Sistemas Electrónicos para el Tratamiento de la Información 2

Jesus M. Hernández Mangas
3 de febrero de 2011

Índice general

Prólogo

Propósito

El propósito de este libro es, como el de todos los libros, dar a conocer un tema apasionante. El autor, en su dilatada trayectoria en la Universidad de Valladolid impartiendo la asignatura de Sistemas Electrónicos para el Tratamiento de la Información 2, ha recopilado sus apuntes de clase y los problemas resueltos en esta obra con el objetivo de que pueda ser utilizada como obra de referencia en la enseñanza de esta asignatura en la Titulación de Ingeniero en Electrónica y, con las adaptaciones pertinentes, en otras.

Dado que los Ingenieros en su trabajo van a utilizar obligatoriamente referencias bibliográficas escritas en inglés, se ha creído conveniente mantener la notación inglesa en la mayoría de las siglas y en algunas descripciones de banderas (*flags*) de los registros empleados.

Agradecimientos

El autor quiere agradecer la colaboración a todas aquellas personas que de una u otra manera han ayudado en la realización de este proyecto.

Valladolid, 3 de febrero de 2011

Parte I

Microcontroladores

Capítulo 1

Introducción

Sistemas electrónicos para el tratamiento de la información

¿Qué entendemos por Información? Es todo aquello que podemos medir y sobre lo que podemos actuar.

¿En qué consiste el tratamiento de la información? Consiste en la adquisición, operación y devolución de la información....

Sistemas electrónicos Pueden ser analógicos, digitales o mixtos, que suele ser lo habitual.

1.1. Características de los microcontroladores

1.1.1. Microprocesador versus microcontrolador

Un **microprocesador** es un dispositivo programable de propósito general, que consta de una unidad central de proceso y de unos registros.

Un **microcontrolador** es un dispositivo programable orientado al control. Típicamente funcionan con un tiempo de respuesta inmediato (tiempo real). Consta de la CPU y los registros como un microprocesador habitual. Además dispone de una serie de periféricos integrados tales como de entrada/salida, temporizadores/contadores, de comunicaciones, etc.

Campos de aplicación

- Los microprocesadores se utilizan en los computadores y disponen de una gran capacidad de gestión de memoria y de proceso de datos. Suele ser un proceso *off-line*
- Los microcontroladores se emplean en el control de procesos, por lo general en tiempo real (*on-line*). Disponen de una gestión de tiempos incorporada, atención de eventos y periféricos integrados adaptados a tareas específicas.

1.1.2. Arquitectura Harvard vs. Von Neumann

La arquitectura Harvard surgió a partir de una competición entre las Universidades de Princeton y Harvard. Hay que decir que Princeton ganó porque el MTBF (tiempo

medio entre fallos) de su arquitectura con una única memoria era mejor, aunque era una arquitectura más lenta. Con el desarrollo de los transistores y de los circuitos integrados, la arquitectura Harvard se ha puesto en su sitio.

Básicamente la arquitectura Von Neumann consiste en disponer de una memoria común para almacenar tanto las instrucciones como los datos. Este sistema tiene como ventajas el ahorro del coste y de la complejidad del diseño.

En la arquitectura Harvard existen memorias diferentes para datos y para instrucciones. Cada memoria se direcciona mediante buses diferentes, e incluso es posible que la memoria de datos tenga distinta anchura de palabra que la memoria de programa. Este diseño es más rápido que el anterior ya que el sistema puede ejecutar simultáneamente la lectura de datos de una instrucción con la lectura y decodificación de la instrucción siguiente., disminuyendo el tiempo total de ejecución para cada instrucción.

En algunos microcontroladores y en la mayoría de los DSP se utiliza una arquitectura Harvard modificada de 3 buses: uno de programa y dos de datos, lo cual permite que la CPU lea una instrucción y dos operandos a la vez (aunque no dos posiciones de memoria a la vez, ya que sería necesaria una memoria de doble puerto).

1.1.3. RISC versus CISC

La clasificación de los microcontroladores se puede hacer también atendiendo al conjunto de instrucciones que implementan.

CISC (Complex Instruction Set Computers)

- Conjunto de instrucciones amplio. Con un buen número de modos de direccionamiento.
- Facilita la programación.

RISC (Reduced Instruction Set Computers)

- Conjunto reducido de instrucciones.
- Menor tamaño, menor número de pines y menor consumo.
- Más complejo de programar.

SISC (Specific Instruction Set Computer)

- Conjunto específico de instrucciones para facilitar y agilizar la E/S.
- Manejo de interrupciones.
- Manipulación de bits.

