períodos de encendido de los dos dispositivos no se deben de solapar para evitar una falla eléctrica. Por tanto, se requieren dos salidas PWM complementarias no solapadas y que enciendan y apaguen correctamente los dos dispositivos. Sin embargo, en la práctica, los tiempos de encendido de un transistor son más rápidos que los de apagado, por tanto, es necesario introducir un retardo en el encendido y apagado de un transistor para evitar el solapamiento de operación con el otro transistor, es decir, acortar el período de ON de la señal de PWM de un transistor activo. Por lo tanto, si se insertan tiempos muertos (o banda muerta) entre el cambio a OFF de un transistor y el cambio a ON del otro transistor, este tiempo de retardo requerido es especificado por las características de los transistores y la carga de la aplicación. El generador de banda muerta puede ser habilitado o deshabilitado por cada unidad de comparación individualmente. Este efecto se puede observar en la figura 8.15, donde dos transistores TR1 y TR2 son disparados por señales complementarias PWM para alimentar a un motor.

Cuando se requiere utilizar banda muerta en una señal PWM, se habilitan los bits respectivos en los registros DBTCONA/B. El DSP C28x permite generar señales PWM al insertar tiempos muertos utilizando la unidad de comparación, ésta permite generar bandas muertas en las transiciones de la señal PWM, como se observa en la figura 8.15. En la primera y segunda transición de PWM se inserta un retardo, el cual se determina por el registro de control de banda muerta DBTCONx.

Dependiendo del dispositivo de "switcheo" se requiere más o menos banda muerta, el tiempo en ciclos de reloj del CPU para el retardo, o banda muerta, se puede calcular como:

$$\text{Banda muerta en ciclos del reloj de CPU} = \frac{bits(8 \rightarrow 11)\text{de DBTCONx}}{\text{reloj preescalador de eventos}} \tag{8.8}$$

El circuito generador de banda muerta produce dos salidas complementarias (con y sin zona muerta), para cada señal de salida de la unidad de comparación. El estado de salida del generador de banda muerta es reconfigurable por medio del registro doble buffereado ACTRx.

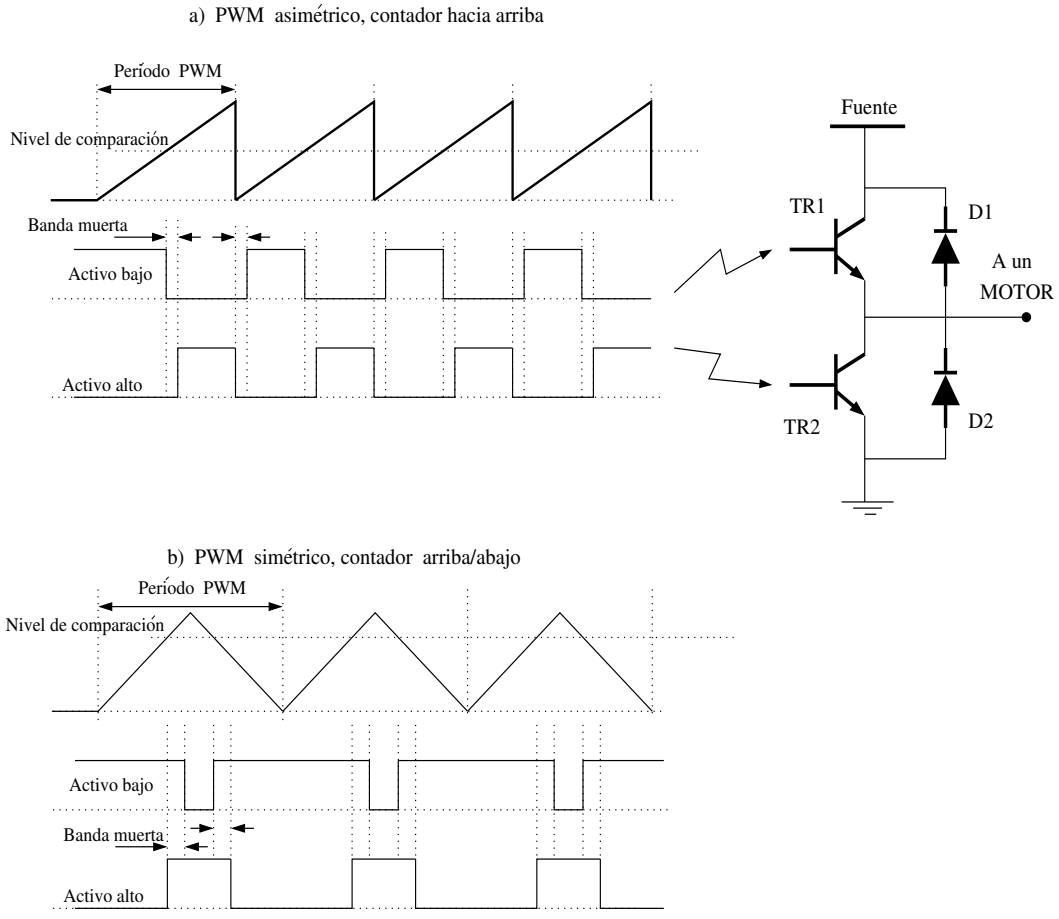


Figura 8.15. Generación de banda muerta en señales PWM

8.9.5. Módulo codificador en cuadratura de pulso QEP

El módulo codificador en cuadratura de pulso (QEP) es utilizado para acoplarse directamente con codificadores ópticos, incrementales lineales o rotatorios, y así obtener información de posición, dirección y velocidad de una máquina rotatoria para ser utilizadas en sistemas de control de movimiento y posición de alto desempeño. Algunas aplicaciones típicas se encuentran en robótica y mouses de esfera de una computadora, en este tipo de mouse la esfera gira en dos ejes, izquierdo derecho y arriba abajo, estos ejes son conectados a un codificador óptico que se comunica con la computadora para indicarle que tan rápido se mueve el mouse y en que sentido. El QEP detecta señales cuadradas desfasadas [2] 90 grados eléctricos, por lo que en dispositivos rotatorios pueden identificar arranque, paros e inversiones de giro [19], [29].

Un codificador óptico en cuadratura se construye con un disco conectado al eje de un motor, el disco está ranurado homogéneamente para producir espacios de luz y obscuridad, lo que genera una señal cuadrada. En la orilla del disco se colocan fotosensores espaciados en un cuarto del período angular, en estos sensores se reciben las señales cuadradas y desfasadas 90 grados (QEPA y QEPB), adicionalmente se le puede agregar al disco una ranura en otra posición para detectar un giro completo. Las señales asociadas a QEP se muestran en la figura 8.16

Cuando el circuito QEP es habilitado, los capturadores CAP1 y CAP2 para EVA, CAP4 y CAP5 de EVB, pueden utilizarse con el circuito codificador de pulso en cuadratura QEP. La sincronización de estas entradas se realiza en el chip. La base de tiempo para QEP está dada por los temporizadores GP 2 para EVA y GP 4 para EVB. Los temporizadores GP deben operar en modo de conteo arriba/abajo.

Decodificación QEP

Los pulsos codificados en cuadratura son dos secuencias de pulso con frecuencia variable y corrimientos de fase fija a un cuarto de período, es decir, 90 grados. Cuando estas señales son generadas por un codificador óptico en el eje de un motor, la dirección de rotación del motor puede ser determinada detectando cuál de las dos secuencias está adelantada. La posición angular y la velocidad pueden ser determinadas por el conteo de pulsos y la frecuencia.
De acuerdo al sentido de rotación de una rueda dentada conectada al eje de un motor, se obtienen las señales en cuadratura QEPA y QEP, con diferencia de tiempo de un cuarto de ciclo, de las que se puede establecer una codificación binaria con estados 00, 10, 11, 01, de acuerdo con las señales QEPA/B y que se pueden sincronizar al reloj del periférico para determinar el estado del motor, de acuerdo con la figura 8.16:

- Giro positivo, se presenta la secuencia ..., 10, 11, 01, 00, 10, 11, ... a tiempos de reloj constantes, el motor está girando a velocidad constante en sentido positivo.

[2] El término "fase" con "s" se utiliza en este trabajo para denotar retardos entre señales, por otro lado cuando se utiliza con "c" se refiere a acoplamientos entre dispositivos de hardware o elementos de software.

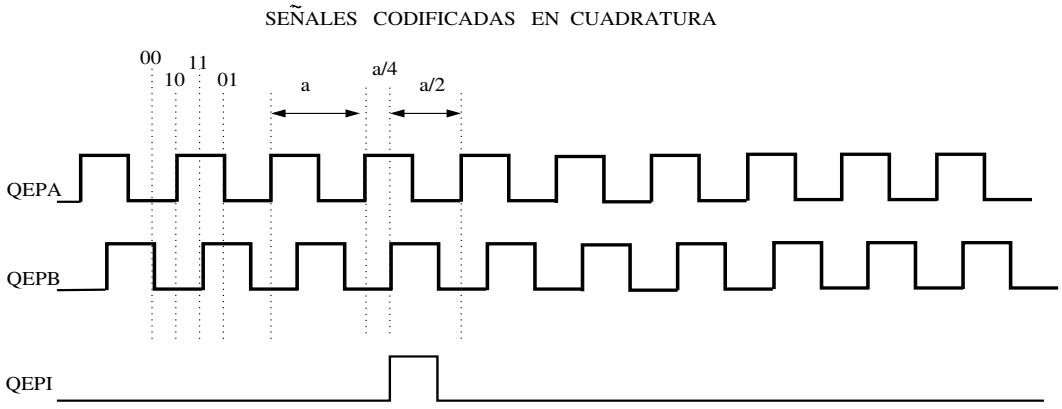


Figura 8.16. Señales QEP y decodificación

- Giro negativo, se presenta la secuencia ..., 10, 00, 01, 11, 10, 00, ... a tiempos de reloj constantes, el motor está girando a velocidad constante en sentido negativo.

- Aumento de velocidad en un sentido de giro: los estados binario conservan su secuencia cambiando más rápido.

- Disminución de velocidad en un sentido de giro: los estados binarios conservan su secuencia cambiando más lento.

- Paro: el estado actual permanece constante indefinidamente.

- Inversión de sentido de giro: la secuencia de estados cambia a un estado anterior.

La lógica de detección de QEP en el módulo EV determina cuál de las secuencias adelanta a la otra, entonces se genera una señal como dirección de entrada a los temporizadores GP2 o 4. El contador de temporizador cuenta hacia arriba si la entrada CAP1/QEP1 en EVA es la secuencia adelantada, o cuenta hacia abajo si CAP2/QEP2 es la secuencia de entrada adelantada. Esta lógica se determina por una máquina de estados cuyo diagrama está en la figura 8.16, también se puede determinar la dirección y velocidad de giro del motor.

Ambos flancos de las dos secuencias QEP de entrada son contados por el circuito QEP, por tanto, la frecuencia del generador de reloj por la lógica QEP de los temporizadores GP2 o 4 es cuatro veces mayor que las secuencias QEP de entrada. El reloj en cuadratura es conectado al reloj de los temporizadores GP2 o 4.

Los temporizadores GP 2 o 4 siempre inician el conteo de un valor actual. Un valor deseable puede ser cargado en los temporizadores GP, previamente a la habilitación del modo QEP. Cuando el circuito QEP es seleccionado como fuente de reloj, el temporizador ignora las entradas TDIRA/B y TCLKINA/B.

Para su configuración, se deben cargar los dos temporizadores contadores GP, el período y el registro de comparación con los valores deseados, configurar T2CON para fijar el modo hacia arriba/abajo del temporizador GP 2 con la fuente de reloj, habilitar y seleccionar el temporizador.

Entradas eQEP

- Contiene dos entradas para las señales (reloj) en cuadratura, una de índice y una de "strobe".

- Entradas QEPA y QEPB desfasadas en 90 grados eléctricos para determinar la dirección de rotación del disco ranurado. En modo dirección de conteo, las señales de dirección y reloj vienen de una fuente externa, estas fuentes tienen algunos codificadores en lugar de salida en cuadratura, entonces en la entrada QEPA se alimenta el reloj y en QEPB la dirección.

- Señal QEPI, es utilizada como señal índice para asignar una posición de inicio a partir de la cual la información del codificador se incrementa utilizando los pulsos en cuadratura.
- Entrada QEPS o de "strobe", con esta señal se puede inicializar o registrar la posición de un contador en la ocurrencia de un evento deseado.

Funciones del módulo QEP

- Entrada calificadora programable (GPIO MUX).
- Unidad decodificadora en cuadratura (QDU). Las entradas del reloj y dirección que posicionan el reloj son seleccionados utilizando los bits QSRC del registro QDECCTL, basándose en los requerimientos de la entrada:
  - Contador en modo cuadratura.
  - Contador en modo dirección.
  - Contador en modo ascendente.
  - Contador en modo descendente.
- Unidad de posición y control para medir posición (PCCU). Esta unidad contiene dos registros de configuración (QEPCTL y QEPPOSCNT) para configurar los modos de operación del contador de posición, inicialización de los modos del contador de posición y la lógica de comparación de posición del generador de la señal de sincronía.
- Unidad de captura de flanco de captura para medida de bajas velocidades (QCAP). Sirve para medir el tiempo transcurrido entre dos eventos para el cálculo de velocidad.
- Unidad de base de tiempo para medir velocidad y frecuencia (UTIME). Incluye un temporizador de 32 bits (QUTMR) que es alimentado por el reloj SYSCLKOUT para generar interrupciones periódicas en el cálculo de velocidades. La interrupción de salida de la unidad de tiempo es fijada por el bit UTO del registro QFLG, cuando el temporizador QUTMR sea igual al registro de período QUPRD.
- Temporizador "watchdog" para detectar una ranura (QWDOG). Contiene 16 temporizadores "watchdog" que permiten monitorear la cuadratura de reloj para indicar la operación correcta de un sistema de monitoreo y control QEP.

Resumen

En este capítulo se han presentado los periféricos más importantes de los DSP C28X, como ya lo hemos manifestado, en muchas aplicaciones no es suficiente tener máquinas de alto desempeño en cuanto al procesamiento, sino que también es necesario tener buenas interfaces a periféricos que permitan la transferencia de información de forma eficiente. Aquí se han expuesto los periféricos de una forma muy genérica, sin embargo, los interesados pueden ampliar la información en los manuales de cada periférico enumerados en la bibliografía.

Capítulo 9

Puertos seriales

En los ambientes modernos de sistemas digitales y comunicaciones, las transmisiones seriales se utilizan ampliamente debido a que con un par de líneas es suficiente para transmitir gran cantidad de información. Probablemente en la actualidad los dispositivos digitales programables no pueden coexistir sin puertos de comunicación con el mundo externo, por lo que los puertos seriales han sido una solución eficiente y de pocos recursos. La familia de DSP C28x ha integrado a sus dispositivos varios periféricos que utilizan transmisión serial.

En este capítulo se han agrupado varios periféricos de la familia C28x que tienen la característica de transmisión serial y presentan similitudes.

9.1. Interfaz de comunicación serial

La interfaz de comunicación serial (SCI) es un puerto serial asíncrono de dos alambres y es conocido como UART (universal asynchronous receiver/transmitter). La SCI soporta comunicación digital entre el CPU y otros periféricos asíncronos que utilizan el formato estándar de no retorno a cero (NRZ). Esta interfaz es utilizada entre dispositivos para distancias largas y baja razón de transmisión [18], [31].

El receptor y transmisor SCI tienen cada uno 16 niveles de registros FIFO para evitar consumos de tiempo e independientemente tienen su propio habilitador de interrupciones. Puede operar en comunicación half-duplex o full-duplex.

Para asegurar la integridad de la transferencia de los datos, la SCI verifica los datos recibidos, la paridad, sobreescritura y errores de trama. La razón de bit es programable a diferente velocidad a través de un registro selector de baudaje de 16 bits. Un diagrama general del puerto SCI se observa en la figura 9.1.

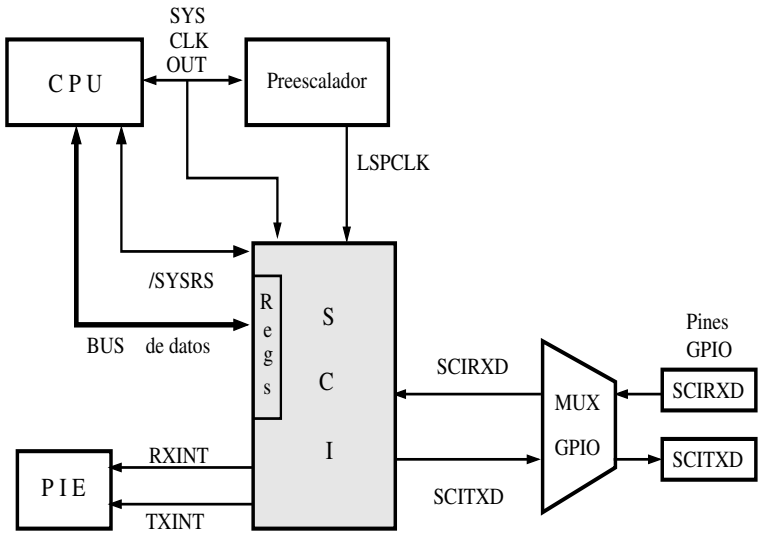


Figura 9.1. Diagrama del módulo SCI del C28x

9.1.1. Características

- Dos pines externos para la comunicación:
  - SCITXD: pin de salida de transmisión
  - SCIRXD: pin de recepción de entrada
  Ambos pines pueden ser utilizados como pines GPIO cuando no los use la SCI.

- Razón programable de hasta 64 Kbauds.

- Formato de palabra:
  - Un bit de inicio
  - Longitud de dato programable: de uno a ocho bits
  - Bit de paridad: par, impar o sin paridad
  - Uno o dos bits de paro
  - Un bit extra para distinguir dirección de dato

- Cuatro banderas de detección de error: paridad, sobreescritura, de trama y de ruptura.

- Dos modos de levantado del modo multiproceso: línea inactiva o en espera y dirección de bit.

- Operación “full” o “half duplex”.

- Funciones de doble buffereado para la transmisión y recepción.

- La transmisión y recepción puede operar a través de interrupciones o encuesta.

- Habilitador de interrupciones independientemente para transmisión y recepción.
- Formato NRZ.
- Reloj implícito.
- 13 registros para el control del módulo localizados a partir de la dirección 7050h. Todos estos registros son de ocho bits, la parte alta se llena con ceros.
- Lógica de auto detección del baudaje de hardware.
- 16 niveles FIFO para la transmisión y recepción.

El CPU escribe el dato a transmitir en el registro SCITXBUF, este dato es transferido al registro de corrimiento de transmisión TXSHF y el dato es corrido hacia el pin de salida de dato SCITXD un bit por cada ciclo de reloj. En la recepción, los datos arriban a través del pin SCIRXD un bit por cada ciclo de reloj, los bits son corridos en el registro de corrimiento RXSHF, una vez realizado el corrimiento del dato completo, éste es copiado a los registros SCIRXBUF y SCIXEMU, de donde lo puede leer el CPU.

En la tabla 9.1 se resumen los registros de la interfaz SCI, enseguida se describirán brevemente estos registros y el funcionamiento de la SCI. Las señales de la SCI se describen en la tabla 9.2

9.1.2. Comunicación por la interfaz SCI

El formato de comunicación asíncrona SCI utiliza una línea simple (una ruta) o dos líneas (dos rutas). Como se observa en la figura 9.2, en este modelo se utiliza la trama que consta de:

- Un bit de inicio
- Ocho bits de dato
- Un bit de paridad opcional
- Uno o dos bits de paro

donde existen ocho pulsos del reloj SCICLK por bit del dato. En la *recepción* se inicia la operación en un período válido del bit inicio. Este bit de inicio es identificado por cuatro períodos consecutivos del reloj SCICLK con el bit en cero, de lo contrario si alguno de los bits es no cero la inicialización termina y se reinicia la detección de otro bit de inicio. Como se observa en la figura 9.2, para los bits que siguen al de inicio, el proceso determina el valor del bit muestreando tres veces en medio de este bit. El muestreo ocurre en el cuarto, quinto y sexto período del reloj SCICLK y determina el valor del bit válido en al menos dos de esos tres ciclos SCICLK. El receptor se sincroniza con la trama de entrada, por lo que no es necesario un reloj externo de sincronía, el reloj se debe generar localmente.

Tabla 9.1. Registros de la SCI

Registro	**Dirección (h)**	**Descripción**
SPICCR	00 7050	Control de comunicación de la SCI-A
SPICTL1	00 7051	Control de SCI-A
SPIHBAUD	00 7052	Baudaje, parte alta SCI-A
SPILBAUD	00 7053	Baudaje, parte baja SCI-A
SPICTL2	00 7054	Control 2, de SCI-A
SPIRXBST	00 7055	Estado de recepción SCI-A
SPIRXEMU	00 7056	Emulación de datos recibidos SCI-A
SPIRXBUF	00 7057	Dato recibido SCI-A
SPITXBUF	00 7059	Dato transmitido SCI-A
SPIFFTX	00 705A	FIFO de transmisión SCI-A
SPIFFRX	00 705B	FIFO de recepción SCI-A
SPIFFCT	00 705C	FIFO de control SCI-A
SPIPRI	00 705F	Control de prioridad SCI-A
SPICCR	00 7750	Control de comunicación de la SCI-B
SPICTL1	00 7751	Control de SCI-B
SPIHBAUD	00 7752	Baudaje, parte alta de la SCI-B
SPILBAUD	00 7753	Baudaje, parte baja de la SCI-B
SPICTL2	00 7754	Control 2, de la SCI-B
SPIRXBST	00 7755	Estado de recepción de la SCI-B
SPIRXEMU	00 7756	Emulación de datos recibidos SCI-B
SPIRXBUF	00 7757	Dato recibido de la SCI-B
SPITXBUF	00 7759	Dato transmitido de la SCI-B
SPIFFTX	00 775A	FIFO de transmisión de la SCI-B
SPIFFRX	00 775B	FIFO de recepción de la SCI-B
SPIFFCT	00 775C	FIFO de control de la SCI-B
SPIPRI	00 775F	Control de prioridad de la SCI-B

Tabla 9.2. Señales de la interfaz SCI

Señal	**Descripción**
Externas	
SCIRXD	Recepción de datos
SCITXD	Transmisión de datos
Control	
Reloj de baudaje	Reloj preescalador LSPCLK
Interrupción	
TXINT	Interrupción de trasmisión
RXINT	Interrupción de recepción

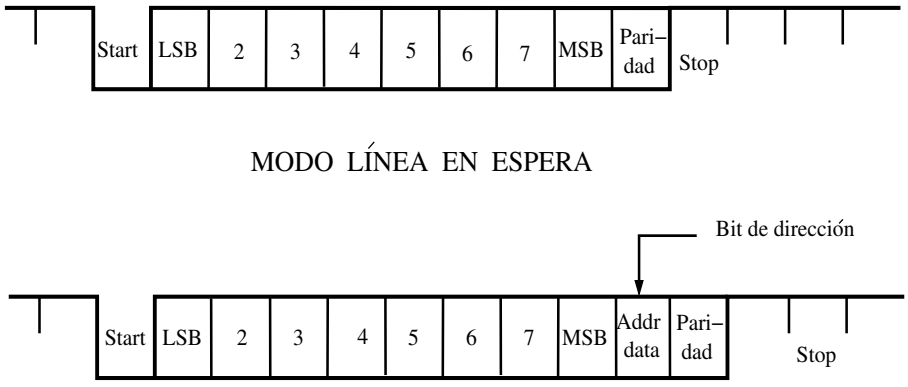


Figura 9.2. Formato de bits la interfaz SCI

Proceso de recepción

En modo de dirección de levantado (wake-up):
• Cuando la bandera RXENA (bit 0 del registro SCICTL1) se va a alto, habilita la recepción.
• Un dato es recibido en el pin SCIRXD, cuando se detecta el bit de inicio.
• El dato es corrido en el registro RXSHF y escrito en SCIRXBUF, se emite un requerimiento de interrupción. La bandera RXRDY (bit 6 del registro SCIRXST) se va alto para indicar que un caracter ha arribado.
• Por programa se lee el registro SCIRXBUF y la bandera RXRDY es limpiada automáticamente.
• El nuevo dato arriba en el pin SCIRXD. El bit de inicio es detectado y borrado.
• El bit RXENA va a un nivel bajo para deshabilitar la recepción. Un dato continúa siendo corrido en RXSHF, pero no es transferido al registro de recepción.

Proceso de transmisión

La transmisión se realiza de la siguiente forma:
• El bit TXENA (bit uno del registro SCICTL1) va a alto habilitando la transmisión de envío de datos.
• El registro SCITXBUF es escrito con un dato de un byte, si la transmisión no está vacía y el TXRDY va a nivel bajo.
• La SCI transfiere el dato al registro de corrimiento de transmisión TXSHF, el transmisor está listo para enviar un segundo caracter (TSRDY va a alto), se presenta un requerimiento de interrupción (se pone en uno el bit TX INT ENA del registro SCICLT2).
• El programa escribe un segundo caracter a SCITXBUF después de que TXRDY va a alto. TXRDY va a nivel bajo de nuevo después que se escribe el segundo caracter en SCITXBUF.
• La transmisión del primer caracter está completa, la transferencia del segundo caracter inicia a través del registro de corrimiento TXSHF.
• El bit TXENA va a nivel bajo para deshabilitar la transmisión, la SCI termina la transmisión del caracter actual.
• La transmisión del segundo caracter es completada, la transmisión está vacía y lista para un nuevo caracter.

9.1.3. Interrupción a la interfaz SCI

La recepción y la transmisión de la SCI pueden ser controladas por interrupciones. El registro SCICTL2 contiene un bit de bandera TXRDY que indica la condición activa de interrupción, y el registro SCIRXST contiene los bits de banderas RXRDY y BRKDT, más la bandera de interrupción RX ERROR, que es una lógica OR de las condiciones FE, OE y PE.

Los bits de habilitación y los vectores de interrupciones para la transmisión y recepción son independientes. Los requerimientos de interrupción pueden priorizarse, esto se indica en el controlador de interrupciones PIE. Cuando la transmisión y la recepción se requieren con la misma prioridad, la recepción siempre adquiere una prioridad más alta que la transmisión reduciendo la posibilidad de un error de sobreescritura en la recepción.

