TEMA 2. FAMILIA DE DSPs TMS320C3x DE TEXAS INSTRUMENTS. CONCEPTOS BÁSICOS DEL HARDWARE.

1. Introducción.
2. Descripción general del sistema.
3. Unidad central de control de procesos (CPU).
   3.1 Registros internos de la CPU.
4. Operaciones internas del bus.
5. Organización de la memoria.
   5.1 Reset y Vectores de Interrupción.
   5.2 Periférico Memoria Caché.
   5.3 Arranque programado en el TMS320C31: *Boot-Loader*.
6. Operaciones externas del bus.
   6.1 Estructura *Master-eslave* en un sistema microprocesador.
   6.2 Diagramas de tiempos y estados de espera.
   6.3 Tiempo de ejecución y tiempo de acceso.
   6.4 Peculiaridades en el acceso a periféricos externos.

1. INTRODUCCIÓN

Hoy en día, algunos de los DSPs de menor coste, que se pueden encontrar en el mercado, son los que pertenecen a la familia **TMS320** del fabricante **Texas Instruments**. Estos DSPs ofrecen un elevado nivel de integración de dispositivos o periféricos internos. Esta última característica disminuye, considerablemente, el coste del hardware (son pocos los integrados externos que hacen falta para diseñar un producto final) del sistema completo y mejora su fiabilidad (cuanto menor sea el número de periféricos externos necesarios, menor será la probabilidad de equivocarse al diseñar el producto final). Texas Instruments tiene en el mercado diferentes estructuras de DSPs:

- DSPs diseñados para una aplicación específica (de tipo **SISC**), por ejemplo, aquellos que se dedican a funciones de contestador automático, a telefonía móvil, etc.
- DSPs programables o de propósito general (de tipo **CISC**), diseñados para suministrar al usuario una elevada flexibilidad para el desarrollo de aplicaciones específicas.

En la figura 2.1 se muestra las diferentes familias de DSPs de aplicación general del fabricante Texas Instruments. Se distingue entre DSPs que disponen de operadores aritméticos de coma fija[1] y los que disponen de operadores aritméticos de coma variable. Obsérvese que, en todos los casos, los operadores aritméticos incluyen operaciones con números flotantes algo que no ocurría en el M68HC11 de Motorola, estudiado anteriormente, donde el operador aritmético asociado a la CPU incluía, únicamente, la posibilidad de realizar operaciones con enteros.

A lo largo de este tema nos centraremos en el análisis de la familia de DSPs TMS320C3x, que incluyen operadores de coma flotante. Esta familia de DSPs está constituido por tres dispositivos, todos con una estructura interna (unidad central de procesos) común pero con diferentes periféricos internos. El DSP más completo de la familia es el TMS320C30. Posteriormente aparecieron los DSPs TMS320C31 y TMS320C32 que son versiones reducidas en la circuitería interna respecto de la del TMS320C30 y, por tanto, de muy bajo coste. Además, están especialmente preparados para simplificar los costes, en el desarrollo del hardware externo, que precisa el usuario para diseñar y probar cualquier tipo de aplicación basada en el TMS320C30.

[1] Los operadores aritméticos del DSP emplean, como operandos, números flotantes con un decimal representado por un número de bits fijo (COMA FIJA) o variable (COMA VARIABLE).

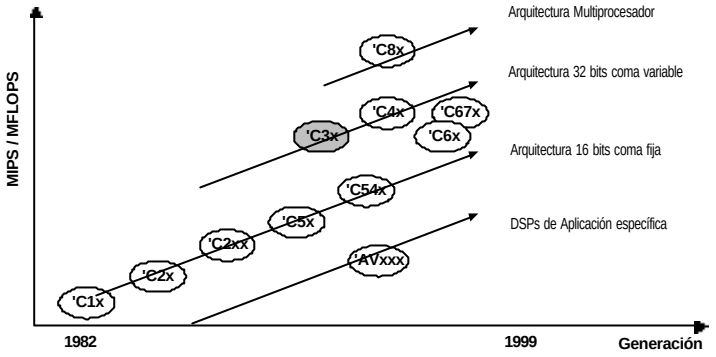


Figura 2.1. Procesadores Digitales de Señal del fabricante Texas Instruments[2].

2. DESCRIPCIÓN GENERAL DEL SISTEMA

En la figura 2.2 se muestra un diagrama de bloques de la estructura interna de la familia TMS320C3x.

Incorpora una arquitectura microprocesadora interna clásica de tipo Harvard (internamente ofrece buses de datos y direcciones independientes que permiten indistintamente el acceso a memoria de programa y dato). La estructura interna de la CPU de este DSP, le permite, en principio, realizar, simultáneamente y en un único ciclo máquina[3], operaciones de multiplicación y suma, así como operaciones con enteros o flotantes de coma variable. Los accesos al exterior se realizan, sin embargo, utilizando una arquitectura microprocesadora de tipo Von-Neuman (selección y acceso a los datos e instrucciones por los mismos buses de dirección y datos).

[2] MIPS: **M**illones de **I**nstrucciones **P**or **S**egundo. MFLOPS: **M**illones de **O**peraciones con **FL**otantes **P**or **S**egundo.

[3] El **ciclo máquina** hace referencia al tiempo efectivo de ejecución de una instrucción. A partir del oscilador, cristal de cuarzo, del sistema digital se obtiene el reloj del sistema, cuya frecuencia es la mitad de la del oscilador, $f_{CLK} = f_{oscilador}/2$. El ciclo máquina de este DSP coincide con el periodo de la señal de reloj, $T_{CLK} = 2 \cdot T_{oscilador}$.

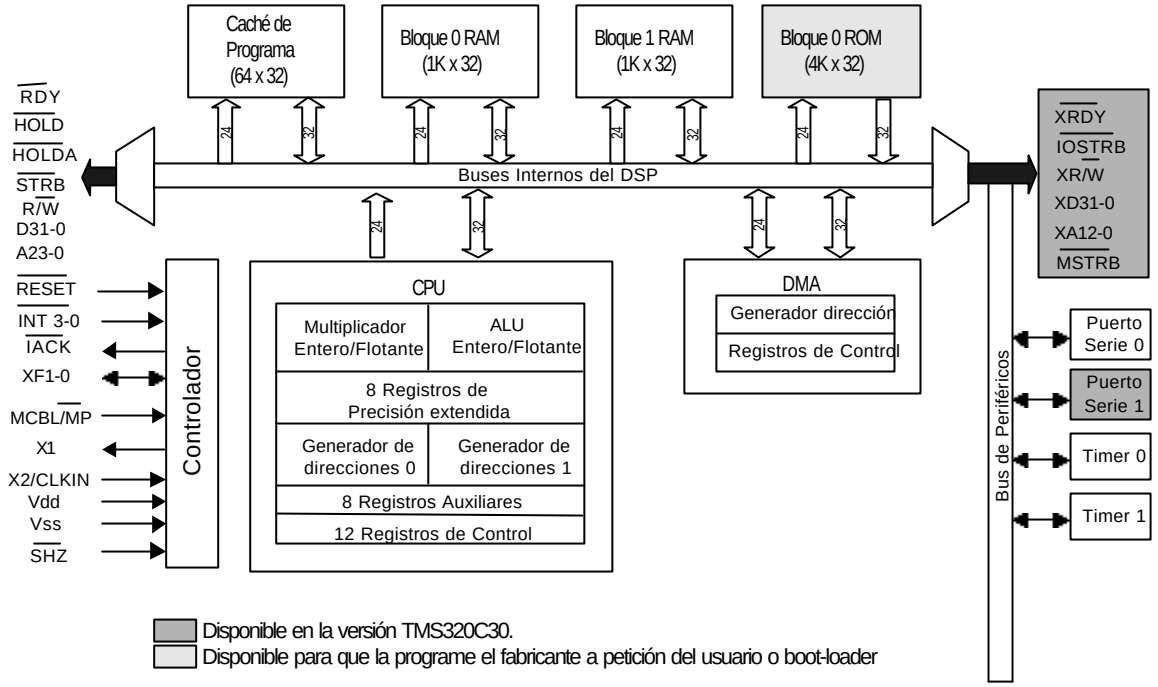


Figura 2.2. Diagrama de bloques de la estructura interna del DSP TMS320C3x

La CPU dispone de una serie de registros internos, R0-R7, registros que hacen las veces de registros acumuladores de propósito general. Su función es la de ser el operando de las diferentes operaciones aritméticas que puede realizar la CPU. Juntos con estos registros aparecen otros en la CPU, los denominados ARi, que se emplean como operandos de unas unidades aritméticas auxiliares.

El bus de datos de este sistema microprocesador es de 32 bits, siendo el bus de direcciones de 24 bits, lo que permite una capacidad de direccionamiento total del dispositivo de 16 *Mwords* (obsérvese que la *word* para este sistema es de 32 bits).

Texas Instruments ofrece (como ocurre la mayoría de las veces con cualquier microprocesador, microcontrolador o DSP de cualquier fabricante) DSPs de esta familia que incorporan modos de funcionamiento con ahorro energético: los modos IDLE2 y LOPOWER. En modo *IDLE2*, se congela el reloj hasta que aparezca una de las cuatro interrupciones externas que lo reactivan. En modo *LOPOWER*, la CPU y la DMA continúan ejecutando instrucciones y transfiriendo datos a memoria, pero a una frecuencia de reloj mucho más baja (se divide por 16 la entrada de pulsos proveniente del oscilador).

Se observa, además, que esta familia de DSPs dispone de una serie de periféricos internos:

- Memoria interna: Memoria RAM (dos bloques de 1*Kw* cada uno) y ROM (4*Kw* de tipo OTP-programable una única vez por el fabricante). El tiempo de acceso a estos periféricos es de un ciclo máquina.
- Periférico de memoria caché de programa. El tiempo de acceso a estos periféricos es de un ciclo máquina. El usuario lo emplea por ejemplo, para disminuir los accesos a instrucciones ubicadas en memoria externa, con tiempos de acceso más elevados que los que tienen las instrucciones que se ubican en memoria interna (de tipo RAM o ROM).
- Periférico DMA interno.
- Puertos serie con protocolo síncrono. Dos puertos serie disponibles en el TMS320C30, únicamente uno en el TMS320C31 y TMS320C32.
- Dos periféricos temporizadores programables de 32 bits (*Timers*).
- Periférico de control y gestión de interrupciones que hace este proceso transparente a la CPU.
- Periféricos de control de acceso a memoria. Estos periféricos se encargan de gestionar los accesos a posiciones de memoria ubicadas en el interior o exterior del integrado. El acceso al exterior se realiza mediante dos buses independientes entre sí (para el caso del TMS320C30), el bus principal y el de expansión, cada uno formado por un bus de datos, otro de direcciones y, finalmente, un bus de control. Podemos, por tanto, controlar el acceso al exterior del dispositivo por dos canales, independientes entre sí, cada uno con su protocolo, sus tiempos característicos, etc. Las versiones TMS320C31 y TMS320C32 únicamente disponen del bus principal para gestionar los accesos al exterior.

El DSP, mediante una unidad multiacumuladora (*MAC*), puede ir preparando la próxima instrucción a ejecutar mientras procesa (opera con) dos o más datos, en el mismo ciclo de reloj (estructura interna de tipo Harvard).

Los cuatro componentes básicos de esta familia son:

- ✓ **TMS320C30** con versiones que funcionan a diferentes frecuencias máximas de reloj externo (oscilador o cristal de cuarzo): 40 y 50 MHz. Por ejemplo, el de 40 MHz de frecuencia máxima tendría, funcionando con un cristal de cuarzo de dicha frecuencia conectado al DSP, 50ns de tiempo de ejecución de una instrucción (el doble del periodo del oscilador) y la posibilidad de realizar hasta 40 MFLOPS (millones de operaciones de coma flotante por segundo) y ejecutar 20 MIPS (millones de instrucciones en un segundo). Nótese que estos dos últimos valores, obtenidos del catálogo del fabricante, se refieren al caso hipotético en que el sistema ejecute múltiples instrucciones, cada una con una duración de un ciclo máquina, e implementando, cada una de esas instrucciones, dos operaciones en coma flotante.

- ✓ **TMS320C31** con versiones a 40, 50, 60 y 80 MHz. Básicamente es una versión económica del TMS320C30. Dispone de un programa denominado autocarga o *boot-loader* en la ROM interna (ROM de 4*Kw* que no se encuentra disponible, en este caso, para que el usuario le encargue al fabricante su reprogramación), de un puerto serie y no dispone de acceso al exterior mediante el bus de expansión.

- ✓ **TMS320C32** con versiones a 40, 50 y 60 MHz. Como característica fundamental destaca la posibilidad de configurar el ancho del bus de datos para que funcione, de forma transparente al usuario, a 16 ó 32 bits en el acceso a instrucciones externas (memoria de programa) y a 8, 16 ó 32 bits en el acceso a datos externos (memoria de datos), manteniendo la estructura interna de 32 bits. Esto le permite al usuario disminuir los costes de desarrollo (menor número de dispositivos de memoria externos necesarios para que el sistema digital funcione) inicialmente asociados al diseño de sistemas basados en este DSP. Dispone de ROM interna ya programada (*boot-loader*), de dos dispositivos de RAM interna de 256x32bits, un puerto serie, dos periféricos timers y otros dos periféricos DMA. Las interrupciones externas pueden generarse por nivel o flanco. Finalmente, dispone, al contrario que los otros DSPs de la familia, de la posibilidad de reubicar la

tabla de 256 vectores de interrupción de tipo TRAP. Los periféricos puerto serie y temporizador, de este DSP, tienen una estructura y comportamiento idénticos a los de los otros DSPs de la familia. Los periféricos DMA son equivalentes pero añaden la posibilidad de configurar (si aparece un conflicto por acceso a las mismas zonas de memoria) la prioridad en los accesos gestionados desde la CPU y la DMA.

- ✓ **TMS320C33** con versiones de 120 y 150 MHz, último DSP desarrollado por Texas Instruments de la gama TMS320C3x (año 1998). Este DSP se basa en el TMS320C31. Las diferencias son que dispone de un único temporizador, de más RAM interna (hasta 34Kw) y aparece en versiones de bajo consumo (se alimenta a tensiones de menos de $5V_{DC}$ lo que empieza a ser común en muchos dispositivos electrónicos digitales). El coste de este DSP es superior al del TMS320C32 e inferior al del TMS320C31.

En adelante, analizaremos con detenimiento las características del TMS320C30, el padre de la familia (por llamarlo de alguna forma) y comentaremos algunas de las peculiaridades de los otros dos elementos de la misma. En la figura 2.3 se muestra, en detalle, la estructura interna de la familia de DSPs de Texas Instruments TMS320C3x.

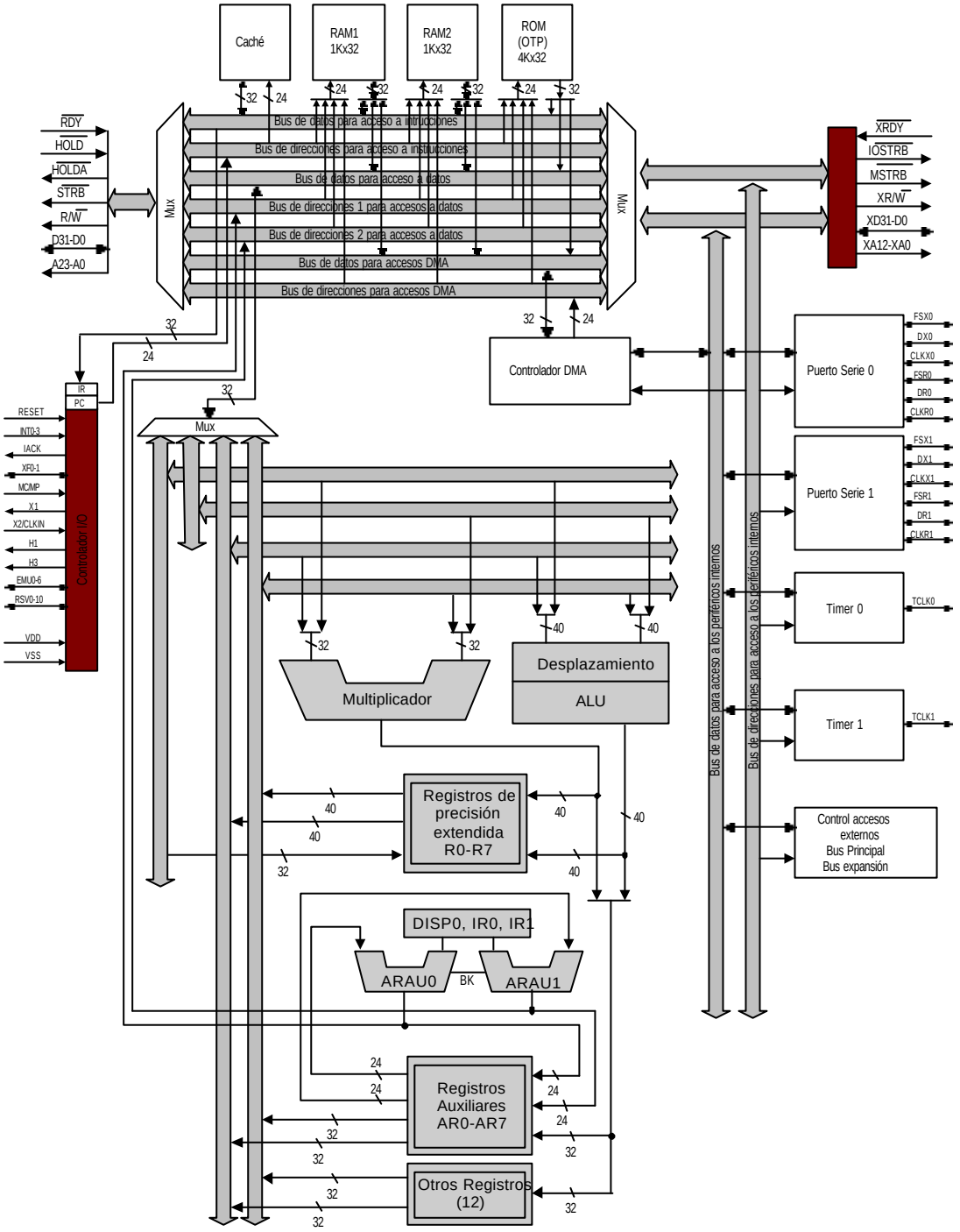


Figura 2.3. Estructura Interna de los DSPs de la familia TMS320C3x del fabricante Texas Instruments.

Las características generales de este tipo de DSPs, se pueden enumerar y resumir como sigue:

1. Características de memoria interna.

    - 4Kx32bits de ROM interna con un solo ciclo máquina de acceso. En el caso del TMS320C30 disponible para que el fabricante se la programe al usuario, mientras que, en los casos del TMS320C31 y TMS320C32 ya está programada de fábrica.
    - Dos dispositivos de 1Kx32 bits de RAM interna, con un único ciclo máquina de acceso. En el caso del TMS320C32, dos dispositivos de 256x32bits.
    - 64 registros de 32 bits de memoria cache.
    - 28 registros internos a la CPU. 8 registros acumuladores de precisión extendida, R0-R7, de 40 bits en lugar de los 32 bits habituales.

2. Operadores aritméticos.

    - Multiplicador interno en punto flotante (operandos en 32 bits y resultado en 40 bits) o entero (operandos en 24 bits y resultado en 32 bits).
    - Unidad aritmética y lógica (ALU). Realiza operaciones lógicas y de desplazamiento con realimentación de 32 bits y operaciones aritméticas con datos enteros (de 32 bits) o flotantes (de 40 bits).
    - Operaciones de multiplicación y de la ALU en paralelo en un sólo ciclo.