1.2. Aplicaciones de los microcontroladores

Tiene numerosas aplicaciones ampliamente utilizadas y que nos rodean. Proporcionan la *inteligencia* a los sistemas electrónicos.

- **Instrumentos portátiles.** Hay muchas aplicaciones: el polímetro, un medidor ultrasónico de distancias, una balanza electrónica, etc.
- **Subfunciones de instrumentos.** Por ejemplo: pantallas táctiles, teclado, ratón, display LCD, On-Screen-Display, etc.
- **Dispositivos periféricos.** Impresoras láser o de tinta, modems, plotters, etc.
- **Dispositivos autónomos.** Fotocopiadora, video doméstico, teléfonos, etc.
- **Aplicaciones en automoción.** Como pueden ser la inyección electrónica, los frenos ABS, la tarificación en los Taxis, cuadro de instrumentos, GPS, etc.

Control en tiempo real

¿Qué es respuesta tiempo real? Se da este nombre a aquellos sistemas en los que las acciones de salida deben estar disponibles en un intervalo de tiempo reducido y acotado desde que se tienen las entradas. Algunos ejemplos de sistemas de control en tiempo real son: la lavadora automática, el control de presencia, la instrumentación electrónica, la robótica, los satélites artificiales, etc. En contraste algunos sistemas que no son en tiempo real: el control de nóminas, la elaboración de estadísticas, las simulaciones *off-line*, etc.

Sin embargo, no todos los sistemas en tiempo real son críticos y su falta respuesta a tiempo no ocasiona graves perjuicios.

- *Sistemas en tiempo real* **críticos**. Son aquellos en que el incumplimiento del plazo temporal de respuesta provoca un fallo catastrófico en el sistema. Por ejemplo: un satélite artificial, un brazo de un robot, un misil anti-misil, etc.
- *Sistemas en tiempo real* **no críticos**. El incumplimiento de los plazos temporales no ocasiona, necesariamente, un fallo en el sistema. Ejemplo: una cafetera, un semáforo, un instrumento de medida en el laboratorio, etc. Por supuesto todo es relativo.

Las necesidades, por tanto, de un sistema en tiempo real pueden ser genéricamente clasificadas como:

- *Estructura temporal*. Deben disponer de generadores de base de tiempos. así como contadores y temporizadores, que pueden ser fijos o programables.
- *Entradas y salidas al exterior*. Tanto analógicas como digitales, que pueden ser individuales o agrupadas, que proporcionen o no alta corriente, y que sean reprogramables en tiempo de ejecución.
- *Capacidad de atención al entorno*. A través de un sistema de interrupciones *hardware* o *software*, que permitan la gestión de eventos.

- *Capacidad de comunicaciones.* Esto es imprescindible cuando tratamos con sistemas distribuidos, y tienen multitud de formatos: serie, paralelo, bucle de corriente, modems, etc. Por supuesto deben permitir la interoperabilidad con equipos de otros fabricantes y con las instalaciones previas existentes, y deben implementar algunos protocolos de comunicaciones.

Las etapas de diseño de un sistema digital en tiempo real son:

1. Conocimiento de los dispositivos aplicables y sus recursos. Es necesario conocer los microcontroladores, microprocesadores, memorias, teclados, displays, sensores, actuadores, etc, que pueden ser empleados en nuestro sistema

2. Conocimiento de los algoritmos y procedimientos idóneos para resolver los problemas típicos: máquinas de estado, teoría de autómatas, redes de Petri, sistemas operativos en tiempo real, etc.

3. Diseño del sistema digital: hardware y software. Es un proceso iterativo y recurrente el que se emplea para llevar a cabo los algoritmos de control.

En resumen, siempre hay que tener presente que el sistema digital debe ser:
Lo más compacto posible, lo más barato posible y con un tiempo de desarrollo lo más corto posible.
Si cabe en un solo chip, y es más barato, y ya estaba hecho antes, ¡mejor!

Algunas consideraciones prácticas

- *Alimentación.* Típicamente a 5 V estabilizados, aunque suelen tener un margen de variación amplio. Son importantes aquellos micros de baja tensión (3 V), que son idóneos para los equipos portátiles (pilas o baterías).

- *Prestaciones asociadas a la alimentación.* Detección de anomalías en el suministro primario de energía: caídas de tensiones, cortes esporádicos. etc. El objetivo es la prevención de fallos catastróficos o pérdidas de datos. Debe permitir la reinicialización en caso de mal funcionamiento.