Si el bit RX/BK INT ENA del registro SCICTL2 está en uno, se presenta un requerimiento de interrupción cuando:
• La SCI recibe una trama completa y ha transferido el dato completo de RXSHF al registro SCIRXBUF. En esta acción se fija la bandera RXRDY (bit 6 del registro SCIRXST) y se inicia el proceso de interrupción.
• Ocurre una condición de ruptura (SCIRXD permanece en bajo durante 10 períodos de bit seguidos a un bit de paro perdido, esto fija la bandera BRKDT (bit 5 del registro SCITXBUF) y se inicia la interrupción.

Si el bit 0, TX INT ENA del registro SCCICTL2 es puesto a uno, se emite un requerimiento de interrupción de transmisión aunque el dato en el registro SCITXBUF sea transferido al registro de corrimiento TXSHF, indicando que el CPU puede escribir a SCITXBUF. Esta acción fija la bandera TXRDY (bit 7 del registro SCICTL1) e inicia la interrupción.

9.1.4. Comunicación multiprocesos por SCI

La comunicación multiprocesos permite al C28x enviar a otros procesadores bloques de datos a través de una misma línea serial, de todos los procesadores conectados a la línea sólo uno de ellos envía a la vez y los otros se identifican para recibir el dato que les corresponde.

En esta forma de comunicación, el primer byte enviado contiene una dirección que es leída por todos los receptores, sólo los receptores que se identifican con esa dirección programada de antemano pueden ser interrumpidos para recibir el dato que está en el segundo byte de la comunicación, los otros procesadores no pueden recibir el dato.

Todos los procesadores conectados en serie ponen en uno el bit SCI SLEEP (bit 2 del registro SCICTL1), para que puedan interrumpirse cuando la dirección correspondiente sea detectada, cuando el procesador lee el bloque de dirección y es igual a la programada, se borra el bit SLEEP y genera una interrupción de recepción del dato.

Reconocimiento del byte de dirección

El C28X utiliza dos modos para reconocer el byte de dirección en comunicación serial multiprocesos: modo inactivo en línea y modo bit de dirección. Para la operación multiprocesos se selecciona vía software a través del bit ADDR/IDLE MODE (bit 3 del registro SCICCR). En ambos modos se utiliza la bandera TXWAKE (bit 3 del registro SCICTL1), la bandera RXWAKE (bit 1 del registro SCIRXST) y la bandera SLEEP (bit 2 del registro SCICTL1).

Secuencia de recepción

En ambos modos la secuencia de recepción es:
• En la recepción de un bloque de dirección, el puerto SCI despierta y emite una interrupción (el bit RX/BK INT ENA, bit uno del registro SCICTL2 debe estar habilitado para que se presente la interrupción), se lee la trama de dirección que contiene la dirección destino.
• Una rutina de software verifica la dirección entrante y es comparada con la dirección del dispositivo almacenada en memoria.
• Si se trata de la dirección del CPU correspondiente, el CPU borra el bit SLEEP y lee el resto del bloque, de lo contrario el bit SLEEP permanece en uno y no se recibe la interrupción.

Modo de línea inactiva o espera

En este modo, el bit ADDR/IDLE MODE = 0 (bit 3 del registro SCICCR), los bloques son separados por un tiempo inactivo más largo que entre tramas de bloques. Este modo deja un espacio antes de la dirección byte, no se agrega un bit extra para indicar dirección o dato, y es más eficiente que el modo bit de dirección para el manejo de bloques que contienen más de 10 bytes de datos. Este modo es el que debe usarse para comunicaciones típicas multiprocesos SCI y es mostrado en la figura 9.3.

Los pasos para este modo son:
• El puerto SCI despierta después de recibir la señal de inicio de bloque.
• El procesador reconoce la interrupción de SCI.
• La rutina de atención de interrupción compara la dirección recibida con la dirección propia.
• Si se trata del CPU direccionado, la rutina de interrupción limpia el bit SLEEP y recibe el resto del bloque de datos.
• De lo contrario, si no se trata de su dirección, el bit SLEEP permanece en uno, el CPU continúa ejecutando su programa principal sin ser interrumpido hasta que se presente otro bloque de inicio.

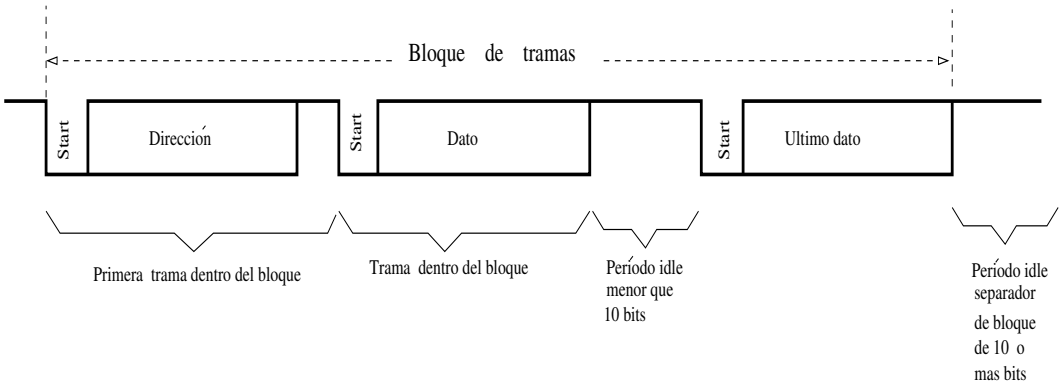


Figura 9.3. Formato de comunicación multiprocesos en modo inactivo en línea

Modo bit de dirección

En este modo, el bit ADDR/IDLE MODE = 1. Este modo agrega un bit extra a la dirección en cualquier byte para distinguir entre dirección y dato, el bit agregado vale uno para la dirección. Este modo es más eficiente para el manejo de bloques pequeños de datos, aunque diferente al modo inactivo, no tiene que esperar entre bloques de datos. Sin embargo, para transmisiones de alta velocidad, no es suficientemente rápido para evitar 10 bits inactivos en el flujo de bits de transmisión. La forma de transmisión en este modo es mostrado en la figura 9.4.

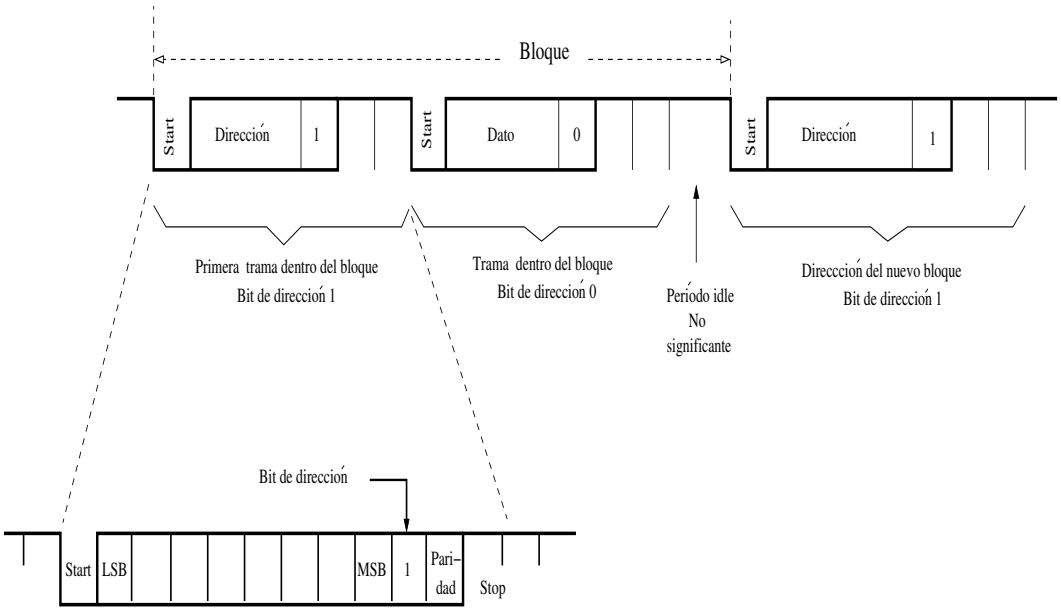


Figura 9.4. Transmisión multiprocesos en modo bit de dirección

9.2. Interfaz de puerto serie (SPI)

La interfaz SPI es un puerto serial síncrono de entrada/salida (I/O) que permite transferencias de alta velocidad de 1 a 16 bits, esta se utiliza normalmente para la comunicación de DSP con otros periféricos externos en distancias cortas [18][32]. Su diagrama general se muestra en la figura 9.5.

Características generales

• Dos modos de operación: maestro y esclavo con combinaciones.
• Utiliza un reloj explícito.
• Razón de baudaje: hasta 125 diferentes modos programables. El máximo baudaje está limitado por la máxima velocidad de los registros de I/O utilizados en los pines SPI.
• Longitud de palabra de 1 a 16 bits.
• Cuatro esquemas de reloj:

- Flanco de bajada sin retardo: el reloj SPICLK es activo alto, el puerto SPI transmite un dato en el flanco de bajada de la señal SPICLK y recibe datos en el flanco de subida de la misma señal.

- Flanco de bajada con retardo: el reloj SPICLK es activo alto. La SPI transmite un dato en la mitad del ciclo adelante en el flanco de bajada de la señal SPICLK y recibe datos en el flanco de caída de la misma señal.

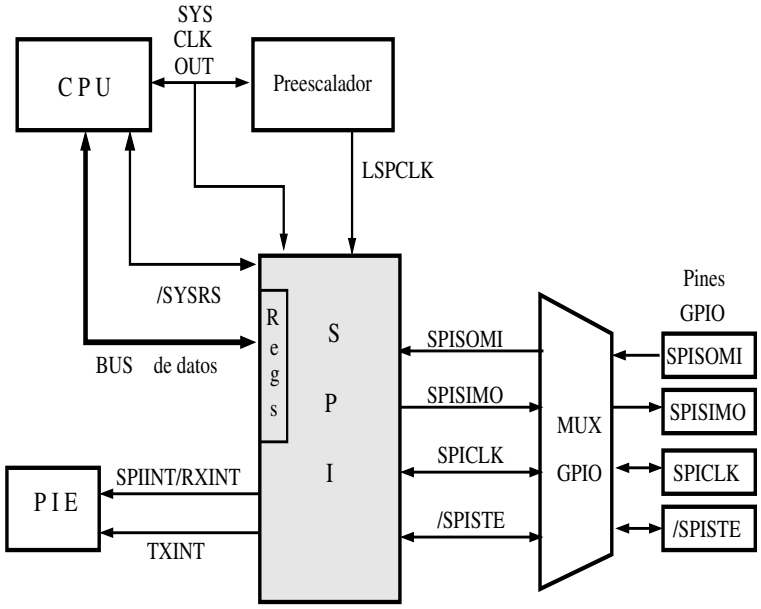


Figura 9.5. Diagrama general del módulo SPI del C28x

- Flanco de subida sin retardo: el reloj SPICLK es inactivo bajo. La SPI transmite un dato en el flanco de subida de la señal SPICLK y recibe datos en el flanco de caída de la misma señal.
- Flanco de subida con retardo: el reloj SPICLK es inactivo bajo. La SPI transmite un dato en la mitad del ciclo adelante en el flanco de caída de la señal SPICLK y recibe datos en el flanco de subida de la misma señal.

• Operación simultánea de recepción y transmisión.
• La transmisión y recepción pueden operarse por interrupciones o encuesta.
• 12 registros de control localizados a partir de la dirección 7040h.
• 16 niveles de transmisión recepción FIFO.
• Control de retardo de transmisión.

9.2.1. Registros de SPI

En la tabla 9.4 se resumen los registros asociados al funcionamiento del puerto SPI, y enseguida se describirán brevemente estos registros y su funcionamiento.

En su forma más simple, el puerto SPI en esencia funciona como un registro de corrimiento programable y controlable. Los datos son corridos y enviados a través del puerto utilizando el registro SPIDAT, transmitiendo primero el bit MSb, los 16 bits transmitidos de SPIDAT son copiados en el registro SPIBUF, después de que la transmisión es completada.

Tabla 9.3. Señales de la interfaz SPI

Señal	**Descripción**
Externas	
SPICLK	Reloj SPI
SPISIMO	SPI entrada esclavo, salida maestra
SPISOMI	SPI salida esclavo, entrada maestra
SPISTE	SPI habilita transmisión esclava (opcional)
Control	
Razón de reloj de SPI	LSPCLK
De interrupción	
SPIRXINT	Interrupción de transmisión/recepción modo no FIFO Interrupción de recepción en modo FIFO
SPITXINT	Interrupción de transmisión en modo FIFO

Tabla 9.4. Registros de la SPI

Registro	**Dirección (h)**	**Descripción**
SPICCR	00 7040	Control de configuración SPI
SPICTL	00 7041	Control de operación
SPISTS	00 7042	De estado
SPIBRR	00 7044	Razón de baudaje
SPIRXEMU	00 7046	Emulación de buffer de recepción
SPIRXBUF	00 7047	Buffer de entrada
SPITXBUF	00 7048	Buffer de salida
SPIDAT	00 7049	De dato serial
SPIFFTX	00 704A	FIFO de transmisión
SPIFFRX	00 704B	FIFO de recepción
SPIFFCT	00 704C	FIFO de control
SPIPRI	00 704F	Control de prioridad

Un dato a transmitir es escrito directamente en el registro SPIDAT, un dato recibido es almacenado en el registro SPIBUF y puede ser leído por el CPU.

SPICCR: Registro de control de configuración SPI, contiene los bits de control para la configuración de la SPI, dentro de estas funciones están:
• Bit 7: en cero resetea el módulo SPI.
• Bit 6: selección de polaridad del reloj SPICLK.
• Bit 5: reservado.
• Bit 4: habilita modo "loopback" en uno, para pruebas del puerto y sólo es válido en modo maestro.
• Bits 3..0: cuatro bits para la configuración de la longitud de palabra, 0000 corresponde a un bit, hasta 1111 para 16 bits.

SPICTL: Registro de control de operación, para transmisión de datos.
• Dos bits para la habilitación de interrupciones SPI.
• Selección de fase SPICLK.
• Modo de operación esclavo/maestro.
• Habilitación de transmisión de datos.

SPISTS: Registro de estado, contiene dos bits de estado para recepción y un bit de estado para transmisión:
• Sobreflujo en la recepción.
• Bandera de interrupción.
• Bandera de buffer de transmisión lleno.

SPIBRR: Registro de razón de baudaje (bits 6:0), contiene siete bits que determinan la razón de transferencia. La razón de baudaje se calcula con:

$$\text{SPI razón de baudaje} = \frac{LSPCLK}{SPIBRR + 1} \tag{9.1}$$

donde el registro SPIBRR, contiene una constante de 3 a 127. Con señal de reloj SPCLK

$$SPICLK = \frac{LSPCLK}{SPIBRR + 1} \tag{9.2}$$

Es decir, puede tener hasta 125 tasas de baudaje.

SPIRXEMU: Registro de emulación de buffer de recepción, contiene el dato recibido. Este registro es utilizado sólo para propósitos de emulación.

SPIRXBUF: Registro de buffer de entrada, buffer de recepción serial, contiene el dato recibido.

SPITXBUF: Registro de buffer de salida, contiene el próximo dato a transmitir.

SPIDAT: Registro de dato serial, contiene el dato a ser transmitido, actúa como un registro de corrimiento de recepción y transmisión.

SPIPRI: Registro de control de prioridad, especifica la prioridad de interrupciones y determina la operación SPI durante la suspensión del programa por emulación XDS.

9.2.2. Operación de la interfaz SPI

Para la operación maestro esclavo de la interfaz SPI entre dos C28x, se configuran de antemano ambos dispositivos. La comunicación inicia cuando el maestro envía una señal de SPICLK. En ambos modos maestro y esclavo, el dato es desplazado en el registro de corrimiento SPIDAT, bit a bit por cada flanco del reloj SPICLK y detectado en el otro registro de corrimiento en el otro flanco del reloj SPICLK. En este caso, ambos controladores envían y reciben información simultáneamente.

Posibles métodos de transmisión:
• El maestro envía datos, el esclavo envía datos comodín.
• El maestro envía datos, el esclavo también envía datos.
• El maestro envía datos comodín, el esclavo envía datos.

Modo maestro

En este modo MASTER/SLAVE = 1, la interfaz SPI provee el reloj serial vía el pin SPICLK para la comunicación. El dato de salida se presenta en el pin SPISIMO y es registrado en el pin de entrada SPISOMI. El registro SPIBRR determina la transmisión y la recepción para ambos pines, este registro puede seleccionar hasta 123 razones de transferencia.

El dato escrito en SPIDAT o SPIXBUF inicia la transmisión vía el pin SPISIMO, el bit MSb se envía primero. Simultáneamente, el dato recibido en el esclavo es corrido a través del pin SPISOMI, entrada en el bit LSb de SPIDAT. Cuando el número de bits seleccionados han sido transmitidos, el dato recibido es transferido al buffer de recepción SIPRXBUF para su lectura por el CPU. Hasta este momento han ocurrido los eventos:
• SPIDAT contiene el dato transmitido a SPIRXBUF.
• La bandera SPI INT, bit 6 del registro SPISTS es puesto a uno.
• Si existe un dato válido en el buffer de transmisión SPITXBUF, indicado por el bit TXBUF FULL de SPISTS, el dato es transferido a SPIDAT y transmitido, de lo contrario, SPICLK se detiene después de que todos los bits han sido corridos fuera de SPIDAT.
• Si el bit 0, SPI INT ENA de SPICTL es fijado a uno, se emite una interrupción.

Modo esclavo

En este modo, el bit MASTER/SLAVE = 0, los datos son corridos hacia afuera en el pin SPISOMI y de entrada en el pin SPISIMO. El dato escrito en SPIDAT o SPITXBUF es transmitido en el flanco apropiado de la señal SPICLK del maestro. El dato escrito en SPITXBUF será transferido a SPIDAT cuando todos los bits del caracter transmitido han sido corridos fuera de SPIDAT.
Si ningún caracter está siendo transmitido cuando SPITXBUF es escrito, el dato será transferido inmediatamente a SPIDAT. Para recibir un dato, SPI espera que el maestro le envíe la señal de reloj y entonces realiza el corrimiento del dato en el pin SPISIMO hacia SPIDAT. Si un dato va a ser transmitido por el esclavo simultáneamente y SPITXBUF no ha sido cargado previamente, el dato debe ser escrito a SPITXBUF o SPIDAT antes del inicio de la señal SPICLK.

9.2.3. Interrupciones de SPI

El registro SPICTL es el encargado de configurar y controlar las interrupciones de SPI a través de sus bits:
• Bit 0, SPI INT ENA: en cero deshabilita interrupciones y en uno las habilita.
• Bit 6, SPI INT FLAG: en uno indica que un caracter ha sido puesto en el buffer receptor SPI y el dato está listo para ser leído. Esta bandera puede ser reseteada por varias fuentes, en el reset del CPU, cuando se lee el dato en SPIRXBUF, el reset del SPI o el dispositivo entra en un ciclo IDLE2 o HALT.
• Bit 4, OVERRUN INT ENA: habilita una interrupción de sobreescritura en la recepción. Habilitado en uno, deshabilitado en cero.
• Bit 7, RECEIVER OVERRUN FLAG: esta bandera se fija a uno cuando un nuevo caracter es recibido y cargado en SPIRXBUF antes que el caracter previo haya sido leído. Este bit se limpia por software.

9.3. Controlador de red de área

El controlador de red de área mejorado (eCAN) es una versión del módulo controlador de red de área (CAN), este último sólo contiene 16 buzones, mientras que eCAN dispone de 32. El eCAN implementado en el DSP C28x es compatible con el estándar eCAN 2.0B, es un módulo versátil y robusto para comunicación serial en ambientes industriales. Utiliza un protocolo de comunicación serial con otros controladores en ambientes ruidosos y contiene 32 buzones de correo configurables. El controlador eCAN minimiza la carga y consumo de tiempo del CPU en las comunicaciones [18], [30].

Características

El módulo eCAN tiene las siguientes características:
• Completamente compatible con el protocolo CAN 2.0B.
• Soporta una razón de transferencia de datos de hasta 1Mbps.
• Contiene 32 buzones de correo configurables.
• Configurable como receptor o transmisor.
• Configurable con identificadores estándares o extendibles.
• Soporta datos y tramas remotos.
• Soporta de 0 a 8 bytes de datos.
• Utiliza 32 bits de figura o estampa de datos en la recepción y transmisión de mensajes.
• Protección contra nuevos mensajes.
• Permite programación dinámica de prioridades en la transmisión de mensajes.
• Emplea un esquema programable de interrupciones con dos niveles de interrupción.
• Emplea programación de interrupciones en la transmisión y recepción de tiempos fuera.
• Modos de baja potencia.
• Despertado programable en la actividad de bus.
• Forma automática de repetición de un mensaje remoto requerido.
• Retransmisión automática de una trama en caso de pérdida de arbitraje o un error.
• Contador de figura de tiempo 32 bits para sincronizar mensajes específicos.
• En los dos extremos del bus se recomienda conectar una resistencia de 120 Ω.

9.3.1. Módulo eCAN

El controlador eCAN utiliza un protocolo de comunicación serial multimaestro que eficienta el control distribuido en tiempo real, con un alto nivel de seguridad y una razón de comunicación de hasta 1 Mbps. El bus eCAN es ideal para aplicaciones que operan en ambientes ruidosos, tales como los automóviles e industria. Prioriza mensajes de hasta seis bytes de longitud de datos y los puede enviar a través de un bus serial multimaestro utilizando protocolo de arbitraje y mecanismos de detección de errores para alto nivel de integridad.

Protocolo eCAN

El protocolo eCAN soporta cuatro diferentes tipos de trama para comunicación:
• Tramas de datos que portan los datos de un nodo transmisor a nodos receptores.
• Tramas remotas que son transmitidas por un nodo que requiere transmisión de trama de datos con igual identificador.
• Errores de trama que son transmitidos por cualquier nodo o detección de error de bus.
• Sobrecarga de tramas que proveen un retardo extra entre trama de datos anterior y posterior.

Además, las especificaciones de la versión 2.0B de eCAN, definen dos formatos diferentes en la longitud del identificador de campo: trama estándar con 11 bits de identificación y tramas extendidos con 29 bits de identificación.

La trama estándar de datos eCAN contiene de 44 a 108 bits y el extendido eCAN contiene de 64 a 128 bits. Además, hasta 23 bits extras de relleno que pueden ser insertados en un trama estándar, y 28 bits de relleno en el formato extendido. La máxima trama de datos puede ser de hasta 131 bits en eCAN estándar y 156 en el extendido, el formato con sus campos de bits puede observarse en la figura 9.6.

- Inicio de trama.
- Campo de arbitraje que contiene un identificador y tipo de mensaje a enviar (11 a 29 bits).
- Campo de control que contiene el número de datos.
- Campo de datos de hasta ocho bytes.
- Campo de verificación de redundancia cíclica (CRC).
- Reconocimiento.
- Fin de trama.

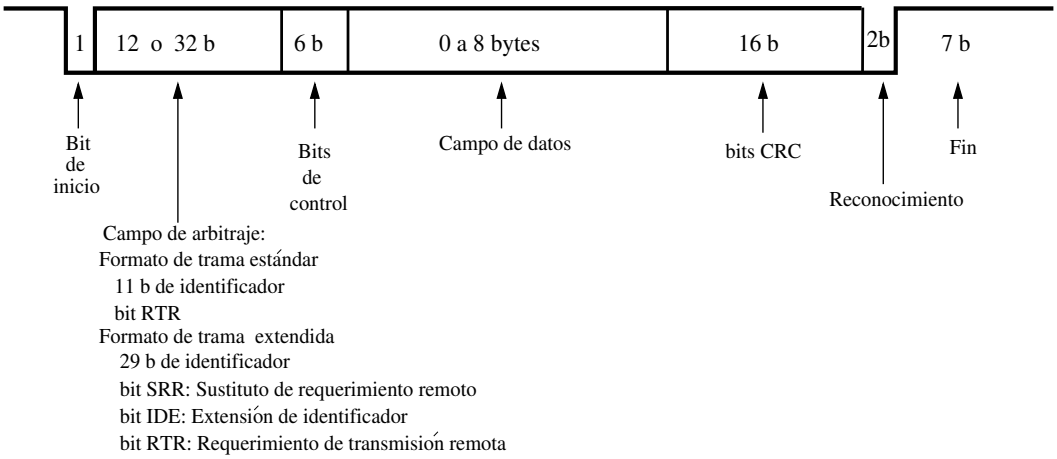


Figura 9.6. Diagrama del formato eCAN del C28x

9.3.2. Funcionamiento del controlador eCAN

La comunicación del módulo eCAN con el CPU se realiza a través del controlador de mensajes y un kernel del protocolo eCAN (CPK), como se muestra en la figura 9.6. Las funciones del CPK son:

- Decodificar todos los mensajes recibidos en el bus eCAN de acuerdo con el protocolo eCAN y transferir mensajes a un buffer receptor en la inicialización. El CPU especifica al controlador de mensajes de todos los identificadores de mensajes a utilizar en la aplicación.

- Transmitir mensajes sobre el bus eCAN de acuerdo con el protocolo CAN.

En la inicialización, el CPU especifica al controlador de mensajes todos los identificadores de mensajes a utilizar en la aplicación. El controlador de mensajes eCAN:

- Es responsable de determinar si el mensaje recibido por el CPK debe ser preservado por el CPU o descartado.
- Es responsable de enviar el próximo mensaje a transmitir del CPK de acuerdo con la prioridad de mensajes.
- Contiene una unidad manejadora de memoria (MMU), incluye una interfaz al CPU, control de recepción y una unidad de manejo de tiempo.
- Buzón RAM que habilita el almacenamiento de 32 mensajes.
- Registros de control y de estado.

Después de la *recepción* por CPK de un mensaje válido, la unidad de control de recepción de mensajes determina si el mensaje recibido debe ser almacenado en los 32 mensajes objeto RAM en el buzón. La unidad de control de recepción verifica el estado, el identificador y la máscara de todos los mensajes objeto para determinar su localización apropiada en el buzón. El mensaje recibido pasa por un filtro de aceptación para ser escrito en la primera localidad del buzón, si la unidad de control no encuentra un buzón de almacenamiento, el mensaje es descartado.

Cuando un mensaje va a ser *transmitido*, el controlador lo transfiere al buffer de transmisión del CPK para empezar la transmisión del mensaje en próximo estado inactivo del bus. Cuando existe más de un mensaje, el mensaje con más alta prioridad está listo para ser transmitido y se transfiere en el CPK por el controlador de mensajes. Si dos mensajes tienen la misma prioridad, entonces se transmite primero el buzón con el número mayor.