3. Periféricos internos.

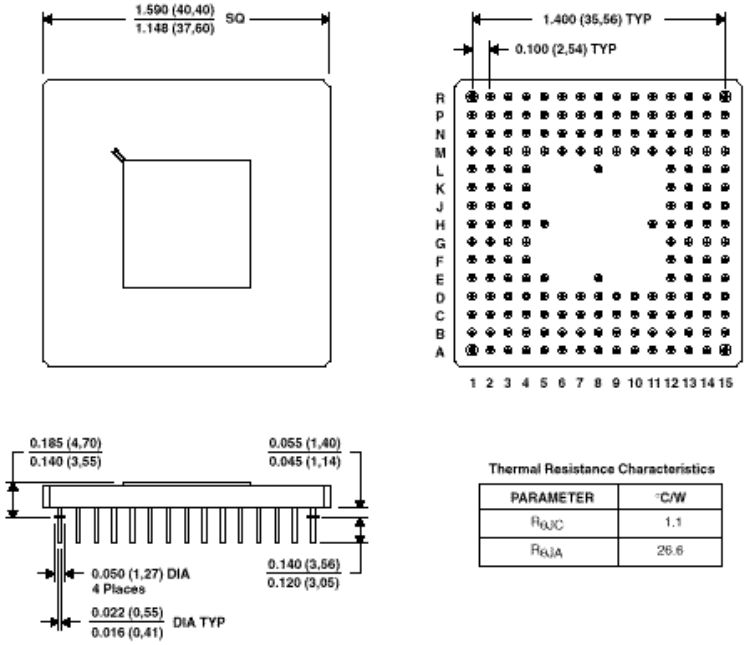
    - Controladores de eventos DMA internos.
    - Dos puertos digitales de entrada/salida de propósito general: XF0, XF1.
    - Temporizadores de 32 bits.
    - Puertos serie síncronos.
    - Dos periféricos independientes (bus principal y bus de expansión) que controlan los accesos al exterior. Cada periférico gestiona un bus de datos de 32 bits, un buses de

direcciones (de 24 y 13 líneas respectivamente) y un bus de control (de 5 y 4 líneas respectivamente).

4. Otras características.
   - Instrucciones:
     - Posibilidad de utilizar dos y más operandos.
     - Existencia de instrucciones de repetición de un bloque de códigos.
     - Posibilidad de ejecutar saltos en un solo ciclo máquina.
     - Existencia de instrucciones CALL y RET condicionales.
     - Existencia de instrucciones con interbloqueo para multiproceso (varios DSPs conectados en paralelo controlando un único proceso)
     - Instrucciones complejas que ejecutan dos operaciones simples en paralelo.
   - Interrupciones externas.
   - Posibilidad de operar en modos de bajo consumo energético (IDLE2 ó LOPOWER).

En definitiva, las características de paralelismo en operaciones, la gran cantidad de memoria interna, la unidad de punto flotante, el controlador DMA interno y los periféricos temporizadores, permite su empleo en sistemas capaces de implementar sofisticados algoritmos matemáticos en tiempo real.

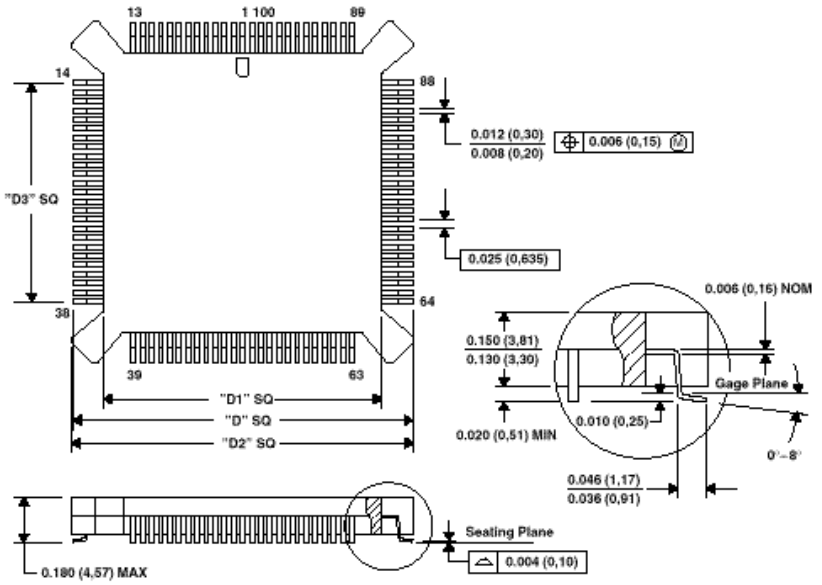
Los principales encapsulados (forma y características del ASIC) que tienen estos microprocesadores se muestran en la figura 2.4 (se muestra el micro en encapsulado PGA de 181 pines, existe también en encapsulado PQFP de 208 pines), para el TMS320C30, figura 2.5 (encapsulado PQFP de 132 pines), para el TMS320C31 y figura 2.6 para el TMS320C32 (encapsulado CPGA de 144 pines).



Thermal Resistance Characteristics

PARAMETER	°C/W
$R_{\theta JC}$	1.1
$R_{\theta JA}$	26.6

Figura 2.4. Encapsulado de tipo PGA con 181 pines, TMS320C30.



DIM	LEADS ***	100	132
"D"	MAX	0.890 (22,61)	1.090 (27,69)
	MIN	0.870 (22,10)	1.070 (27,18)
"D1"	MAX	0.766 (19,46)	0.966 (24,54)
	MIN	0.734 (18,64)	0.934 (23,72)
"D2"	MAX	0.912 (23,16)	1.112 (28,25)
	MIN	0.888 (22,56)	1.088 (27,64)
"D3"	NOM	0.600 (15,24)	0.800 (20,32)

Figura 2.5. Encapsulado de tipo PQFP con 100 pines. El encapsulado típico del TMS320C31 es como el de la figura pero con 132 pines.

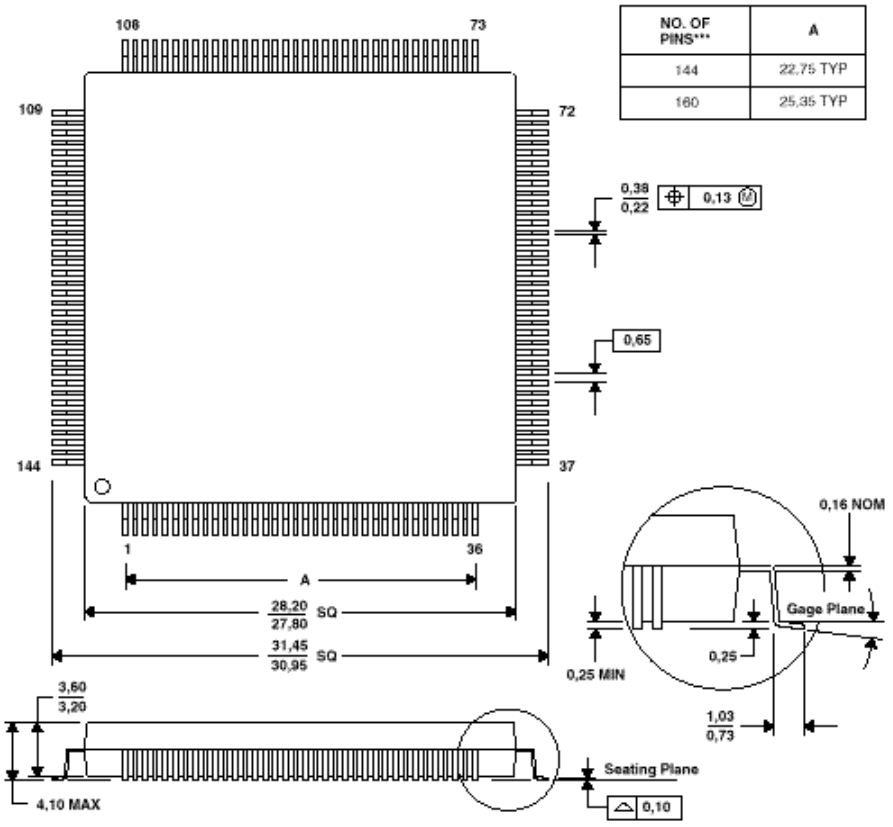


Figura 2.6. Encapsulado de tipo CPGA con 144 pines, TMS320C32. El encapsulado del TMS320LC33 es parecido (PQFP de 144 pines).

3. UNIDAD CENTRAL DE CONTROL DE PROCESOS (CPU)

Este módulo es el que controla y gestiona el ASIC. Está constituido por dos partes:

1. Una parte encargada del **manejo de las instrucciones**. Formada por las máquinas de estado síncronas que controlan todos los recursos del sistema. Las instrucciones que maneja esta CPU tienen un tamaño fijo, 32 bits y la CPU tiene asignados dos registros para manejar las instrucciones, el contador de programa o registro PC (registro contador que se emplea como puntero a la dirección de la siguiente instrucción que se debe ejecutar) y el registro de instrucción o registro IR (registro que no es accesible para el usuario y que almacena la instrucción leída por la CPU).

Las máquinas de estados, encargadas de la gestión de los recursos, emplean el registro PC para apuntar a la próxima instrucción que se debe ejecutar. Este registro se conecta directamente con el denominado bus de dirección de las instrucciones. La instrucción se recoge mediante el bus de datos de las instrucciones y se almacena en el registro IR. Una vez leída la instrucción, la máquina de estado la interpreta (genera, en el tiempo, la secuencia de operaciones básicas que lleve asociadas). Estas operaciones básicas se clasifican en cinco tipos y se describirán con detalle en el apartado "*operaciones internas del bus"*.

2. La otra parte es la que se encarga del **manejo de los datos**. Es un bloque que incluye los operadores aritméticos que existen en el DSP. El bloque de operadores aritméticos de la CPU consta, a su vez, de cinco elementos:

   - Un Multiplicador.
   - Una unidad aritmético lógica, ALU.
   - Unidades aritméticas de registros auxiliares, ARAU0 y ARAU1.
   - Los registros internos de la CPU, en total 30 registros. De ellos, 22 son de 32 bits y el resto son registros de 40 bits.
   - Cuatro buses de datos internos: CPU1, CPU2, REG1 y REG2. Acceso a los dos buses de dirección de los datos de que dispone el DSP.

   Todos los registros se conectan a dos de los cuatro buses internos (REG1, REG2) y pueden ser empleados como operandos del multiplicador y la ALU de la CPU. Las salidas de los operadores ARAUs, que se conectan a los dos buses de dirección para el acceso a datos, permiten la generación simultánea de dos direcciones de otros tantos datos que son recogidos por la CPU mediante los buses CPU1 y CPU2. Este paralelismo en el acceso a los operandos permite la realización de operaciones simultáneas en la ALU y el multiplicador. En la figura 2.7 se representa la estructura de los operadores de la CPU. En esta figura pueden observarse todos los bloques comentados anteriormente. Las características más relevantes de estos módulos son:

- **Multiplicador**: Puede realizar multiplicaciones, con números enteros o con números flotantes, en un solo ciclo máquina de la CPU. Tiene, además, la posibilidad de operar mientras, en paralelo, opera la ALU en un ciclo máquina. Cuando realiza un producto en punto flotante, las entradas son números flotantes codificados en 32 bits, mientras que la salida es otro número flotante pero codificado en 40 bits. Cuando la multiplicación se realiza con números enteros, las entradas son enteros de 24 bits mientras que la salida es un entero de 32 bits.

- **Unidad aritmético lógica (ALU) y registro de desplazamiento**. Realizan operaciones en un ciclo máquina de la CPU, operaciones de tipo lógicas (con datos de 32 bits), enteras (con datos de 32 bits) y flotantes (con datos de 40 bits). Incluyen la operación de conversión de entero a flotante o viceversa. También pueden realizar desplazamientos cíclicos de hasta 32 bits hacia derecha o izquierda (lo que por ejemplo sirve para realizar multiplicación o división entera por potencias de dos).

- Los **buses internos CPU1, CPU2, REG1 y REG2** pueden transportar dos operandos recogidos de la memoria y otros dos operandos que proceden de los registros internos de la CPU. De esta manera, la CPU permite multiplicación o suma paralela sobre cuatro operandos, flotantes o enteros, en el mismo ciclo máquina.

- **Unidades aritméticas de registros auxiliares (ARAU0 y ARAU1)**. Son dos operadores aritméticos que se emplean en la generación de direcciones. Estas unidades aritméticas operan en paralelo con el multiplicador y la ALU. Pueden generar dos direcciones en un ciclo máquina. Permiten implementar instrucciones de direccionamiento indexado mediante dos registros índice (IR0, IR1), soportando modos de direccionamiento por desplazamiento circular y bit-inverso que se verán en el tema 16.

- **Registros de la CPU**. La CPU incluye 29 registros accesibles al usuario (28 asociados a la parte de manejo de datos y 1, el PC, ligado a la parte de manejo de

instrucciones de la CPU) que constituyen un archivo de registros multipuerto (todos estos registros pueden ser accedidos desde el multiplicador, la ALU y por otros registros de propósito general). Los registros que posee la CPU del TMS320C30 se resumen en la tabla 2-1.

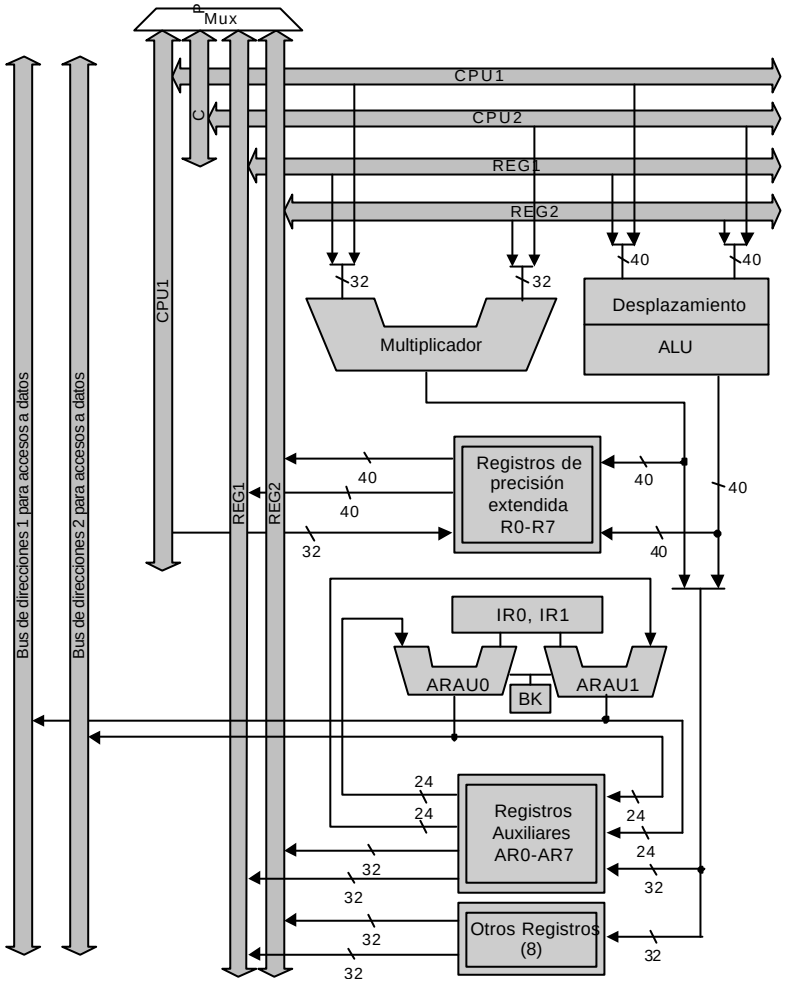


Figura 2.7. Zona dedicada al manejo de datos de la Unidad Central de Control de Procesos (CPU) de la familia TMS320C3x.

Nombre del Registro	**Función asignada**
Registros para manejo de datos: Registros acumuladores	
R0-R7	Registros de precisión extendida (40 bits)
AR0-AR7	Registros auxiliares (32 bits)
Registros para manejo de datos: Modos de direccionamiento	
DP	Puntero a la página de datos (32 bits)

IR0	Registro de índice 0 (32 bits)
IR1	Registro de índice 1 (32 bits)
BK	Registro de tamaño de bloque (32 bits)
Registros de estado y control del sistema	
SP	Puntero a la pila del sistema (32 bits)
ST	Registro de estado (32 bits)
IE	Habilitador de interrupciones CPU/DMA (32 bits)
IF	Banderas de interrupción de la CPU (32 bits)
IOF	Registro de control de las banderas I/O (32 bits)
Registros de control de la ejecución de un bloque de instrucciones: Modo Repetición	
RS	Repetición de una o varias instrucciones: Dirección de comienzo (32 bits)
RE	Repetición de una o varias instrucciones: Dirección final (32 bits)
RC	Repetición de una o varias instrucciones: Contador de repetición (32 bits)

Tabla 2-1. Registros de la CPU.

3.1 Registros Internos de la CPU

Veamos las funciones de los distintos registros internos de la CPU.

1. **Registros de precisión extendida (R0-R7)**. Son registros de propósito general, empleados por el multiplicador y la ALU de la CPU. El tamaño de ellos es de 40 bits (obsérvese que el bus de datos del sistema es de 32 bits, de ahí el nombre de registro de precisión extendida, con 40 bits se obtiene mayor precisión que con 32 bits). Estos registros son los únicos, dentro de la CPU, que pueden ser empleados como operandos flotantes (si un usuario desea trabajar con números flotantes debe almacenar dichos números en estos registros antes de operar con ellos). Si se operase con alguno de estos registros en modo flotante se usarían los 40 bits del mismo. Si las operaciones a realizar son de enteros se usan los bits menos significativos (del 0 al 31), el resto (bits 32 al 39) permanecen inalterados (no se utilizan al operar con enteros), figura 2.8.

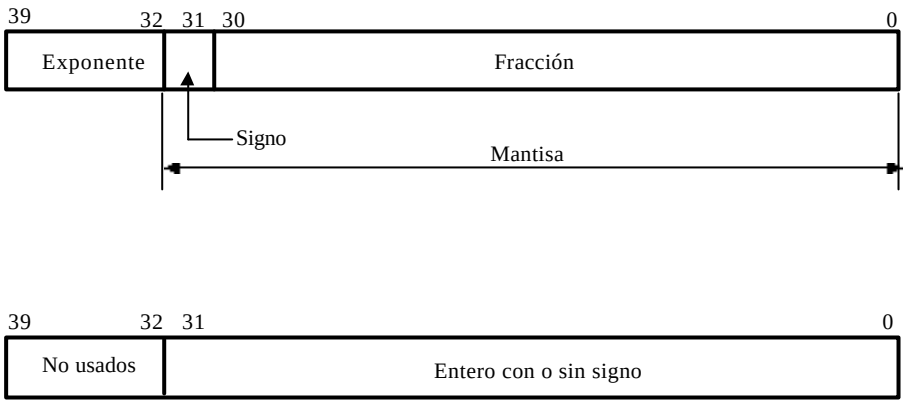


Figura 2.8. Formato en punto flotante y entero (con o sin signo), de los registros de precisión extendida (R0-R7).

2. **Registros auxiliares (AR0-AR7)**. Pueden ser modificados por las *ARAUs*, la ALU y el multiplicador. Son registros de propósito general y 32 bits que pueden emplearse como operandos de tipo entero en operaciones aritméticas o como generadores de direcciones de 24 bits.

3. **Puntero a la página de datos (DP)**. Es un registro de 32 bits de los cuales sólo los 8 bits menos significativos son utilizados. El registro se usa, para acceder a operandos utilizando el modo de direccionamiento directo, como selector o puntero selector a una página de datos (definida con los 8 LSBs del registro) a la que se está accediendo en un instante dado. Cada página ocupa 64 *Kwords* de 32 bits de memoria existiendo un total de 256 páginas (la memoria tiene una capacidad de direccionamiento total de 16 *Mwords*).

4. **Registros de índice (IR0 e IR1)**. Son registros de 32 bits empleados por las ARAUs para calcular las direcciones de los datos en modo de direccionamiento indexado.

5. **Registro de dimensión de bloque (BK)**. Es un registro de 32 bits utilizado por las ARAUs para fijar la dimensión del bloque de datos en el modo de direccionamiento denominado circular (se verá en el tema siguiente). Está especialmente diseñado para implementar colas mediante la técnica de matriz circular, donde determina el tamaño de la

matriz. La implementación de colas circulares gestionadas automáticamente por hardware (de forma casi transparente al usuario) por el DSP es una de las características que ofrecen este tipo de microprocesadores, pensados para realizar, en tiempo real, algoritmos de tipo repetitivos (FFTs, filtros digitales...).

6. **Puntero a la pila (SP)**. Como en cualquier microprocesador, este DSP dispone de una pila. El registro SP es un registro de 32 bits que contiene la dirección de la cima de la pila (apunta al último elemento introducido en la pila). Este registro es alterado por interrupciones, excepciones de tipo software, llamadas y retornos de subrutinas e instrucciones de tipo PUSH y POP (almacena datos en la pila o recoge datos de la pila).