- *Sistemas con baterías.* Que sean de bajo consumo. Que permitan la autodesconexión en caso de inactividad para así aumentar la duración de las baterías. Se debe detectar la condición de batería baja para prevenir fallos en el sistema.

- *Criterio de diseño.* Si el sistema puede fallar: fallará, en el peor momento posible.

- *Precauciones elementales.*

  - Prever medios alternativos de alimentación. Detectar los fallos de alimentación con antelación suficiente y reaccionar en consecuencia (grabando información vital o activando el reset hasta que se recupere la situación normal, para volver a empezar.
  - Si hay datos vitales para el funcionamiento del sistema deben guardarse en memoria no volátil (configuración, programa, seguridad).

- Hay que garantizar el adecuado arranque del sistema y la inicialización de sus entradas y salidas en un estado conocido.
- Hay que controlar el sistema para que no alcance nunca un estado indeterminado. Para ello se puede utilizar un temporizador *perro guardián* (watch-dog).
- Hay que controlar el reset del sistema y el arranque del oscilador de reloj. Los circuitos RC para el reset dejan mucho que desear en la mayoría de los sistemas. Es mejor un reset temporizado predecible, con una supervisión continua del circuito de alimentación, con un control adecuado del flujo de programa y que permitiera convertir una memoria estática en no volátil. La solución son los circuitos de supervisión. (MC3316P, TL7705, MAX690, MAX691).

Capítulo 2

Características Generales

2.1. Memoria

Los microcontroladores incorporan en su diseño memorias para almacenar los programas y memorias para datos, que pueden ser volátiles o no.

La memoria de programa almacenará las instrucciones del programa de control. Algunos microcontroladores soportan la posibilidad de utilizar memoria exterior, para lo cual incorporan entre sus pines buses de direcciones y datos (que se suelen multiplexar utilizando las mismas líneas) y un bus de control. En otros, como es el caso del PIC16F88 no permiten añadir memoria exterior de una forma directa.

La memoria de datos debe ser de lectura/escritura y puede ser volátil o no. Normalmente los microcontroladores no incorporan grandes cantidades de memoria de datos. Además no existen registros como tales, sino que son direcciones de memoria.

La memoria que implementan los microcontroladores puede ser de diferentes tipos dependiendo de las prestaciones y el coste de los mismos.

2.1.1. Memoria de datos

RAM

Es una memoria de acceso aleatorio (*Random Access Memory*). Se utiliza para almacenar los datos de los programas. En algunos casos puede almacenar el programa mismo (arquitectura Von Neumann). La RAM puede ser dinámica (típico cuando es memoria externa, necesita de un circuito de refresco) o estática (típico cuando la memoria es interna), que es más rápida. La RAM interna no suele ser muy grande (hasta 2 Kbytes).

Bancos de registros

En muchas familias diferentes no existen registros específicos como en los microprocesadores, sino que existe un banco de registros que suelen estar mapeados en la memoria de datos. De entre todas las posiciones de la RAM algunas tienen una funcionalidad específica (*SFR: Specific Function Registers*) y el resto de de propósito general (*GPR: General Purpose Registers*). El número de registros es grande en comparación con los microprocesadores.

2.1.2. Memoria de programa

ROM

La memoria ROM (*Read Only Memory*) almacena el programa de control y se graba en el chip durante el proceso de fabricación. Se utilizan máscaras para ello. El coste de diseño es muy elevado, aunque el coste de producción es muy barato para grandes cantidades (mayor de 500). Se tarda más tiempo en disponer del producto final (8 a 44 semanas).

EPROM

La memoria EPROM (*Erasable Programmable ROM*) se utiliza para la memoria de programa. Es una memoria que puede grabarse eléctricamente multitud de veces, aunque para su borrado es necesaria la exposición de la memoria a rayos ultra-violeta, con unos tiempos de borrado notables (del orden de minutos). Los microcontroladores que incorporan este tipo de memorias disponen de una ventana de cristal para permitir la entrada de los rayos UV. Este tipo de memoria es interesante en la fase de diseño y depuración.

OTP

La memoria OTP (*One Time Programmable*) suele ser una memoria EPROM que carece de la ventana de borrado, por lo que tan solo se puede grabar una vez. Su coste es inferior al de las memorias EPROM y pueden ser una buena idea para el proceso de producción cuando se fabrican cantidades pequeñas o medias o cuando la aplicación no puede esperar.

EEPROM

La memoria EEPROM (*Electrically Erasable Programmable ROM*) permite su borrado de forma eléctrica un número bastante grande (1.000.000) de veces por lo que la hace ideal para diseñar prototipos o para la enseñanza. Su tiempo de borrado es relativamente grande comparado con la memoria de lectura/escritura y necesitan tensiones de programación mayores que la tensión de funcionamiento.