La unidad de manejo de tiempo se encarga de componer las figuras o estampas de tiempo para recibir y transmitir. Esta unidad genera una interrupción cuando un mensaje no ha sido recibido o transmitido durante el período permitido de tiempo.

Para iniciar la transferencia de datos, el bit de requerimiento de transmisión debe ser puesto a uno en el registro de control eCAN correspondiente. El buzón puede configurarse para interrumpir al CPU después de que la transmisión o recepción de mensajes ha sido exitosa.

Mapa de memoria eCAN

El módulo eCAN contiene dos segmentos de dirección mapeados en la memoria del DSP C28x. El primer segmento es utilizado para acceder a los registros de control, registros de estado, máscaras de aceptación, estampa de tiempo y el tiempo del mensaje objeto. El acceso a los registros de control y estado está limitado a 32 bits de ancho. La máscara

local de aceptación, el registro estampa y registros de tiempo pueden ser accedidos a 8, 16 y 32 bits. Los 32 buzones de mensajes son de longitud de 8 bytes (256 bits), que pueden ser configurados para transmisión o recepción y cada uno tiene una máscara individual de aceptación. Los registros LAMn, MOTSn, MOTOn y los registros de buzón no utilizados en una aplicación pueden ser utilizados como registros de propósito general (deshabilitados en el registro CANME). En las tablas 9.5 y 9.6, se describen los registros del controlador eCAN y sus direcciones.

Buzones

El módulo eCAN tiene 32 diferentes mensajes objeto, cada uno puede ser configurado como transmisión o recepción y tiene su propia máscara de aceptación. Un mensaje objeto consiste de un mensaje en el buzón con:
• 29 bits de identificador de mensaje.
• Registro de control de mensaje.
• 8 bytes del dato mensaje.
• Una máscara de aceptación de 29 bits.
• 32 bits de estampa de tiempo.
• 32 bits de tiempo de salida.

Transmisión de un buzón

El CPU almacena un dato que será transmitido en un buzón configurado para transmitir. Después de haber sido escrito el dato y su identificador de RAM, el mensaje es enviado si el bit correspondiente TRS[n] ha sido puesto a uno, habilitando al buzón al fijar su bit ME.n, $n = 0, \cdots, 31$.
Si se quieren enviar más de dos buzones, los mensajes se envían de acuerdo con la prioridad. La prioridad de los buzones de transmisión dependen de fijar el campo TPL en el registro de control MSGCTRL.
Si la transmisión falla debido a la pérdida de arbitraje o error, la transmisión del mensaje será reintentado, sin embargo, antes se verifica si existe otro requerimiento de transmisión y envía el buzón con mayor prioridad.

Recepción de un buzón

El identificador de cada mensaje que ingresa es comparado con el identificador que está en el buzón utilizando su máscara apropiada. Cuando se detecta que son iguales, se escribe en su localidad RAM el identificador recibido, los bits de control y los bytes de datos. Al mismo tiempo, el bit de mensaje recibido pendiente RPM[n] (RMP.31..0) es fijado a uno y se genera una interrupción si está habilitada. Después de que el CPU lee el dato, el bit RPM[n] (RMP.31..0) se limpia.

Tabla 9.5. Registros del controlador eCAN

Registro	**Dirección (h)**	**Descripción**
CANME	00 6000	Habilitación de buzón
CANMD	00 6002	Dirección de buzón
CANTRS	00 6004	Fija requerimiento de transmisión
CANTRR	00 6006	Resetea requerimiento de transmisión
CANTA	00 6008	Reconocimiento de transmisión
CANAA	00 600A	Aborta el reconocimiento
CANRMP	00 600C	Mensaje de recepción pendiente
CANRML	00 600E	Mensaje recibido perdido
CANRFP	00 6010	Trama remota pendiente
CANGAM	00 6012	Máscara global de aceptación
CANMC	00 6014	Control maestro
CANBTC	00 6016	Configuración de bit de tiempo
CANES	00 6018	Error y estado
CANTEC	00 601A	Contador de error de transmisión
CANREC	00 601C	Contador de error de recepción
CANGIF0	00 601E	Banderas de interrupción global 0
CANGIM	00 6020	Banderas de máscara de interrupción
CANGIF1	00 6022	Banderas de interrupción global 1
CANMIM	00 6024	Máscara de interrupción de buzón
CANMIL	00 6026	Nivel de interrupción de buzón
CANOPC	00 6028	Control de protección de sobreescritura
CANTIOC	00 602A	Control TX I/O
CANRIOC	00 602C	Control RX I/O
CANTSC	00 602E	Contador de estampa de tiempo (Modo SCC)
CANTOC	00 6030	Control de tiempo de salida (Modo SCC)
CANTOS	00 6032	Control de estado de salida (Modo SCC)
RESERVADO	00 6034	
...	...	
RESERVADO	00 603F	

Tabla 9.6. Registros del controlador eCAN (continuación)

Registro	**Dirección (h)**	**Descripción**
LAM	00 6040 ... 607F	Máscara de aceptación local (32 x 32 bits en RAM)
MOTS	00 6080 ... 60BF	Mensajes objeto de estampa de tiempo (32 x 32 bits en RAM)
MOTO	00 60C0 ... 60FF	Mensajes objeto de tiempo de salida (32 x 32 bits en RAM)
Buzón 0	6100 ... 6107	32 buzones de 128 bits
Buzón 1	6108 ... 610F	
Buzón 2	6110 ... 6117	
...	...	
Buzón 29	61E8 ... 61EF	
Buzón 30	61F0 ... 61EF	
Buzón 31	61F8 ... 61FF	
MSGID	61E8...61E9	Identificador de mensaje (32b)
MSGCTRL	61EA...61EB	Control de mensaje (32b)
MDL	61EC...61ED	Dato mensaje parte baja (4 bytes)
MDH	61EE...61EF	Dato mensaje parte alta (4 bytes)

Cuando el mensaje es recibido, el controlador de mensaje empieza a buscar en los buzones uno que coincida, empezando por la prioridad más alta.
Si un segundo mensaje es recibido en el mismo buzón y el bit RPM[n] (RMP.31..0) no se ha limpiado, el mensaje entrante se pierde y se pone el bit RPL[n] (RML.31..0).

9.4. Controlador I2C

El módulo I2C (Inter-integrated circuit) está disponible únicamente en las familias F2823x, F2833x y F280xx, éste módulo permite realizar una interface entre estos periféricos C28x y dispositivos que acepten el estándar I2C V2.1 de Philips Semiconductores. Este módulo simplifica las conexiones seriales, ya que utilizan sólo dos líneas para la recepción y la transmisión de 1 a 8 bits de datos [33], [19].

Como se observa en la figura 9.7, el módulo I2C soporta dispositivos maestros o esclavos compatibles. La comunicación se establece a través de dos líneas seriales, una para datos (SDA) la otra para reloj (SCL) conectados a la alimentación Vcc a través de resistencias de "pull up", de tal forma que en reposo permanecen en nivel alto. Ambos pines SDA y SCL son bidireccionales.

Características

• Compatible con el bus Philips V2.1:
• Formato de transferencia de 8 bits.
• Modo de direccionamiento de 7 y 10 bits.
• Modo de byte START.
• Soporta múltiples maestros transmisores y esclavos receptores.
• Soporta múltiples esclavos transmisores y maestros receptores.
• Combina modos maestro transmisor/receptor y receptor/transmisor.
• Transferencia de datos de 10 Kbps hasta 400 Kbps.
• 16 registros FIFO para recepción y 16 FIFO para transmisión.
• Una interrupción para ser utilizada por el CPU, con las condiciones:
- Transmisión de dato listo.
- Recepción de dato listo.
- Registro de acceso listo.
- No se reconoció la recepción.
- Pérdida de arbitraje.
- Detección de paro.
- Direccionamiento como esclavo.
• Interrupción adicional que puede usar el CPU en modo FIFO.
• Capacidad de habilitar y deshabilitar el módulo.
• Modo de formato de dato libre.

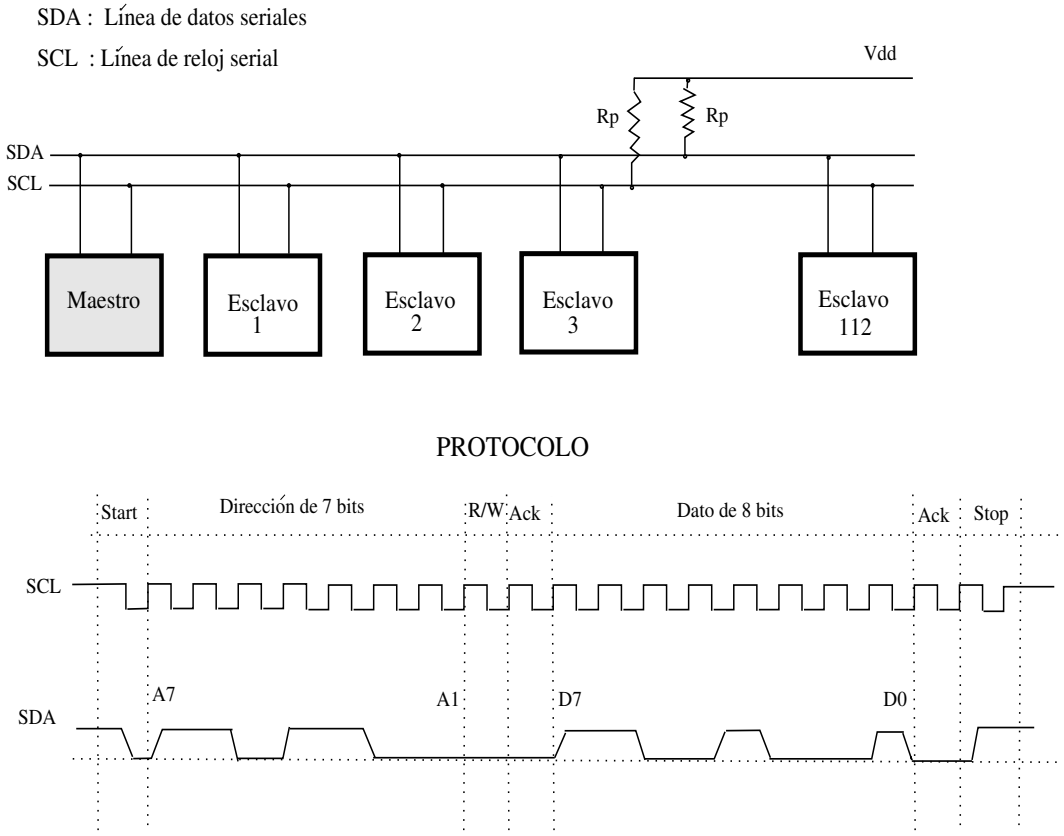


Figura 9.7. Conexión entre dispositivos con el módulo I2C y el protocolo de comunicación

9.4.1. Funcionamiento

A cada dispositivo conectado al bus I2C se le reconoce por una dirección de siete bits, cada dispositivo puede ser receptor o transmisor dependiendo de la función deseada del dispositivo. Un dispositivo maestro es el que inicia la transferencia de datos en el bus y genera la señal de reloj para la transferencia, durante esta transferencia cualquier dispositivo direccionado por el maestro es considerado esclavo.

Técnicas de transferencia

• Modo estándar: envía exactamente "n" valores de datos, donde "n" es el valor programado en el módulo I2C a través de registros.
• Modo repetición: cuida el envío de valores de datos hasta que por software se inicie la condición de STOP o una nueva condición de START.

Bloques principales

• Interface serial: pin de datos SDA y pin de reloj SCL.
• Registros de datos y registros FIFO para retener temporalmente datos recibidos y transmitidos que viajan entre el pin SDA y el CPU.
• Registros de control y estado.
• Interfaces a buses de periféricos que habilitan al CPU el acceso a los registros del módulo y los FIFO.
• Reloj de sincronía de entrada I2C en el pin SCL para la transferencia de datos.
• Preescalador que divide la frecuencia de reloj para el manejo del módulo I2C.
• Un filtro antirruido en cada pin SDA y SCL.
• Arbitraje entre el módulo I2C que interactúa con otro maestro.
• Lógica de generación de interrupciones.
• Lógica de generación de interrupciones FIFO, el acceso a los registros FIFO puede ser sincronizada a los datos de recepción y transmisión del módulo I2C.

El CPU escribe un dato en I2CDXR para transmitir y leer los datos recibidos en I2CDRR. Para la operación serial, estos registros están conectados a registros de corrimiento.

Generación de reloj

El generador de reloj utiliza una señal de reloj de CPU y la escala a través de los registros PLLCR, los divisores IPSC, ICLL e ICCH, para enviarla al bus I2C a través del pin SCL, como se observa en la figura 9.7.
El modulo I2C tiene cuatro modos básicos de operación que soporta la transferencia como maestro o como esclavo.

Como maestro: inicia la transmisión enviando una dirección para un dispositivo esclavo particular. Para recibir un dato del receptor, éste debe cambiar a modo maestro receptor.

Como esclavo: se inicia como esclavo receptor y envía un reconocimiento cuando reconoce su dirección. Si el maestro envía un dato, el módulo debe permanecer como esclavo receptor. Si el maestro hace la petición de un dato, el módulo debe cambiar a modo esclavo transmisor.

Es decir, que de las combinaciones anteriores se obtienen cuatro modos de operación:

- **Modo esclavo receptor:** el módulo se comporta como esclavo y recibe datos del maestro, todos los modos esclavos inician de esta forma. En este modo un dato serial es recibido en el pin SDA y es corrido hacia el interior sincronizado con el reloj generado por el maestro.
- **Modo esclavo transmisor:** transmite un dato al maestro, el esclavo puede pasar a este modo sólo del modo esclavo receptor. El esclavo entra en modo transmisor si el byte de dirección esclava es igual a su propia dirección en el registro I2COAR y el maestro tiene su bandera de transmisión R/W = 1.
- **Modo maestro receptor:** el maestro recibe datos del esclavo. En este modo se puede ingresar sólo después de que el maestro estuvo en modo transmisor. El módulo I2C entra en modo maestro receptor después de que el esclavo le transmitió la dirección y R/W = 1.
- **Modo maestro transmisor:** el módulo I2C transmite información de control y datos al esclavo. Todos los maestros inicializan en este modo.

9.4.2. Condiciones START y STOP

Estas condiciones pueden ser generadas por el módulo I2C cuando es configurado como esclavo. Las señales se pueden observar en figura 9.7.

- La condición de START es definida como una transición de alto a bajo en la línea SDA mientras SCL está en alto. El maestro maneja esta condición para indicar el inicio de la transferencia de datos.
- La condición STOP es definida como una transición de bajo a alto en la línea SDA mientras SCL está en alto. El maestro maneja esta condición para indicar el fin de la transferencia de datos.

Después de la condición de START y antes de la condición STOP subsecuente, el bus I2C es considerado ocupado y el bit BB del registro I2CSTR es puesto en uno. Después de un STOP el bus es considerado libre y BB = 0. Para el módulo I2C la transferencia de datos

inicia con la condición STA1RT, los bits MST (modo maestro) y condición de START STT de I2CMDR se deben poner en uno. La transmisión termina cuando el bit de condición de STOP STP se pone en uno.

Formato de datos seriales de siete bits

El módulo I2C soporta transferencia de datos de uno a ocho bits. Cada bit puesto en la línea SDA debe corresponder a un pulso de reloj SCL, los datos son transferidos enviando primero el bit más significativo MSb. El número de datos a ser transmitidos o recibidos son sin restricción. En la figura 9.7 se muestra el formato de señales SDA y SCL, donde se envía después del START una dirección de siete bits, un bit R/W, un bit de reconocimiento y un dato de ocho bits con un bit de reconocimiento. Si R/W = 0, el maestro envía una dirección hacia el esclavo; si R/W = 1, el maestro lee un dato del esclavo. Si un dispositivo I2C esclavo se identifica con la dirección, emite un bit de reconocimiento ACK y lee el siguiente byte de datos, después de que el número de bits "n" hayan sido transferidos, el receptor inserta un bit de reconocimiento ACK. "n" es un número contador de bits de uno a ocho, y se indica en el campo BC del registro I2CMDR.

9.4.3. Transmisión en formato libre

En este formato no se necesita el envío de dirección, el primer byte enviado después de la condición de START es un byte de dato, se inserta un bit de reconocimiento después de cada byte de dato o un dato de uno a ocho bits dependiendo de campo BC de I2CMDR. El transmisor y el receptor deben aceptar este formato de datos y la dirección destino del dato debe ser la misma durante la transferencia. Para la selección de este formato se pone un uno en el bit FDF (formato libre de dato) del I2CMDR.

9.4.4. Requerimiento de interrupciones

El módulo I2C puede generar el requerimiento de siete interrupciones, todas las interrupciones generadas son multiplexadas a través de un arbitro de prioridades para enviar un solo requerimiento al CPU. Cada interrupción tiene un bit o bandera asociado en el registro de estado I2CSTR y un bit de habilitación o máscara en el registro I2CIER.

Interrupciones de I2C:

- XRDYINT: condición de listo para transmitir. El registro de transmisión de datos (I2CDXR) está listo para aceptar un nuevo dato, ya que el dato previo ha sido copiado al registro de corrimiento I2CXSR.

- RRDYINT: condición de recepción lista. El registro receptor de datos I2CDRR, está listo para leerse, ya que un dato ha sido transferido del registro de corrimiento I2CRSR.

- ARDYINT: condición de acceso de registros lista. Los registros de I2C están listos para ser accesados porque los valores previos de dirección, dato y comandos han sido usados.
- NACKINT: condición de transferencia no reconocida. El módulo I2C configurado como maestro transmisor no recibió un reconocimiento del esclavo receptor.
- ALINT: pérdida de arbitraje. El módulo I2C ha perdido el contexto de arbitraje con otro maestro transmisor.
- SCDINT: detección de STOP. Se detectó una condición STOP en el bus.
- AASINT: condición esclavo. El I2C ha sido direccionado como dispositivo esclavo por otro maestro.

Las condiciones de interrupción anteriores también pueden verificarse encuestando los bits de configuración de I2C. También existen las interrupciones de FIFO, donde los registros FIFO de transmisión se pueden configurar para que interrumpan después de haber transmitido un número de bytes (hasta 16), los registros FIFO de recepción se pueden configurar para que interrumpan cuando hayan recibido un número de bytes (hasta 16).

Resumen

En este capítulo se han abordado varios puertos seriales de la familia C28x, como se puede observar, estos DSP manejan una gran cantidad de puertos de este tipo, que le permite al DSP, además de estar realizando operaciones de control y procesamiento, tener una amplia gama de posibilidades para interactuar con otros dispositivos de una manera versátil y económica.

Capítulo 10

Puerto serie multicanal buffereado

Los puertos serie multicanal buffereado (McBSP) proveen una comunicación directa a dispositivos tales como codecs, convertidores A/D y D/A con puerto serial, otros sistemas seriales, aplicaciones multiproceso a través del puerto serial multiplexado por división de tiempo (TDM), etc. Las familias F2823x, F2833x y F2806x pueden contener hasta dos puertos McBSP similares a los de otras familias de DSP [25].

En este capítulo se describen las características más importantes de los puertos McBSP, partiendo desde el modo de operación convencional, el puerto serie buffereado, operación en modo TDM, hasta su forma multicanal buffereada que involucra múltiples modos de operación.

10.1. Generalidades

El puerto McBSP se comunica con el CPU desde el mundo externo por medio de cuatro pines o señales, internamente maneja la información por los registros de 32 bits DRR(1,2) para la recepción y DXR(1,2) para la transmisión. Los datos pueden provenir de periféricos internos o externos y ser transferidos por el controlador de acceso directo a memoria (DMA).

Características

• Comunicación full-duplex.
• Doble-buffer para transmisión y triple-buffer para recepción, permitiendo el envío continuo de datos.
• Reloj independiente y sincronía de trama para recepción y transmisión.
• Envío de interrupciones al CPU o al controlador de DMA, tanto en recepción como en transmisión.
• 128 canales para transmisión y recepción.
• Modos de selección multicanal para habilitar o deshabilitar bloques de transferencia en cada canal.

• Interfaz directa a codecs, interfaces analógicas y dispositivos A/D y D/A.
• Soporta generador externo de reloj y señales de sincronía.
• Generador de razón de muestreo programable para la generación de reloj interno y señales de sincronía.
• Interfaz directa a:
– Tramas T1/E1.
– Dispositivos IOM-2.
– Dispositivos AC97.
– Dispositivos I2S.
– Dispositivos SPI.
• Tamaños de palabra: 8, 12, 16, 20, 24, y 32 bits
• Transmisión de bits en orden inverso: LSb primero.
• Justificación de la palabra a la izquierda o derecha con o sin extensión de signo.
• Utiliza ley μ y ley A para expansión en la recepción y compresión en la transmisión.
• Operación en una o dos partes, es decir, tramas compuestas de dos partes, una primera con un conjunto de palabras a una longitud y la segunda con otra cantidad de palabras de otra longitud.

Pines y señales de los puertos McBSP-A y McBSP-B

Pines:
MCLKRA/B: fuente de reloj I/O para recepción, también alimenta el generador de razón de muestreo.
MCLKXA/B: fuente de reloj I/O para transmisión, también alimenta el generador de razón de muestreo.
MDRA/B: pin de recepción serial (I).
MDXA/B: pin de transmisión serial (O).
MFSRA/B: sincronía de trama de recepción (I/O).
MFSXA/B: sincronía de trama de transmisión (I/O).

Señales:
MRINT: interrupción al CPU en la recepción.
MXINT: interrupción al CPU en la transmisión.
REVT: sincronización de eventos en la recepción con DMA.
XEVT: sincronización de eventos en la transmisión con DMA.

10.2. Compresión logarítmica

Algunas señales como la voz, se caracterizan por presentar con más frecuencia amplitudes pequeñas que amplitudes grandes ocasionando redundancia de información. En cuantización

PCM cada muestra de la señal es codificada independientemente de las otras muestras, es decir, que una cuantización uniforme produce un mismo espaciamiento entre niveles sucesivos a través del intervalo dinámico de la señal. Para una cuantización de L bits, una mejor aproximación consiste en hacer estos niveles más cerrados, es decir, menos espaciados en las amplitudes pequeñas y más espaciados en las amplitudes grandes. Esto produce una cuantización no uniforme del error, ocasionando un valor medio cuadrático menor del error. Una cuantización no uniforme es obtenida pasando la señal a través de un dispositivo no lineal que comprime la amplitud de la señal. Este tipo de codificador permite comprimir una señal de entrada de 13 bits a una señal de salida con intervalo dinámico de 7 bits. Para la codificación de señales de voz, en USA, Canadá y Japón se ha adoptado el estándar de ley μ, resultando una reducción de 24 db en la cuantización de la potencia del ruido relativo a la cuantización uniforme [3], [11]. En la reconstrucción de la señal se utiliza una relación logarítmica inversa para expandir la señal. La relación compresor-expansor se conoce como "compander".

El estándar para USA y Japón es llamado Ley μ está definido por la ecuación (10.1):

$$|y| = \frac{log(1+\mu|x|)}{log(1+\mu)} \tag{10.1}$$

donde:
|x|: magnitud de entrada
|y| : magnitud de salida
μ: parámetro seleccionado para las características de compresión igual a 255.

El estándar europeo es llamado Ley A y está definido por la ecuación (10.2):

$$|y| = \begin{cases} \frac{A|x|}{1+ln(A)} & \text{para } 0 \leq x \leq \frac{1}{A} \\ \frac{Ln(1+A|x|)}{ln(1+A)} & \text{para } \frac{1}{A} \leq x \leq 1 \end{cases} \tag{10.2}$$

donde A=87.6

Los puertos McBSP pueden ser configurados para expandir una palabra de 8 bits que venga comprimida bajo la ley μ o ley A, y luego el dato ingrese al CPU o se transfiera vía el módulo DMA expandido a 16b en complemento a dos. Para la transmisión, el dato a enviar puede comprimirse en cualquiera de estas dos leyes antes de enviarlo en formato a 8 bits. Para su selección se utilizan los bits XCOMPAND y RCOMPAND en los registros de control de transmisión y recepción.

10.3. Proceso de recepción de un puerto serie síncrono convencional

Los siguientes puntos describen cómo los datos que ingresan por el pin DR llegan al CPU o los maneja el DMA, las señales y registros se pueden observar en la figura 10.1, los protocolos de señales, tanto para la recepción como para la transmisión, se presentan en la figura 10.2, donde las señales FSR, FSX, DX y DR corresponden a los pines de la figura 10.1:

1. El McBSP espera un pulso de la señal de sincronía FSR.
2. Cuando arriba el pulso FSR, el McBSP introduce el retardo seleccionado en los bits RDATDLY del registro RCR2.
3. El puerto acepta los bits de datos en el pin DR, y va corriendo un bit por cada ciclo de reloj en el registro de corrimiento RSR1 si la palabra es de 16 bits o menor, en RSR(1,2) si la palabra es mayor de 16 bits.
4. Cuando una palabra completa es recibida, el puerto McBSP copia el dato de los registros RSR(1,2) a los registros buffer RBR(1,2) previendo que RBR no contenga un dato previo.
5. Si no se configuró la expansión, el dato es copiado en los registros DDR(1,2) sin expansión, de lo contrario se realiza la ley de expansión y se copia el dato expandido en DDR1. Cuando el nuevo dato está listo en DDR, se pone el bit de RRDY en el registro SPCR1. Esto indica que se tiene un dato listo.
6. El CPU o el controlador de DMA leen el dato recibido, el bit RDDY se limpia y una nueva transferencia RBR a DRR puede iniciarse.

10.3.1. Transmisión

1. La transmisión en el puerto McBSP inicia cuando el CPU o el controlador de DMA escribe un dato en los registros DXR(1,2), el bit XRDY del registro SPCR2 es limpiado para indicar la transmisión y que aún no está listo para aceptar otro dato. Tan pronto como el dato fue escrito en DXR, el mismo dato es copiado en el registro de corrimiento XSR. Si se habilitó el modo compresión, el dato se transfirió por la ley de compresión indicada.
2. Cuando el nuevo dato es escrito en DXR, el McBSP copia el contenido en el registro de corrimiento de transmisión XSR, y pone el bit XRDY en uno, esto indica que el puerto está listo para aceptar otro dato para transmitir.
3. El puerto McBSP espera un pulso de la señal de sincronía de transmisión FSX.