7. **Registro de estado (ST)**. Contiene información global sobre el estado de la CPU en cada instante. Su estructura se muestra en la figura 2.9. En la tabla 2-2 se definen cada uno de los bits de este registro. En este registro aparecen 8 bits (del bit 0 o LSB al bit 7) activados por hardware por la parte de manejo de datos de la CPU. Cada vez que se realiza alguna operación aritmético-lógica, en la que intervengan la ALU o el multiplicador, se activa o no alguno de estos bits. El resto son bits de control del periférico caché (bits CF, CE y CC), del modo de repetición de instrucciones (bit RM) y para la habilitación global de las interrupciones (bit GIE).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xx	xx	GIE	CC	CE	CF	xx	RM	OVM	LUF	LV	UF	N	Z	V	C
		R/W	R/W	R/W	R/W		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

xx = Bit Reservado
R = Bit accesible en lectura
W = Bit accesible en escritura

Figura 2.9. Registro de Estado (ST).

Bit	Nombre	Valor de Reset	Función que realiza
0	C	0	Bandera de acarreo (Carry flag). a) Cuando se realiza una operación aritmética de suma con manejo de enteros, C=1 si se produce acarreo en el MSB del resultado. Si se trata de una resta, C=1 si es necesario introducir un bit en el MSB del resultado. b) Cuando se realiza una operación con flotantes, el acarreo no se ve afectado. c) Cuando se realizan operaciones de desplazamiento sobre un registro, el bit de carry se actualiza con el valor desplazado fuera del registro.
1	V	0	Bandera de desbordamiento (Overflow flag). Si se produce desbordamiento, V=1. El desbordamiento de entero implica que el resultado de la operación de enteros es menor que -2^{31} o mayor de $2^{31}-1$. En el caso de flotante, se produce desbordamiento cuando el resultado de una operación de flotantes tiene un exponente mayor de 127.
2	Z	0	Bandera de cero (Zero flag). Si el resultado de una operación es cero, Z=1.
3	N	0	Bandera de negativo (Negative flag). Si el resultado de una operación es negativo, N=1.
4	UF	0	Desbordamiento flotante por debajo (Underflow flag). Sólo para operaciones con flotantes. Se produce desbordamiento por debajo cuando el resultado de una operación de flotantes tiene un exponente menor o igual de –128 y en ese caso, UF=1.

5	LV	0	Bandera de desbordamiento latcheado (Latched overflow flag). Si se ha producido alguna vez un desbordamiento de tipo V, LV=1. Sólo puede borrarlo el usuario escribiendo un 0.
6	LUF	0	Latched floting-point underflow flag. Si se ha producido alguna vez un desbordamiento de tipo UF, LUF=1. Sólo puede borrarlo el usuario escribiendo un 0.
7	OVM	0	Bandera de modo de desbordamiento (Overflow Mode flag). Esta bandera afecta exclusivamente a las operaciones enteras. Si OVM=0, el entero que resulte del desbordamiento no es tratado de manera especial. Si es OVM=1: a) Los enteros que desborden en dirección positiva se saturan al mayor número positivo en complemento a dos (7FFFFFFFh). b) Los enteros que desborden en dirección negativa se saturan al número más negativo en complemento a dos (80000000h). Nota: las funciones V y LV son independientes del valor de OVM.
8	RM	0	Bandera de modo de repetición. Si RM=1, el contador de programa se está modificando en los modos de repetición, tanto de bloque como simple.
9	Reservado	0	Se lee siempre un cero.
10	CF	0	Fija la Caché. Cuando CF=1, se impide su modificación, esté (se permite el acceso a recoger instrucciones pero no se permite que se actualice) o no habilitada. Se puede realizar un Caché Clear si CF=1.
11	CE	0	Habilita la Caché cuando CE=1. Se puede realizar un Caché Clear si CE=0.
12	CC	0	Caché Clear. Limpia la Caché. Si CC=1, se reinicializa la Caché, aunque ésta se encuentre congelada (CF=1).
13	GIE	0	Habilitación global de interrupciones (Global Interrupt Enable).
15-14	Reservado	0	Se leen siempre ceros.
31-16	Reservado	0-0	Valores Indefinidos.

Tabla 2-2. Funciones de los bits del registro de estado.

8. **Registro de habilitación de interrupciones de CPU y de programación del sincronismo de la DMA (IE)**. Cuando un bit del registro se pone a cero, se inhabilita la interrupción de CPU asociada a él o se impide la sincronización con dicha interrupción de

los eventos (escritura y/o lectura) del periférico interno DMA[4]. Obsérvese que todas las **interrupciones** de tipo hardware que tiene disponibles el usuario en este microprocesador son **enmascarables**.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
xx	xx	xx	xx	xx	EDINT (DMA)	ETINT1 (DMA)	ETINT0 (DMA)	ERINT1 (DMA)	EXINT1 (DMA)	ERINT0 (DMA)	EXINT0 (DMA)	EINT3 (DMA)	EINT2 (DMA)	EINT1 (DMA)	EINT0 (DMA)
					R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xx	xx	xx	xx	xx	EDINT (CPU)	ETINT1 (CPU)	ETINT0 (CPU)	ERINT1 (CPU)	EXINT1 (CPU)	ERINT0 (CPU)	EXINT0 (CPU)	EINT3 (CPU)	EINT2 (CPU)	EINT1 (CPU)	EINT0 (CPU)
					R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

xx = Bit Reservado
R = Bit accesible en lectura
W = Bit accesible en escritura

Figura 2.10. Registro de habilitación de interrupciones (IE).

Bit	Nombre	Valor de Reset	Función que realiza
0	EINT0	0	Habilita la interrupción externa 0 (CPU) si vale 1.
1	EINT1	0	Habilita la interrupción externa 1 (CPU) si vale 1.
2	EINT2	0	Habilita la interrupción externa 2 (CPU) si vale 1.
3	EINT3	0	Habilita la interrupción externa 3 (CPU) si vale 1.
4	EXINT0	0	Habilita la interrupción de transmisión por el puerto serie 0 (CPU) si vale 1.
5	ERINT0	0	Habilita la interrupción de recepción por el puerto serie 0 (CPU) si vale 1.
6	EXINT1 (*)	0	Habilita la interrupción de transmisión por el puerto serie 1 (CPU) si vale 1.
7	ERINT1 (*)	0	Habilita la interrupción de recepción por el puerto serie 1 (CPU) si vale 1.
8	ETINT0	0	Habilita la interrupción del Timer 0 (CPU) si vale 1.
9	ETINT1	0	Habilita la interrupción del Timer 1 (CPU) si vale 1.
10	EDINT	0	Habilita la interrupción del controlador de DMA (CPU) si vale 1.
15-11	Reservados	0-0	Valores indefinidos.
16	EINT0	0	Habilita la sincronización del evento DMA con la interrupción externa 0 si vale 1.
17	EINT1	0	Habilita la sincronización del evento DMA con la interrupción externa 1 si vale 1.

[4] La fuente de cualquier interrupción puede emplearse para generar un evento CPU (ruptura de la secuencia normal de ejecución del programa para ejecutar una rutina de servicio de la interrupción) o para generar eventos DMA (sincronización de la transferencia de datos que gestiona y realiza el periférico interno DMA).

18	EINT2	0	Habilita la sincronización del evento DMA con la interrupción externa 2 si vale 1.
19	EINT3	0	Habilita la sincronización del evento DMA con la interrupción externa 3 si vale 1.
20	EXINT0	0	Habilita la sincronización del evento DMA con la interrupción de transmisión del puerto serie 0 si vale 1.
21	ERINT0	0	Habilita la sincronización del evento DMA con la interrupción de recepción del puerto serie 0 si vale 1 si vale 1.
22	EXINT1 (*)	0	Habilita la sincronización del evento DMA con la interrupción de transmisión del puerto serie 1 si vale 1.
23	ERINT1 (*)	0	Habilita la sincronización del evento DMA con la interrupción de recepción del puerto serie 1 si vale 1.
24	ETINT0	0	Habilita la sincronización del evento DMA con la interrupción del Timer0 si vale 1.
25	ETINT1	0	Habilita la sincronización del evento DMA con la interrupción del Timer1 si vale 1.
26	EDINT	0	Habilita la sincronización del evento DMA con la interrupción del propio controlador de DMA si vale 1.
31-27	Reservados	0-0	Valores indefinidos.

Tabla 2-3. Funciones de los bits del registro de habilitación de interrupciones. (*) No disponibles en el TMS320C31.

9. **Registro de las banderas de petición de las interrupciones (IF)**. Cuando alguno de sus bits se pone a uno, es porque se ha producido la interrupción correspondiente. Mientras la CPU no atienda la rutina de servicio de dicha interrupción, la bandera asociada a la misma en el registro IF permanece activa. En cuanto se acceda a la rutina de servicio, la bandera correspondiente se pone a cero, indicando que no se encuentra pendiente la ejecución de la rutina de servicio de la interrupción.

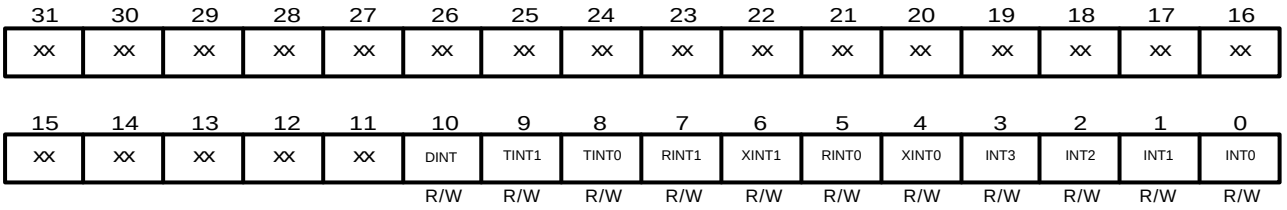


Figura 2.11. Registro de petición de servicio de rutinas de interrupción (IF).

Bit	Nombre	Valor de Reset	Función que realiza
0	INT0	0	Bandera de interrupción externa 0.
1	INT1	0	Bandera de interrupción externa 1.
2	INT2	0	Bandera de interrupción externa 2.
3	INT3	0	Bandera de interrupción externa 3.
4	XINT0	0	Bandera de interrupción de transmisión por el puerto serie 0.
5	RINT0	0	Bandera de interrupción de recepción por el puerto serie 0.
6	XINT1(*)	0	Bandera de interrupción de transmisión por el puerto serie 1.
7	RINT1(*)	0	Bandera de interrupción de recepción por el puerto serie 1.
8	TINT0	0	Bandera de interrupción del Timer0.
9	TINT1	0	Bandera de interrupción del Timer1.
10	DINT	0	Bandera de interrupción del controlador de DMA.
31-11	Reservados	0-0	Valores indefinidos.

Tabla 2-4. Funciones de los bits del registro de banderas de interrupciones activas. (*) No disponibles en el TMS320C31.

10. **Registro de control de las entradas y salidas de propósito general (IOF)**. Controla los pines XF0 y XF1 del DSP (obsérvese que el TMS320C30, en principio, tan sólo dispone de dos líneas digitales de entrada y/o salida de propósito general). El usuario utiliza este registro para indicar la función de los pines XF0 y XF1 (si son líneas de entrada o salida al ASIC) y para leer el valor digital asociado a las dos líneas o escribir en ellas.

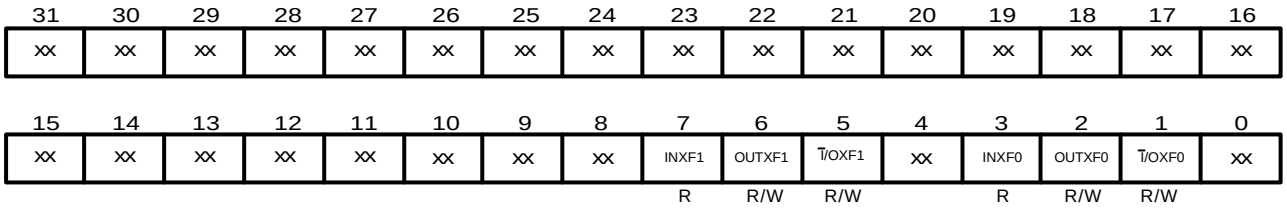


Figura 2.12. Registro de control de los pines XF0 y XF1 (IOF).

Bit	Nombre	Valor de Reset	Función que realiza
0	Reservado	0	Se lee siempre un cero.
1	I/OXF0	0	Si I/OXF0=0, XF0 se configura como pin de entrada de propósito general. Si I/OXF0=1, XF0 se configura como pin de salida de propósito general.
2	OUTXF0	0	Dato que sale por XF0 (Cuando I/OXF0=1).
3	INXF0	0	Dato que se lee de XF0 (Cuando I/OXF0=0).
4	Reservado	0	Se lee siempre un cero.
5	I/OXF1	0	Si I/OXF1=0, XF1 se configura como pin de entrada de propósito general. Si I/OXF1=1, XF1 se configura como pin de salida de propósito general.
6	OUTXF1	0	El dato que sale por XF1 (Cuando I/OXF1=1).
7	INXF1	0	Dato que se lee de XF1 (Cuando I/OXF1=0).
31-8	Reservado	0-0	Se leen siempre ceros.

Tabla 2-5. Funciones de los bits del registro de control de XF0 y XF1.

4. OPERACIONES INTERNAS DEL BUS

Por operaciones internas del bus se entiende el flujo de instrucciones y datos que existe dentro del DSP así como el conjunto de operaciones asociadas a dicho flujo. Este flujo está gobernado por dos subsistemas internos al micro, la CPU (analizada previamente) y la DMA (que se estudiará en detalle en temas posteriores y que se puede entender como un sistema equivalente a la CPU pero que se encarga únicamente del manejo de datos, nunca de las instrucciones).

La familia de DSPs TMS320C3x establece, para completar la ejecución de cada instrucción del sistema, cuatro fases, microinstrucciones u operaciones básicas, asociadas a la CPU, para procesar completamente una instrucción:

1. Microinstrucción ***FETCH*** o de recogida. Hace referencia, como su nombre indica, a la fase o período de tiempo en que la CPU recoge la instrucción de la zona de memoria de programa y actualiza el registro contador de programa.
2. Microinstrucción ***DECODE*** o de decodificación. Hace referencia a la fase o período de tiempo en que la CPU decodifica la instrucción. Determina la secuencia de operaciones que debe realizar el sistema para completar la ejecución de la instrucción y las direcciones de los operandos, si fuesen necesarios, que tiene asociados la instrucción. En esta fase, por tanto, la CPU genera las direcciones de los datos, controla las modificaciones que se precisen en los registros SP y AR0-AR7.
3. Microinstrucción ***READ*** o de lectura de operandos. Hace referencia a la fase o período de tiempo en que la CPU accede a los operandos, cuyas direcciones se actualizaron en la fase anterior, sobre los que opera la instrucción.
4. Microinstrucción ***EXECUTE*** o de ejecución. Hace referencia a la fase o período de tiempo en que la CPU procede a la ejecución de la instrucción. Incluye el cálculo y las operaciones de almacenamiento de los resultados. Si fuese necesario, en esta fase, la CPU accede a recoger los operandos que sean registros internos de la CPU y almacena los resultados en dichos registros o en la zona de memoria de datos

Dispone, además, de una microinstrucción asociada a los eventos de DMA o microinstrucción ***DMA***, relacionada con los eventos de DMA. Esta microinstrucción se ejecuta independiente a los eventos asociados a la CPU. Hace referencia a las operaciones de escritura o lectura de memoria asociadas a los periféricos DMA. Tiene la característica, por tanto, de operar en paralelo con la CPU.

La estructura interna de tipo Harvard en estos DSPs, con la existencia de varios buses internos, posibilita la denominada estructura ***pipelining*** en la ejecución de las instrucciones

que consiste en la ejecución, en paralelo, de los niveles Fetch, Decode, Read y Execute de varias instrucciones básicas. En la figura 2.13 se muestran los pasos básicos asociados a la ejecución de una instrucción y la estructura de *pipeline*. Se pretende la ejecución, si es posible y en el mismo periodo de tiempo, de las cuatro microinstrucciones que *tocan* asociadas a otras tantas instrucciones: *overlaping* o *solapamiento perfecto de instrucciones*.

Ciclo	Fetch	Decode	Read	Execute
k-3	w	-	-	-
k-2	x	w	-	-
k-1	y	x	w	-
k	z	y	x	w
k+1	-	z	y	x
k+2	-	-	z	y
k+3	-	-	-	z

Solapamiento perfecto de instrucciones (*Overlaping*)

Figura 2.13. Estructura *pipeline*: Secuencia de operaciones básicas para la ejecución de cuatro instrucciones (w, x, y, z).

El *overlaping*, ejecución en paralelo pero en fases de microinstrucción diferentes de cuatro instrucciones, permite una gran velocidad de ejecución, hasta el valor máximo de 1 instrucción/ciclo máquina. La familia TMS320C3x ofrece una estructura de control de *pipeline* transparente al usuario, que gestiona todos los posibles conflictos que puedan aparecer. El conflicto aparece cuando una instrucción, que ha ejecutado ya un nivel de *pipeline*, pretende pasar a otro nivel o microinstrucción en el que va a emplear unos recursos que no están aún listos (entender las microinstrucciones como procesos que consumen recursos que han de ser liberados para que otra instrucción pueda hacer uso de los mismos). La instrucción que está en el nivel de *pipeline* de prioridad más baja permanece congelada hasta que la que esté en el orden más alto de prioridad, que antecede en el tiempo a la anterior, libere el recurso que está usando y que precisa la primera instrucción.

En la figura 2.14 aparecen los órdenes de prioridades establecidos para los diferentes recursos o microinstrucciones de la estructura *pipeline*.

Obsérvese que la DMA puede presentar, en los accesos a periféricos de almacenamiento internos, problemas de conflicto a pesar de tener recursos, buses de dirección y datos internos, independientes de los de la CPU (aunque dispone de buses independientes de la CPU, los buses de la CPU y de la DMA están multiplexados en el acceso a los periféricos internos del sistema). Ejemplos de conflictos *pipeline* se pueden nombrar instrucciones de salto, acceso a registros y de acceso a posiciones de memoria. En todos los casos, estos conflictos son transparentes al usuario, los resuelve internamente el propio DSP. Al usuario **sólo** le **afecta en que ralentiza el tiempo de ejecución del programa** puesto que los conflictos se solucionan parando la ejecución de la instrucción que no puede acceder al recurso que está usando otra.

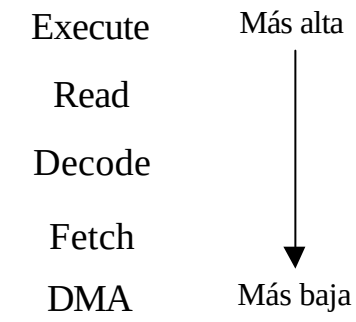


Figura 2.14. Órdenes de prioridades en los recursos de la estructura pipeline.

5. ORGANIZACIÓN DE LA MEMORIA

El espacio total de memoria que es capaz de direccionar la familia TMS320C3x de Texas Instruments, es de 16M (unos 16 millones) de palabras de 32 bits. Dentro de este espacio total que es capaz de direccionar el DSP se incluyen las zonas de memoria internas de las que dispone el sistema (memoria RAM interna, ROM interna y registros de configuración de periféricos internos). En la figura 2.15, se muestran los buses y la estructura de acceso a memoria del sistema.

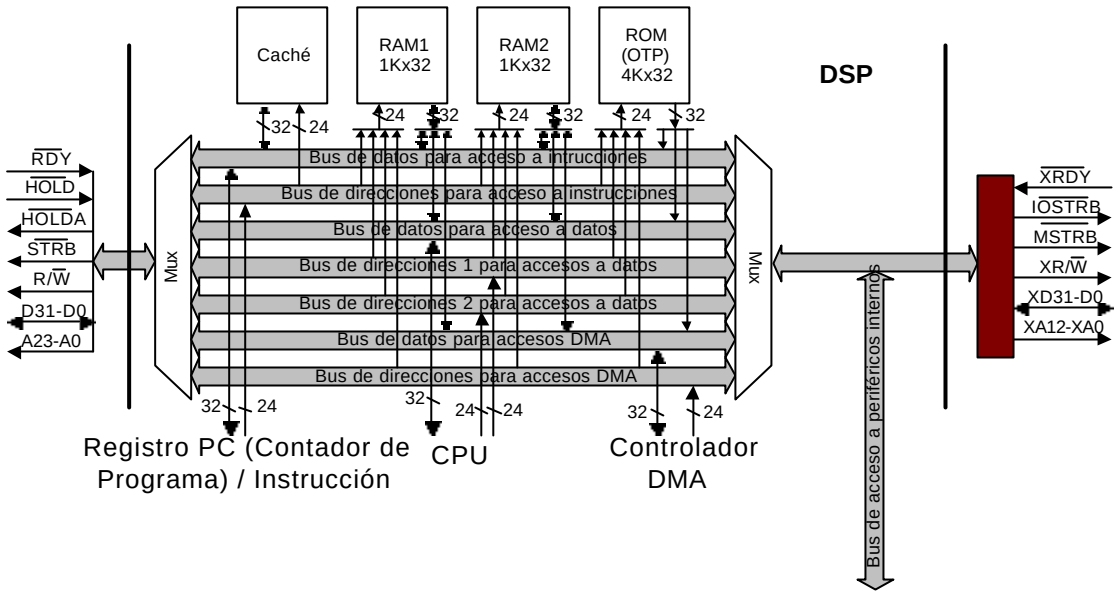


Figura 2.15. Acceso a memoria: Buses del sistema.