FLASH

Las memorias FLASH-ROM son memorias de solo lectura que al igual que las EEPROM permiten su borrado eléctrico. Sin embargo, sólo se permite el borrado de bloques completos de memoria en vez de un único valor como en el caso de las EEPROM. Esto hace que se puedan diseñar memorias con mayor capacidad. Son ideales para el diseño y para permitir el cambio o actualización del software del controlador *firmware*.

2.2. Interrupciones

El mecanismo de las interrupciones es fundamental en los sistemas que responden en tiempo real. Estos tienen que ser capaces de interrumpir lo que estén haciendo para atender un suceso externo o interno urgente.

2.2.1. Tipos de interrupciones

Existen varias formas de implementar la atención de eventos:

- **Polling.** No es realmente una prestación que proporcionen los microcontroladores sino que es lo que hay que hacer cuando se carece de un mecanismo de interrupción hardware. Consiste en preguntar constantemente a los periféricos si necesitan ser atendidos. El periférico lo indicará con algún flag. Este mecanismo es lento (sobretodo cuando hay muchos periféricos a los que preguntar) e impide al microcontrolador realizar otras funciones mientras pregunta.

- **Interrupciones.** Un método mucho más eficiente que el polling, es que los periféricos interrumpan al microcontrolador cuando tienen datos que transmitir. Este, interrumpirá su trabajo, identificará el periférico y saltará a la subrutina de interrupción correspondiente.

  La mayoría de los microcontroladores dispone de al menos una patilla de petición de interrupción externa, de la que se puede seleccionar su activación por flanco (se subida o de bajada) o nivel. Ambos sistemas tienen ventajas: el flanco es insensible a la duración, pero es sensible a rebotes o ruidos(*glitches*); el nivel debe mantenerse durante un tiempo, pero no le afectan los ruidos.

Otra clasificación atendiendo a su enmascaramiento:

- **Interrupciones enmascarables.** Una interrupción enmascarable es aquella que pude ser inhibida o habilitada (enmascarar significa inhibir la interrupción). Es interesante poder desactivar determinadas interrupciones cuando el microcontrolador esté realizando tareas críticas que no conviene que sean interrumpidas. La mayoría de los micros disponen de un bit de habilitación/inhibición global (GIE: *Global Interrupt Enable*).

- **Interrupciones no enmascarables** Son interrupciones que no se pueden inhibir. Siempre se atienden. A veces se denominan NMI (*Non Maskable Interrupts*). Interesantes cuando hay tareas críticas del microcontrolador que deben hacerse por interrupciones.

Atendiendo a su complejidad:

- **Interrupciones simples.** Las interrupciones simples (sin vectores) consisten en que cuando ocurre una interrupción, el contador de programa (PC) se carga con una dirección específica fija. En esta dirección la subrutina de atención comprueba uno a uno qué periférico es el responsable. Es un mecanismo barato y sencillo de implementar en hardware. Sin embargo la atención a una interrupción es lenta, debido a que hay que comprobar una a una todas las fuentes (en micros con menos de 5 fuentes este suele ser el método). Las prioridades las decide el diseñador ya que la primera fuente de interrupción comprobada es la que mayor prioridad va a tener.

- **Interrupciones vectorizadas.** Las interrupciones vectorizadas son un poco más sencillas de poner en marcha, pero el diseñador tiene menos control sobre las prioridades. Cuando ocurre una interrupción, el hardware salta automáticamente a una dirección que depende de la interrupción que haya tenido lugar. Es un mecanismo mucho más rápido.

2.3. Periféricos

2.3.1. Temporizadores/Contadores

Función: contar tiempo o eventos.
Características típicas:

- El número de bits del temporizador o contador: 8, 16, etc.
- La posibilidad de selección de contar eventos internos (reloj) o externos.
- En el caso de contar eventos externos la posibilidad de seleccionar el nivel o flanco activo.
- Deben tener la posibilidad de dividir la frecuencia de contaje con una pre-escala programable.
- Muchos son capaces de generar una interrupción al finalizar la cuenta (bien por desbordarse o por alcanzar el valor de un registro periodo).
- Tienen la posibilidad de modificar y leer el contenido de la cuenta en cualquier momento (habrá que tener un especial cuidado cuando el temporizador se lee o modifica en dos pasos: parte alta y parte baja).
- Algunos tienen la posibilidad de generar salidas regulares a través de alguna patilla del microcontrolador.