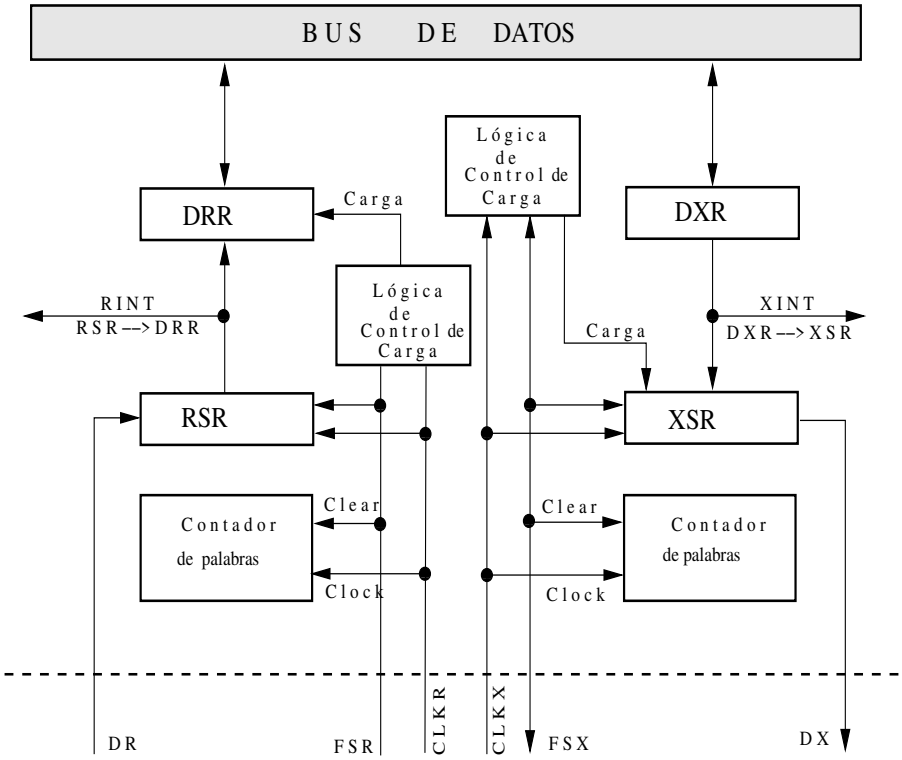


Figura 10.1. Diagrama del puerto serie asíncrono convencional

4. Cuando el pulso de FSX arriba, el McBSP inserta los retardos programados en los bits XDATDLY del registro XCR2.

5. El dato en el registro XSR se empieza a correr un bit por ciclo de reloj en el pin DX.

10.4. Puerto serie buffereado

Los puertos serie buffereados (BSP) permiten realizar interfaces directas con dispositivos externos como otros DSP, codecs y otros dispositivos seriales. Los BSP están basados en las interfaces de puertos serie estándar que se encuentran en los DSP de las familias C5x y C54x. Los puertos BSP constan de las siguientes características:

- Comunicación Full-Duplex.
- Registros de datos de doble buffer que permiten una comunicación de datos continua.
- Señalización de sincronía y reloj independientes para la recepción y transmisión.

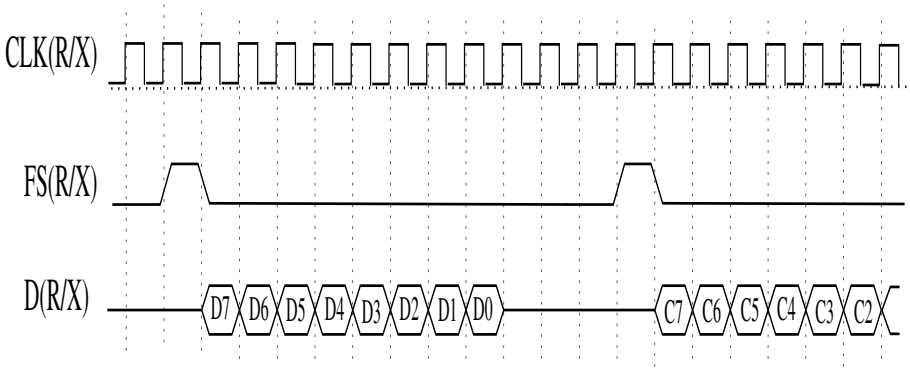


Figura 10.2. Diagrama de tiempos para un puerto serie asíncrono

- Canales separados para la recepción y transmisión que operan de manera independiente.

En la familia C28x este tipo de transferencia se implementa combinando el puerto McBSP con el módulo DMA

10.5. Puerto serie multiplexado por división de tiempo (TDM)

El puerto serie multiplexado por división de tiempo (TDM) permite una comunicación serial con otros siete DSP proveyendo una poderosa interfaz serial para aplicaciones multiproceso. Para la operación del puerto serie TDM se pone un uno en el bit TDM del registro de configuración de puerto serie TDM (TSPC). Si el bit TDM = 0, el puerto TDM puede operar como otro puerto serie convencional del DSP. El registro TSPC es similar al registro SPC del puerto serie a excepción de que el bit 0 (TDM) indica la operación de este puerto. Este tipo de puertos existe en las familias C5x y C54x, en la familia C28x el concepto de división de tiempo se traslada al dominio de los canales.

El concepto de división de tiempo significa que cada dispositivo toma una parte del tiempo disponible a compartir. Para la comunicación entre varios dispositivos a través del puerto TDM requiere juntar las líneas de transmisión de datos (TDX) y de recepción (TRD) en una línea simple llamada línea de datos (TDAT), para que los datos fluyan sobre esta línea. Similarmente, los relojes se unen en una línea de reloj TDM (figura 10.3). Una señal simple de trama es utilizada para indicar el inicio de un evento de 8 palabras TDM.

La línea TADD es manejada por un DSP en particular para un tiempo específico y determina qué dispositivo de la conexión TDM puede efectuar una recepción válida en el

tiempo de "slot". Todos los puertos TDM operan sincronizados por las líneas TCLK y TFRM, las cuales son generadas por cada dispositivo.

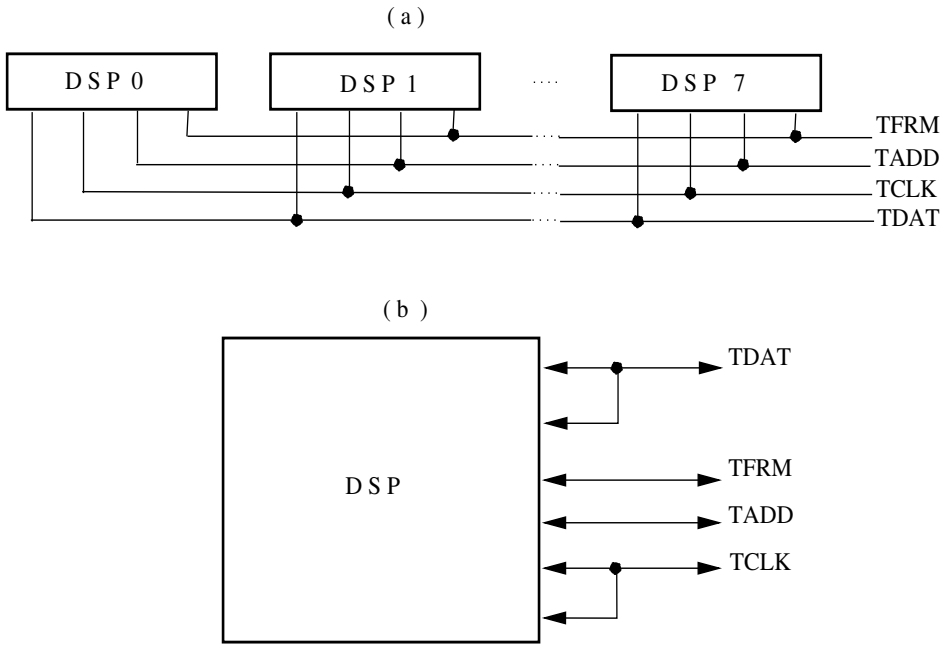


Figura 10.3. Conexión de DSP para operación multiproceso por puerto serie TDM

10.6. Puerto serie multicanal buffereado (McBSP)

Dependiendo de la familia, los DSP pueden tener varios puertos McBSP. Los puertos McBSP además de tener las características de los puertos serie estándares, BSP y TDM tienen otras potencialidades. En la figura 10.4 se tiene un diagrama general de un puerto McBSP, en la parte superior, prácticamente se está hablando de un puerto serie estándar, por lo que los pines de recepción y transmisión son el DR y DX, respectivamente; además, se conserva la misma nomenclatura para las señales de sincronía de trama de recepción FSR y FSX, y los relojes. Los demás elementos de la figura son parte de la potencialidad de los puertos McBSP, entre éstos están la sección de multicanal y la sincronización de eventos con la unidad de DMA. La señal CLKS es de reloj externo.

10.6.1. Registros

Los registros de los puertos McBSP de las familias F2823x, F2833x y F2806x están mapeados, como se muestra en las tablas 10.1 y 10.2

Tabla 10.1. Registros de los puertos McBSP A y B

McBSP A (dir. hex.)	McBSP B (dir. hex.)	Registro	Descripción del registro
5000	5040	DRR2	Recepción de datos 2
5001	5041	DRR1	Recepción de datos 1
5002	5042	DXR2	Transmisión de datos 2
5003	5043	DXR1	Transmisión de datos 1
5004	5044	SPCR2	Control 2
5005	5045	SPCR1	Control 1
5006	5046	RCR2	Control de recepción 2
5007	5047	RCR1	Control de recepción 1
5008	5048	XCR2	Control de transmisión 2
5009	5049	XCR1	Control de transmisión 1
500A	504A	SRGR2	Generador de razón de muestreo 2
500B	504B	SRGR1	Generador de razón de muestreo 1
500C	504C	MCR1	Multicanal 1
500D	504D	MCR2	Multicanal 2
500E	504E	RCERA	Habilita recepción de canal Partición A
500F	504F	RCERB	Habilita recepción de canal Partición B
5010	5050	XCERA	Habilita transmisión de canal Partición A
5011	5051	XCERB	Habilita transmisión de canal Partición B
5012	5052	PCR	Control de pines

Tabla 10.2. Registros de los puertos McBSP A y B (continuación)

McBSP A (dir. hex.)	McBSP B (dir. hex.)	Registro	Descripción del registro
5013	5053	RCERC	Habilita recepción de canal partición C
5014	5054	RCERD	Habilita recepción de canal partición D
5015	5055	XCERC	Habilita transmisión de canal partición C
5016	5056	XCERD	Habilita transmisión de canal partición D
5017	5057	RCERE	Habilita recepción de canal partición E
5018	5058	RCERF	Habilita recepción de canal partición F
5019	5059	XCERE	Habilita transmisión de canal partición E
501A	505A	XCERF	Habilita transmisión de canal partición F
501B	505B	RCERG	Habilita recepción de canal partición G
501C	505C	RCERH	Habilita recepción de canal partición H
501D	505D	XCERG	Habilita transmisión de canal partición G
501E	505E	XCERH	Habilita transmisión de canal partición H
5023	5063	MFFINT	Habilita interrupciones

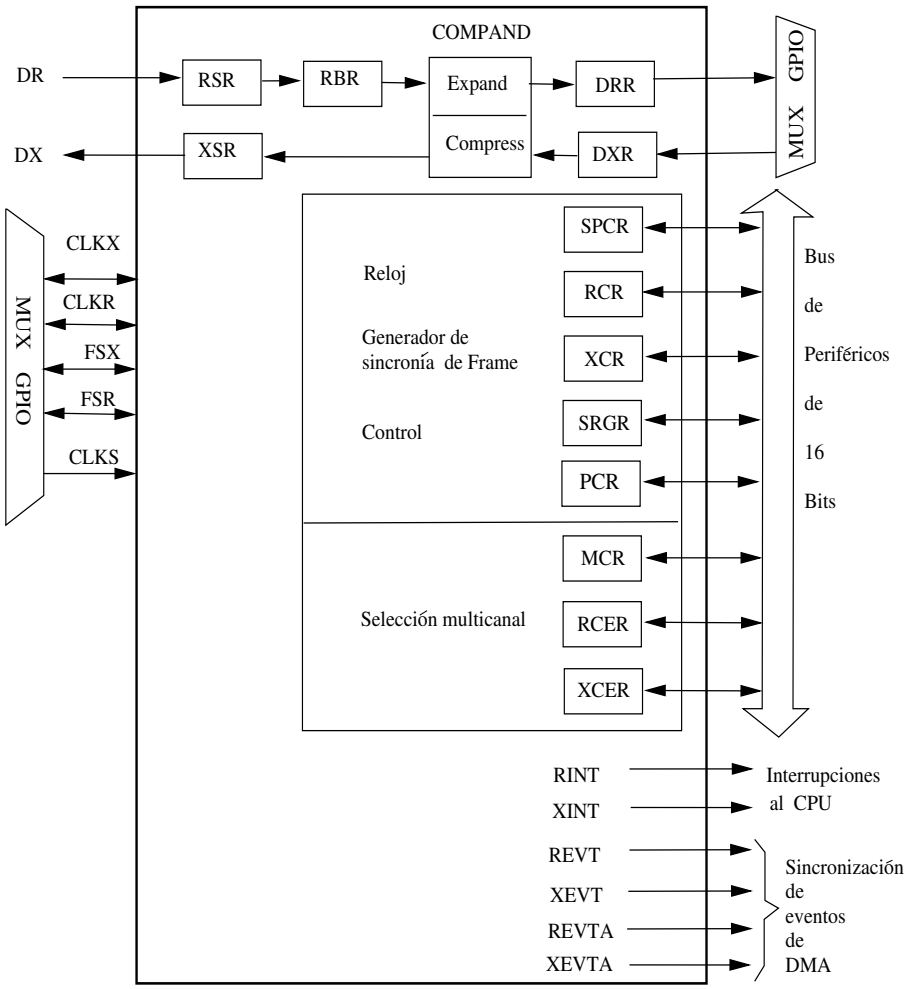


Figura 10.4. Diagrama de bloques del puerto McBSP

10.6.2. Configuración de puertos McBSP

La configuración de la forma de operar de los puertos McBSP se realiza a través de bits o campo de bits de sus registros. Los puertos McBSP se configuran a través de tres registros: dos de control SPCR(1,2) en tablas 10.3 y 10.4 respectivamente, y un registro de control de pines PCR.

Configuración como receptor

• Resetear el puerto.
• Comportamiento global:
– Los pines de recepción deben operar como pines del puerto McBSP, registro PCR.

– Deshabilitar el modo loopback, DLB en SPCR1.
– Habilitar/deshabilitar el modo de paro del reloj, CLKSTP en SPCR1.
– Habilitar/deshabilitar el modo de recepción multicanal, RMCM en MCR1.
• Comportamiento de datos:
– Seleccionar una o dos partes para la trama, RPHASE en RCR2.
– Longitud de palabra, RWDLEN1 en RCR1 y RWDLEN2 en RCR2.
– Longitud de trama, RFRLEN1 en RCR1 y RFRLEN1 en RCR2.
– Habilitar/deshabilitar la sincronía de trama, RFIG en RCR2.
– Modo compand, RCOMPAND en RCR2.
– Retardo de datos, RDATDLY en RCR2.
– Modo extensión de signo y justificación de datos, RJUST en RCR1.
– Modo de interrupción RINTM en RCR1.
• Comportamiento de sincronía de trama:
– Modo de sincronía de trama, FSRM en PCR.
– Polaridad de sincronía de trama, FSRP en PCR.
– Generador de razón de muestreo, período de sincronización (FPER) y ancho de pulso (FWID) .
• Comportamiento de reloj:
– Modo de reloj, CLKRM en PCR.
– Polaridad de reloj, CLKRP en PCR.
– Divisor de reloj en SRG, CLKGDV en SRGR1.
– Modo de sincronización de reloj SGR, GSYNC en SRGR2.
– Modo del reloj SRG, SCLKME en PCR.
– Polaridad del reloj SRG, CLKXP o CLKRP en PCR.
• Habilitar el puerto para recepción.

Configuración como transmisor

De forma similar a la recepción existen registros para configurar los modos de transmisión de los puertos McBSP.
• Comportamiento global:
– Configurar los pines de transmisión para operar como pines del puerto McBSP, registro PCR.
– Deshabilitar modo loopback, DLB en SPCR1.
– Habilitar/deshabilitar el modo de paro del reloj, CLKSTP en SPCR1.
– Habilitar/deshabilitar el modo de transmisión multicanal, XMCM en MCR2.
• Comportamiento de datos:
– Seleccionar una o dos fases para la trama, XPHASE en XCR2.
– Longitud de palabra, XWDLEN1 en XCR1 y XWDLEN2 en XCR2.
– Longitud de trama, XFRLEN1 en XCR1 y XFRLEN1 en XCR2.
– Habilitar/deshabilitar la sincronía de trama, XFIG en XCR2.

– Modo compand, XCOMPAND en XCR2.
– Retardo de datos, XDATDLY en XCR2.
– Poner el modo de retardo en DXENA de SPCR1.
– Modo de interrupción, XINT en SPCR2.
• Comportamiento de sincronía de trama.
– Modo de sincronía de trama, FSXM en PCR y FSGM y SRGR2.
– Polaridad de sincronía de trama, FSXP en PCR.
– Generador de razón de muestreo SRG, período de sincronización, FPER en SRGR2 y ancho de pulso, FWID en SRGR1.
• Comportamiento de reloj:
– Modo de reloj, CLKXM en PCR.
– Polaridad de reloj, CLKXP en PCR.
– Divisor de reloj en SRG, CLKGDV en SRGR1.
– Modo de sincronización de reloj SGR, GSYNC en SRGR2.
– Modo del reloj SRG, SCLKME en PCR.
– Polaridad del reloj SRG, CLKRP en PCR.
• Habilitar el puerto para transmisión.

Tabla 10.3. Registro de control de puerto serie SPCR1

Bit	Nombre	Función
15	DLB	1, Habilita modo digital loop back
14..13	RJUST	Modo de justificación de palabra y signo 00, Justificación a la derecha y llena con ceros los MSbs de DRR(1,2) 01, Justificación a la derecha y extensión de signo 10, Justificación a la izquierda y llena con ceros los LSBs de DRR(1,2) Reservado
12..11	CLKSTP	En modo SPI: relaciona los pines CLKXP y CLKRP 00, CLKXP=0 y CLKXP=0 el reloj se inicializa con flanco positivo sin retardo 01, CLKXP=1 y CLKXP=0 el reloj se inicializa con flanco negativo sin retardo 10, CLKXP=0 y CLKXP=1 el reloj se inicializa con flanco positivo con retardo 11 CLKXP=1 y CLKXP=1 el, reloj se inicializa con flanco negativo con retardo
10..8	Reservado	
7	DXENA	Habilita DX: 0, DX en OFF; 1, DX en ON
6	ABIS	1, habilita modo A-bis
5-4	RINTM	Modo de interrupción de recepción 00 RINT manejado por RRDY y fin de trama modo A-bis 01 RINT generado por fin de bloque o fin de trama en operación multicanal 10 RINT generado por una nueva trama de sincronía 11 RINT generado por RSYNCERR
3	RSYNCERR	Sincronización de error de recepción 0, No hubo error de sincronía 1, Sincronización de error detectado por el puerto McBSP
2	RFULL	Registro de corrimiento de recepción RSR(1,2) lleno 0 RBR(1,2) no hubo sobreescritura 1 DRR(1,2) no leído, RBR(1,2) lleno y RSR(1,2) lleno con nueva palabra
1	RRDY	1, Recepción lista con dato en DRR(1,2) El dato puede ser leído por el CPU o DMA
0	/RRST	0, Resetea recepción/transmisión 1, habilita la recepción/transmisión

Tabla 10.4. Registro de control de puerto serie SPCR2

Bit	Nombre	Función
15..10		Reservado
9	FREE	1, habilita modo corrida libre
8	SOFT	1, habilita modo SOFT
7	/FRST	Generador de reset de sincronía de trama 0, La señal de sincronía de trama FSG no es generada 1, La señal de sincronía de trama es generada después de (FPER+1) ciclos de reloj CLKG
6	/GRST	Generador de reset de razón de muestreo SRG 0, resetea al SRG 1, El CLKG es manejado por el valor en registro SRGR(1,2)
5..4	XINTM	Modo de interrupción de transmisión 00 XINT manejado por XRDY y fin de trama modo A-bis 01 XINT generado por fin de bloque o fin de trama en operación multicanal 10 XINT generado por una nueva estructura de sincronía 11 XINT generado por XSYNCERR
3	XSYNCERR	Sincronización de error de transmisión 0, No hubo error de sincronía 1, Sincronización de error detectado por el puerto McBSP
2	/XEMPTY	Registro de corrimiento de transmisión XSR(1,2) vacío 0 XSR(1,2) están vacíos 1 XSR(1,2) no están vacíos
1	XRDY	1, transmisión lista con dato nuevo en DXR(1,2) Un dato ha sido copiado de DXR(1,2) a XSR(1,2) y DXR(1,2) está listo para copiarle otro dato
0	/XRST	0, Resetea transmisión, 1, habilita la transmisión

10.6.3. Registros de control de transmisión y recepción RCR(1,2) y XCR(1,2)

Estos registros configuran varios parámetros para la operación de recepción y transmisión, se muestran en las tablas 10.5, 10.6, 10.7 y 10.8.

Tabla 10.5. Registro de control de recepción RCR1

Bit	Nombre	Función
15		Reservado
14..8	RFRLEN1	Longitud de trama de recepción multicanal 1 000 0000, una palabra por trama 000 0001, dos palabras por trama ... 111 1110, 127 palabras por trama 111 1111, 128 palabras por trama
7..5	RWDLEN1	Longitud de palabra de recepción 1 000, 8 bits 001, 12 010, 16 011, 20 100, 24 101, 32 11x, reservado
4..0		reservado

Tabla 10.6. Registro de control de recepción RCR2

Bit	Nombre	Función
15	RPHASE	Partes de recepción 0, trama de una parte 1, trama de dos partes
14..8	RFRLEN2	Longitud de trama de recepción multicanal 2 000 0000, una palabra por trama 000 0001, dos palabras por trama ... 111 1110, 127 palabras por trama 111 1111, 128 palabras por trama
7..	RWDLEN2	Longitud de palabra de recepción 2 000, 8 bits 001, 12 010, 16 011, 20 100, 24 101, 32 11x, reservado
4..3	RCOMPAND	Modo expansión de recepción 00, No expande, el dato recibido empieza con el MSb 01, No expande, el dato recibido empieza con el LSb 10, Expande usando ley μ para el dato recibido 11, Expande usando ley A para el dato recibido
2	RFIG	Ignora recepción de trama para modo continuo 0, recibe el pulso RFS después que un RFS restablece transferencia 1, ignora subsecuentes pulsos RFS después del primer RFS
1-0	RDATDLY	Retardo de dato recibido 00, 0 bits de retardo 01, un bit de retardo 10, dos bits de retardo 11, reservado

Tabla 10.7. Registro de control de transmisión XCR1

Bit	Nombre	Función
15		Reservado
14..8	XFRLEN1	Longitud de trama de transmisión multicanal 1 000 0000, una palabra por trama 000 0001, dos palabras por trama ... 111 1110, 127 palabras por trama 111 1111, 128 palabras por trama
7..5	XWDLEN1	Longitud de palabra de transmisión 1 000, 8 bits 001, 12 010, 16 011, 20 100, 24 101, 32 11x, reservado
4..0		reservado

Tabla 10.8. Registro de control de transmisión XCR2

Bit	Nombre	Función
15	XPHASE	Partes de transmisión 0, trama de una parte 1, trama de dos partes
14..8	XFRLEN2	Longitud de trama de transmisión multicanal 2 000 0000, una palabra por trama 000 0001, dos palabras por trama ... 111 1110, 127 palabras por trama 111 1111, 128 palabras por trama
7..5	XWDLEN2	Longitud de palabra de transmisión 2 000, 8 bits 001, 12 010, 16 011, 20 100, 24 101, 32 11x, reservado
4..3	XCOMPAND	Modo compresión de transmisión 00, No comprime, el dato transmitido empieza con el MSb 01, No comprime, el dato transmitido empieza con el LSb 10, Comprime usando ley μ para el dato transmitido 11, Comprime usando ley A para el dato transmitido
2	XFIG	Ignora transmisión de trama para modo continuo 0, transmite el pulso XFS después que un XFS restablece transferencia 1, recibe el pulso XFS, después ignora subsecuentes XFS
1..0	XDATDLY	Retardo de dato transmitido 00, 0 bits de retardo 01, un bit de retardo 10, dos bits de retardo 11, reservado

10.6.4. Registros generadores de razón de muestreo SRGR(1,2)

Cada puerto McBSP contiene un módulo generador de razón de muestro (SRG) que puede ser programado para generar un reloj interno CLKG y una señal de sincronía de trama FSG. El reloj CLKG puede ser utilizado para sincronizar los registros de corrimiento de recepción y transmisión y la señal FSG para la inicialización de las transferencias en los pines DR o DX.

El reloj de SRG puede ser alimentado de tres fuentes: LSPCLK o los relojes externos MCLKX o MCLKR. Estos se seleccionan en los bits PCR y CLKSM del registro de control SRGR2. En el caso de seleccionar como relojes externos MCLKX o MCLKR, se puede seleccionar su polaridad con los bits CLKXP y CLKRP del registro SRGR2. Los registros SRGR(1,2) controlan las características de operación del generador de razón de muestreo, éste se observa en la figura 10.5.