El sistema dispone de dos conjuntos de líneas para acceso al exterior, el denominado **bus principal** (formado por un bus de datos de 32 bits, un bus de direcciones de 24 bits y un bus de control de 5 bits) y el **bus de expansión** (formado por un bus de datos de 32 bits, un bus de direcciones de 13 bits y un bus de control de 4 bits). En la figura 2.16 se muestran los mapas de memoria del TMS320C30 y del TMS320C31. Se observa que cada DSP tiene dos posibles mapas de memoria. La diferencia estriba en la posibilidad de acceso o no a la ROM interna del sistema. La forma de seleccionar uno u otro mapa de memoria es imponiendo un nivel alto o bajo en el pin $MCBL/\overline{MP}$ de entrada a DSP (si $MCBL/\overline{MP}=0$, el sistema funciona en modo microprocesador arrancando desde memoria externa mientras que si $MCBL/\overline{MP}=1$, el sistema funcionaría en modo microcomputador arrancando desde la memoria interna de tipo ROM-OTP).

Obsérvese que el TMS320C31 no dispone de bus de expansión, en el mapa de memoria no hay zonas asignadas al bus de expansión, y que la ROM interna de tipo OTP, ya viene programada de fábrica conteniendo un programa denominado *Boot-Loader* que describiremos más adelante.

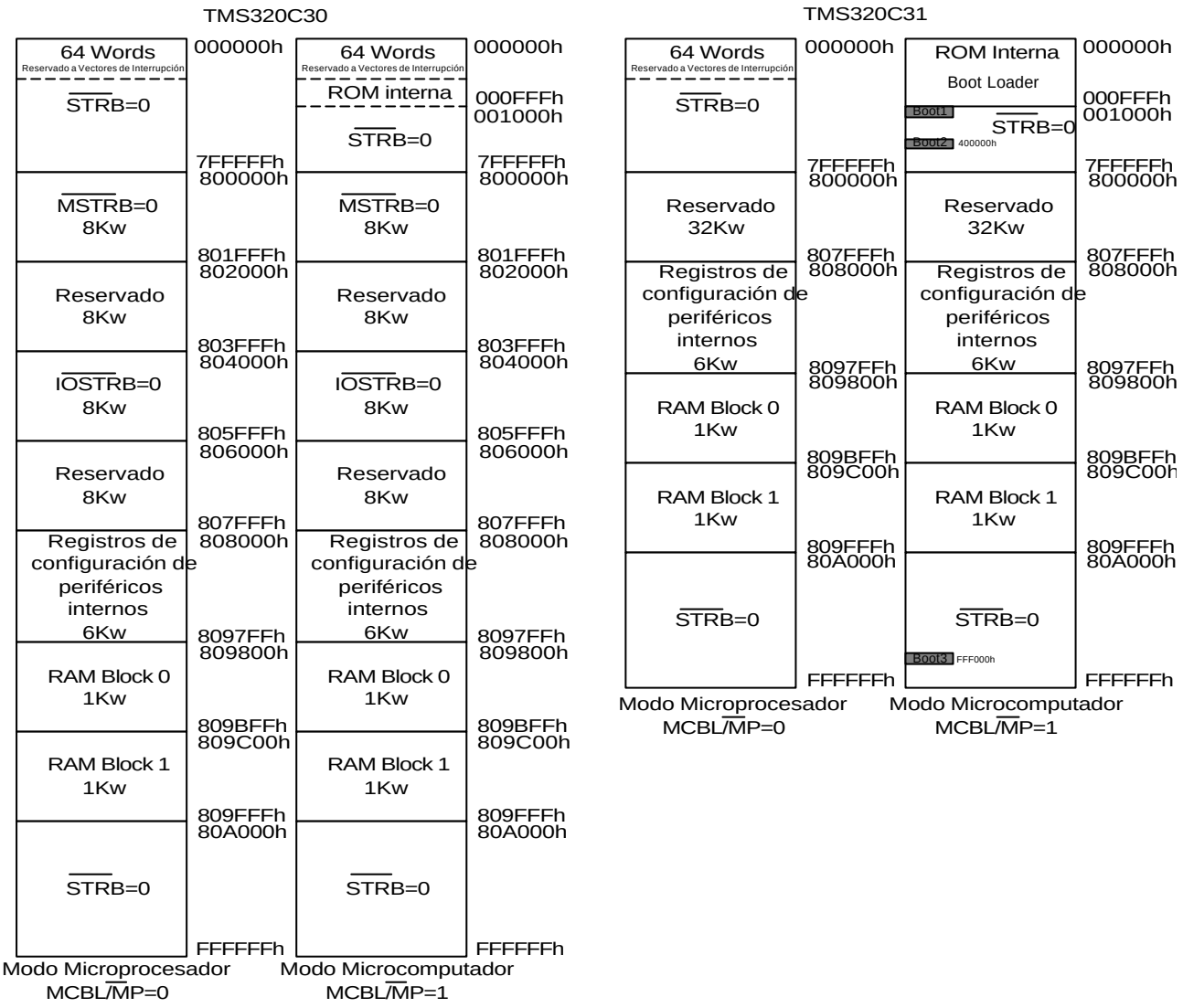


Figura 2.16. Mapas de Memoria de la familia TMS320C3x. Obsérvese que la única diferencia entre modo microcomputador y microprocesador es la habilitación o inhabilitación, en la zona baja del mapa de memoria, del periférico interno de memoria no volátil (ROM).

Como conclusión, destacar que el DSP posee dos modos de operación:

1. **Microprocesador**: El sistema arranca autovectorizando desde la posición 000000H, que debe coincidir en este caso con dispositivos externos de memoria dispuestos (rutados y colocados en el mapa de memoria del sistema microprocesador) y programados por el usuario.

2. **Microcomputador**: El sistema arranca autovectorizando desde la posición 000000H que en este caso coincide con la ROM interna del sistema. Obsérvese que para que el sistema TMS320C30 funcione, será preciso que el fabricante, Texas Instruments, programe en la fábrica, antes de entregar el DSP al usuario, la ROM interna de tipo OTP (*One Time Programmable*). Este proceso requiere que el fabricante genere máscaras especiales para programar la ROM interna con el programa suministrado por un usuario particular. Esto supone el encarecimiento del coste de fabricación del DSP, de forma que sólo será factible para usuarios que requieran un volumen muy elevado de DSPs, nunca para un usuario que sólo desee un DSP programado de fábrica. Para el TMS320C31 el caso es diferente, la ROM interna ya está programada de fábrica (incorpora un programa denominado *Boot-Loader*) y no es posible contactar con el fabricante para su reprogramación (ROM de tipo OTP). Posteriormente analizaremos el programa *Boot-Loader*.

En la figura 2.17 se detalla la zona del mapa de memoria en la que se ubican los distintos periféricos internos que tienen los diferentes miembros de la familia TMS320C3x.

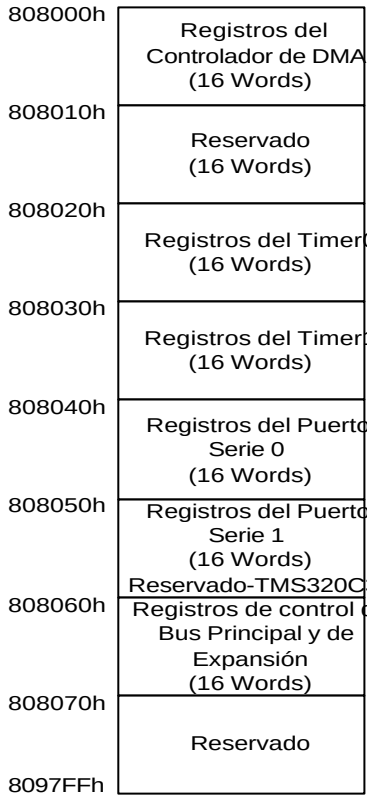


Figura 2.17. Mapa de memoria de los registros de control de los periféricos internos de los DSPs de la familia TMS320C3x.

5.1 *Reset y Vectores de Interrupción.*

Independientemente del modo de funcionamiento del DSP (TMS320C30 ó TMS320C31), la secuencia de operaciones siguientes a un reset hardware del sistema (aplicación de un 0 lógico en la entrada $\overline{RESET}$ del circuito integrado, reset externo no-enmascarable) es siempre la misma. El sistema recoge el vector de reset (posición 000000h en el mapa de memoria del DSP) y copia el valor que existe en ese registro (que se ubica en un periférico de memoria externa suponiendo que el sistema funciona en modo microprocesador o en la ROM interna, si el sistema funciona en modo microcomputador) en el registro de contador de programa o PC de la CPU. Se dice por tanto que el sistema tiene el reset autovectorizado a la posición absoluta 000000h del mapa de memoria del DSP, dado que cuando aparece el reset, el sistema accede a la posición 000000h para recoger la dirección absoluta de la primera

instrucción a ejecutar. Al conectar la alimentación al DSP, el estado de los registros es desconocido. La función fundamental del reset, es llevar los registros del sistema a un estado inicial conocido, para lo cual es preciso mantener a nivel bajo (0 lógico) el pin de entrada $\overline{RESET}$ al circuito integrado durante al menos 10 ciclos de reloj del sistema. Un reset genera la siguiente reacción de los registros y líneas de entrada y salida del circuito integrado:

- Accesos al exterior: Las líneas de los buses de dirección y de datos ($D_0...D_{31}$, $A_0...A_{23}$, $XD_0...XD_{31}$, $XA_0...XA_{12}$) se ponen en alta impedancia, así como las líneas de control del tipo de acceso al exterior ($R/\overline{W}$, $XR/\overline{W}$). Las líneas $\overline{STRB}$, $\overline{MSTRB}$ y $\overline{IOSTRB}$ se ponen a 1. Los registros de control asociados al bus principal y de expansión se inicializan.
- Las líneas de los periféricos internos (temporizadores y puertos serie) así como las entradas o salidas digitales, $XF0$ y $XF1$ se ponen en alta impedancia. Los periféricos internos al DSP y sus registros de control se inicializan.
- Los registros internos de la CPU se inicializan: IE=0, ST=0, IF=0, IOF=0, etc.

Al hablar de interrupciones en un microprocesador se distingue entre fuente que genera la interrupción y servicio que provoca:

1. **Fuentes de interrupción**. La forma de atención de las interrupciones es análoga al proceso de reset. Los DSPs de la familia TMS320C3x disponen de cuatro fuentes de interrupciones externas ($\overline{INT0}...\overline{INT3}$), dos fuentes de interrupción asociadas a los periféricos internos puertos serie síncronos, una fuente asignada a la transmisión y otra a la recepción (XINT0, RINT0 para el puerto serie 0 y XINT1, RINT1 para el puerto serie 1 que no está disponible en el TMS320C31), así como una fuente de interrupción asociada a cada uno de los periféricos internos restantes, temporizadores y controlador de DMA (TINT0, TINT1 y DINT).

El esquema de generación de una interrupción en un DSP de esta familia, se muestra en las figuras 2.19 y 2.20. Las interrupciones externas ($\overline{INTn}$) son activas por nivel bajo y se sincronizan con el ciclo máquina del sistema, H1 ó H3. Para garantizar que son detectadas, deben estar activas al menos durante 1 ciclo de H1 y para ser detectadas sólo una vez, deberán desactivarse antes de dos ciclos de reloj H1. Si permanecen activas más de dos ciclos de H1, pueden ser atendidas más de una vez (se generan lo que se conoce como interrupciones espúreas o indeseadas).

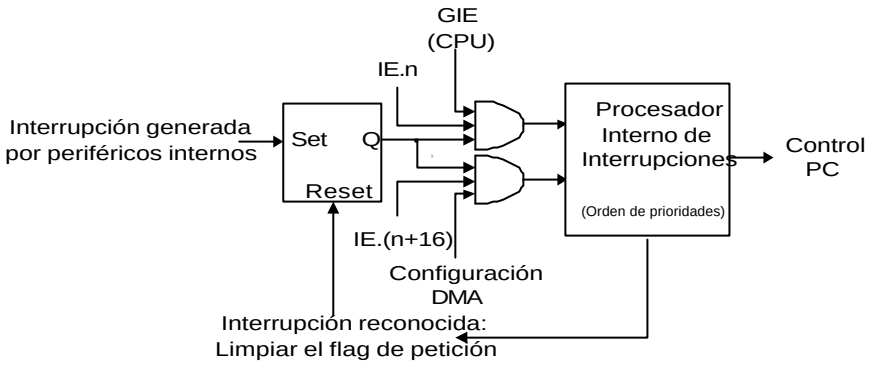


Figura 2.18. Esquema de generación de interrupciones internas.

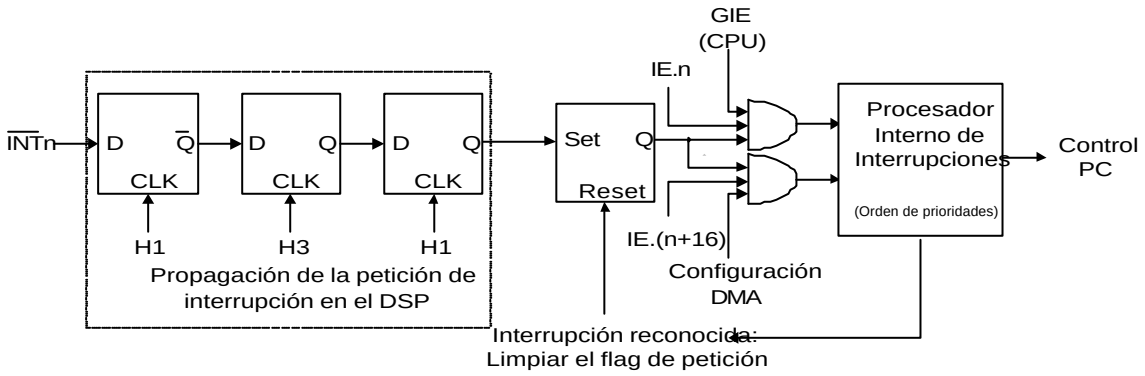


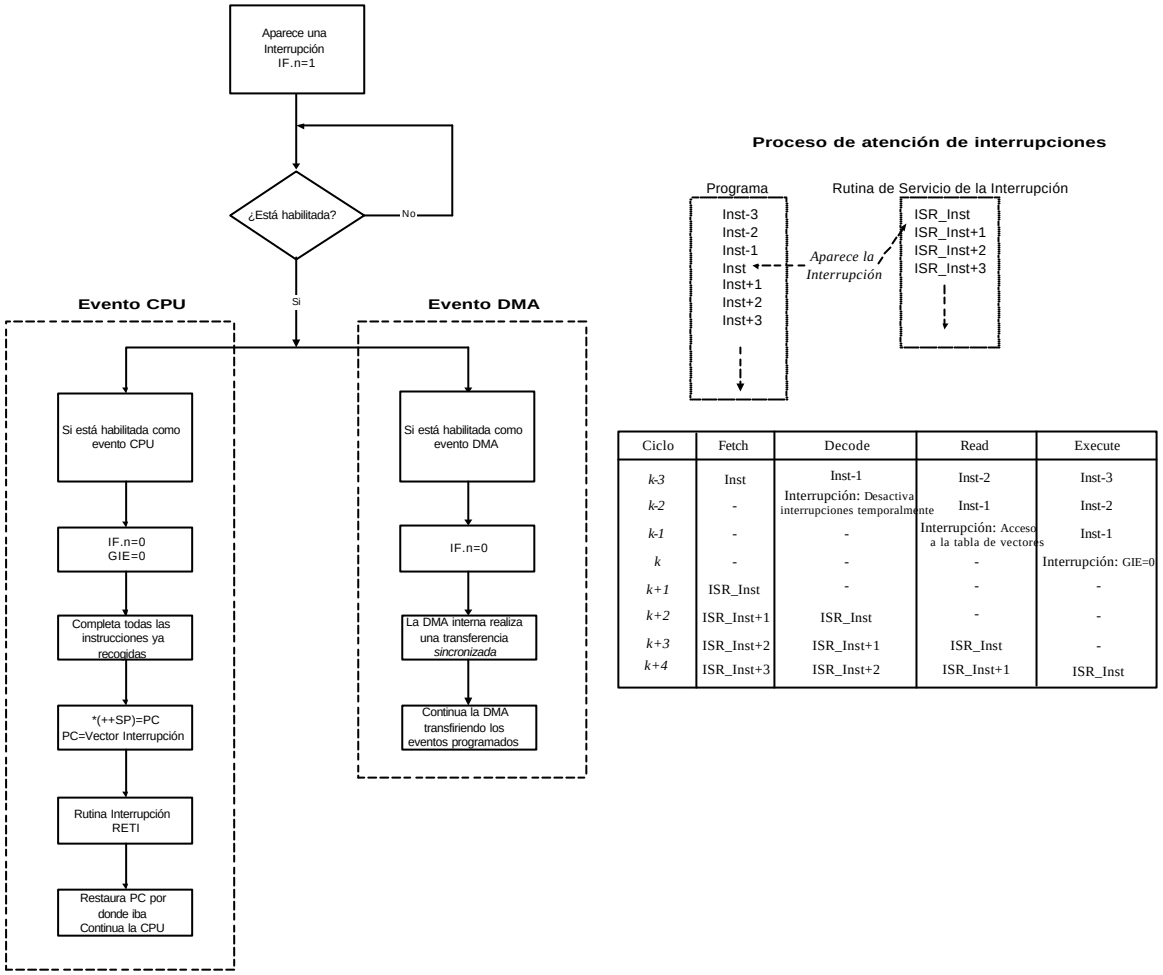
Figura 2.19. Esquema de generación de interrupciones externas.

2. **Servicio de interrupciones**. Cuando una interrupción es detectada (el bit correspondiente a dicha interrupción en el registro IF se pone a 1) el sistema reacciona como se muestra en el diagrama de flujo de la figura 2.21.

- Si la interrupción tiene habilitada la generación de eventos CPU (está a 1 el bit correspondiente del registro IE y además el bit de habilitación global de interrupciones está también a 1) la instrucción que, cuando se detecta la interrupción, está siendo recogida no se ejecuta. La dirección de esta instrucción, registro PC, es almacenada en la dirección apuntada por el registro SP, puntero a la cima de la pila. Inmediatamente después de almacenar en la pila el PC, se pone a cero el bit GIE del registro ST (se desactiva el bit de habilitación global de las interrupciones) y se recoge el vector de interrupción asociado a la interrupción detectada que se almacena en PC. Se empieza, por tanto, a ejecutar la rutina de servicio de la interrupción. Al final de la rutina de servicio de la interrupción deberá realizarse el retorno del contador de programa a la dirección de la instrucción que no se ejecutó. El retorno normal de la rutina de servicio lo realiza la instrucción RETI, que almacena en el registro PC el dato que aparece en la cima de la pila (deberá coincidir con el valor del registro PC que se almacenó en la pila al comenzar el proceso de atención de la interrupción) y que reactiva el bit GIE.

- Si la interrupción tiene habilitada la generación de eventos DMA, está a 1 el bit correspondiente del registro IE y además se permite la sincronización de eventos DMA por interrupciones, sirve para lanzar las transferencias de tipo DMA programadas (se verá en el tema 17).

Obsérvese, por tanto, que una interrupción puede generar un evento de tipo CPU (ruptura de la secuencia normal de ejecución en un programa) y otro de DMA. Ambos procesos son totalmente independientes y se ejecutan en paralelo.



Ciclo	Fetch	Decode	Read	Execute
k-3	Inst	Inst-1	Inst-2	Inst-3
k-2	-	Interrupción: Desactiva interrupciones temporalmente	Inst-1	Inst-2
k-1	-	-	Interrupción: Acceso a la tabla de vectores	Inst-1
k	-	-	-	Interrupción: GIE=0
k+1	ISR_Inst	-	-	-
k+2	ISR_Inst+1	ISR_Inst	-	-
k+3	ISR_Inst+2	ISR_Inst+1	ISR_Inst	-
k+4	ISR_Inst+3	ISR_Inst+2	ISR_Inst+1	ISR_Inst

Figura 2.20. Procesamiento de un evento interrupción: Posibilidad de generar eventos CPU y DMA en paralelo.

Comentarios relacionados con el proceso de atención de interrupciones en la familia TMS320C3x:

1. El flag GIE (del registro ST de la CPU) se pone a 0 al comenzar el proceso de atención de una interrupción. Por tanto, y en principio, no se produce anidamiento entre interrupciones. No es posible que, mientras se esté ejecutando la rutina de servicio de una interrupción, al producirse una nueva interrupción, ésta última se atienda antes de terminar de ejecutarse la rutina de servicio de la primera interrupción que apareció. Al terminar de ejecutar la rutina de servicio de la interrupción habrá que devolver el registro PC al valor

que tenía antes de la llegada de la interrupción y, si se desea permitir la generación de nuevas interrupciones habrá que volver a activar el bit GIE (instrucciones RETI o RETS). Obsérvese que, dentro de la propia rutina de servicio de las interrupciones, el usuario puede poner, vía software, el bit GIE a 1 permitiendo el anidamiento de las interrupciones.

2. Los accesos en escritura al registro IF pueden causar fallos en la generación de las interrupciones (si se ha generado una interrupción y está lista para ser latcheada en IF a la par que se escribe en el mismo registro IF, el proceso de escritura tiene más prioridad, perdiéndose la interrupción generada lo que es especialmente grave si la interrupción generada lo fue por un periférico interno como la DMA). El fabricante desaconseja el uso del *polling* o generación por software (escritura de 1´s lógicos) en los flags del registro IF para provocar interrupciones así como su desactivación por software (escritura de 0´s lógicos), aconsejando la generación de la interrupción y dentro de la rutina de servicio la distinción entre interrupción real o espúrea.

3. La señal que genere una interrupción externa debe tener un 0 lógico (permanecer activa) un tiempo comprendido entre 1 y 2 ciclos H1 ó H3 (entre 2 y 4 ciclos de reloj del sistema) para garantizar que no se produzca más de una interrupción externa en el sistema (el flags del registro IF correspondiente a la interrupción externa lo limpia automáticamente la CPU generando una señal de reset asíncrono al entrar en la rutina de servicio de la interrupción).

4. La atención de las interrupciones provoca que no se ejecute la instrucción que entre en el momento de la aparición de la interrupción en su fase de recogida (Fetch en la estructura de pipeline) siempre que ésta se encuentre en su primer ciclo máquina de recogida. Si la instrucción está en su fase de recogida pero en un ciclo máquina superior al primero (debido, por ejemplo, a la aparición de conflictos en la estructura *pipeline*) se termina de ejecutar antes de atenderse la rutina de servicio de la interrupción. Todo esta casuística, la gestiona la CPU de forma y manera transparente al usuario.

5. Posibilidad de generar interrupciones de tipo software (excepciones). Aparte de las interrupciones antes mencionadas. Se pueden generar hasta 32 interrupciones software empleando la instrucción “TRAPcond n”, con 0<n<31, que generaría una ruptura programada de la secuencia normal de control de forma análoga a como sucede en el

caso de interrupciones de tipo hardware. Este tipo de interrupciones se generan independientemente del valor (0 ó 1) de GIE. Una instrucción de tipo TRAP, provoca además que el flag GIE se ponga a 0 inicialmente (sea o no cierta la condición ***cond***, para generación de la excepción software), lo que puede provocar un error si posteriormente se produce un conflicto en la estructura de pipeline, en las fases de lectura (Read) o decodificación (Decode), asociadas a la propia instrucción TRAP, que dejaría GIE=0 luego de ejecutarse la instrucción, aunque no se cumpla la condición ni se genere la interrupción software. Para evitar este error, el fabricante recomienda la inserción de dos instrucciones NOP antes del uso de la instrucción TRAP. Secuencias de operaciones que pueden generar este error son:

```
Ejemplo1:   LDI  memoria,SP
            TRAPcond  n
Ejemplo2:   LDI  memoria,SP
            NOP
            TRAPcond  n
Ejemplo3:   STI  SP,memoria
            TRAPcond n
Ejemplo4:   STI  memoria,SP
            NOP
            TRAPcond  n
```

En la figura 2.22 se muestra el mapa de memoria de los vectores de reset, interrupciones y excepciones software en la familia TMS320C3x de Texas Instruments.

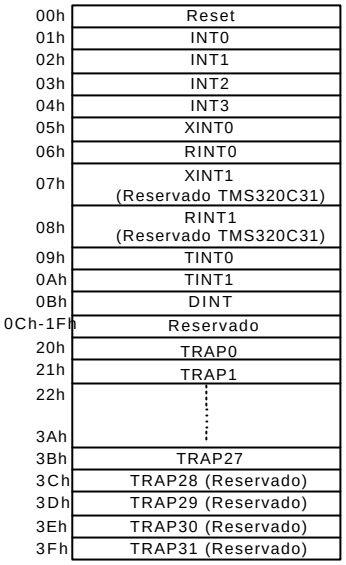


Figura 2.21. Mapa de memoria de los vectores de reset, interrupciones y excepciones software.

5.2 *Periférico Memoria Caché.*

Los DSPs de la familia TMS320C3x de Texas Instruments poseen un periférico interno destinado a la optimización, en caso necesario, de los accesos del sistema a memoria externa de programa. Este periférico consiste en una memoria capaz de almacenar hasta 64 instrucciones (de tamaño 64x32 bits) y un algoritmo de control del acceso a dicha memoria. Se usa para almacenar secciones de código repetidas, para evitarle a la CPU el acceso al exterior a volver a recoger una instrucción que hace relativamente poco tiempo que se recogió. Esto reduce los accesos a memoria externa acelerando, en el tiempo, la ejecución de los programas almacenados en dispositivos de memoria externa lentos (que tienen por otro lado la ventaja de ser los más baratos). Cuando la CPU pretende recoger una instrucción entra en funcionamiento este periférico, siempre que se encuentre habilitado, que decide, de alguna forma, si es o no necesario el acceso al dispositivo externo donde se encuentra físicamente la instrucción.

Una de las principales ventajas e inconvenientes de este periférico, es que actúa de forma independiente y automática, el usuario no interviene. Usa un algoritmo denominado ***LRU*** (*Least Recently Used*) para la actualización del contenido de la memoria y como lógica de control del acceso al periférico.

La arquitectura de la memoria Caché se muestra en la figura 2.23. La memoria Caché se divide en dos segmentos de 32 *words* cada uno, con 32 posibles instrucciones cada uno. A cada segmento se le asocia otro registro de 19 bits denominado ***SSA*** *(Segment Start Address*), **SSA0** para el segmento 0 y **SSA1** para el segmento 1. A cada *word* de la Caché se le asocia un bit, denominado bit P, que indica si está o no presente una instrucción en ella.

Cuando la CPU necesita recoger una instrucción ubicada en la memoria externa, el periférico Caché realiza un chequeo para ver si la instrucción está contenida en su memoria. El Algoritmo de control de acceso a la memoria de la Caché, figura 2.23, se basa en la división de la dirección de la nueva instrucción que se desea recoger, indicada por el registro PC, en dos partes:

- Los 19 MSBs del registro PC (bits más significativos de la dirección). Se usan para seleccionar entre uno de los dos segmentos de que dispone el periférico (SSA0 ó SSA1).
- Los 5 LSBs del registro PC (bits menos significativos de la dirección). Se usan para buscar la instrucción dentro del segmento seleccionado, para seleccionar una posición de memoria de entre 32 posibles que dispone cada segmento.

Los 19 MSBs de la dirección de la nueva instrucción se comparan con los dos registros SSA. Si coincide con alguno de ellos, los 5 LSBs del registro PC se utilizan para comprobar si el bit P del segmento seleccionado está activo. El bit P lo utiliza la CPU para ver si la instrucción que desea recoger está presente en el segmento. Si no se encuentra una coincidencia con los SSA de los 19 MSBs de la instrucción puede reemplazarse uno de los dos segmentos de la memoria Caché con nuevos datos, instrucciones que se recojan de la memoria externa a partir de ese momento.

El algoritmo LRU es el que determina el segmento a reemplazar. Para ello se define una pila de 2 bits, *LRU-Stack*, que se emplea en determinar el segmento, de los dos posibles, menos usado recientemente por el periférico. Cuando hay que reemplazar datos de la Caché se usa el segmento que más tiempo hace que no es accedido, determinado por el *elemento o bit que ocupa la parte inferior de la pila*. Una vez determinado el segmento menos empleado, las 32 banderas P de él son puestas a 0 y el registro SSA correspondiente a ese segmento es reemplazado por los 19 MSBs de la dirección de la instrucción que se está recogiendo en esos momentos. Cada vez que el periférico accede a uno de los segmentos introduce en *la cima de la pila* el número correspondiente al mismo (si se usó el SSA0, se introduce en la parte alta de la pila del LRU el valor 0). Obsérvese que la pila tiene 2 bits, uno de los cuales vale siempre 1 y el otro 0, de forma y manera que si en la parte alta de la pila aparece un 0 en la baja tiene que haber un 1 y viceversa.

Cuando se resetea el sistema microprocesador, el periférico de memoria Caché se inicializa con los siguientes valores:

- *Top pila*=0. El segmento 0 es el segmento que, después de un reset, se define como el último usado.
- *Bottom pila*=1. El segmento 1 es el segmento que, después de un reset, se define como el que no se ha usado recientemente.
- Todas las banderas P (una por cada word de la memoria caché) se ponen a 0 indicando que no hay ninguna instrucción en el registro correspondiente.

El algoritmo de acceso y actualización de la memoria caché se resume como sigue:

1. **CACHE HIT**: Se dice que se ha producido un *Caché Hit* si la memoria del periférico caché tiene la instrucción que la CPU está buscando. En este caso, la CPU no accede al exterior a recoger la instrucción sino que la recoge de la memoria del periférico caché. El algoritmo de acceso a la instrucción se completa con la actualización de la pila de dos bits: En la cima de la pila (*top pila*) se introduce el número del segmento (0 ó 1) donde estaba la instrucción recogida.

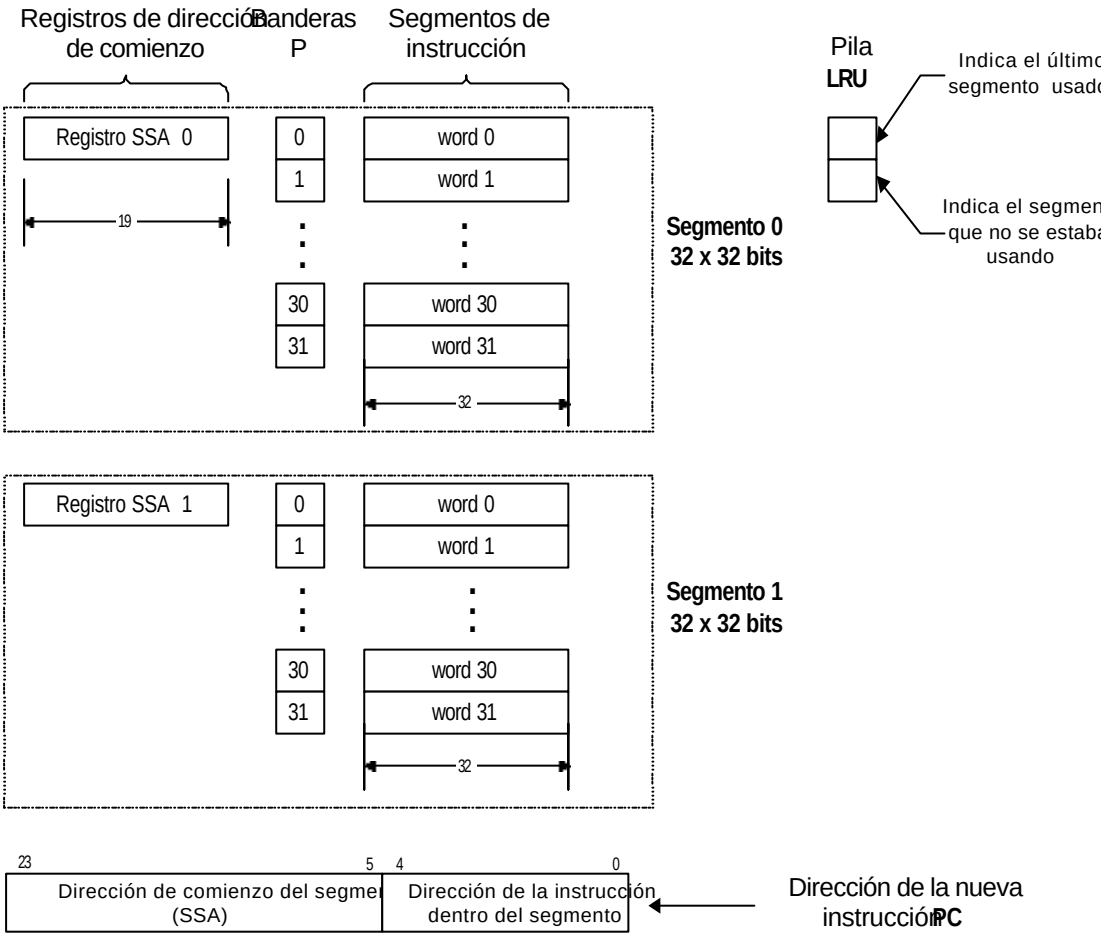


Figura 2.22. Arquitectura del periférico de *memoria caché*.

2. **CACHE MISS**: Se dice que se ha producido un *Caché Miss* si la memoria del periférico caché no tiene la instrucción que la CPU está buscando. En este caso, el periférico no dispone de la instrucción por dos posibles causas:

   - **WORD MISS**: Uno de los segmentos SSA coincide con los 19 MSBs de la dirección de la instrucción que se desea recoger pero la bandera P, asociada a los 5 LSBs de esa instrucción, no está activada. En este caso, la instrucción es recogida por la CPU de la memoria externa y se aprovecha para almacenarla en la propia caché. Los 5 LSBs de la dirección de la instrucción indican la word en la que se almacena la instrucción. La bandera P correspondiente a dicha posición de memoria de la caché se pone a 1. En la

cima de la pila de dos bits aparece el número del segmento en el que se acaba de almacenar la instrucción recogida.

- **SEGMENT MISS**: Los 19 MSBs de la dirección de la instrucción no coinciden con ninguno de los registros SSA del periférico. En este caso, la instrucción se recoge de memoria exterior y, simultáneamente, se aprovecha para almacenarla en la caché. Ahora, al contrario que en el caso anterior, todas las banderas P correspondientes al segmento que indicaba la parte baja de la pila (*bottom pila*) son puestos a 0 (se limpian). Los 19 MSBs de la dirección de la instrucción accedida se copian al registro SSA del segmento inicializado, se activa la bandera P ligada a los 5 LSBs de la dirección de la instrucción recogida y, finalmente, se guarda la instrucción recogida en la posición de memoria seleccionada de la caché. Al final, el número del segmento activado se almacena en la cima de la pila (*top pila*).

NOTA IMPORTANTE: Hay que tener cuidado con programas que modifiquen el código si se ha habilitado, activado, simultáneamente el periférico caché. Las modificaciones que se realicen puede no recogerlas la caché. Nos podemos encontrar con una situación en la que, aunque hemos cambiado el código vía software, la CPU no ejecuta el código nuevo sino el antiguo por estar impidiendo, el periférico caché, el acceso a la memoria donde se ha ubicado el programa.

5.3 *Arranque programado en el TMS320C31:* Boot-Loader.

El TMS320C31 tiene, al igual que todos los DSPs de la familia TMS320C3x, ROM interna del tipo OTP (***One Time Programmable***). A diferencia que en el caso del TMS320C30 esta ROM interna ya viene programada de fábrica con un programa denominado *boot-Loader*[5]. El reset del TMS320C31, funcionando en modo microcomputador (arranque desde memoria ROM interna), provoca que la CPU del DSP lea la posición 000000h de la ROM

[5] Equiparable al Búfalo en el Motorola MC68HC11.

interna. En dicha posición se encuentra la dirección de comienzo del programa denominado *boot-loader* ubicado, a su vez, en la ROM interna del DSP.

En la figura 2.24 se muestran los diagramas de flujo que describen el funcionamiento del programa *boot-loader*.

El programa *boot-loader* **gestiona** el **trasvase de bloques de datos** (código diseñado por el usuario) enviados desde otro sistema microprocesador y recibidos por el puerto serie de que dispone el TMS320C31 o ubicados en algún dispositivo externo de memoria (colocado en el mapa de memoria del DSP en una de las tres posiciones conocidas como BOOT1, BOOT2 ó BOOT3, véase figura 2.16, y normalmente de tipo EPROM) **hacia otras posiciones del mapa de memoria del DSP** ocupadas por dispositivos de almacenamiento RAM.

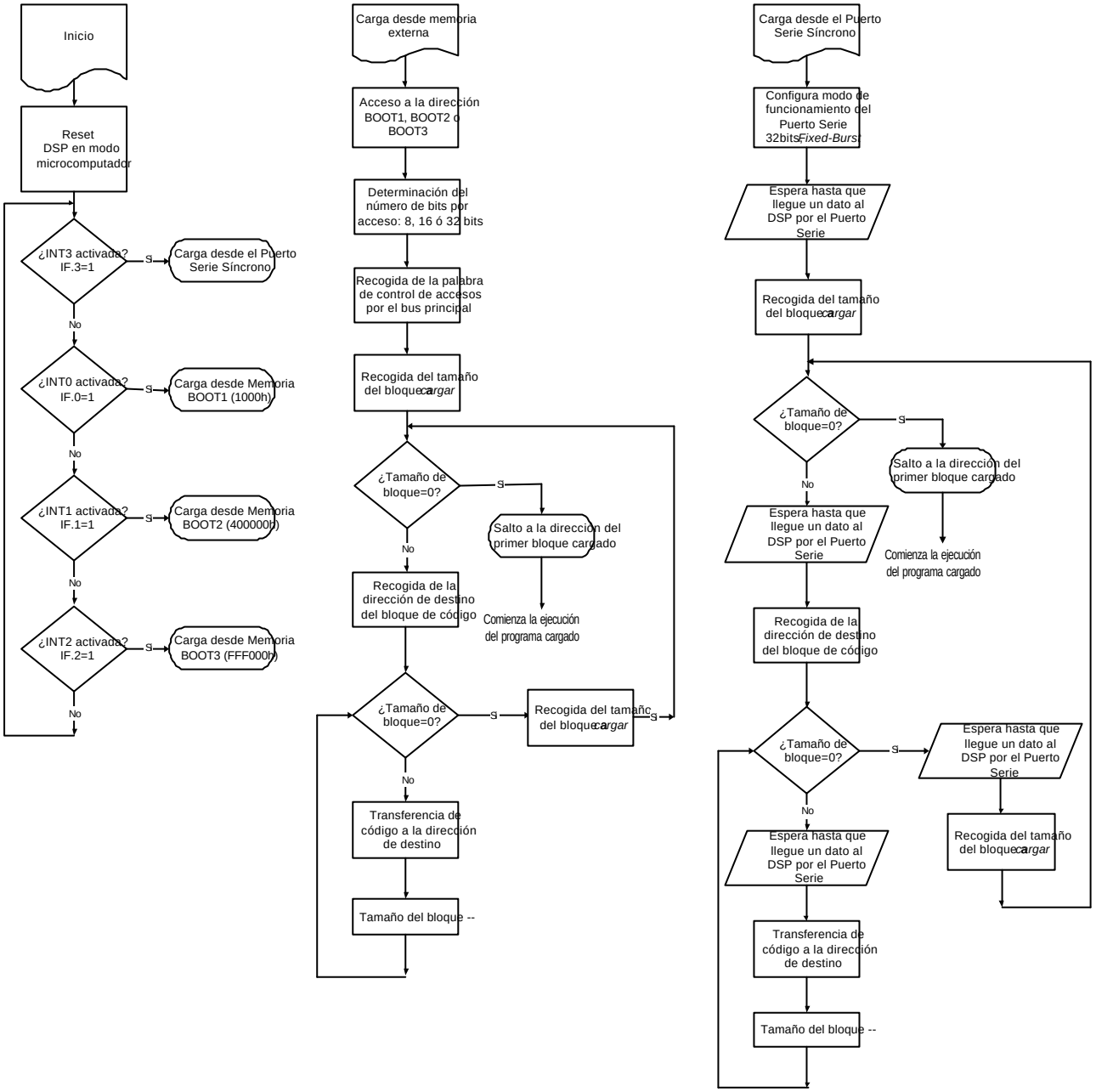


Figura 2.23. Diagrama de flujo del programa Boot-loader (ubicado en ROM interna del TMS320C31).