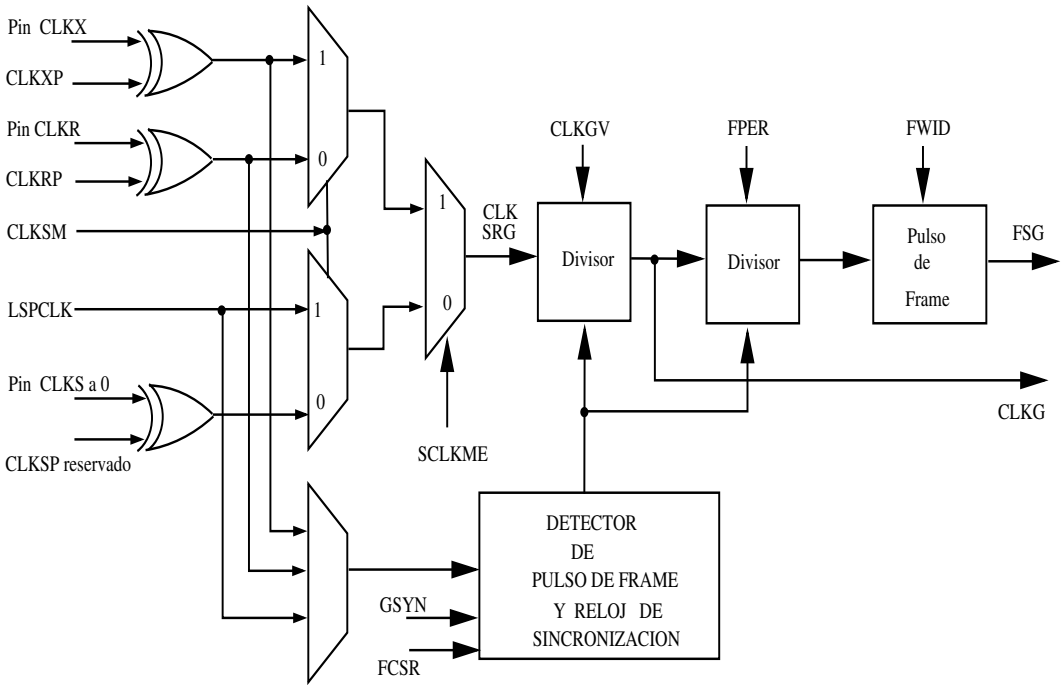


Figura 10.5. Generador de razón de muestreo SRG

La fuente de reloj seleccionada es dividida por los bits CLKGDV (1 a 255) del registro SRGR1 para producir CLKG. El divisor de período es el campo de bits FPER del registro SRGR1 que subdivide a la señal de reloj CLKG y controla el inicio de un pulso de trama hasta el inicio del nuevo pulso. El ancho del pulso de sincronía de trama se controla por el divisor FWID del registro SRGR1. Si se quiere transmitir utilizando la señal de sincronía de trama FSG, se debe poner el bit FSXM = 1 en el registro PCR y FSGM = 1 en SRGR2. Los bits de los registros SRGR1 y SRGR2 se describen en la tabla 10.9.

Tabla 10.9. Registros generadores de razón de muestreo SRGR1 y SRGR2

Bit	Nombre	**Registro** / Función
		Registro SRGR1
15..8	FWID	Ancho de trama. FWID+1 determina el ancho de pulso de sincronía FSG durante su período activo. Intervalo de 1 a 256 períodos de CLKG Es recomendado que FWID <WLEN(1,2)
7..0	CLKGDV	Divisor de CLK CPU o CLKS para obtener SRG
		Registro SRGR2
15	GSYNC	Sincronización de reloj CLKG 0, CLKG corre libremente 1, CLKG corre, pero es resincronizado y la señal de sincronía de trama FSG es generada sólo después de detectar la señal de sincronía de trama de recepción FSR
14		Reservado
13	CLKSM	Modo de CLKG, seleccionado en combinación de SCLKME SCLKME|CLKSM: 00, reservado SCLKME|CLKSM: 01, reloj interno LSPCLK SCLKME|CLKSM: 10, reloj externo por pin CLKR SCLKME|CLKSM: 11, reloj externo por pin CLKX
12	FSGM	Modo de sincronía de transmisión de trama de SRG 0, La señal FSX es debido a la copia DXR a XSR 1, FSX es manejada por la señal de sincronía de FSG
11..0	FPER	Período de trama. FPER+1 determina cuándo se activa la siguiente señal de sincronía. Intervalo 1 a 4096 períodos de CLKG

Los puertos McBSP pueden ser configurados para seleccionar múltiples canales independientes, tanto en la transmisión como para la recepción. Cada trama representa un flujo de datos multiplexados por división de tiempo (TDM) en un protocolo multicanal. El número de palabras por trama RFRLEN1 y XFRLEN1 indican el número de canales disponibles a seleccionar. Cuando se utiliza el concepto de flujo de datos en TDM, el CPU puede necesitar procesar algunos canales. De esta forma, la operación multicanal permite habilitar o seleccionar hasta 32 canales de 128 posibles.

En las familias F2823x, F2833x y F2806x, los 128 canales son divididos en ocho bloques de 16 canales cada uno y son asignados a las letras A-H, como se muestra en la tabla 10.10. En la tabla se observa la existencia de dos formas de asignación que depende del particionamiento deseado. Es posible que los canales A y B operen como receptores y de A-H como transmisores. Cada partición o bloque tiene un registro dedicado para habilitar los canales.

Tabla 10.10. Bloques y canales del puerto McBSP

Bloque	**Canales**	**Pares/impares**	**8 bloques**
0	0 ..15	A	A
1	16 .. 31	B	B
2	32 .. 47	A	C
3	48 .. 63	B	D
4	64 .. 79	A	E
5	80 .. 95	B	F
6	96 .. 111	A	G
7	112 .. 127	B	H

10.6.5. Operación multicanal

La operación multicanal de los puertos McBSP permite la operación TDM cuando se conectan con otros dispositivos similares, tanto para transmitir como para recibir.
Cada partición de canal tiene registros para la habilitación de canales dedicados. Para la selección del modo multicanal asegura:

- Operar a un bloque, RPHASE/XPHASE = 0.
- Escribir la longitud de trama en RFRLEN1/XFRLEN1 que incluya toda la cantidad de canales por utilizar considerando los canales intermedios no usados.
- Seleccionar el modo multicanal con bit RMCME=1, para recepción y XMCME=1 para transmisión.

Seleccionar el modo de dos u ocho particiones:

- En modo de dos particiones, seleccionar qué canales operarán como receptores con bit RPA(A/B)LK en registros RCERA/B.
- En modo de dos particiones, seleccionar qué canales operarán como transmisores con bit XPA(A/B)LK en los registros XCERA/B. En la figura 10.6 se observa la asignación temporal de particiones.
- Los registros de corrimiento tanto en la transmisión como en la recepción sólo operan sobre los canales activos, los canales deshabilitados no son tomados en cuenta.
- En modo de ocho particiones, habilitar los canales para recepción con los bits RCER en registro MCR1 y los bits XMCM para transmisión en el registro XCR2. En la figura 10.7 se observa la asignación temporal de particiones.
- En modo multicanal se puede emitir una interrupción por cada bloque de 16 canales transmitidos si XINTM = 01b, o en la recepción de 16 canales si RINTM = 01b.

En *recepción* opera de la siguiente forma, si un canal de recepción no es habilitado:

- RDDY no se fija en uno en la recepción del último bit de la palabra.
- RBR(1,2) no es copiado a DRR(1,2) en la recepción del último bit de la palabra, es decir que RDDY no se activa, esto también implica que no se generan interrupciones.

En *transmisión* opera de la siguiente forma, si un canal de transmisión no es habilitado:

- El pin DX permanece en estado de alta impedancia.
- No se transfiere DXR(1,2) a XSR(1,2) en la transmisión.
- /XEMPTY y XRDY no son afectados.

Registros para la operación multicanal

- Registros de control multicanal 1 y 2 MCR(1,2), tablas 10.11 y 10.12.
- Registros de habilitación de partición A y B para transmisión XCER(A,B).
- Registros de habilitación de partición A y B para recepción RCER(A,B).

El registro de control de pines PCR, se utiliza para configurar y controlar el funcionamiento de los pines externos y algunas señales internas del puerto McBSP, los bits de este registro se muestran en la figura 10.13.

Tabla 10.11. Registro de control multicanal MCR1

Bit	Nombre	Función
15..10		Reservado
9	RMCME	Habilitación de selección multicanal Opera en conjunto con XMCME RMCME = 0 y XMCME = 0, modo multicanal normal máximo 32 canales habilitados RMCME = 1 y XMCME = 1, modo de 128 canales habilitados
8..7	RPBBLK	Partición del bloque B para recepción 00, Bloque 1 canales 16 .. 31 01, Bloque 3 canales 48 .. 63 10, Bloque 5 canales 80 .. 96 11, Bloque 7 canales 112 .. 127
6..5	RPABLK	Partición del bloque A para recepción 00, Bloque 0 canales 0 .. 15 01, Bloque 2 canales 32 .. 47 10, Bloque 4 canales 64 .. 79 11, Bloque 6 canales 96 .. 111
4..2	RCBLK	Bloque actual recibido 000, Bloque 0 canales 0 .. 15 001, Bloque 1 canales 16 .. 31 010, Bloque 2 canales 32 .. 47 011, Bloque 3 canales 48 .. 63 100, Bloque 4 canales 64 .. 79 101, Bloque 5 canales 80 .. 95 110, Bloque 6 canales 96 .. 111 111, Bloque 7 canales 112 .. 127
1		Reservado
0	RMCM	Habilitación de recepción multicanal 0, habilita los 128 canales 1, deshabilita todos los canales por defecto Se habilitan por separado con RPBBLK o RPABLK y RCER(A/B)

Tabla 10.12. Registro de control multicanal MCR2

Bit	Nombre	Función
15..10		Reservado
9	XMCME	Habilitación de selección multicanal para transmisión Opera en conjunto con RMCME RMCME = 0 y XMCME = 0, modo multicanal normal máximo 32 canales habilitados RMCME = 1 y XMCME = 1, modo de 128 canales habilitados
8..7	XPBBLK	Partición del bloque B para transmisión 00, Bloque 1 canales 16 .. 31 01, Bloque 3 canales 48 .. 63 10, Bloque 5 canales 80 .. 95 11, Bloque 7 canales 112 .. 127
6..5	XPABLK	Partición del bloque A para transmisión 00, Bloque 0 canales 0 .. 15 01, Bloque 2 canales 32 .. 47 10, Bloque 4 canales 64 .. 79 11, Bloque 6 canales 96 .. 111
4..2	XCBLK	Bloque actual transmitido 000, Bloque 0 canales 0 .. 15 001, Bloque 1 canales 16 .. 31 010, Bloque 2 canales 32 .. 47 011, Bloque 3 canales 48 .. 63 100, Bloque 4 canales 64 .. 79 101, Bloque 5 canales 80 .. 95 110, Bloque 6 canales 96 .. 111 111, Bloque 7 canales 112 .. 127
1..0	XMCM	Habilitación de transmisión multicanal 00, habilita todos los canales sin máscara 01, deshabilita todos los canales por defecto y mascarados Se habilitan por separado con XPBBLK o RPABLK y XCER(A/B) 10, deshabilita todos los canales con máscara Se selecciona por separado con XPBBLK o RPABLK y XCER(A/B) estando mascarados 11, deshabilita todos los canales por defecto y mascarados Se habilitan por separado con XPBBLK o RPABLK y XCER(A/B)

Tabla 10.13. Registro de control de pines PCR

Bit	Nombre	Función
15-12		Reservado
11	FSXM	Modo de sincronía de trama de transmisión 0, sincronizado a señal externa 1, determinada por SRG, FSR es un pin de salida excepto cuando GSYNC = 1, en SRGR
10	FSRM	Modo de sincronía de trama de recepción 0, sincronizado a la señal externa, FSR pin de entrada 1, determinada internamente por SRG bit FSRGM en SRGR2
9	CLKXM	Modo del reloj de transmisión 0, es manejado por un reloj externo, CLK(R/X) es entrada 1, CLK(R/X) es un pin de salida y manejado por el SRG
8	CLKRM	Modo del reloj de recepción 0, es manejado por un reloj externo, CLKR es entrada 1, CLKR es un pin de salida y manejado por el SRG
7	SCLKME	Bit de selección de modo de reloj de muestro Opera en conjunto con CLKSM SCLKME|CLKSM 00, reservado SCLKME|CLKSM 01, reloj interno LSPCLK SCLKME|CLKSM 10, reloj externo en pin CLKR SCLKME|CLKSM 11, reloj externo en pin CLKX
6	CLKS_STAT	Refleja el estado del pin CLKS cuando es seleccionado como entrada de propósito general Reservado en algunas versiones
5	DX_STAT	Refleja el estado del pin DX cuando es seleccionado como salida de propósito general
4	DR_STAT	Refleja el estado del pin DR cuando es seleccionado como entrada de propósito general
3	FSXP	Polaridad de señal FSXP 0, FSXP activo alto 1, FSXP activo bajo
2	FSRP	Polaridad de señal FSRP 0, FSRP activo alto 1, FSRP activo bajo
1	CLKXP	Polaridad del reloj de transmisión 0, El dato es muestreado en el flanco positivo de CLKX 1, El dato es muestreado en el flanco negativo de CLKX
0	CLKRP	Polaridad del reloj de recepción 0, El dato es muestreado en el flanco positivo de CLKR 1, El dato es muestreado en el flanco negativo de CLKR

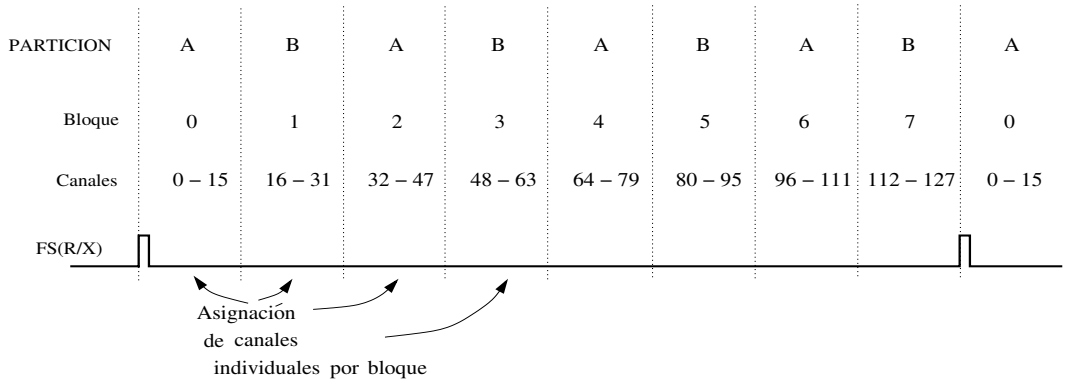


Figura 10.6. Operación multicanal del puerto McBSP en modo dos particiones

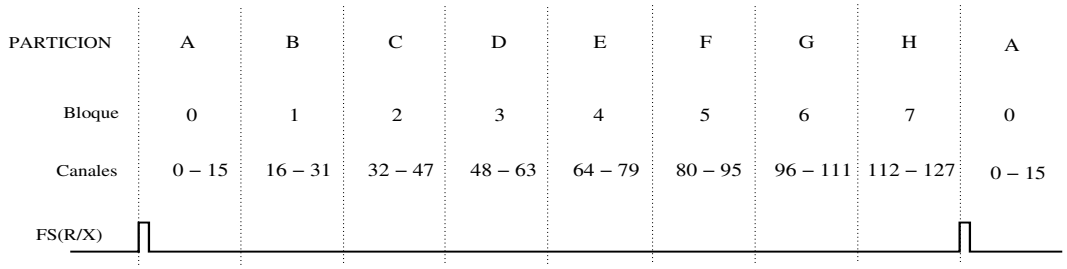


Figura 10.7. Operación multicanal del puerto MCBSP en modo de ocho particiones

Registros habilitadores de canales para recepción y transmisión

Los registros de habilitación de las particiones del canal A y B para la recepción RCER(A/B) y para la transmisión XCER(A/B), son utilizados para habilitar individualmente cada uno de los 32 canales de recepción y transmisión. Los 32 canales se dividen en bloque A y B de 16 cada uno y se habilitan poniendo en uno el bit respectivo, donde los canales se enumeran en orden ascendente del bit LSb al 0000 bit MSb.

Resumen

Se ha presentado una introducción al puerto serie multicanal McBSP, como se ha visto, éste constituye un puerto serial con múltiples posibilidades de operación, desde un puerto serial convencional, un puerto serial multiplexado por división de tiempo, un puerto serial buffereado hasta un puerto multicanal. Este tipo de puertos es usado ampliamente en las comunicaciones que demandan altos volúmenes de información y sobre todo cuando se quieren transmitir varios canales.

Tabla 10.14. Registros habilitadores de canales para recepción y transmisión (R/X)CER(A/B)

Bit	Nombre	Descripción
	Registro RCERA-RCERH	Habilitación de canales n:15,14,...,0 para recepción en particiones A..H
15..0		0, deshabilita canal n para recepción de la partición A..H 1, habilita canal n para recepción de la partición A..H
	Registro XCERA-XCERH	Habilitación de canales n:15,14,...,0 para transmisión en particiones A..H
15..0		0, deshabilita canal n para transmisión de la partición A..H 1, habilita canal n para transmisión de la partición A..H

Capítulo 11

Transferencia por DMA

El acceso directo a memoria (DMA) es una técnica de procesamiento en paralelo que permite controlar la transferencia de bloques de memoria sin la intervención del CPU. El DMA permite el movimiento de memoria a memoria, de memoria a periféricos internos o externos y viceversa, también puede realizar reordenamiento de datos. Este proceso trabaja en paralelo con el CPU y lo libera en tareas de movimiento o transferencia de grandes bloques de información. Como se observa en la figura 11.1, la transferencia por DMA se realiza a través de los buses del CPU; para este proceso un controlador de DMA se encarga de solicitar los buses al CPU para efectuar la operación de transferencia cuando sea necesario, una vez transferida la información, el DMA debe liberar los buses e indicarle al CPU la finalización de la transferencia.

En este capítulo se exponen los conceptos generales de DMA y sus características en los DSP C28x.

11.1. Generalidades de operación del DMA

- Lectura: un canal DMA lee un dato de una fuente que puede ser memoria o periférico localizados en memoria programa, dato o en el espacio I/O.

- Escritura: un canal de DMA escribe un dato que fue leído durante la transferencia de lectura a su destino en una localización en memoria. El destino puede ser memoria programa, dato, o un periférico en el espacio I/O.

- Transferencia de elemento: la combinación de lectura escritura transfiere un elemento, éste corresponde a una palabra de memoria.

- Transferencia de trama: cada canal de DMA tiene un número de elementos programables por cada trama, el DMA mueve todos los elementos en una trama.

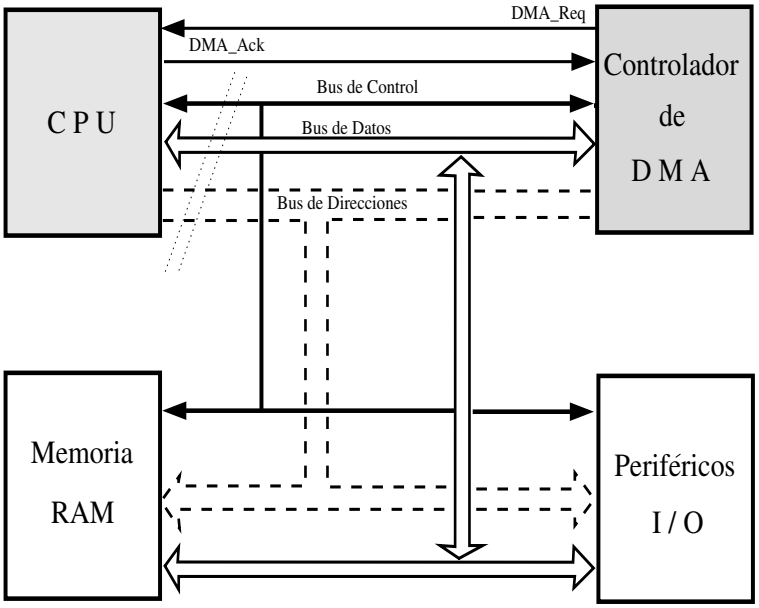


Figura 11.1. Diagrama de transferencia DMA

- Transferencia de bloques: en cada canal de DMA puede programarse un número de tramas por bloque, el DMA mueve el total de número de tramas definidos en el bloque.
- La máxima velocidad de operación es una palabra por ciclo del reloj.

En la figura 11.2 se observan los tiempos necesarios para la transferencia por DMA, en primera instancia el controlador DMA le envía una señal de requerimiento de DMA al CPU (HOLD_Req), cuatro ciclos de reloj después el CPU envía al DMA una señal de reconocimiento de DMA (HOLD_Ack) y libera los buses de datos y de direcciones para que el DMA tome el control de éstos y realice la transferencia. Cuando el controlador de DMA ha realizado la transferencia envía a un estado bajo la señal de HOLD_Req y de nuevo le devuelve el control de los buses al CPU.

Características de operación

- Operación en segundo plano: una vez programado el DMA, opera independiente y en paralelo con el CPU.
- Se pueden programar todos los canales independientemente.
- Transferencia multitrama: cada bloque a transferir se puede considerar como múltiples tramas.
- Programación de prioridades: los canales de DMA se pueden programar independientemente con prioridad alta o baja.

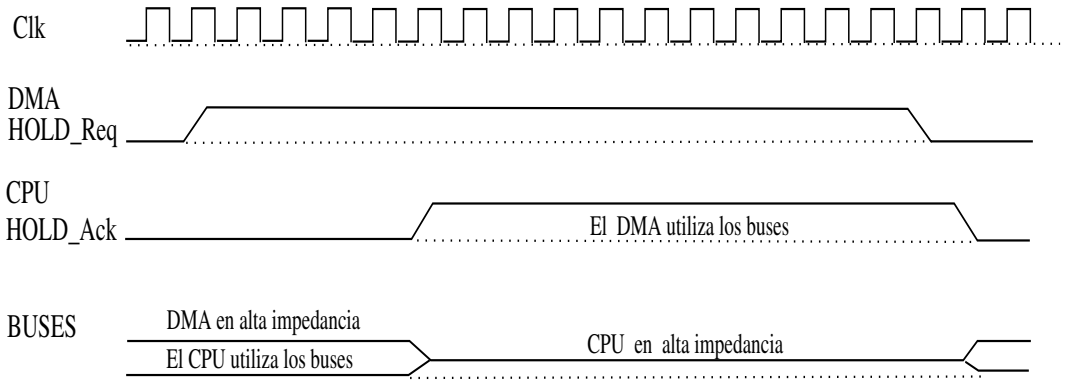


Figura 11.2. Diagrama de tiempos de DMA

- Generador de direcciones programables: cada registro de canal de dirección fuente y destino puede tener índices para cada transferencia de lectura y escritura.
- Intervalo de dirección completa: el DMA puede acceder a todas las direcciones extendidas en el DSP.
- Ancho programable de transferencia: cada canal puede configurarse independientemente para transferir una palabra en modo simple o una palabra doble (32 bits)
- Autoinicialización: una vez que un bloque fue transferido, el canal de DMA se puede configurar para que reinicialice la transferencia de un nuevo bloque.
- Sincronización de eventos: cada elemento transferido puede ser inicializado por eventos seleccionados de antemano.
- Generación de interrupciones: cuando se completa la transferencia de cada trama o un bloque, cada canal de DMA puede enviar una interrupción de DMA.

Operaciones básicas de DMA

En la figura 11.3 se muestran la tres operaciones básicas que realiza un controlador de DMA:

- Transferencia de un bloque de memoria fuente a uno destino: por cada ciclo de reloj de DMA se transfiere una palabra del bloque fuente al destino. Para el bloque fuente existe un registro apuntador que contiene la dirección del dato a transferir (dir_F) que se autoincrementa por cada palabra transferida. Para el bloque destino existe un registro apuntador (dir_D) a la localidad donde se va a escribir el dato transferido, este registro también se autoincrementa por cada transferencia.

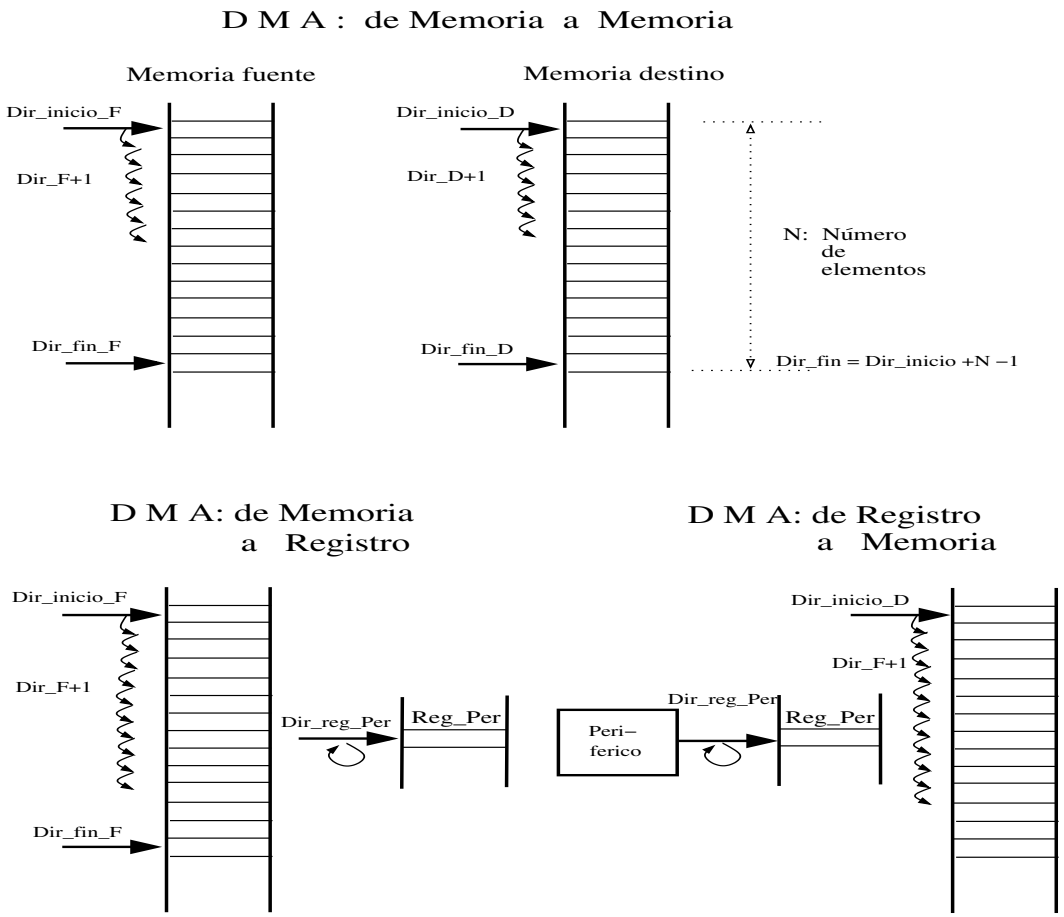


Figura 11.3. Modos básicos de transferencia por DMA

- Transferencia de memoria a registro: en este caso, el bloque fuente funciona similar al caso anterior, el destino es una localidad fija y normalmente corresponde al registro de transferencia a algún periférico, el periférico debe transferir de inmediato el dato recibido para evitar la sobreescritura.

- De periférico a memoria: similar al caso anterior, sólo que ahora se ha invertido el sentido de la transferencia.

En la figura 11.4 se muestra la transferencia de elementos y tramas entre bloques de memoria.