El programa es realmente sencillo. Consiste en un bucle que espera una señal, generada por el usuario, que le indique que debe iniciar el trasvase de bloques de datos. Mientras el programa se encuentre en el bucle de espera, la única tarea que realiza el DSP es comprobar si el usuario ha generado alguna de las señales de inicio de trasvase de bloques de datos. El usuario dispone de cuatro líneas de entrada al DSP, $\overline{INT0}$ a $\overline{INT3}$, como señales para iniciar el trasvase de los bloques de datos. Inicialmente estas señales se usan como líneas

digitales de entrada al programa y no como líneas de petición de interrupciones externas, y le sirven al usuario para indicarle al *boot-loader* hacia donde debe ir a buscar el código que va a cargar en memoria RAM ubicada en el mapa de memoria del DSP. Cuando una de estas líneas es activada por el usuario, el programa *boot-loader* inicia la carga programada, bien desde memoria externa (desde una dirección de tres posibles, **BOOT1**, **BOOT2** ó **BOOT3**), bien mediante el puerto serie (el DSP estaría esperando datos que otro sistema digital, que puede ser otro DSP, le envía empleando un protocolo serie síncrono).

1. **Carga desde Memoria Externa**. El programa *boot-loader* accede a un dispositivo de memoria, normalmente de tipo EPROM, ubicado en BOOT1, BOOT2 ó BOOT3 en el mapa de memoria para realizar el trasvase de un bloque de datos (programa diseñado por el usuario) hacia otras posiciones del mapa de memoria del DSP. Antes de trasladar los bloques de código el programa *boot-loader* precisa que se le pasen una serie de parámetros de configuración o programación del tipo y modo de transferencia que debe realizar (cabecera del bloque de datos, primeros cuatro registros de 32 bits). La cabecera de un bloque de datos se muestra en la tabla 2-6. Estos valores de configuración son, el **tipo de dispositivo de memoria** al que se accede a recoger el bloque de datos (con un ancho de bus programable de 8, 16 ó 32bits), la **palabra de control para la configuración del acceso** a dichos periféricos de memoria a través del bus principal, el **tamaño del bloque de datos** que se desea transferir y, finalmente, la **dirección de destino** o de ubicación del bloque. Posteriormente aparece la primera instrucción del bloque de datos que se trasladará a otra zona del mapa de memoria del DSP.

   En definitiva, primero se configura el modo y tipo de transferencia, luego se realiza la misma. Se pueden realizar transferencias de múltiples bloques de datos desde las zonas *BOOTs* hacia otras zonas en el mapa de memoria del DSP, figura 2.24. El *boot-loader* se termina añadiendo, al final del último bloque transferido, el código 00000000h que representa que el tamaño del siguiente bloque a transferir es 0 (no existe tal bloque en realidad). Obsérvese que, al menos, el primer bloque debe tener longitud mayor de cero para que tenga sentido invocar al programa *boot-loader*, si no fuese así el sistema arrancaría de forma impredecible.

Posición	Descripción	Valores válidos
0	Tipo de memoria boot (8, 16 ó 32)	0x8, 0x10, 0x20
1	Configuración accesos por el bus principal	
2	Tamaño del bloque del programa (blk)	0 < Tamaño bloque < 2^{24}
3	Dirección de destino	Cualquier dirección de 24 bits es válida
4	Código de inicio del programa	Cualquier instrucción o dato de 32 bits

Tabla 2-6. Registros de configuración del programa *boot-loader*: Cabecera del bloque de código en memoria externa (BOOT1, BOOT2 ó BOOT3).

Posición	Valor	Descripción
BOOTn+0	0x08	Memoria externa de ancho 1 byte, 0x8
BOOTn+1	0x00	
BOOTn+2	0x00	
BOOTn+3	0x00	
BOOTn+4	0x58	Tipo de acceso: 0x1058 • SWW = 11 • WCNT = 2
BOOTn+5	0x10	
BOOTn+6	0x00	
BOOTn+7	0x00	
BOOTn+8	0xFF	Tamaño bloque de código a trasladar: 0x1FF
BOOTn+9	0x01	
BOOTn+10	0x00	
BOOTn+11	0x00	
BOOTn+12	0x00	Dirección de destino del código: 0x809C00
BOOTn+13	0x9C	
BOOTn+14	0x80	
BOOTn+15	0x00	

Tabla 2-7. Cabecera del bloque de código en memoria externa (BOOT1, BOOT2 ó BOOT3) con tipo de memoria boot de un ancho de 1 byte (8 bits).

Posición	Valor	Descripción
BOOTn+0	0x0010	Memoria externa de ancho 2 bytes, 0x10
BOOTn+1	0x0000	

BOOTn+2	0x1058	Tipo de acceso: 0x1058 • SWW = 11 • WCNT = 2
BOOTn+3	0x0000	
BOOTn+4	0x01FF	Tamaño bloque de código a trasladar: 0x1FF
BOOTn+5	0x0000	
BOOTn+6	0x9C00	Dirección de destino del código: 0x809C00
BOOTn+7	0x0080	

Tabla 2-8. Cabecera del bloque de código en memoria externa (BOOT1, BOOT2 ó BOOT3) con tipo de memoria boot de un ancho de 2 bytes (16 bits).

Posición	Valor	Descripción
BOOTn+0	0x00000020	Memoria externa de ancho 4 bytes
BOOTn+1	0x00001058	Tipo de acceso: 0x1058 • SWW = 11 • WCNT = 2
BOOTn+2	0x000001FF	Tamaño bloque de código a trasladar: 0x1FF
BOOTn+3	0x00809C00	Dirección de destino del código: 0x809C00

Tabla 2-9. Cabecera del bloque de código en memoria externa (BOOT1, BOOT2 ó BOOT3) con tipo de memoria boot de un ancho de 4 bytes (32 bits).

En las tres tablas anteriores, tablas 2-7, 2-8 y 2-9, se muestra la secuencia de acceso del programa *boot-loader* a cualquiera de las posiciones BOOT1, BOOT2 ó BOOT3 con memoria externa configurada con un ancho de 1, 2 ó 4 bytes.

2. **Carga desde el puerto serie**. El *boot-loader* configura automáticamente el puerto serie en modo de trasferencia de datos de ancho 32 bits y en modo de transferencia de bits *fixed-burst* (se verá cuando se estudie el periférico puerto serie síncrono del DSP). Como en el caso de la carga desde memoria externa se precisa de una cabecera para iniciar y programar la operación de carga. La cabecera, en este caso, sólo incluye el tamaño del bloque y la dirección de destino del bloque de datos.

Comentarios sobre el programa *boot-loader*:

- Genera una pila de 1x32 bits (una *word*) que se encuentra en la posición 809800h (dentro de la RAM interna, bloque 0). No puede aparecer como destino de una instrucción esa dirección (cuando se traslade el control al DSP, a saber que puede haber en esa posición).

- Las banderas de las interrupciones (registro IF) que se activen se ponen a 1, no las limpia el programa *boot-loader*, de forma que hay que comenzar limpiándolas antes de habilitar las interrupciones externas si se desea hacer uso, en el programa del usuario (bloque de código trasladado por el *boot-loader*), de las mismas y no se desea que aparezca una interrupción espúrea.

- En modo microcomputador el TMS320C31 trae definidos, de fábrica, los vectores de interrupciones y excepciones software (la ROM interna OTP la programó el fabricante) para que apunten a direcciones fijas del bloque 1 de la RAM interna. Si se desea usar alguna interrupción debe ubicarse, en la posición del bloque 1 de RAM asignada por el fabricante, una instrucción de salto a las correspondientes rutinas de servicio de las interrupciones, figura 2.25.

- El arranque efectivo se realiza desde la dirección de comienzo del primer bloque transferido, una vez efectuada la transferencia de los bloques de datos en el mapa de memoria del DSP.

Dirección	Descripción
809FC1h	INT0
809FC2h	INT1
809FC3h	INT2
809FC4h	INT3
809FC5h	XINT0
809FC6h	RINT0
809FC7h	Reservado
809FC8h	Reservado
809FC9h	TINT0
809FCAh	TINT1
809FCBh	DINT
809FCC-809FDFh	Reservado
809FE0h	TRAP0
809FE1h	TRAP1
	⋮
809FFBh	TRAP27
809FFC-809FFFh	Reservado

Figura 2.24. Redireccionamiento de los vectores de interrupción y de las excepciones software en el TMS320C31 en modo microcomputador.

6. OPERACIONES EXTERNAS DEL BUS

En este apartado se analizan las diferentes formas de acceso del DSP, a través de los buses de dirección y datos de que dispone, a dispositivos de memoria y otros periféricos externos.

6.1. Estructura *Master-slave* en un sistema microprocesador.

Un sistema microprocesador debe entenderse como sistema digital en el que existe un único *maestro o master* (el microprocesador) y multitud de *esclavos o slaves* (periféricos de almacenamiento de programas y datos, de conversión analógico-digital y digital-analógico, etc.). El *maestro* es el dispositivo que se encarga de gestionar la comunicación en el sistema digital y de controlar la transferencia de información por los buses del sistema. Toda transferencia de información que se realice en el sistema digital comienza cuando lo decida el microprocesador y concluye cuando lo desee el mismo microprocesador que comenzó la comunicación. De esta forma, el microprocesador activa, en modo lectura o escritura, y termina el acceso al periférico al que desee acceder mediante las líneas asociadas a los buses

de control y dirección, siguiendo un estricto diagrama de tiempo suministrado por el fabricante, leyendo (o escribiendo) información del (en el) bus de datos del sistema.

El diseño de un sistema microprocesador requiere, por parte del diseñador, la consideración de dos aspectos. El primero, ya analizado en el primer parcial, consiste en evitar conflictos en los buses del sistema (dos dispositivos no pueden, bajo ningún concepto, activar, en el mismo instante de tiempo, la misma línea pues crearían un cortocircuito). El segundo aspecto está relacionado con la estructura *maestro-esclavo* del sistema microprocesador. Se ha comentado que es el *maestro* el que controla, en todo momento, la comunicación. La transferencia de los datos es iniciada y terminada por el *maestro* del sistema microprocesador, en nuestro caso, el DSP. Por ello, y para que la comunicación sea fiable, es preciso que al *esclavo* le dé tiempo a responder al *maestro*.

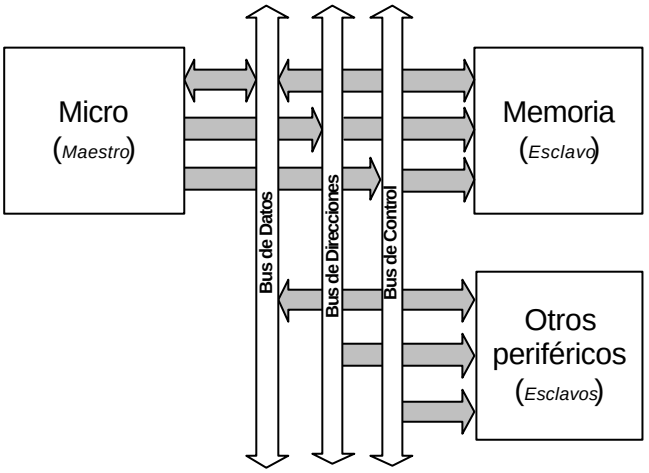


Figura 2.25. Estructura *maestro-esclavo* en un sistema microprocesador.

Para que la comunicación sea fiable o efectiva (y, por tanto, el *maestro* consigue leer el dato o instrucción que requiere del *esclavo* o escribir el dato que desea almacenar en el *esclavo*) es necesario que el sistema digital se diseñe de forma que el tiempo que transcurre desde que se inicia el acceso hasta que éste termina ($T_{acceso-DSP}$) sea mayor que el tiempo que tarda en responder el *esclavo* ($T_{respuesta-prférico}$). Es decir, si el microprocesador desea leer un dato o una instrucción de un periférico externo y, para ello, comienza un acceso externo a través del

bus principal o del de expansión, al periférico le tiene que dar tiempo a poner el dato requerido en el bus de datos antes de que el DSP concluya el acceso al exterior para que el DSP lea correctamente el dato. Si el microprocesador quiere escribir un dato en un periférico externo, debe de mantener el dato en el bus de datos el tiempo que al periférico le haga falta para almacenarlo en su interior. En caso contrario, el DSP no leería la instrucción o el dato deseado y no almacenaría el dato en el periférico externo y, por tanto, el sistema digital no funcionaría, o al menos no lo haría correctamente.

La segunda condición básica necesaria para un diseño adecuado de un sistema microprocesador se puede resumir, por tanto, en el cumplimiento de la siguiente desigualdad: $T_{acceso\ -DSP} > T_{respuesta\ -prférico}$.

6.2. Diagramas de tiempos y estados de espera.

Un *diagrama de tiempo* es un gráfico que representa, en el tiempo, como se realiza un acceso en lectura o escritura en un dispositivo (microprocesador o periférico). De su correcta interpretación depende que un diseñador realice un diseño adecuado de un sistema microprocesador, desde el punto de vista de los tiempos de acceso en la estructura *maestro-esclavo* ($T_{acceso\ -DSP} > T_{repuesta\ -prférico}$). El diagrama de tiempo en un microprocesador representa cómo se inicia y concluye la comunicación e indica el tiempo de acceso del *maestro*.

Asociado a un periférico, el diagrama de tiempo establece el tiempo de respuesta del *esclavo* que, en condiciones ideales, coincide con el tiempo de acceso al periférico. La diferencia entre tiempo de respuesta y de acceso a un esclavo depende del tiempo de retraso, T_{delay} , asociado a la lógica de selección del propio dispositivo, $T_{respuesta\ -prférico} = T_{acceso\ -periférico} + T_{delay}$. El microprocesador inicia el acceso pero el periférico al que desea acceder no se entera del comienzo del acceso hasta unos pocos nanosegundos después, T_{delay} , debido a que todo

periférico tiene asociada una lógica de selección que, a su vez, lleva asociada un tiempo de respuesta mayor de *0nseg*.

Los microprocesadores pequeños en prestaciones y potencia de cálculo, caso de la familia MC68HC11 y la mayoría de familias de microcontroladores de 8 bits, no permiten al usuario controlar el tiempo de acceso del microprocesador a periféricos externos. El fabricante del microprocesador suministra un tiempo de acceso, del *maestro* a los *esclavos*, fijo. Esto es debido a que el ciclo máquina del microprocesador suele ser lo suficientemente grande (el acceso del microprocesador lo suficientemente lento) como para que la mayoría de los periféricos puedan ser accedidos sin problemas de tiempo por parte del *maestro*. El caso del DSP es diferente. El ciclo máquina de este microprocesador es muy pequeño (del orden de los *40nseg* o incluso menos). No todos los periféricos que se pueden encontrar en el mercado presentan unos tiempos de respuesta tan bajos (por ejemplo, y para que el lector se haga una idea, el tiempo de respuesta y estabilización de la salida en una función lógica elemental −OR, AND...− asociada a un dispositivo diseñado con tecnología LS −TTL Schottky de bajo consumo− está en torno a los *10nseg*). Se presenta, por tanto, un problema que cuando se estudió la familia MC68HC11 no se tenía. ¿Es posible emplear cualquier periférico externo en un sistema digital basado en un DSP de la familia TMS320C3x?. Y si es así, ¿cómo es capaz de gestionar el usuario el acceso del DSP al periférico?. Impedir, al usuario, el empleo de un periférico, por lento que éste sea, no deja de ser una limitación para el diseñador de un sistema microprocesador. Los fabricantes, para vender sus microprocesadores, diseñan de la manera más versátil posible, lo que implica permitir el empleo del mayor número de periféricos externos posibles. La respuesta, por tanto, a la primera de las preguntas anteriores es que sí, será posible emplear cualquier periférico externo en un sistema digital basado en un DSP de la familia TMS320C3x. Obviamente, para permitir esta característica es necesario que el fabricante permita que el usuario controle el tiempo que tarda el DSP en realizar un acceso al exterior. *Los estados de espera* representan, en un sistema microprocesador, la manera que tiene el usuario de controlar el tiempo de acceso de un *maestro* a un *esclavo*. El acceso del microprocesador al periférico se va a poder ralentizar cierto tiempo, múltiplo entero del ciclo máquina del microprocesador. Ese múltiplo entero del ciclo máquina que ralentiza el acceso del microprocesador a los periféricos son los *estados de espera* del

acceso. Obsérvese que, este aumento de la flexibilidad en el acceso exterior del sistema va a suponer un aumento en la complejidad del diseño, como se podrá observar en este apartado. El DSP ofrece la posibilidad de generar *estados de espera* en los accesos a los periféricos externos programando, convenientemente, un registro de control por cada bus del sistema. Esta característica permite al sistema DSP, *maestro* en un sistema digital, acceder a dispositivos externos, *esclavos* en dicho sistema digital, con tiempos de acceso altos, superiores a los que realmente necesitaría el *maestro* para recoger una información previamente requerida al *esclavo.*

En el caso del TMS320C30 el acceso, como veíamos, puede realizarse a través de dos interfaces diferentes, el **bus primario** (consta de 24 líneas de dirección A0-A23, 32 líneas de datos D0-D31 y varias líneas de control $\overline{RDY}$ **,** $\overline{HOLD}$, $\overline{HOLDA}$, $\overline{STRB}$ y $R/\overline{W}$) y el **bus de expansión** (consta de 13 líneas de dirección XA0-XA12, 32 líneas de datos XD0-XD31 y otras líneas de control diferentes de las del bus principal, $\overline{XRDY}$, $\overline{MSTRB}$, $\overline{IOSTRB}$ y $XR/\overline{W}$). En el caso del TMS320C31 estos accesos sólo pueden realizarse a través de una interface, el bus primario, figura 2.15.

Las características fundamentales de los accesos externos de los DSPs de la familia TMS320C3x de Texas Instruments son:

- ✓ **Bus principal**. Bus de datos de 32 bits, bus de direcciones de 24 bits (capacidad de direccionamiento de 2^{24} words ó 16 *Mwords*) y bus de control con cinco líneas de las que tres gestionan el acceso al exterior (las demás líneas se estudiarán más adelante):
  - $\overline{RDY}$: línea externa de control de la generación de estados de espera de acceso a través del bus.
  - $\overline{STRB}$: señal que indica que se inicia un acceso a través del bus principal)
  - $R/\overline{W}$: señal que indica si el acceso, a través del bus principal, es en lectura para recoger del bus un dato que requiere el DSP o en escritura para poner en el bus un dato que lee algún periférico externo.

- ✓ **Bus de expansión**. Bus de datos de 32 bits, bus de direcciones de 13 bits (capacidad de direccionamiento de 2^{13} word ó 8 *Kwords*) y bus de control con cuatro líneas, $\overline{XRDY}$ (línea externa de control de la generación de estados de espera de acceso a través del bus), $\overline{MSTRB}$ e $\overline{IOSTRB}$ (señales que indican que se inicia un acceso a través del bus de expansión, distinguiéndose entre dos posibles protocolos de acceso) y por último la señal $XR/\overline{W}$ (indica si el acceso a través del bus de expansión es en lectura para recoger del bus un dato que requiere el DSP o en escritura para poner en el bus un dato que lee algún periférico externo).

El dispositivo dispone de dos periféricos internos que gestionan los accesos, a través de los buses del sistema, a los dispositivos externos. El modo de funcionamiento de estos periféricos se configura con los registros de control del bus principal y del bus de expansión, figuras 2.28 y 2.29, tablas 2-10 y 2-11.

Dirección	Descripción
808060h	Registro de Control Bus Expansi
808061h	Reservado
808062h	Reservado
808063h	Reservado
808064h	Registro de Control Bus Principa
808065h	Reservado
808066h	Reservado
808067h	Reservado
808068h	Reservado
808069h	Reservado
80806Ah	Reservado
80806Bh	Reservado
80806Ch	Reservado
80806Dh	Reservado
80806Eh	Reservado
80806Fh	Reservado

Figura 2.26. Registros de control de los buses del sistema, familia TMS320C3x.

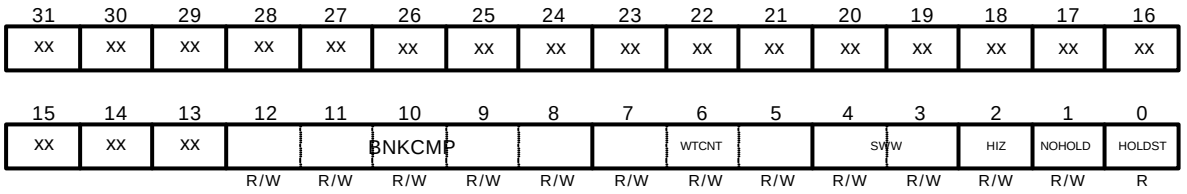


Figura 2.27. Registro de control del bus principal.

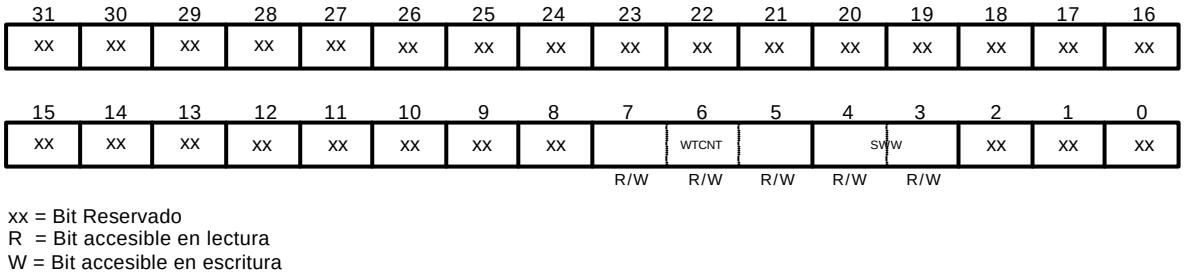


Figura 2.28. Registro de control del bus de expansión.

Bit	Nombre	Valor de Reset	Función que realiza
0	HOLDST	0 ó 1	Bit indicativo del estado HOLD. Si HOLDST=1, indica que el bus principal está en alta impedancia. Si HOLDST=0, indica que el bus principal no está en alta impedancia. Este bit es válido tanto si el bus está en HOLD porque algún periférico externo lo ha solicitado por hardware o si ha entrado en dicho estado vía software.
1	NOHOLD	0	Permite que el bus entre en alta impedancia si lo pide la señal externa de entrada $\overline{HOLD}$. Si NOHOLD=1 el TMS320C3x gobierna las líneas del bus independientemente de que otro dispositivo pida el control de dichas líneas. Obviamente, en ese caso no se activa nunca la señal HOLDA de reconocimiento de haber cedido el control de las líneas del bus, sin embargo, el control y la señal HOLDA pueden cederse y activarse respectivamente si el control lo cede el DSP vía software (se genera un HOLD interno, HIZ=1).
2	HIZ	0	Generación de HOLD interno, por software. Cuando HIZ=1, el bus principal se pone en alta impedancia, otro periférico podría gobernar esas líneas sin generar conflictos. $\overline{HOLDA}$ se pone a 0 cuando el bus principal se pone en alta impedancia.

3-4	SWW	11	*Software Wait mode*. Estos dos bits definen, junto con WTCNT, el modo de generación de los estados de espera en los accesos a través del bus principal.
5-7	WTCNT	111	*Software Wait mode*. Estos tres bits especifican el número de ciclos internos de estados de espera. Se relaciona con la GENERACIÓN DE ESTADOS DE ESPERA INTERNOS. El rango es de 0 (WTCNT=000) a 7 (WTCNT=111).
8-12	BNKCMP	10000	*Bank Compare*. Estos cinco bits especifican el número de MSBs bits de la dirección que se emplean para definir el tamaño de un banco de conmutación programable.
13-31	Reservados	0-0	Se leen como ceros.

Tabla 2-10. Funciones de los bits del registro de control del bus principal.

Bit	**Nombre**	**Valor de Reset**	**Función que realiza**
0-2	Reservados	000	Se leen como ceros
3-4	SWW	11	*Software Wait mode*. Estos dos bits definen, junto con WTCNT, el modo de generación de los estados de espera en los accesos a través del bus principal.
5-7	WTCNT	111	*Software Wait mode*. Estos tres bits especifican el número de ciclos internos de estados de espera. Se relaciona con la GENERACIÓN DE ESTADOS INTERNOS DE ESPERA. El rango es de 0 (WTCNT=000) a 7 (WTCNT=111).
8-31	Reservados	0-0	Se leen como ceros.

Tabla 2-11. Funciones de los bits del registro de control del bus de expansión.

Los accesos del DSP para recoger datos e instrucciones de periféricos externos se caracterizan por los *diagramas de tiempo de acceso al exterior*. En el DSP se distinguen **dos protocolos diferentes** de acceso al exterior, uno que corresponde a los accesos a través del bus principal, con $\overline{STRB}$ activo, o del bus de expansión, con $\overline{MSTRB}$ activo, y el segundo que se corresponde con los accesos a través del bus de expansión con $\overline{IOSTRB}$ activo.

En las figuras 2.30, 2.31 y 2.32, se muestran accesos a periféricos externos con cero estados de espera (el acceso externo que realiza el DSP es el más rápido posible). Cuando se realiza el acceso a través del bus principal o del de expansión con $\overline{MSTRB}$ activo, el acceso en lectura tarda un ciclo máquina (salvo el caso particular en que un acceso en lectura sea inmediatamente posterior a otro en escritura, en cuyo caso el acceso en lectura tarda dos ciclos máquina) y el de escritura dos ciclos máquina. Cuando el acceso se realiza a través del bus de expansión con $\overline{IOSTRB}$ activo, tanto el acceso en lectura como el de escritura tardan dos ciclos máquina. Recuérdese que los accesos en lectura y escritura a RAM interna del DSP eran aún más rápidos, ambos se producían en un ciclo máquina.

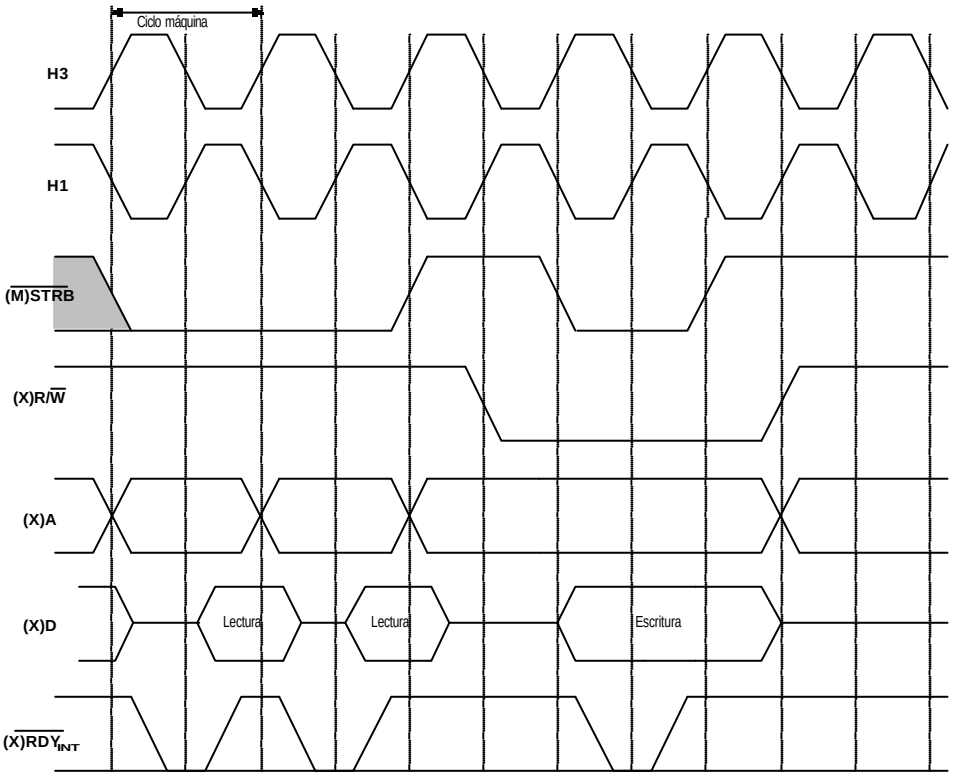


Figura 2.29. Accesos a través del bus principal ($\overline{STRB}$ activo) o del bus de expansión ($\overline{MSTRB}$ activo) con cero estados de espera: Lectura + Lectura + Escritura.

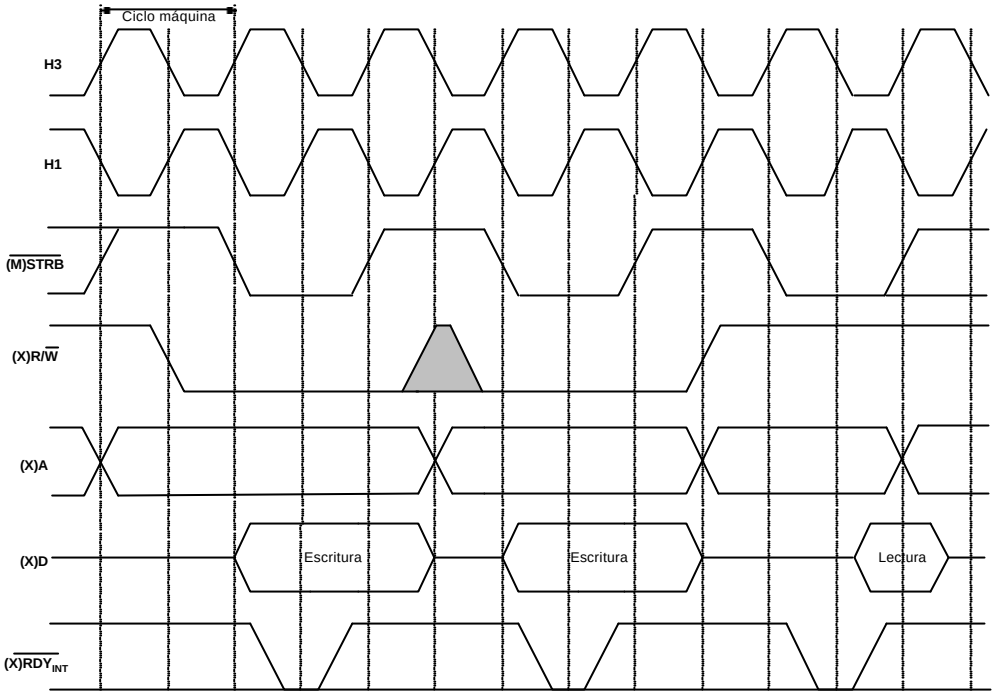


Figura 2.30. Accesos a través del bus principal ($\overline{STRB}$ activo) o del bus de expansión ($\overline{MSTRB}$ activo) con cero estados de espera: Escritura + Escritura + Lectura.

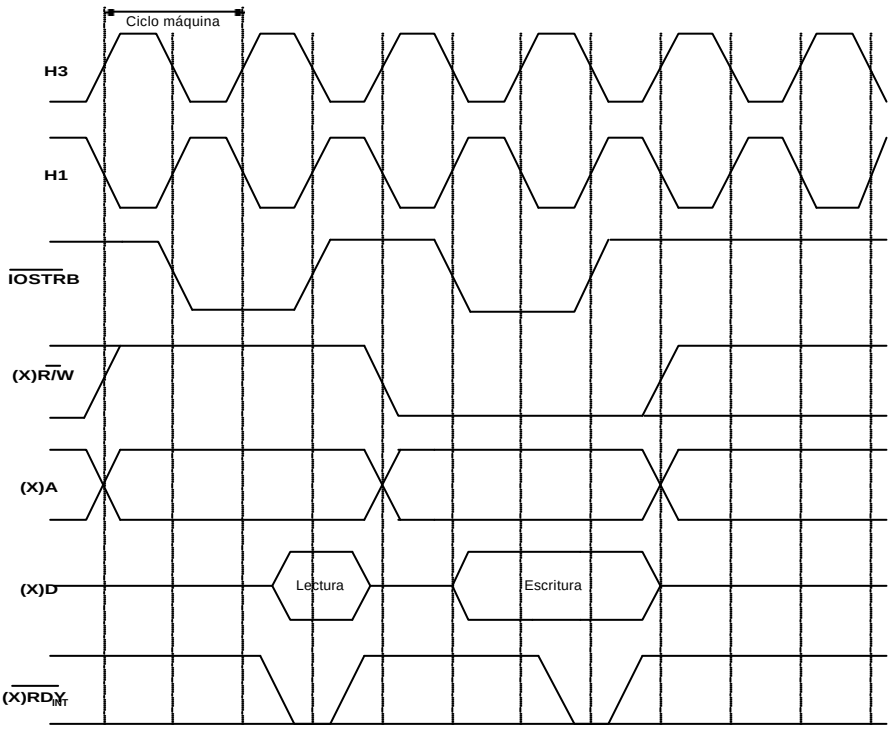


Figura 2.31. Accesos a través del bus de expansión ($\overline{IOSTRB}$ activo) con cero estados de espera: Escritura + Lectura.

En las figuras 2.33, 2.34 y 2.35 se muestran accesos, con un estado de espera, a través del bus principal ($\overline{STRB}$ activo) y del bus de expansión ($\overline{MSTRB}$ activo) o a través del bus de expansión ($\overline{IOSTRB}$ activo).

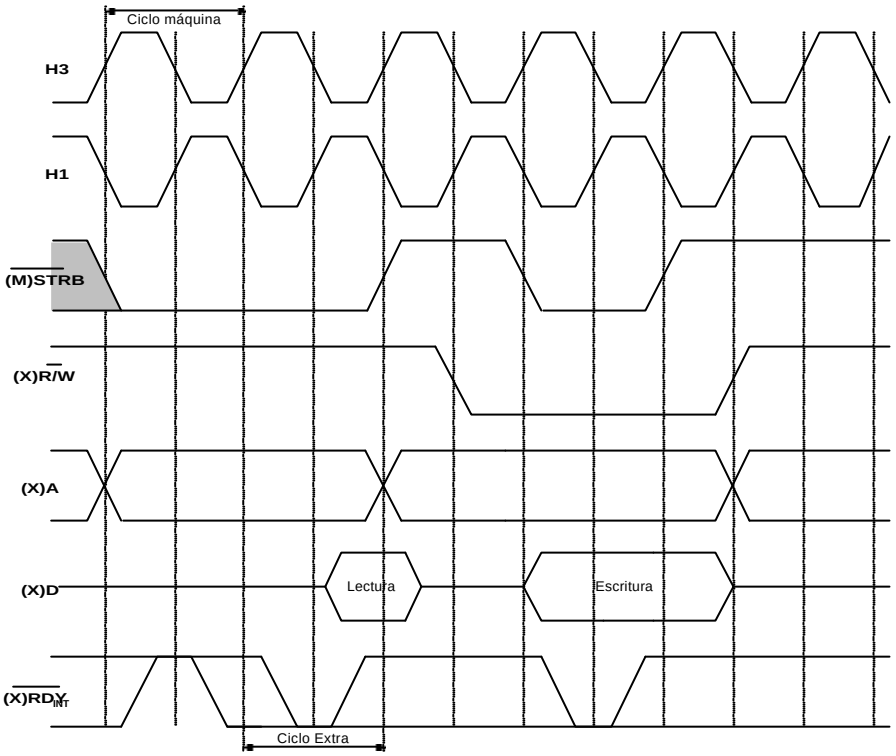


Figura 2.32. Accesos en lectura y escritura a través del bus principal ($\overline{STRB}$ activo) o del bus de expansión ($\overline{MSTRB}$ activo) con un estado de espera en lectura y cero en escritura.

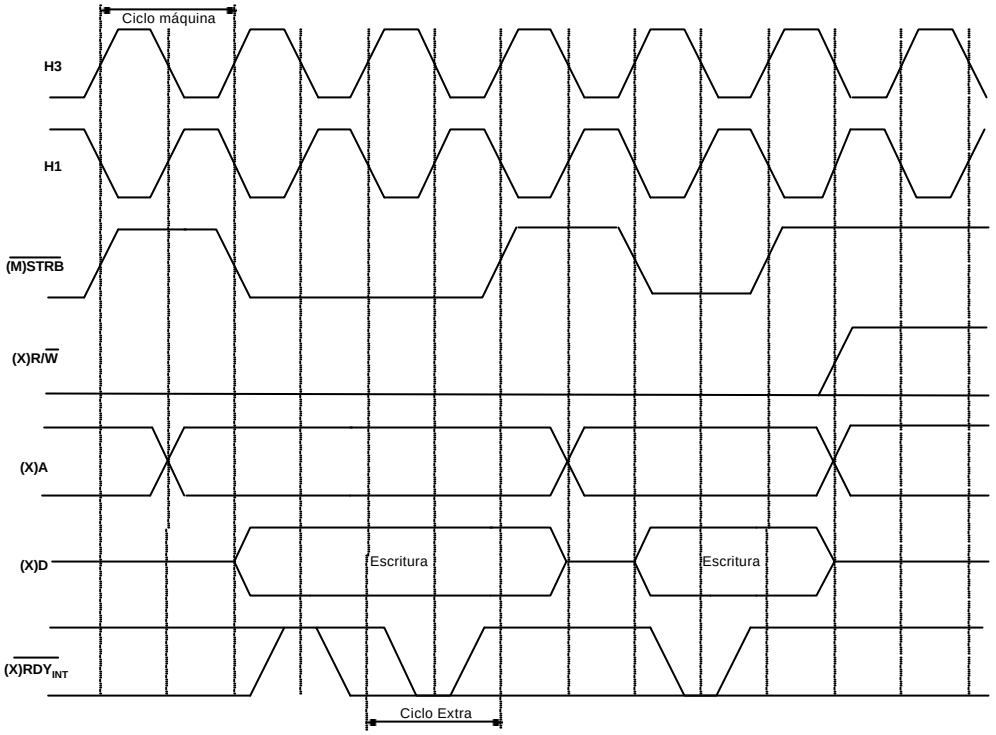


Figura 2.33. Accesos en escritura a través del bus principal ($\overline{STRB}$ activo) o del bus de expansión ($\overline{MSTRB}$ activo), primero con un estado de espera, luego con cero.

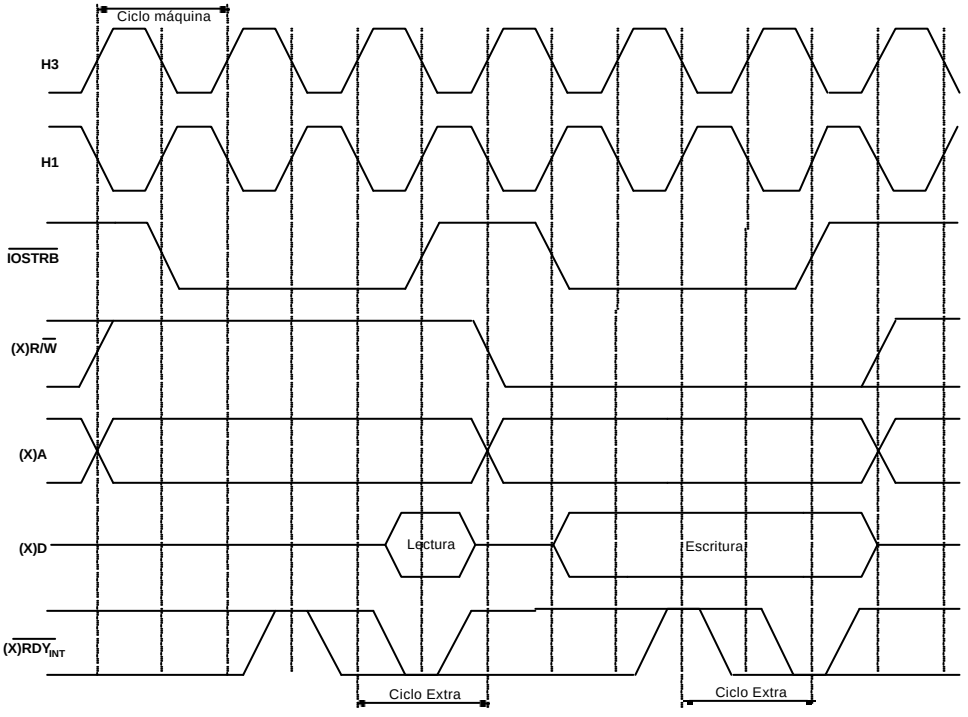


Figura 2.34. Accesos en lectura y escritura a través del bus de expansión ($\overline{IOSTRB}$ activo) con un estado de espera.