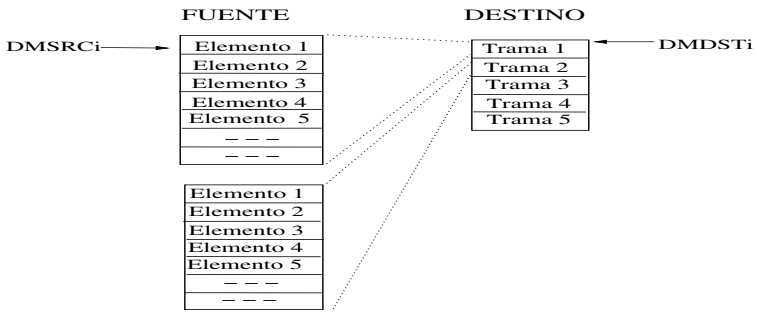


Figura 11.4. Transferencia de tramas

Operación y configuración de DMA

La operación y configuración de los canales de DMA se realizan escribiendo a los registros de control de DMA que normalmente operan en modo protegido. En la figura 11.5 se muestra un diagrama de flujo del procedimiento de configuración y operación de DMA, de esta figura se observa que la operación de DMA está ligada a las señales de la figura 11.2.

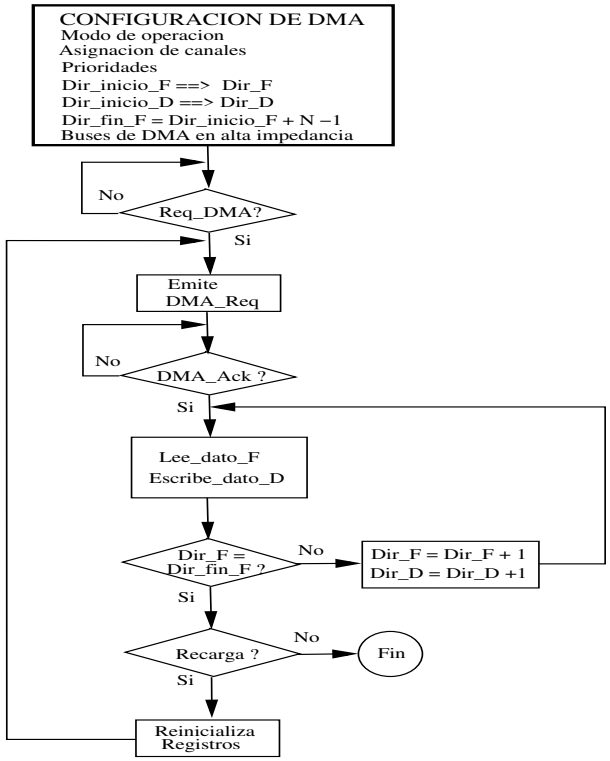


Figura 11.5. Diagrama de flujo de configuración y operación de DMA

11.2. Ejemplos de aplicación de DMA

En la figura 11.6 se muestra la forma de cómo transponer una matriz cuyos valores están ubicados en memoria. En la figura 11.7 se ilustra la forma de submuestrear una componente de color de una imagen a un formato 4 a 1, utilizando transferencia por DMA.

En el caso de transponer una matriz que está ordenada en memoria con las líneas consecutivamente, el resultado debe quedar en memoria ordenada por columnas. Como se observa en la figura 11.6, se necesita un apuntador a una localidad inicial destino Dir_inicio_D, el cual debe ir avanzando una localidad por dato. Por otro lado, en los datos fuente se necesita un apuntador a una localidad inicial fuente Dir_inicio_F, este apuntador deberá saltar de línea en línea, y en el caso particular de la figura 11.6, de cuatro en cuatro. Para una matriz de MxN, cuando complete todos los elementos de una columna debe retornar a la primera línea y siguiente columna. La forma de efectuar esta transferencia se puede configurar por DMA a través de sus registros.

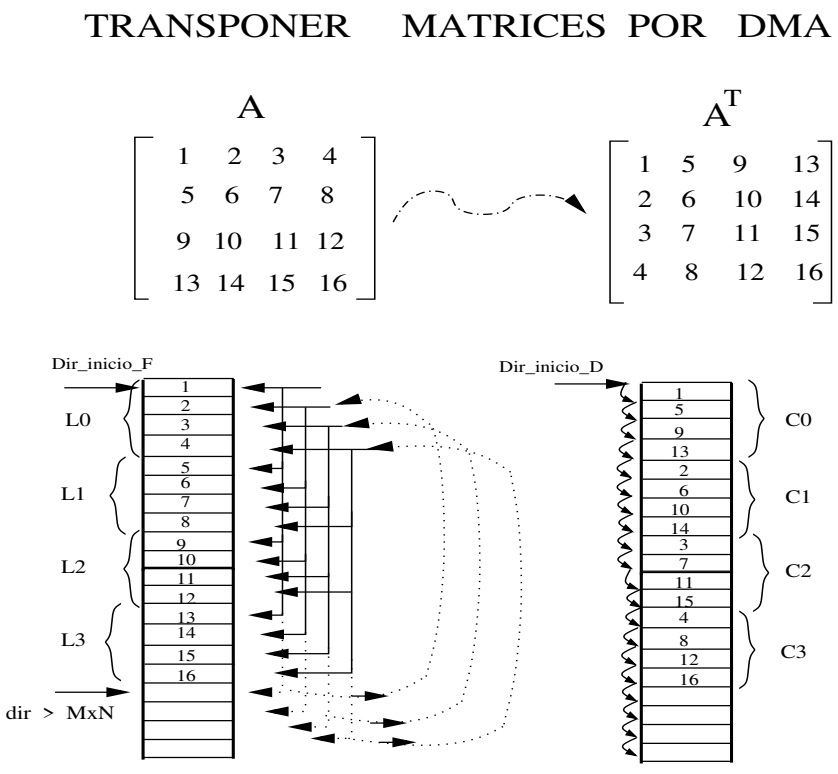


Figura 11.6. Transposición de matrices

En el proceso de submuestreo 4:1 de imágenes, se tiene una imagen de MxN y se requiere extraer únicamente los pixeles en color negro de la figura 11.7 y dejarlos en forma orde-

nada por líneas en localidades destino. El apuntador a los datos destino, como se muestra en la figura 11.7, debe irse incrementando en una localidad, si la imagen fuente de MxN está en memoria ordenada en forma de líneas, se apunta inicialmente por un apuntador Dir_inicio_F, el cual debe saltar cada dos localidades dentro de una línea par de longitud N, las líneas pares de la imagen deben saltarse y reiniciar el direccionamiento en líneas pares.

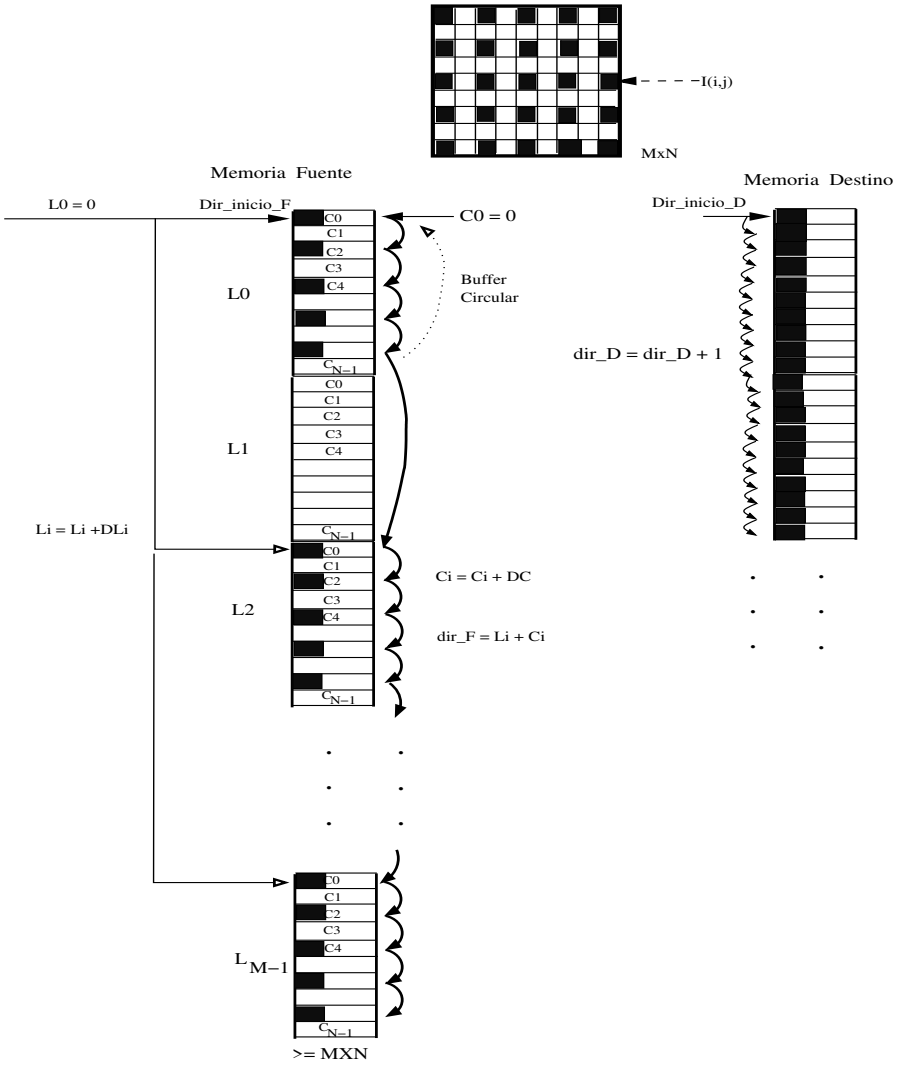


Figura 11.7. Submuestreo de imágenes 4:1

11.3. Módulo DMA del DSP C28x

El módulo DMA de la familia C28x lo encontramos en familias C2833x y F2833x y en algunos DSP de las familia F282xx y F2806x . Este módulo está basado en eventos de máquina, es decir, que requiere de interrupciones de periféricos para iniciar la transferencia. Estos DSP contienen seis canales de DMA que pueden ser configurados independientemente y cada uno contiene su propia interrupción por el módulo PIE. Sólo un canal puede ser configurado con alta prioridad y los otros cinco con prioridad baja [18], [25], [26], [35].

Características generales

• 6 canales con interrupción de PIE independiente.
• Fuentes de interrupción por periféricos:
- Secuenciador ADC 1 y 2.
- Puertos McBSP A o B, en transmisión o recepción.
- Interrupciones XINT1-7 y XINT13.
- Temporizadores de CPU.
- Señales ePWM1-6 ADCSOCA y ADSOCB.
- Por software.
• Datos fuente/destino:
- Bloques SARAM L4-L7 de 16Kx16bits.
- Todas las zonas XINTF.
- Registros resultados de ADC.
- Registros de transmisión y recepción de puertos McBSP A o B.
- Registros ePWM1-6/HRPWM1-6.
• Tamaño de palabra de 16 bits incluyendo los McBSP A o B.
• Cuatro ciclos por palabra (5 para lectura de McBSPs).

Fuentes de interrupción

En el campo de bits PERINTSEL del registro MODEn de cada canal "n", se selecciona la fuente de inicialización del canal DMA. La interrupción del periférico es guardada en el bit correspondiente PERINTFLG del registro de CONTROL, y si la interrupción del canal es habilitada (bit PERINTE del registro MODE y bit RUNSTS de CONTROL), entonces se debe dar la transferencia DMA. El bit PERINTFLG permanece pendiente hasta que la lógica de prioridad le permite atender al canal de DMA. Una vez atendido la bandera se limpia.

Si todos los canales tienen la misma prioridad, se acceden en forma circular en el orden: CH1 → CH2 → CH3 → CH4 → CH5 → CH6 → CH1 → CH2 → ..

Si se requiere un canal de mayor prioridad, se activa la opción y la operación de los canales queda:

CH1: con mayor prioridad.
Los otros canales operan con baja prioridad y son atendidos en forma circular:

CH2 → CH3 → CH4 → CH5 → CH6 → CH2 → ...

Bus de DMA

Consiste de un bus de 22 bits de dirección, un bus de lectura de datos de 32 bits y otro bus de escritura de datos de 32 bits.

Control de la transferencia

• El registro BURST_SIZE permite una transferencia máxima de 32 palabras en una trama.

• El registro TRANSFER_SIZE define cómo es transferida una trama entera.

• La interrupción puede configurarse para que ocurra al inicio o al fin de la transferencia, en el registro MODE bit CHINTMODE.

• El canal de DMA transfiere un bloque de datos cada vez que se recibe la interrupción de periférico.

• Al inicio de la transferencia, los registros sombra o de respaldo de cada apuntador son copiados en los registros respectivos activos. Durante el ciclo de transferencia, los registros activos SRC/DST_ADDR se les suma el valor de BUSRT_STEP.

• Al finalizar la transferencia, los registros apuntadores de dirección se pueden modificar por dos métodos: sumándoles un valor signado contenido en el registro SRC/DST _TRANSFER _STEP o recargados con los apuntadores activos. Cada canal contiene dos apuntadores sombra de retorno SRC_BEG_ADDR y DST_BEG_ADDR, permitiendo a la fuente y al destino retornar independientemente.

• Similar a los registros SRC_ADDR y DST_ADDR, los registros activos SRC/DST _BEG_ADDR son cargados por sus registros sombras al inicio de la transferencia.

11.3.1. Descripción de registros de DMA

Apuntadores fuente/destino (SRC/DST_ADDR)

Los valores escritos en los registros sombra son las direcciones inicio de la primera localización de datos para lectura o escritura. Al inicio de la transferencia, los registros sombra son copiados en los registros activos, estos registros mantienen la dirección actual apuntada.

Registros apuntadores fuente/destino de dirección inicio (SRC/DST_BEG_ADDR)

Son apuntadores de reinicialización, el valor escrito en sus respectivos registros sombra son cargados en los registros activos al inicio de la transferencia. En la condición de reinicialización, el registro activo será incrementado por el valor en el registro SRC/DST_WRAP_STEP previo al inicio de la carga en el registro SRC/DST_ADDR.

Control de transferencia

Por cada canal, el proceso de transferencia puede ser controlado por los siguientes valores:

- Tamaño de trama fuente/destino (BURST_SIZE):
  Especifica el número de palabras a ser transferidas. Este valor es cargado en el registro BURST_COUNT al inicio de cada trama. El registro BURST_COUNT decrece por cada palabra que es transferida, cuando alcanza cero, la trama está completa e indica que otro canal de DMA puede ser atendido.
  El comportamiento del canal es definido por el bit ONE_SHOT del registro MODE, el tamaño máximo de trama es determinado por el tipo de periférico. Para el ADC, el tamaño de trama puede ser de hasta 16 registros. Para el puerto McBSP, el tamaño de trama está limitado a uno debido a que solo contiene un registro FIFO.

- Tamaño de transferencia fuente/destino (TRANSFER_SIZE):
  Especifica el número de tramas a ser transferidas por interrupción de CPU. El registro TRANSFER_SIZE es cargado en el registro de conteo TRANSFER_COUNT al inicio de cada transferencia.

- Tamaño para reinicialización de transferencia fuente/destino (SRC/DST_WRAP_SIZE):
  Especifica el número de tramas a ser transferidas antes de que el apuntador de dirección retorne al inicio. Esto es utilizado para implementar funciones de direccionamiento circular. Este valor es cargado en el registro SRC/DST_WRAP_COUNT al inicio de la transferencia.

Por cada apuntador fuente/destino, el cambio de la dirección es controlada por los valores de salto:

- Paso de trama fuente/destino (SRC/DST_BURST_STEP):
  Con cada transferencia de trama, el paso del salto de la dirección fuente y destino son especificadas por estos registros. El valor puede ser positivo o negativo, para incrementos o decrementos. Si no se requiere que la dirección salte como en caso de los puertos McBSP, se escribe con cero.

- Paso de transferencia fuente/destino (SRC/DST_TRANSFER_STEP):
  Especifica la dirección offset del inicio de transferencia de próxima trama después de completar la transferencia de la trama actual. Esto se presenta cuando la localización de los datos están espaciados a intervalos constantes. El valor puede ser negativo o positivo.

- Paso de retorno fuente/destino (SRC/DST_WRAP_STEP):
  Cuando el contador de retorno alcanza cero, este valor especifica el número de palabras a sumar o restar del apuntador BEG_ADDR y se fija la nueva dirección inicio. Se utiliza para implementar direccionamiento circular. Puede ser positivo o negativo.

11.3.2. Modos de transferencia

Existen tres modos de control de la forma de transferencia: ya sea para ciclos de trama o ciclos de transferencia.

- Modo One Shot (ONESHOT):
  Si se habilita este modo cuando ocurre una interrupción de un evento, el DMA continuará su transferencia hasta que TRANSFER_COUNT llegue a cero. Si se deshabilita este modo, entonces se requiere una interrupción de un evento por cada transferencia de trama en el canal hasta que TRANSFER_COUNT llegue a cero.

- Modo continuo (CONTINUOUS):
  Si el modo continuo es deshabilitado, el bit RUNSTS en el registro de control es limpiado al final de la transferencia deshabilitando el canal de DMA. El canal se puede rehabilitar poniendo el bit RUN en uno en el registro de CONTROL antes que otra transferencia de canal suceda.

- Modo de interrupción de canal (CHINTMODE):
  Este modo es seleccionado cuando se requiere que una interrupción sea generada al inicio o final de una nueva transferencia.

11.3.3. Tabla de registros

En la tabla 11.1 se enumeran los registros generales del controlador de DMA y los registros correspondientes al canal uno. Todos los registros de configuración de DMA son protegidos. Para los canales restantes, la cantidad de registros y nombres son similares y están ubicados en las direcciones:
Canal 2: 1040h a 105Fh, canal 3: 1060h a 107Fh, canal 4: 1080h a 109h, canal 5: 10A0h a 10BFh y canal 6: 10C0h a 10DFh.

Tabla 11.1. Registro de configuración y control del canal uno de DMA

Dirección (h)	**Registro**	**Descripción**
	Registros generales	
1000	DMACTRL	Control
1001	DEBUGCTRL	Depuración
1002	REVISION	Revisión de periféricos
1003	Reservado	–
1004	PRIORITYCTRL1	Control de prioridad
1005	Reservado	–
1006	PRIORITYSTAT	Estado de prioridad
1007	Reservado	–
...	...	...
101F	Reservado	–
	Canal de DMA 1	
1020	MODE	Modo
1021	CONTROL	Control
1022	BURST_SIZE	Tamaño de trama
1023	BURST_COUNT	Contador de trama
1024	SRC_BURST_STEP	Tamaño de paso de trama fuente
1025	DST_BURST_STEP	Tamaño de paso de trama destino
1026	TRANSFER_SIZE	Tamaño de transferencia
1027	TRANSFER_COUNT	Contador de transferencia
1028	SRC_TRANSFER_STEP	Tamaño de paso de transferencia fuente
1029	DST_TRANSFER_STEP	Tamaño de paso de transferencia destino
102A	SRC_WRAP_SIZE	Tamaño de retorno fuente
102B	SRC_WRAP_COUNT	Contador de retorno fuente
102C	SRC_WRAP_STEP	Tamaño de paso de retorno
102D	DST_WRAP_SIZE	Tamaño de paso de retorno destino
102E	DST_WRAP_COUNT	Contador de retorno destino
102F	DST_WRAP_STEP	Tamaño de paso de retorno destino
1030	SRC_BEG_ADDR_SHADOW	Sombra de registro fuente
1032	SRC_ADDR_SHADOW	Dirección de apuntador actual
1034	SRC_BEG_ADDR	Dirección fuente de inicio activa
1036	SRC_ADDR	Apuntador de dirección fuente
1038	DST_BEG_ADDR_SHADOW	Sombra de registro destino
103A	DST_ADDR_SHADOW	Apuntador de dirección destino
103C	DST_BEG_ADDR	Dirección inicio de destino activo
103E	DST_ADDR	Apuntador de dirección destino
103F	Reservado	

Resumen

En este capítulo se ha expuesto brevemente la técnica de DMA, que en la actualidad existe en muchas aplicaciones como sistemas embebidos, comunicaciones, telefonía, multimedia y procesamiento de señales en general. Se sugiere al usuario tomar en consideración esta técnica para comprobar la eficiencia en el desempeño de sus aplicaciones.

Hasta este capítulo se ha expuesto la arquitectura y módulos más importanes de los DSP de la familia C28x, el capítulo siguiente es un complemento teórico para conocer más a fondo sobre cómo realizar aplicaciones utilizando formatos numéricos.

Apéndice A

Formatos numéricos

En cualquier sistema digital o una computadora, los números son representados como una combinación finita de números binarios o bits que toman valores cero y uno, lo que ocasiona errores por los efectos de precisión finita debido a los efectos de cuantización. Los bits son organizados en conjuntos de ocho llamados bytes o 16 bits llamados words. La representación más frecuente de los números en una computadora es a través de dos tipos de notación o formatos: *de punto fijo y punto flotante*.

Los conceptos vertidos en esta parte están directamente relacionados a resolver problemas de aritmética y realizar aplicaciones de PDS. Se analizan errores de precisión numérica, los formatos numéricos de punto fijo y punto flotante así como las operaciones básicas que se utilizan.

A.1. Errores numéricos

En el diseño e implementación de un sistema de procesamiento digital de señales (PDS) en una arquitectura dedicada como un procesador digital de señales (DSP), los algoritmos son implementados en un hardware digital en arquitecturas de longitud finita, por lo que se debe tener mucho cuidado en los errores aritméticos, por tanto, debe existir un compromiso entre el intervalo dinámico de las variables y la precisión de las mismas. Esto también es válido en procesadores de punto flotante, pero es específicamente más sensible para procesadores de punto fijo.

Cuando se efectúan operaciones aritméticas en un sistema de procesamiento digital de señales representado en formatos numéricos de precisión finita se tienen tres tipos de errores:

- Efecto de conversión de una señal analógica a digital.
- La representación de los coeficientes.
- El truncamiento o redondeo de los resultados cuando se almacenan después de las operaciones.

Los efectos de precisión numérica son no lineales y aleatorios, por lo que su análisis es muy complejo.

A.2. Formatos de punto fijo

En una arquitectura digital toda la información está escrita con ceros y unos, y cada quien podría interpretar la información de muchas maneras, dando lugar a diferentes formatos numéricos, sin embargo, existen formatos binarios ya establecidos de amplio uso.

Para **números enteros positivos** tenemos el formato:

- No signado: utiliza los L bits de una palabra digital para representar una magnitud. La interpretación de un número se realiza de acuerdo con la posición de los bits y los pesos binarios en potencia de dos. Al bit menos significativo (LSb) le corresponde el peso unitario (2^0) y al bit más significativo (MSb), el peso mayor 2^{L-1}. Cualquier número entero positivo X se puede representar por la suma de pesos binarios:

$$X = \sum_{n=0}^{L-1} 2^n b_n = b_0 + 2b_1 + 2^2 b_2 + 2^3 b_3 + \cdots + 2^{L-1} \tag{A.1}$$

  donde b_n representa el valor binario, cero o uno, en la posición n.

Para la representación de **números enteros positivos y negativos**, el bit MSb normalmente se utiliza como signo, y por convención, un cero en la posición MSb representa un número positivo, y un uno, un número negativo [7]. Los formatos utilizados para la representación de este tipo de números son:

- Signo magnitud (SM): el bit más significativo (MSb) es utilizado para representar el signo del número y el resto de bits representa la magnitud.
- Complemento a uno (C1): los números positivos se representan de forma similar al formato magnitud signo y los negativos en complemento a uno.
- Complemento a dos (C2): los números positivos se representan de forma similar al formato magnitud signo y los negativos en complemento a dos.

En los formatos de complemento, el bit MSb conserva la convención de signo cuando se representa un número negativo, esto se puede apreciar en la tabla A.1 para estos tres formatos.

De la tabla A.1, se observa que en los formatos SM y C1, existe la representación para el cero positivo y negativo, mientras que en el formato C2 sólo existe una representación para el cero, por otro lado, en los formatos SM y C1 no se puede representar el -8 y en C2 sí, por lo que el formato C2 optimiza la cantidad de bits que utiliza para representar al menos un número más [7]. Desde el punto de vista del hardware, cuando se opera aritméticamente en formato MS, se requieren dos unidades diferentes, una para la manipulación del signo

y otra para la magnitud, lo que hace más complejo el hardware, en aritmética en C2 se hace menos complicada la manipulación del signo. En formatos SM y C1, además existe el inconveniente de probar el valor cero con dos signos [8], [15]. Por tanto, la forma más común de realizar operaciones aritméticas por hardware es representar los datos en complemento a dos, debido a la eficiencia de los cálculos [10]. Sin embargo, se debe tener en consideración dos casos especiales, si complementamos en C2 un cero binario en L bits obtenemos el mismo resultado en L bits con acarreo en una posición a la izquierda del bit MSb que debe ignorarse. El segundo caso es cuando complementamos un número de L bits con un uno en la posición MSb y ceros en los *L-1* bit restantes, entonces obtenemos el mismo número, esta situación de complemento debe evitarse.

Tabla A.1. Representación de formatos binarios de punto fijo con $L = 4$ bits

Valor numérico	**Signo magnitud (SM)**	**Complemento a uno (C1)**	**Complemento a dos (C2)**
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	—
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1100	1001	1010
-7	1111	1000	1001
-8	—	—	1000

A.2.1. Representación de números fraccionarios en punto fijo

En las operaciones del PDS a nivel hardware se requiere la representación de números positivos y negativos con parte entera y fraccionaria, es decir, número reales, por lo que es necesario tener un formato que pueda representar este tipo de números. Existen diferentes notaciones para formatos de punto fijo cuando se representan números reales, sin embargo,

en una palabra de longitud L bits, utilizan el bit MSb para el signo, una cantidad de bits para la parte entera y el resto de bits para la parte fraccionaria. Existe un punto llamado *punto fijo o entero*, que separa la parte entera de la fraccionaria, este punto está ubicado a la derecha del bit de posición con peso binario unitario, éste punto es hipotético y no utiliza ningún recurso de hardware para su representación y manipulación, es responsabilidad del diseñador o programador saber donde está ubicado.

A.2.2. Formatos numéricos de punto fijo Qi

La representación numérica en formato digital de punto fijo es similar a la representación de números decimales, es decir, como un conjunto de dígitos con un punto decimal. En este formato la cantidad total de bits de la palabra digital se particiona en:

- QE: número de bits para la parte entera
- $QF = Qi$: los bits para la parte fraccionaria
- S: un bit de signo

es decir, que

$$L = 1 + QE + QF \tag{A.2}$$

Como se muestra en la figura A.1, entre la parte entera y la parte fraccionaria existe un *punto hipotético o punto fijo*, que sólo lo interpreta el programador.