La **generación de estados de espera** es gobernada, en todos los casos, por una señal interna del DSP que se denomina $\overline{RDY_{INT}}$ cuando el acceso se realiza por el bus principal y $\overline{XRDY_{INT}}$ cuando el acceso se realiza por el bus de expansión.

Cuando el acceso al exterior se realiza con cero estados de espera, un ciclo de reloj externo después de que el DSP (*maestro*) inicie el acceso al dato (selección en lectura o escritura del *esclavo*), la señal interna $\overline{(X)RDY_{INT}}$ se activa (se pone a cero lógico indicándole a la CPU del DSP que el acceso al exterior –en lectura o escritura– puede completarse) lo antes posible. En el caso del bus principal o del de expansión con $\overline{MSTRB}$ activo, el acceso se completa en un ciclo máquina para la lectura y en dos ciclos máquina para la escritura mientras que, en el caso del bus de expansión con $\overline{IOSTRB}$ activo, en dos ciclos máquina se completan tanto la lectura como la escritura.

Cuando el acceso al exterior se realiza con un estado de espera, un ciclo de reloj externo después de que el DSP inicie el acceso al dato, la señal interna $\overline{(X)RDY_{INT}}$ se desactiva, se pone a uno lógico, indicándole a la CPU del DSP que el acceso al exterior (en lectura o escritura) no puede completarse inmediatamente sino que debe relentizarse en un ciclo máquina (ciclo máquina extra) el acceso al exterior.

Esa señal interna, que hemos denominado $\overline{(X)RDY_{INT}}$, se genera a partir de otras dos señales, una que se corresponde con los pines de entrada al circuito integrado ($\overline{RDY}$ para el bus principal y $\overline{XRDY}$ para el bus de expansión) y otra que se genera internamente en el DSP mediante la programación de dos *temporizadores internos* de ciclos máquina de tres bits, uno asociado al bus principal y otro al de expansión, ($\overline{RDY_{WTCNT}}$ y $\overline{XRDY_{WTCNT}}$). La señal $\overline{RDY_{INT}}$ se obtiene como una función lógica programable de las señales $\overline{RDY}$ y $\overline{RDY_{WTCNT}}$. La señal $\overline{XRDY_{INT}}$ se obtiene como una función lógica programable de las señales $\overline{XRDY}$ y $\overline{XRDY_{WTCNT}}$. El tipo de función lógica se programa con los bits SWW del registro de control de los buses, tanto principal como de expansión.

La forma de generar las señales $\overline{(X)RDY_{INT}}$ se muestra en la figura 2.36:

1. Primero se programa el modo de generación de las señales $\overline{(X)RDY_{WTCNT}}$ (**R**ea**DY** **W**ai**TC**ou**NT**er). Para ello se emplean los registros de control de los buses:

   - **WTCNT** (bits 5-7 del registro de control del bus). Programa el valor inicial a un *temporizador interno* indicando el número de ciclos máquina de espera. Cuando se inicia el acceso al dispositivo externo se carga el valor de WTCNT (3 bits, valor que varía de 0 a 7) en un contador, *temporizador interno*, que se decrementa cada ciclo máquina hasta que llega a 0. Cuando llega a cero activa la señal $\overline{(X)RDY_{WTCNT}}$ y permanece en ese valor hasta el comienzo de otro acceso externo. Mientras que el contador esté a cero, $\overline{(X)RDY_{WTCNT}} = 0$. Cuando tenga un valor distinto de cero, $\overline{(X)RDY_{WTCNT}} = 1$.

   - **SWW** (bits 3-4 del registro de control del bus). Selecciona el tipo de función lógica programable para la generación de estados de espera. Se tienen las siguientes opciones:

     ✓ SWW=00. $\overline{(X)RDY_{INT}}$ se genera sólo en función de los pines de entrada al circuito integrado $\overline{RDY}$ ó $\overline{XRDY}$. Se ignora la señal $\overline{(X)RDY_{WTCNT}}$ generada internamente en el DSP.

     ✓ SWW=01. $\overline{(X)RDY_{INT}}$ se genera sólo en función de la señal $\overline{(X)RDY_{WTCNT}}$. Se ignoran los pines de entrada al circuito integrado, $\overline{RDY}$ ó $\overline{XRDY}$.

     ✓ SWW=10. $\overline{(X)RDY_{INT}}$ se genera como una AND lógica entre los pines de entrada al circuito integrado ($\overline{RDY}$ ó $\overline{XRDY}$) y la señal $\overline{(X)RDY_{WTCNT}}$. Los estados de espera se están generando, retrasándose los accesos a dispositivos

externos, mientras no se active (aparezca un cero lógico) o bien en la señal $\overline{(X)RDY_{WTCNT}}$ o en los pines de entrada al DSP $\overline{RDY}$ ó $\overline{XRDY}$.

- ✓ SWW=11. $\overline{(X)RDY_{INT}}$ se genera como una OR lógica entre los pines de entrada al circuito integrado $\overline{RDY}$ ó $\overline{XRDY}$ y la señal $\overline{(X)RDY_{WTCNT}}$. Los estados de espera se están generando, retrasándose los accesos a dispositivos externos, mientras no se cumpla que tanto la señal $\overline{(X)RDY_{WTCNT}}$ como los pines de entrada $\overline{RDY}$ ó $\overline{XRDY}$ estén a cero. Esta condición es la más restrictiva y la que aparece en el arranque del DSP (configuración por defecto de los bits SWW=11).

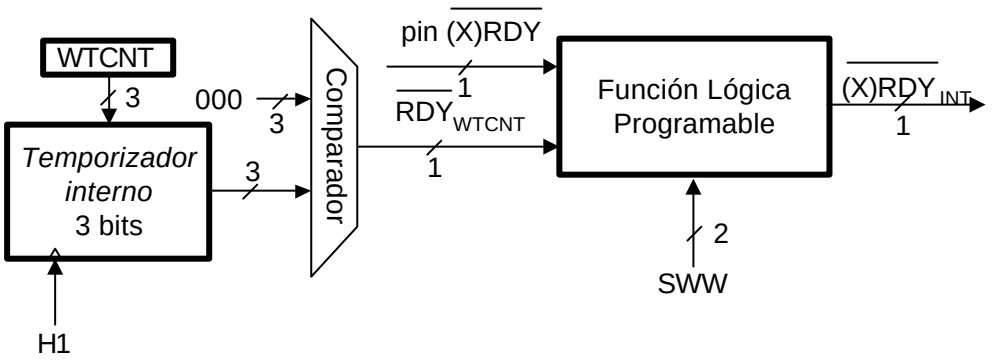


Figura 2.35. Generación de la señal $\overline{(X)RDY_{INT}}$ y de los estados de espera en los accesos externos del DSP.

6.3. Tiempo de ejecución y tiempo de acceso.

En el diseño de un sistema microprocesador basado en un DSP de la familia TMS320C3x, el usuario deberá de garantizar el cumplimiento de la siguiente expresión o desigualdad: $T_{acceso-DSP} > T_{respuesta-periféricos}$ (el maestro o *master* en el sistema microprocesador, el DSP, accede al esclavo o *slave* de manera fiable).

En el acceso del DSP al exterior se distinguen dos tiempos:

1. **Tiempo de ejecución del acceso en lectura o escritura**. Este tiempo hace referencia al tiempo que tarda la CPU del DSP en realizar la operación básica de lectura (de datos o instrucciones) o escritura (de datos):

   - Accesos a través del bus principal ($\overline{STRB}$ activo) o del bus de expansión ($\overline{MSTRB}$ activo). Si los estados de espera se programan a cero, las operaciones básicas de lectura se realizan en 1 ciclo máquina y las de escritura en 2 ciclos máquina (esto ocurre siempre salvo que el acceso en lectura venga después de un acceso en escritura, en cuyo caso el acceso en lectura requiere 2 ciclos máquina). Estos accesos hacen referencia al tiempo que tardaría la CPU del DSP en completar las micro-operaciones (una lectura de instrucción en la fase *fetch*, o de dato en la fase *read* o una escritura de dato en la fase *execute*) en las que hay que realizar una lectura o escritura. En general, y salvo la excepción comentada anteriormente, el tiempo de ejecución de una operación básica de lectura del DSP a través del bus principal o del de expansión con $\overline{MSTRB}$ activo es de $H_1 + N_{WS} \cdot H_1$ (donde N_{ws} es el número de estados de espera que se haya programado) y el de una operación básica de escritura, $2 \cdot H_1 + N_{WS} \cdot H_1$.
   - Accesos a través del bus de expansión con $\overline{IOSTRB}$ activo. Si los estados de espera se programan a cero, las operaciones básicas de lectura y escritura se realizan en 2 ciclos máquina (cuatro ciclos del reloj externo). En este caso, la CPU del DSP tarda más tiempo, que el que tardaba usando el bus principal o el bus de expansión con $\overline{MSTRB}$ activo, en completar las operaciones básicas de lectura y el mismo tiempo en completar las operaciones básicas de escritura. Por tanto, en general, el tiempo de ejecución de una operación básica de lectura y escritura del DSP a través del bus de expansión con $\overline{IOSTRB}$ activo es de $2 \cdot H_1 + N_{WS} \cdot H_1$.

2. **Tiempo de acceso del DSP**. Este tiempo hace referencia al tiempo en el que la CPU del DSP está seleccionando, en lectura o escritura, a un periférico externo y es el tiempo en el que alguna de las señales $\overline{STRB}$, $\overline{MSTRB}$ ó $\overline{IOSTRB}$ está activa. En cualquier caso y

siempre que se haya programado 0 estados de espera, las lecturas y escrituras a periféricos externos tardan 1 único ciclo máquina. En general, el tiempo de acceso en lectura o escritura del DSP a través del bus principal o del de expansión es de $H_1 + N_{WS} \cdot H_1$. Este es el tiempo que hay que considerar en el cumplimiento de los criterios de diseño del sistema microprocesador. El usuario debe garantizar el cumplimiento de la desigualdad: $T_{acceso-DSP} > T_{respuesta-periféricos}$. Para ello, impondrá un número de estados de espera determinado.

Memoria	**Tiempo de acceso del DSP**	**Tiempo de ejecución del DSP**
Interna	$T_{READ-WRITE} = H_1$	$T_{READ-WRITE} = H_1$
Externa ($\overline{STRB}$ ó $\overline{MSTRB}$)	$T_{READ-WRITE} = H_1 + N_{WS} \cdot H_1$	$T_{READ} = H_1 + N_{WS} \cdot H_1$ $T_{WRITE} = 2 \cdot H_1 + N_{WS} \cdot H_1$
Externa ($\overline{IOSTRB}$)	$T_{READ-WRITE} = H_1 + N_{WS} \cdot H_1$	$T_{READ-WRITE} = 2 \cdot H_1 + N_{WS} \cdot H_1$

Tabla 2-12. Cuadro resumen de los tiempos de acceso del DSP a los periféricos ubicados en el mapa de memoria del sistema microprocesador y de ejecución de las operaciones básicas de lectura o escritura.

6.4. Peculiaridades en el acceso a periféricos externos.

Como comentarios, cabe señalar:

1. Después de completar los accesos a través de los buses externos, el bus en cuestión se pone en estado inactivo ($\overline{IOSTRB}$, $\overline{MSTRB}$, $\overline{STRB}$ a valor lógico alto). Además, y como anécdota, cabe señalar que la transición o paso de un acceso en lectura o escritura a través del bus de expansión con $\overline{MSTRB}$ activo al bus de expansión con $\overline{IOSTRB}$ activo genera, por la necesidad de sincronizar las señales de control del bus ($XR/\overline{W}$, $\overline{MSTRB}$ e $\overline{IOSTRB}$), estados de espera adicionales..

2. **Desconexión temporal del DSP de los buses externos conectados al bus principal del microprocesador**. El microprocesador ofrece la posibilidad de ceder el control de los buses asociados al bus principal a un dispositivo externo. La forma de ceder los buses es mediante un protocolo acuse de recibo asociado a las líneas $\overline{HOLD}$ y $\overline{HOLDA}$ (de entrada y salida, respectivamente, del DSP). Esto permitiría a otro dispositivo diferente del DSP gobernar los buses asociados al bus principal (el DSP dejaría de ser, temporalmente, el *maestro* del sistema microprocesador). Por defecto, es el DSP el que gobierna el bus principal. Si el usuario lo permite (bandera NOHOLD=0 en el registro de control del bus principal) el DSP, cuando reciba una petición de ceder el bus principal ($\overline{HOLD} = 0$) se desconecta del bus principal (pone todos los buses asociados al bus principal en triestado) y, cuando ya se ha desconectado totalmente, avisa al dispositivo externo de que puede gobernar el bus principal ($\overline{HOLDA} = 0$). A partir de ese momento el DSP no puede realizar accesos al exterior por el bus principal hasta que no deje de ser maestro el periférico que hizo la petición de HOLD y el DSP no detecte $\overline{HOLD} = 1$. Esta característica está muy relacionada con periféricos de tipo DMA (*Direct Memory Access*) y se comprenderá mejor cuando se estudie el periférico DMA interno del DSP (Tema 17).

3. Finalmente, asociados también al bus principal, aparecen los **bancos de conmutación programables.** Permiten conmutar entre dos dispositivos externos de memoria sin necesidad de insertar estados de espera adicionales desde el exterior.

   Esta característica se asocia con los tiempos de acceso a periféricos externos tan bajos que permite el DSP. Los periféricos de almacenamiento de datos y programas que necesita el DSP para alcanzar toda su potencia de operación deben tener tiempos de acceso muy pequeños. Esto encarece el coste de dichos integrados en el sistema digital. Los dispositivos de memoria externa pueden haberse seleccionado muy al límite en sus tiempos de acceso, respecto de los que precisa el DSP, de manera que dos accesos consecutivos al mismo periférico se realicen correctamente, dentro del tiempo de acceso al periférico. Podría, sin embargo, necesitarse (al hacer un acceso a un periférico seguido

de un acceso a otro periférico diferente del anterior y para garantizar la desconexión del primero antes de la conexión del segundo) la inserción de ciclos extra de acceso al exterior. Se evitan, de esta forma, posibles conflictos en el acceso al bus. Es en este caso donde resulta interesante el empleo de los *bancos de conmutación programables*. El tamaño de dichos *bancos* se programa en los bits BNKCMP del registro de control del bus principal. Un *banco de conmutación programable* se define por el número de MSBs de la dirección que van a ser examinados:

- ✓ Si BNKCMP=16, los 16 MSBs de la dirección se usan para definir los *bancos de conmutación programables* mientras que el resto, los 8 LSBs de la dirección, determinan el tamaño de cada *banco*. En este caso, cada *banco* es de 2^8 =256 *words* de 32 bits. Si fuese BNKCMP>16, BNKCMP=16 (no se permiten *bancos de conmutación programables* de un tamaño inferior a 256 *words*).

- ✓ Si BNKCMP=N<16, los N MSBs de la dirección se usan para definir los *bancos de conmutación programables* mientras que el resto (los 24-N LSBs de la dirección) determinan el tamaño de cada *banco* (en este caso cada *banco* es de 2^{24-N} *words* de 32 bits).

El tamaño de los bancos varía, por tanto, de 2^8 a 2^{24} (caso extremo con BNKCMP=0 y un único *banco*).

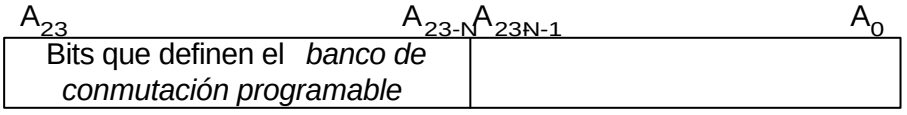
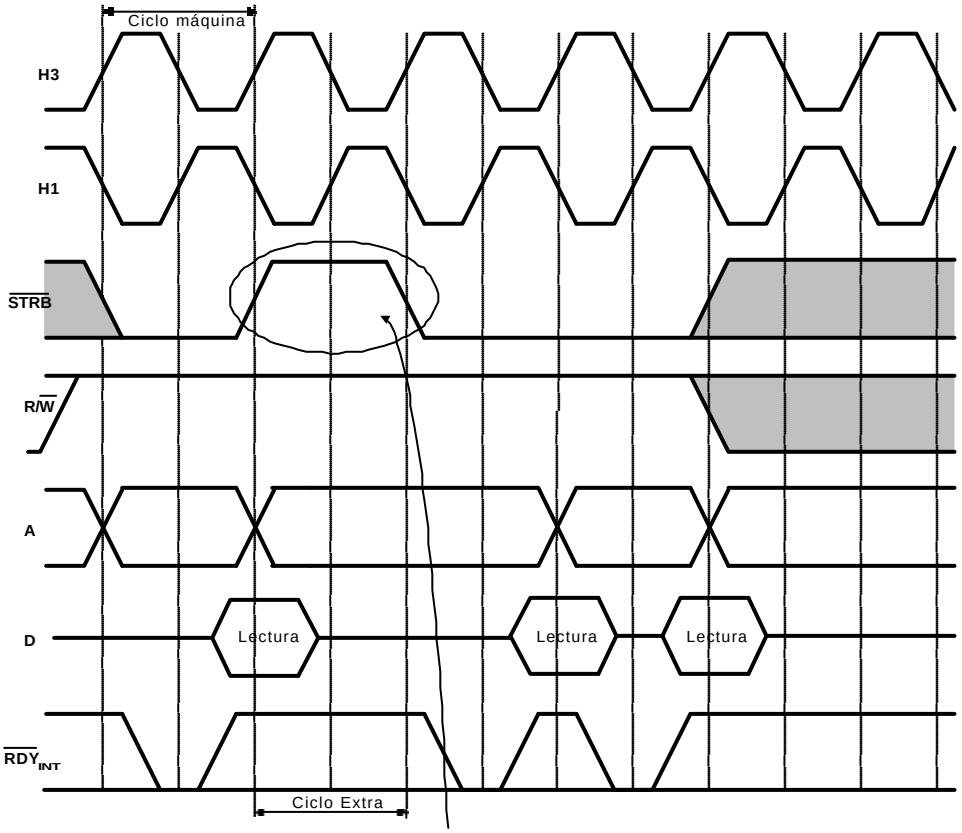


Figura 2.36. Bits de la dirección que definen los *bancos de conmutación programables*.

El DSP dispone de un registro interno que contiene los MSBs, definidos por BNKCMP, de la dirección externa que se accedió por última vez. Si valor de este registro no coincide con los MSBs de la nueva dirección a acceder se inserta, si se está efectuando una operación de acceso externo en lectura que sigue a otra que se acaba de realizar, un ciclo

extra en el acceso (si el acceso fuese en escritura no se insertaría ningún estado de espera). En ambos casos, acceso en lectura o escritura, y durante ese ciclo máquina se cambia el registro interno del DSP que contiene los MSBs de la dirección accedida (se copian en él los MSBs de la última dirección externa que fue accedida). En el acceso en lectura a un banco de conmutación programable distinto del anterior la dirección cambia cuando le corresponde (estados de espera programados) pero la señal de control $\overline{STRB}$ no permanece activa sino que, durante un ciclo máquina, es desactivada (similar a una lectura que sigue a una escritura). Obsérvese que el registro interno que contiene los MSBs del DSP cambia en cada acceso en lectura o escritura, asociado al bus principal, a la memoria externa. Sin embargo, mientras que en los accesos en lectura, los más rápidos, se inserta un estado de espera adicional al conmutar de *banco*, en un acceso en escritura, más lento que el de lectura, no se inserta el ciclo extra al producirse la conmutación del *banco*. En la figura 2.38 se muestra el efecto que produce el cambio de *bancos de conmutación programables* en dos accesos consecutivos en lectura a través del bus principal.



Cambio de Banco de conmutación: Una lectura seguida de otra lectura provoca que la señal $\overline{STRB}$ se inhabilite un ciclo máquina

Figura 2.37. Modo de operación con *bancos de conmutación programables*: Acceso en lectura con cambio de *banco*.