Un número X positivo se puede representar en formato de punto entero como:

$$X = \sum_{i=-QE}^{QF} b_{-i} r^{-i} = b_{QE} r^{QE} + b_{QE-1} r^{QE-1} \cdots, b_1 r^1 + b_0 r^0 + \cdots b_{QF+1} r^{-QF+1} + b_{QF} r^{-QF} \tag{A.3}$$

donde:

$-Q_F \leq b_i \leq Q_E$
bi: representa el dígito binario "0" o "1" en la posición correspondiente
r: la base igual a 2 para el caso binario.

En general cualquier número real X en punto fijo en complemento a dos para $L = QE + QF + 1$ se puede escribir para $QF \neq 0$ [1]:

$$X = (-1)^S b(QE + QF) + \sum_{i=1}^{QE} b(QF - 1 + i) 2^{i-1} + \sum_{i=1}^{QF} b(QF - i) 2^{-i} \tag{A.4}$$

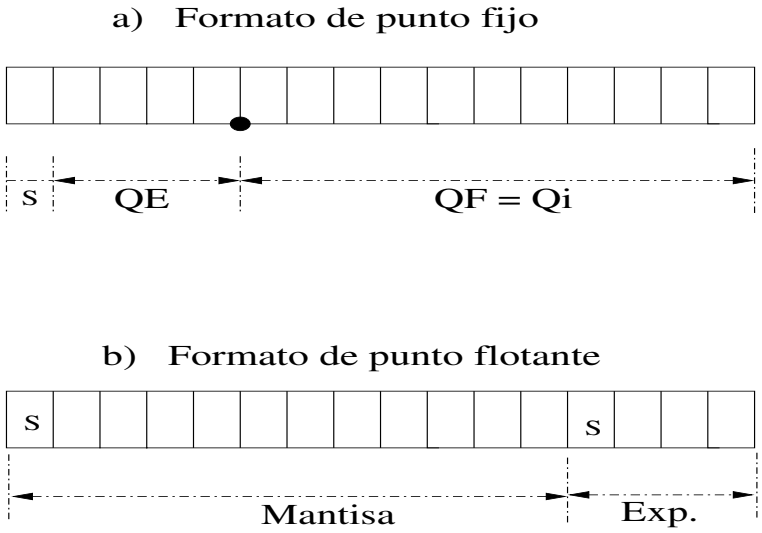


Figura A.1. Formatos numéricos

Recordando que el bit más significativo (MSb) corresponde al signo, y está en la posición $L-1 = QE + QF$, $S = 0$ para números positivos, $S = 1$ para números negativos y $b(.)$ es un número binario 0 o 1 en la posición correspondiente. En la ecuación A.4 las posiciones binarias $b(.)$ están reflejadas a la forma usual de referirse a una palabra digital de longitud L, es decir, $b_{L-1}, b_{L-1}, b_{L-1}, \cdots, b_1, b_0$

Ejemplos

Interpretación de números binarios en formato en punto fijo en complemento a dos

```
0101.1001  = 0x2^3+1x2^2+0x2^1+1x2^0 +1x2^{-1}+ 0x2^{-2}+ 0x2^{-3}
             + 1x2^{-4}  = 5.5625
1110.1100  = -1x2^3+1x2^2+1x2^1+0x2^0 +1x2^{-1}+ 1x2^{-2}+ 0x2^{-3}
             + 0x2^{-4} = -1.25
```

En los formatos de representación numérica es importante tener presente algunos parámetros del sistema para lograr mejores desempeños, estos son, *el intervalo dinámico de las variables, la precisión numérica y la resolución*:

- *El intervalo dinámico (ID)* de un sistema numérico se define como la diferencia entre el número mayor y el número menor que se pueda expresar en el formato, es decir la magnitud numérica más grande a representar. Cualquier número fuera de este intervalo se considera como sobreflujo. El ID en decibeles está definido como la razón entre el

número mayor y el menor (excluyendo al cero) y se expresa [7]

$$ID_{db} = 20log_{10}(\frac{Max}{Min}) \tag{A.5}$$

Por ejemplo, en un formato de punto fijo a 16 bits su intervalo dinámico es 90 db.

- *La precisión numérica (p)*, es el número real más pequeño a representar, se define como la diferencia entre dos números consecutivos y está determinado por el bit menos significativo (LSb). En los dos ejemplos anteriores la precisión numérica está dada por $2^{-4} = 0.0625$.
- *Resolución (Δ)*, si con L bits se puede representar una variable X con un valor máximo y un mínimo, entonces la resolución es [10].

$$\Delta = \frac{Xmax - Xmin}{2^L - 1} \tag{A.6}$$

Por lo tanto, en formato de punto entero debe existir un compromiso entre la precisión y el intervalo dinámico de los números o variables a representar. Es decir, si el intervalo dinámico es muy grande, la precisión no es muy buena y viceversa. Una característica del formato en punto entero es que su resolución es fija y el incremento de la resolución es proporcional al incremento del intervalo dinámico.

Comúnmente el formato de punto fijo más utilizado es el formato llamado Qi, que se refiere a la cantidad de bits para la parte fraccionaria, donde i significa el número de bits para la parte fraccionaria.

A.2.3. Intervalos dinámicos y precisión numérica

Para el formato Qi, el intervalo dinámico (ID) se puede escribir en función de los valores QE y QF como:

$$-2^{QE} < ID < 2^{QE} - 2^{-QF} \tag{A.7}$$

y la precisión:

$$p = 2^{-QF} \tag{A.8}$$

En la tabla A.2 se observa que para $L = 16$ bits, el formato $Q0$ tiene el mayor ID pero con menor precisión, por el contrario el formato $Q15$ tiene la mejor precisión numérica con el menor ID.

En aplicaciones con DSP de 16 bits utilizando formato de punto fijo es muy usual utilizar el formato $Q15$ debido a su precisión, por lo que es necesario normalizar todas la variables, constantes y números para utilizar este formato. Esto se realiza dividiendo cualquier valor entre el máximo número de un conjunto de valores, para que todos los datos queden entre -1 y 0.9999, por tanto el formato $Q15$ es muy utilizado en DSP.

Tabla A.2. Intervalo dinámico y precisión numérica de formatos Qi con $L = 16$ bits

Formato Qi	**Mínimo**	**Máximo**	**Precisión**
Q15	-1	0.9999694	0.0000305175
Q14	-2	1.9999389	0.0000610351
Q12	-8	7.9997558	0.0002441140
Q8	-128	127.96093	0.0039062500
Q4	-2,048	2047.9375	0.0625000000
Q1	-16,384	16,383.5	0.5000000000
Q0	-32,768	32,767	1.0000000000

En la tabla A.3 se muestran los valores para el intervalo dinámico y la precisión numérica, los cálculos se realizan de manera similar al caso de $L = 32$ bits, donde puede observarse que para este ancho de palabra podemos manejar un gran intervalo dinámico y una mejor precisión numérica, esto nos da un mejor balance en el formato que elijamos para nuestras variables. En la tabla A.3 se observa la gran ventaja de manejar anchos de palabra de 32 bits, por estas razones es probable que en muchas aplicaciones actuales algunas arquitecturas de procesadores como los ARM sean tan exitosas y otras marcas estén evolucionando a 32 bits. Como se ha mencionado en este trabajo, las familias de DSP C28x son arquitecturas que operan a $L = 16$ y 32 bits.

Tabla A.3. Intervalo dinámico y precisión numérica de formatos Qi con $L = 32$ bits

Formato Qi	**Mínimo**	**Máximo**	**Precisión**
Q31	-1	0.999999999	0.00000000046
Q30	-2	1.999999999	0.000000001
Q28	-8	7.999999996	0.000000004
Q24	-128	127.999999940	0.000000060
Q20	-2048	2047.999999046	0.000000954
Q16	-32768	32767.999984741	0.000015259
Q12	-524288	524287.999755859	0.000244141
Q8	-8388608	8388607.99609375	0.003906250
Q4	-134217728	134217727.937500	0.062500000
Q1	-1073741824	1073741824.50000	0.500000000
Q0	-2147483648	2147483647.00000	1.000000000

A.2.4. Asignación de variables

En la asignación de variables, el programador debe estar consciente del formato de representación para que en las operaciones de cargar o salvar variables sea coherente y reproduzca el formato deseado. Es por eso que en la mayoría de los DSP existen varias unidades de corrimientos que permiten flexibilizar estas operaciones.

En la figura A.2 se observan dos variables X e Y en formatos Qix y Qiy respectivamente, si se escribe X en la localidad de Y y se requiere representarla en el formato de Y, entonces hay que efectuar los corrimientos $Qix - Qiy$ a la derecha y por lo tanto tenemos más bits para representar la parte entera, pero existe un truncamiento en los bit LSb, por lo tanto existe una pérdida de precisión, pero un aumento en el intervalo dinámico de X.
Si $Qix > Qiy$, entonces:

$$Y = X\text{shift-R}(Qix - Qiy) = X \gg (Qix - Qiy) = X \cdot 2^{-(Qix-Qiy)}$$

Cuando se realizan corrimientos a la derecha se debe tener cuidado en conservar el signo del dato o variable.

El caso contrario es cuando la variable Y se quiere escribir con el mismo formato de X, en este caso se tiene un problema de sobreflujo, si la parte entera de Y no cabe en el formato que tiene la variable X, entonces:

$$X = Y\text{shift-L}(Qix - Qiy) = Y \ll (Qix - Qiy) = X \cdot 2^{(Qix-Qiy)}$$

Cuando se dispone de un número limitado de bits L, la representación numérica de una variable o constante fraccionaria ocasiona errores de precisión, es decir, que en la escritura en memoria del número se debe recortar a L bits, lo que ocasiona un truncamiento.

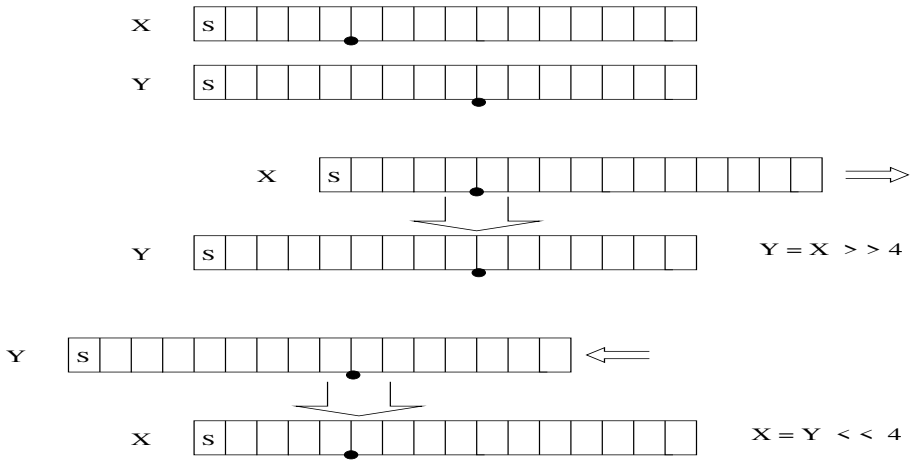


Figura A.2. Asignación numérica en formato de punto fijo

Truncamiento y redondeo

Cuando se representa una variable que pertenece a los números reales en un formato numérico binario de precisión finita, no se puede representar toda la parte decimal de la variable, es decir, que los decimales menos significativos pueden perderse en la representación. En este proceso la pérdida de precisión se puede dar a través de dos formas: por truncamiento y redondeo.

- **Truncamiento:** un número real en precisión infinita es convertido a formato de precisión finita utilizando los bits disponibles, es decir, que existe un recorte de los decimales menos significativos a representar.
- **Redondeo:** se le agrega un "1" binario en una posición más allá del bit LSb a preservar y luego se trunca. Si el bit adicional es "1" entonces el bit LSb se incrementará en uno, si el bit adicional es "0", entonces el redondeo genera el mismo efecto que el truncamiento.

Los procesos de truncamiento y redondeo se pueden emplear en asignación de variables y después de efectuar operaciones aritméticas para guardar los resultados en memoria. El proceso de redondeo es más preciso, pero requiere más operaciones.

A.2.5. Operación suma en punto fijo

Para efectuar una suma entre dos variables representadas en punto entero se efectúan los corrimientos necesarios en una o las dos variables para alinear el punto entero y posteriormente efectuar la suma, es decir, tener las dos variables en el mismo formato Qi. Obviamente hay que tener cuidado en el formato de representación de las variables y del resultado de la suma para evitar el sobreflujo. En la figura A.3 se realiza la suma entre dos variables X e Y en su respectivo formato Qi y L bits, donde el registro resultado es de $2L$ bits. En el primer caso, la suma se realiza $Y + X >> (Qix - Qiy)$ y el resultado queda en Qiy; y en el segundo caso, $X + Y << (Qix - Qiy)$ y el resultado queda en Qix.

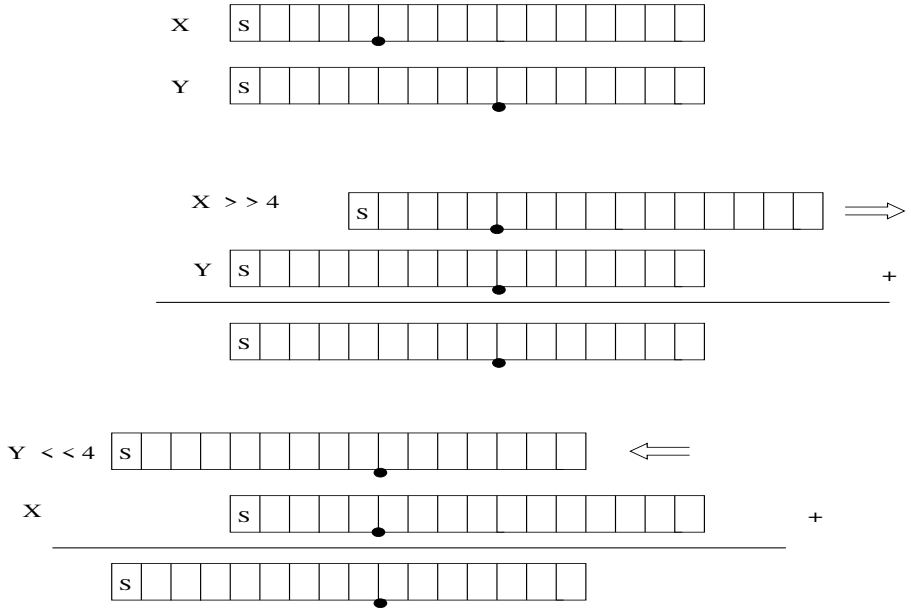


Figura A.3. Suma en aritmética de punto fijo

A.2.6. Operación de multiplicación en punto fijo

Cuando se efectúa una multiplicación de dos variables representadas en $L = 16$ bits, el resultado queda en $2L$ bits. Si se multiplican dos valores con formato Qix y Qiy, el resultado queda en $Qix + Qiy$, además se generan dos bits de signo (ver figura A.4).

En la figura A.4 se ha realizado la multiplicación entre la variable X en $Q11$ y la variable Y en $Q7$, ambas con la misma longitud de palabra L, el resultado de la multiplicación queda en $Q18$ con doble signo S, para efectos de salvar el resultado a memoria se puede eliminar un signo, salvar la parte alta en longitud de palabra L y el resultado quedaría en formato $Q2$. En algunos DSP existe una forma de programación de la salida del multiplicador para correr un bit a la izquierda y eliminar un bit de signo, y así ajustar la salida a formato $Q15$, o también se tiene la posibilidad de un corrimiento de cuatro bits a la izquierda para trabajar en formato $Q12$ [16].

En el siguiente ejemplo se realizan dos multiplicaciones en forma manual, una entre dos números reales positivos y la otra entre un positivo y un negativo. En el primer caso, se realiza la multiplicación de forma similar al procedimiento de la multiplicación entre números reales, y al final se ajusta la suma de posiciones decimales de ambos números. En el segundo caso, el número negativo es el multiplicador, el procedimiento se realiza en forma similar a la multiplicación entre positivos, salvo que el último sumando que corresponde al bit de signo del multiplicando se complementa a dos. Por facilidad se utiliza $L = 4$ bits.

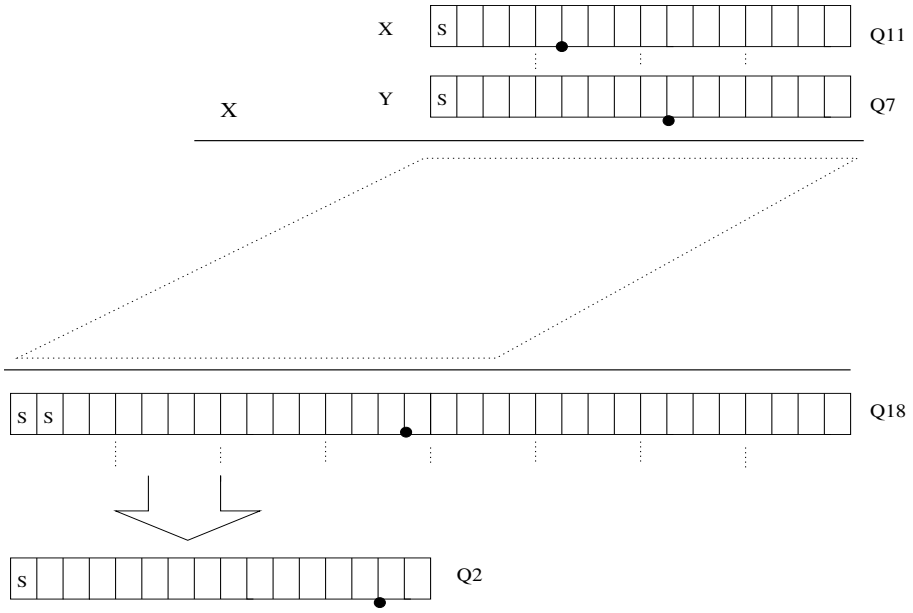


Figura A.4. Multiplicación en aritmética de punto fijo

```
L = 4 bits
A = 2.5  en Q1        010.1          A = 2.5  en Q1          010.1
B = 1.75 en Q2      x 01.11          B =-1.75 en Q2 C2     x 10.01
                  ------------                            ------------
                        0101                                    0101
                       0101                                    0000
                      0101                                    0000
                  +   0000                                +   1011   (C2 de A)
                  -------------                           ---------------
                      0100011          Ax(-B)=-AxB =-4.375     1011101
      AxB=4.375 =   0100.011           Ax(-B) en C2= 4.375     0100.011
```

Existen varias posibilidades de efectuar la multiplicación entre números binarios en aritmética de punto entero dependiendo si son positivos o negativos. La multiplicación binaria entre números positivos se realiza de forma similar a la multiplicación en números decimales.

Algoritmo por hardware de la multiplicación de dos números positivos $M1$ de L bits y el multiplicador M2 de L bits es:

1. Al inicio se llena con ceros un registro de acumulación A de longitud $2L$.

2. Enseguida se verifican los bits LSb del multiplicador $M2$, si es uno entonces $A =$

$A + M1 << L$, de lo contrario $A = 0$. El corrimiento en L bits de $M1$ a la izquierda significa que las sumas se realizan en la parte alta del registro A.

3. Se realiza un corrimiento lógico de A a la derecha, se verifica el siguiente bit LSb de $M2$, si es "uno" entonces $A = A + M1$, de lo contrario $A = 0$.

4. Se repite el paso anterior hasta que se verifique el bit MSb de $M2$.

5. El resultado queda en A con longitud $2L$ bits.

Cuando algún número o ambos son negativos, se pueden convertir a positivos, operarlos como positivos y para el resultado se aplica ley de signos. Esto complica el hardware por lo que se requieren mejores algoritmos o hardware que realicen todas las operaciones. El algoritmo de Booth toma en consideración todas las combinaciones positivo negativo [15].

La división entre números binarios necesita repeticiones de restas y desplazamientos. En los DSP de las familias C5x y C28x existe la instrucción SUBC, que significa restas condicionales y permite realizar divisiones en punto entero [16], [18].

En resumen, cuando se realizan operaciones aritméticas en punto entero, es necesario hacer una simulación para determinar los valores máximos y mínimos que toman los resultados de la aplicación para determinar la dinámica de las variables y elegir los formatos numéricos adecuados. La aritmética de punto entero es la más utilizada en DSP debido a su velocidad de operación y su economía.

A.3. Formato numérico de punto flotante

El formato de punto flotante hace uso del concepto de la notación científica para incrementar el intervalo dinámico. La representación y operación de números en formato de punto flotante es importante en problemas donde el intervalo dinámico de las variables excede la capacidad de su representación en punto entero, ya que en este formato no se pueden representar números muy grandes o números fraccionarios muy pequeños. El procesador en punto entero puede ejecutar más rápido las operaciones para la misma precisión, sin embargo, su intervalo dinámico es pequeño [8].

El formato numérico de punto flotante puede ser empleado para cubrir un amplio intervalo dinámico, sin embargo, la resolución decrece con un incremento en el tamaño de los números sucesivos, es decir, que la distancia entre números sucesivos de punto flotante se incrementa, la resolución variable resulta en un gran intervalo dinámico [10]. Es decir, que los números que se representan en este formato no están espaciados homogéneamente a lo largo de una línea recta real, ya que están más cercanos entre sí cuando se aproximan a cero y más separados cuando se alejan del origen.

En formato de punto flotante, el número a representar se escala para obtener la representación de la mantisa en rango completo, esto se hace incrementando o decrementando el

valor del exponente. El intervalo dinámico del formato está dado por el tamaño del exponente, y la precisión está dada por el número de bits para representar a la mantisa [7].

La representación numérica de punto flotante también es llamada notación real y cualquier número X puede expresarse como $X = M2^E$ donde:

M: es la mantisa

E: el exponente

Ambos se expresan en forma binaria tal como se tiene en la figura A.1.b, es decir, si se tiene un ancho de palabra L, se asignan b_M bits para la mantisa con su signo y b_E bits para el exponente que puede ser negativo o positivo, entonces $L = b_M + b_E$, la mantisa es representada en forma fraccionaria normalizada en el intervalo $0 \leq M < 1$.

Para una longitud de palabra L dada, la representación de un número de punto flotante tiene un mayor intervalo dinámico que para su representación en punto entero con la misma L, debido a que el exponente escala el valor de punto flotante a un valor muy grande o muy pequeño. Sin embargo, como el exponente utiliza parte de L el resto de bits se utilizan para la mantisa que determina la precisión.

El diseñador del formato decide la ubicación de la mantisa y el exponente dentro de la palabra digital, o su representación en varias palabras, comúnmente se utiliza la proporción $b_M = 3L/4$. El número de bits para el exponente determinan el intervalo dinámico y el número de bits de la mantisa, la precisión.

Algunos formatos de punto flotante utilizan un bit implícito que es igual a uno, debido a que en la normalización de un número positivo siempre se tendrá un uno que representa el peso más significativo del número. Esto consiste en que un número expresado en una palabra binaria de longitud L se hacen los corrimientos necesarios del número para obtener la mantisa normalizada, entre $1/2 \leq M < 1$ y esos corrimientos se toman en cuenta para escribirlos en el campo del exponente, en un número el primer uno más significativo siempre va a existir, por lo que pasa a ser el bit implícito. La utilización del bit implícito para la representación o almacenamiento del dato permite aumentar la precisión en un bit, sin embargo, cuando el dato guardado se lee y se va a operar este bit se tiene que hacer explícito, ya que es el más significativo.

Ejemplo de un número normalizado positivo $L = 8$:

```
     X  =  28.79546    En binario      0001 1100 . 1100 1011 1010 (1C.CBAh)
     Normalizando X para que  mantisa 0 < M < 0.99999999
    Xn = M  = 000.1110 0110 0101 1101   e = 5 (0101)
     Extrayendo el bit     implícito    e = 4
    Xn  = 0001.1100 1100 1011 1010      =  1.m,  1 < 1.m < 1.999999
           __/  \__________    _______
 Bit implícito              \/
                           mantisa
```

El exponente que resulta del número de corrimientos al normalizar la mantisa se representa con un desplazamiento, ya que si se representara en complemento a dos, los exponentes negativos aparecerían siendo mayores que los exponentes positivos. Con la representación de desplazamiento para el estándar IEEE 754 en precisión simple, el exponente más negativo queda en 0000 0001 y el más positivo en 1111 1110, donde el desplazamiento utilizado es 127 [5].

A.3.1. Suma de punto flotante

Para efectuar la suma entre dos variables representadas en formato de punto flotante se requiere que los exponentes sean iguales, es decir, se debe hacer un ajuste de uno de los exponentes. Esto se realiza por el corrimiento de la mantisa del número más pequeño a la derecha compensando el incremento de su exponente, en este proceso de reajuste del exponente existe una pérdida de precisión. Es decir, la representación numérica de punto flotante provee una resolución fina para números pequeños, pero para números grandes la resolución es burda, en contraste con el formato en punto entero que provee una resolución uniforme a través de todo el intervalo de valores para un mismo Qi.

Ejemplo:
Dados dos números X e Y de punto flotante:
si $X = (-1)^{sx} X_M 2^{Ex}$
$Y = (-1)^{sy} Y_M 2^{Ey}$ y si $E_y > E_x$
entonces, para efectuar la suma de punto flotante se deben ajustar las mantisas de los operandos para obtener un mismo exponente, esto se logra desplazando a la derecha el punto del número más pequeño con lo que se incrementa su exponente o desplazando a la izquierda el punto del número más grande (disminuyendo su exponente). Sin embargo, para preservar la precisión numérica, el desplazamiento se realiza sobre el número con exponente menor (E_x) al exponente mayor (E_y), es decir, que se tiene que efectuar una comparación previa de los exponentes, entonces

$$W = [(-1)^{sy} Y_M + (-1)^{sx} X_M >> (E_y - E_x)] 2^{Ey} \tag{A.9}$$

renormalizar la mantisa de W y ajustar su exponente, el resultado W queda de punto flotante.

A.3.2. Multiplicación de punto flotante

En este caso se multiplican las mantisas y los exponentes se suman. Puede existir sobreflujo cuando la suma de los exponentes excede el intervalo dinámico de la representación en punto entero del exponente.

Ejemplo:
Multiplicar los número X e Y de punto flotante:

$$W = [(-1)^{sx} X_M \cdot (-1)^{sy} Y_M] 2^{E_X + Ey} \tag{A.10}$$

después se debe renormalizar el resultado.

A.3.3. División de punto flotante

Se dividen las mantisas normalizadas y se restan los exponentes. En este caso también puede haber problemas en la representación del exponente del resultado.

Ejemplo:
Dividir X/Y

$$W = \frac{(-1)^{sx} X_M}{(-1)^{sy} Y_M} 2^{E_X - E_y} \tag{A.11}$$

después se debe renormalizar el resultado.

Para maximizar el desempeño de las implementaciones a 32 bits en los DSP C28x, Texas Instruments ha desarrollado un conjunto de librerías optimizadas llamadas *IQmath* para operaciones en punto entero a 32 bits programando en lenguaje C/C++ [34]. Estas librerías simplifican el diseño y desarrollo de tareas del PDS, su implementación parece ser como si estuviera trabajando de punto flotante, pero realmente se trabaja en punto entero, por tanto, algunas veces se le llama punto flotante virtual.

A.4. Formatos estándares de punto flotante

En realidad no existe una norma estricta en cuanto a la asignación de bits para la mantisa y el exponente, ni la posición dentro del ancho de palabra, incluso el programador del hardware puede crear su propio formato. Sin embargo, existen ciertos estándares de instituciones y compañías de gran prestigio, y obviamente, estos formatos pueden presentar sus propias ventajas.

A.4.1. Formato de punto flotante estándar IEEE 754

Es un estándar en la mayoría de aplicaciones científicas y de ingeniería, como se muestra en la figura A.5. IEEE especifica cuatro formatos de punto flotante [5]:

- Precisión simple a 32 bits
- Precisión simple extendida $\geq$43 bits. Se utilizan para cálculos intermedios para evitar que los resultados finales de las operaciones se deterioren debido a errores de redondeo. Manejan más bits para el exponente y la mantisa.

- Precisión doble a 64 bits
- Precisión doble extendida $\geq$79 bits

Formato de punto flotante IEEE de precisión simple

Este formato es ampliamente utilizado en máquinas de 32 bits, donde un número de punto flotante con este formato se representa:

$$X = (-1)^S M 2^{E-127}$$

donde $0.M$ es una fracción y $1.M$ es un número mixto con un bit para la parte entera (bit implícito) y 23 bits para la parte fraccionaria. El intervalo dinámico de esta representación es de:

$$\pm 2^{-126} \quad a \quad \pm\,(2 - 2^{-23}2^{127}) \qquad o \qquad \pm\,1.18x10^{-38} \quad a \quad \pm\,3.40x10^{38}$$

y su precisión $p = 2^{-23}$ es decir, que cualquier número fuera de este intervalo resulta en sobreflujo.

Ejemplos de conversión de números a formato IEEE de precisión simple :

```
Número         3.14159     signo positivo
Binario        11.001001000011111001111
Normalizado    1.10010010000111111001111       exponente = 1
signo positivo s = 0
E = 1 + 127 = 129 = 1000 0000
m = 10010010000111111001111100000001101110000
Número convertido  s E m   a   32 bits
            0  1000 0000  10010010000111111001111
Agrupando en dígitos hexadecimales a 32 bits
            0100 0000 0100 1001 0000 1111 1100 1111  (40490FC0h)

Para -3.14159 sólo se cambia el bit de signo a 1, entonces
            1100 0000 0100 1001 0000 1111 1100 1111  (C0490FCFh)

Número      0.000018965    positivo
Binario     0.00000000000000010011111000101110000011100000010 1
Normalizado 1.00111110001011100000111000000101010 1  e = -16
Signo positivo  s= 0
E = -16 + 127 = 111 decimal = 01101111 b
Número convertido  s E m   a   32 bits
          0  01101111  00111110001011100000111000000101010 1
Agrupando en dígitos hexadecimales
```

```
0011 0111 1001 1111 0001 0111 0000 0111  (379F1707h)
```

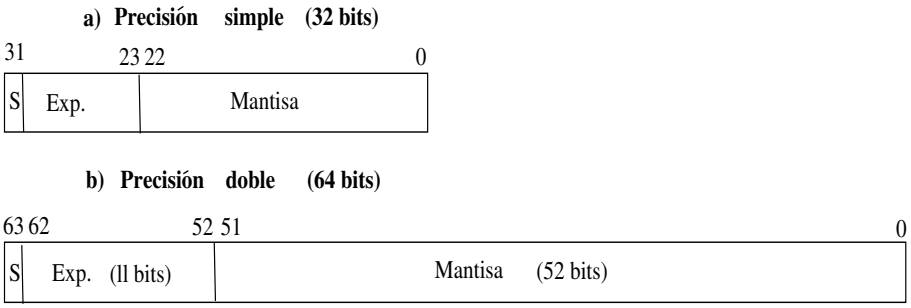


Figura A.5. Formatos numéricos de punto flotante, IEEE 754

El formato IEEE tiene la siguiente interpretación y casos especiales [10]:

```
Si E=255 y M != 0 , entonces X no es un número     (NAN)
Si E=255 y M = 0 , entonces X = (-1)^s(infinito),  (X = infinito)
Si 0 < E < 255,  entonces X = (-1)^s(1.M)2^(E-127)
Si E=0 y M != 0, entonces X=(-1)^s (0.M)2^(E-126), (X aprox 0)
Si E=0 y M = 0,  entonces X=(-1)^s(0),             (X = 0)
```

El formato de precisión doble es similar sólo que para la mantisa se utilizan 52 bits más el bit implícito entre los bits 51 y 52, el exponente se representa en 11 bits con desplazamiento aditivo de 1023. Con intervalo dinámico de 2^{-1022} a 2^{1024} y precisión de 2^{-52}

A.5. Formato de punto flotante de Microsoft

En este formato el exponente tiene una desviación aditiva (offset) de 129, por lo tanto para obtener el exponente verdadero, se debe restar al exponente de 129. El signo de la mantisa está en el bit 23 y la mantisa tiene un bit implícito entre los bits 22 y 23 en precisión simple [4].

Cualquier número X se representa en precisión simple

$$X = (-1)^S M 2^{E-129}$$

Para precisión doble en este formato la mantisa utiliza 52 bits, el exponente se representa en 11 bits con desplazamiento aditivo de 1025, como se muestra en la figura A.6.

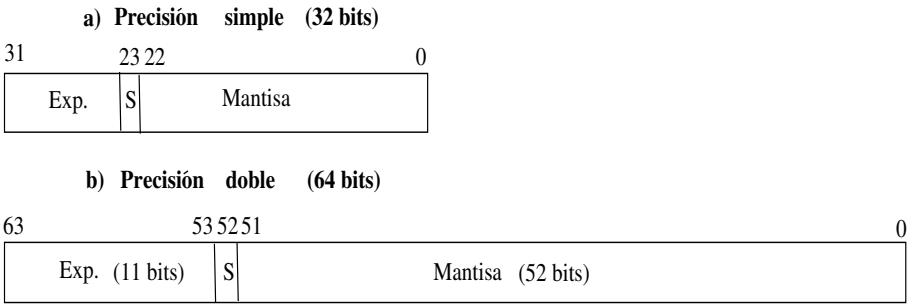


Figura A.6. Formatos numéricos de punto flotante de Microsoft

A.6. Otros formatos

La compañía Intel diseñó coprocesadores matemáticos (familia 80xx7) de tal forma que pudieran manejar operaciones aritméticas con un formato de 80 bits con un intervalo dinámico de $10^{-4,932}$ a $10^{4,932}$ [4]. La dificultad de este formato es la conversión adecuada para utilizar los resultados de las operaciones por otros dispositivos en aplicaciones reales. Como se puede observar, dada una longitud de palabra L, podemos diseñar nuestro propio formato de punto flotando, sin embargo, éste debe ser versátil y eficiente en cuanto su manejo.

Resumen

En este capítulo se han expuesto los formatos numéricos binarios más utilizados en las arquitecturas digitales y en aplicaciones del procesamiento digital de señales. Cuando se realizan simulaciones de aplicaciones en lenguajes de alto nivel, comúnmente se realizan en formatos de punto flotante, luego se vuelve a simular en formatos de punto entero para verificar su comportamiento en una máquina digital. Se debe hacer notar que un diseñador de sistemas de procesamiento digital de señales debe conocer ampliamente el manejo de formatos numéricos para garantizar el buen funcionamiento de los algoritmos sobre arquitecturas reales.

Nota final

El autor espera haber aportado un pequeño grano de arena a la comunidad de estudiantes y profesionales que se dedican a explorar ese inmenso universo digital, donde al retornar en la curvatura del espacio, día a día nos encontramos con nuevas arquitecturas.

Apéndice B

Glosario

A
AC97: "audio codec 97".
ACC: registro acumulador.
ADC: convertidor análogo digital.
AH: acumulador parte alta, bits 31 a 16.
AL: acumulador parte baja, bits 15 a 0.
ALU: unidad aritmética lógica.
ARAU: unidad aritmética de registros auxiliares.
ARi: registros auxiliares o apuntadores de 16 bits, i:0,1,...,7. Parte baja de los XARi.
ARM: microprocesador avanzado RISC.
ARMA: filtros autorregresivos de movimiento promedio.

B
BIOS: sistema básico de entrada salida
Big endiand: forma de organizar los bytes o palabras en memoria, si un dato es de dos palabras de longitud, se almacena en la localidad más baja la palabra MSW y en seguida la LSW.
BOS: localidad baja de la pila.
bps: bit por segundo.
BR: "bit reverse" o acarreo inverso.
BSP: puerto serie buffereado.
bu: bits sin signo.

C
C1: complemento a uno.
C2: complemento a dos.
C28x: procesador digital de señales TMS320C28xx de TI.
CISC: conjunto complejo de instrucciones.

CLA: unidad aceleradora de operaciones matemáticas, sólo en algunas versiones C28x.
CMOS: tecnología de circuitos integrados "complementary metal oxide semiconductor".
CCS: "code composer studio", ambiente integrado de desarrollo.
COND: condición.
CRC: verificación de redundancia cíclica.
CSM: módulo de código de seguridad.
CPK: kernel del protocolo eCAN.
Compandor: compresor expansor.
cte: constante.

D
DAB: bus de direcciones de datos.
DARAM: memoria de doble acceso en un ciclo.
dat: dato.
DDB: bus de lectura de datos.
DFT: transformada discreta de Fourier.
dir: dirección.
DMA: acceso directo a memoria.
dma: dirección de memoria dato.
DMAC: multiplicación acumulación dual o doble de 16x16 bits.
DP: registro apuntador de página.
DSP: procesadores de señales digitales.
DSC: controladores de señales digitales.
DTMF: "dual tone modulation frequency".
Δ: resolución.

E
EEPROM: memoria borrable ROM.
EOC: señal de fin de conversión.
EV: módulo manejador de eventos.
EOS: fin de conversión de una secuencia.

F
FA: filtro analógico.
FD: filtro digital.
FFT: transformada rápida de Fourier.
FIFO: tipo de registros o memoria con acceso primer dato en entrar, primer dato en salir.
FIR: filtros de respuesta finita al impulso.
FPU: unidad de punto flotante.

G
GIOP: entradas y salidas de propósito general.
Gw: giga palabras = mil millones de palabras.

H
h: símbolo posfijo de una cantidad en hexadecimal.
HRPWM: PWM de alta resolución.

I
I2C: interfaz "inter-integrated-circuit".
I2S: interfaz "integrated interchip sound".
IEEE: asociación internacional de ingenieros eléctricos y electrónicos.
ID: intervalo dinámico.
IDE: ambiente integrado de desarrollo.
IIR: filtros de respuesta infinita al impulso.
IOM-2: "oriented modular interface revision 2, bus-compliant device"
I/O: entrada/salida.

J
JTAG: "joint test action group".

K
Kw: kilo palabras = mil palabras.
Khz: kilo Hertz = mil Hertz.

L
Little endian: forma de organizar los bytes o palabras en memoria, si un dato es de dos palabras de longitud, se almacena en la localidad más baja la palabra LSW y enseguida la MSW.
LSb: bit menos significativo.
LSB: byte menos significativo.
LSW: palabra menos significativa.
loc: localidad en memoria.
loc16: localidad de 16 bits.
loc32: localidad de 32 bits.

M
MA: filtros de movimiento promedio.
MAC: operación multiplicación acumulación.
McBSP: puerto serial multicanal buffereado.
Mhz: mega Hertz = un millón de Hertz.

MIPS: millones de instrucciones por segundo.
MMU: unidad manejadora de memoria.
MSPS: millones de muestras por segundo.
MSb: bit más significativo.
MSB: byte más significativo.
MSW: palabra más significativa.
MUX: multiplexor o selector.
Mw: mega palabra = un millón de palabras.

N
NRZ: no retorno a cero.
ns: nanosegundos.

O
OTP: circuitos programables una sola vez: "one time programmable".
OMAP: plataformas abiertas para aplicaciones multimedia.

P
P: registro producto de 32 bits.
p: precisión numérica.
PAB: bus de direcciones de programa.
PDB: bus de datos de programa.
PC: contador de programa.
PDS: procesamiento digital de señales.
PIE: interfaz de expansión de interrupción de periféricos.
PH: 16 bits parte alta de P.
PL: 16 bits parte baja de P.
PLL: mallas de fase amarrada.
pma: dirección de memoria programa.
prog: programa.
PWM: modulación por ancho de pulso.

Q
QE: número de bits para la parte entera.
Qi: formato de punto entero y número de los bits para la parte fraccionaria.
QF: número de los bits para la parte fraccionaria.

R
RAM: memoria de acceso aleatoria.
RISC: conjunto reducido de instrucciones.
ROM: memoria de sólo lectura.

RPC: retorno del contador de programa.

S
S: bit de signo.
SARAM: memoria RAM de simple acceso.
SCI: interfaz de comunicación serial.
SCL: línea para reloj de I2C.
SDA: línea para datos de I2C.
SH(i): corrimientos de entrada y sobre el ACC.
SH(o): corrimientos de salida del ACC.
SH(m): corrimientos de salida del multiplicador.
SLITD: sistema lineal e invariante en el tiempo discreto.
SM: signo magnitud.
SNR: relación señal a ruido.
SPM: modo de corrimiento de la salida del multiplicador.
SP: apuntador de pila.
SOC: inicio de conversión.
SPI: interfaz de puerto serie.
ST0: registro de estado 0 de 16 bits.
ST1: registro de estado 1 de 16 bits.
sig: signo o signado.

T
T: 16 bits parte alta del registro XT.
TI: Texas Instruments, compañía productora de DSP.
TL: 16 bits parte baja del registro XT.
TOS: localidad alta de la pila.

U
MMU: unidad manejadora de memoria.

V
V: volts.
Vcc: voltaje de alimentación.
VCO: osciladores controlados por voltaje.
VCU: unidad Viterbi, de matemáticas complejas y de CRC.
VLWI: "very large word instruction", palabra de instrucción muy larga.

X
XARi: registros auxiliares o apuntadores de 32 bits, i:0,1,...,7.
XT: registro temporal de 32 bits.

W
w: palabra de 16 bits.

Bibliografía

[1] Escobar S. L. & Alcántara S. R. *Fixed Point Arithmetic Using Digital Signal Processors*. International Conference on Telecommunications, ICT2000, Acapulco, Guerrero, México, mayo 2000.

[2] Escobar S. L. *Diseño de Filtros Digitales*. Facultad de Ingeniería, UNAM, noviembre del 2006. 200 pags.

[3] Escobar S. L. *Conceptos Básicos de Procesamiento Digital de señales*. Facultad de Ingeniería, UNAM, México D. F. 2009. 195 pags.

[4] Godfrey J. T. *Lenguaje ensamblador para microcomputadoras IBM, para principiantes y avanzados*. Prentice Hall, México 1991.

[5] IEEE *IEEE Standar for Binary Floating-Point Arithmetic*. IEEE Standar 754-1985.

[6] Kuo S. & Lee B. *Real Time Digital Signal Processing Algorithms, implementation, applications and Experiments with th TMS320C55x*. John Wiley & Sons, England 2001.

[7] Kuo M. S. & Woon-Seng G. *Digital signal Processors, Architecture, implementations and applications*. Prentice-Hall, New Jersey, USA 2005.

[8] Lynch M. A. *Microprogrammed state machine design*. CRC Press, USA 1993.

[9] Mitra S. K. *Digital Signal Processing. A computer Based approach*. Second edition, McGraw-Hill, New York 2001.

[10] Proakis J. G & Manolakis D. G. *Digital Signal Processing, Principles, Algorithms and Applications*. Macmillan, New York 1992.

[11] Proakis J. G. *Digital Communications*. McGraw Hill, New York 1995.

[12] Rabiner L. & Gold B. *Theory and applications of digital signal processing*. Prentice Hall, USA 1975.

[13] Silva L. A. *Detector de ángulo de arribo de una señal en un arreglo de micrófonos*. Tesis de licenciatura, Facultad de Ingeniería, UNAM. México D.F. 2010.

[14] Smith S. W. *The Scientist and engineer's guide to digital signal processing.* 2d. Edition, California Technical Publishing. San Diego California, USA 1999.

[15] Stallings W. *Organización y arquitectura de computadoras.* Séptima Edición, Prentice Hall, España 2006.

[16] Texas Instruments. *TMS320C5x, User's Guide.* USA 1993.

[17] Texas Instruments. *TMS320C2802x, 2803x Piccolo Analog-to-Digital Converter (ADC) and Comparator Reference Guide.* SPRUGE5F, USA 2011.

[18] Texas Instruments. *TMS320C28x CPU and Instruction Set Reference. SPRU430.* USA 2004.

[19] Texas Instruments. *TMS320C28xx, DSP Peripheral Reference Guide.* SPRU566. USA 2003.

[20] Texas Instruments. *TMS320F28x DSP, System Control and Interrupts Reference Guide. SPRU078A.* USA 2003.

[21] Texas Instruments. *TMS320F2802x/TMS320F2802xx Piccolo System Control and Interrupts Reference Guide.* SPRUFN3C. USA 2009.

[22] Texas Instruments. *TMS320x2833x, Analog-to-Digital Converter (ADC) Module.* SPRU812. USA 2007.

[23] Texas Instruments. *TMS320F28x DSP Event Manager (EV) Reference Guide. SPRU065A.* USA 2003.

[24] Texas Instruments. *TMS320x2833x, 2823x Boot ROM Reference Guide. SPRU963.* USA 2008.

[25] Texas Instruments. *TMS320x2833x, 2823x Multichannel Buffered Serial Port (McBSP) Reference Guide.* SPRU061A. USA 2003.

[26] Texas Instruments. *TMS320x2833x, 2823x Direct Memory Access (DMA) Module Reference Guide. SPRUFB8.* USA 2009.

[27] Texas Instruments. *TMS320x2833x, 2823x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide. SPRUG04.*

[28] Texas Instruments. *TMS320x2833x, 2823x Enhanced Capture (eCAP) Module Reference Guide. SPRUFG4.* USA 2009.

[29] Texas Instruments.*TMS320x2833x, 2823x Enhanced Quadrature Encoder Pulse (eQEP) Module Reference Guide. SPRUG05.* USA 2008.

[30] Texas Instruments.*TMS320x2833x, 2823x Enhanced Controller Area Network (eCAN) Reference Guide. SPRUEU1.* USA 2003.

[31] Texas Instruments.*TMS320x2833x, 2823x Serial Communications Interface (SCI) Reference Guide. SPRUFZ5.* USA 2003.

[32] Texas Instruments. *TMS320x2833x, 2823x DSC Serial Peripheral Interface (SPI) Reference Guide. SPRUEU3.* USA 2003.

[33] Texas Instruments. *TMS320x2833x, 2823x Inter-Integrated Circuit (I2C) Module Reference Guide. SPRUG03B.* USA 2011.

[34] Texas Instruments. *IQmath Library, a virtual floating point engine.* USA 2002.

[35] Texas Instruments. *TMS320F28335, TMS320F28334, TMS320F28332, TMS320F28235, TMS320F28234, TMS320F28232. Digital Signal Controllers (DSCs), Data Manual.* SPRS439I. USA 2007.

[36] Texas Instruments. *TMS320x2806x Piccolo, Technical Reference Manual.* SPRUH18C. USA 2011.

[37] Texas Instruments. *Piccolo Microcontrollers. SPRS698B.* USA, nov. 2010, Rev. jul. 2011.

[38] Vázquez S. S. *Reducción de ruido en señales de voz mediante un formador de haz.* Tesis de licenciatura, Facultad de Ingeniería, UNAM. México D.F. 2012.

Bibliografía recomendable

Alcántara S. R. & Escobar S. L. *Dynamic Range and Scaling Evaluation in Adaptive Filtering Algorithms for DSP fixed-point implementation.* International Symposium on Information Theory and its Applications (ISITA98). México, octuber 14-16, 1998.

Alcántara S. R. & Escobar S. L. *A comparative TMS DSP Fixed-Point Implementation of FRLS Adaptive Filtering Algorithms Family.* The International Conference on Signal Processing Applications and Technology (ICSPAT). November 1-4, 1999. Orlando, Florida USA.

Barr M. *Programming Embedded Systems in C and C++.* O'Really USA 1999.

Barreto G. F, Toral M. S. y Ruiz G. M. *Procesadores Digitales de Señales de altas prestaciones de Texas Instruments $^{(TM)}$, de la familia TMS329C3x a la TMS320C600.* McGraw Hill España 2005.

Chassaing R. *DSP Application Using C and the TMS320C6x.* John Wiley & Sons, USA 2002.

Dahnoum N. *DSP Implementation using the TMS320C6000 DSP plataform.* Prentice Hall, Londres UK, 2000.

Escobar S. L. *Arquitecturas de DSPs, familia TMS320 y el TMS20C50.* Facultad de Ingeniería, UNAM, México D. F., agosto del 2000.

Escobar L., Psenicka B. y Molero. A. M. *Arquitecturas de DSPs, familias TMS320C54x y TMS320C54xx y aplicaciones.* Facultad de Ingeniería, UNAM, octubre del 2005. 191 pags.

Escobar S. L. y Alcántara S. R. *Fixed Point Arithmetic Using Digital Signal Processors.* International Conference on Telecommunications, ICT2000, Acapulco Guerrero, México, mayo 2000.

Hamming R. W. *Digital Filters.* Second Edition, Prentice-Hall Englewood Cliffs, New Jersey 1983.

Haykin S. *Modern Filters.* MacMillan, New York 1989.

Haykin S. *Adaptive Filter Theory.* Englewood Cliffs, NJ Prentice-Hall, 1991.

Edited by Simon Haykin. *Blind Deconvolution.* Prentice Hall, Englewood Cliffs, New York 1994.

Hayes M. *Statistical Digital Signal Processing and modeling.* John Wiley & Sons, USA 1996.

Hsu H. P. *Análisis de Fourier.* Addison Wesley, USA 1973.

Liu D. *Embedded DSP processor Design, Application Specific Instruction Set Processors.* Elsevier, USA 2008.

Morris M. M. *Arquitectura de computadoras.* Prentice Hall. México 1983.

Oppenheim A. V. & Schafer R. W. *Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, New Jersey 1975.

Oppenhein A. V. *Applications of digital signal processing.* Prentice Hall, USA. 1978.

Oppenheim A. V., Schafer R. W. & Buck J. R. *Discrete time Signal Processing.* Prentice-Hall, USA 1999.

Proakis J., Lin R. C. and Nikias C. *Advanced Digital Signal Processing.* Macmillan-Maxwell, Ontario Canada 1992.

Prado J. & Alcántara S. R. *A Fast Square-Rooting Algorithm Using a Digital Signal Processor*. Proceedings of IEEE, USA, Vol. 75, No.2, pg. 262-264, Feb. 1987.

Proakis J. G, Rader M. MC., Ling F. & Nikias K. *Advanced Digital Signal Processing*. Macmillan, New York 1992.

Miano J. *Compressed Image File Formats JPEG, PNG, GIF, XBM and BMP*. Addison Wesley, New York 2000.

Proakis J. & Manolakis D. *Tratamiento Digital de Señales: Principios, algoritmos y aplicaciones*. Prentice Hall, USA 1998.

Psenicka B. *Apuntes de Procesamiento Digital de Señales*. Facultad de Ingeniería, UNAM, México D.F., 1996.

Psenicka B. y Escobar S. L. *Procesamiento Digital de Señales, segunda parte, Microcontroladores y realización de los filtros digitales con TMS320Cxx*. Facultad de Ingeniería, UNAM, México D.F., julio de 1998.

Rabiner L. & Schafer R. W. *Digital Processing of Speech Signals*. Prentice-Hall Inc. USA 1978.

Savage C. J. y Vázquez Gabriel. *Diseño de Microprocesadores*. Facultad de Ingeniería, UNAM, marzo del 2004. 347 pags.

Texas Instruments. *TMS320C3x, User's Guide*. USA 1994.

Texas Instruments. *TMS320C4x, User's Guide*. USA 1993.

Texas Instruments. *TMS320C6x, User's Guide*. USA 1999.

Texas Instruments. *TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide*. SPRU72C. USA 2006.

Texas Instruments *Digital Signal Processing Applications with the TMS320 Family, Theory, Algorithms, and implementations*. Vol.1,2 y 3. USA 1990.

Texas Instruments. *TMS30C1x, User's Guide*. USA 1990.

Texas Instruments. *TMS30C2x, User's Guide*. USA 1991.

Texas Instruments. *TMS30C3x, User's Guide*. USA 1992.

Texas Instruments. *TMS30C5x, User's Guide*. USA 1997.

Texas Instruments. *TMS320C28x Floating Point Unit and Instruction Set Reference Guide. SPRUEO2*. USA 2008.

Texas Instruments. *TMS320x2833x, 2823x DSC External Interface (XINTF) Reference Guide. SPRU949*. USA 2003.

Texas Instruments. *TMS320x2833x, 2823x High-Resolution Pulse Width Modulator (HRPWM) Reference Guide. SPRUG02*.

Texas Instruments. *TMS320C28x Assembly Language Tools v5.0.0 User's Guide. SPRU513.* USA 2001.

Texas Instruments. *TMS320C28x Optimizing C/C++ Compiler v5.0.0 User's Guide.* SPRU514. USA 2001.

Texas Instruments. *TMS320C28x Instruction Set Simulator Technical Overview. SPRU608.* USA 2002.

Texas Instruments. *TMS320C28x DSP/BIOS 5.32 Application Programming Interface (API) Reference Guide. SPRU625.* USA 2002.

Toliyat H. A. & Campbell S. *DSP Based Electromechanical motion control.* CRC Press.USA 2004.