2.3.2. Entrada/salida de propósito general

Función: acceso digital en paralelo a entradas y salidas.
Características típicas:

- Sentido: entrada, salida, bidireccional.
- Posibilidad de cambio de sentido en tiempo de ejecución.
- Capacidad para alimentar LEDs (mA que es capaz de generar o recibir por pin y en conjunto).
- Capacidad de generación de interrupciones al cambiar su estado.
- Salida con colector/drenador abierto.

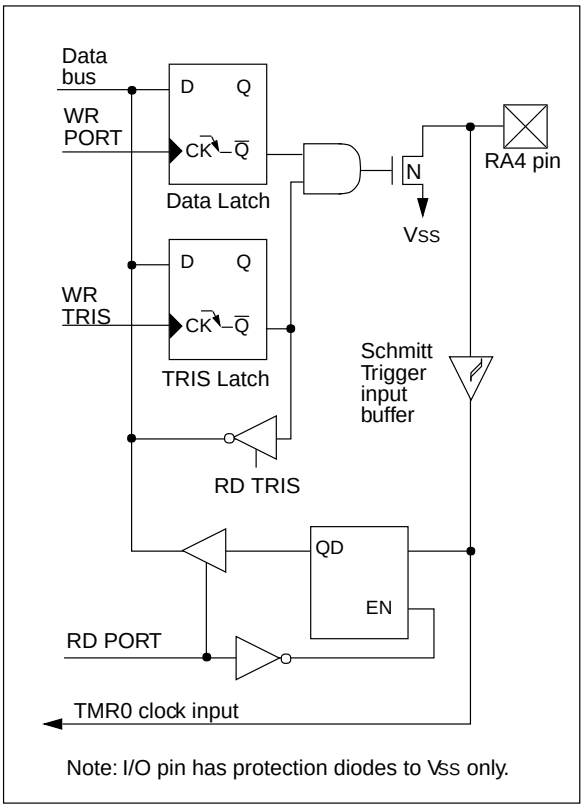
  Es una característica que nos podemos encontrar en algunas patillas cuando funcionan como salida. Hablaremos de drenador abierto si se emplean transistores MOSFET y de colector abierto si se emplean transistores bipolares.

  Consiste en que en el *driver* de salida de la patilla no aparece el transistor que conduce el uno lógico de manera que el microcontrolador es capaz de poner el cero lógico y cuando se trata desde la lógica interna de poner un uno lógico en la salida lo que se consigue es que la patilla se encuentre en alta impedancia.

Para lograr que la señal suba al uno lógico será necesario colocar externamente una resistencia de *pull-up* de un valor apropiado.

Este tipo de patillas viene muy bien para el conexionado tipo *AND* necesario para conectarse a un bus y evitar el cortocircuito entre dos salidas. Todas saben poner un cero lógico y cuando todas ponen un uno lógico es cuando aparece en el bus.

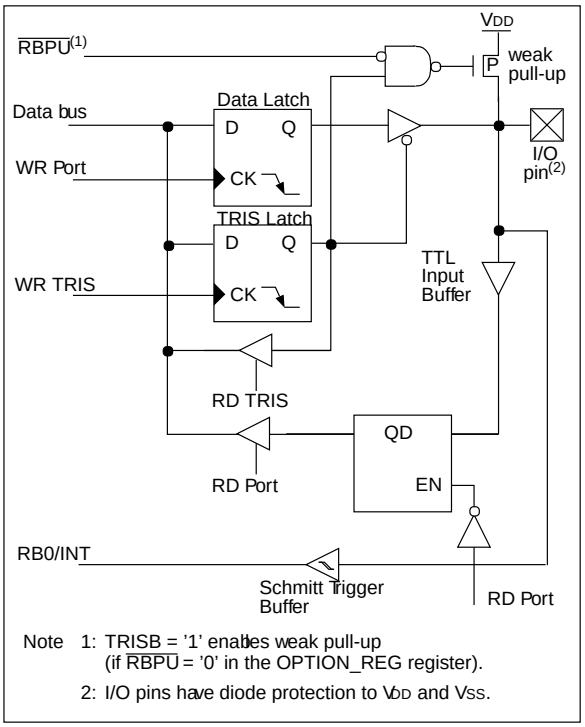
En la figura aparece el diagrama de una patilla de salida con drenador abierto:



- Pull-up entrada programable.

  En ausencia de señal eléctrica el pin de entrada toma el valor lógico 1. Se usa si la señal de entrada va a estar en alta impedancia en algún momento.

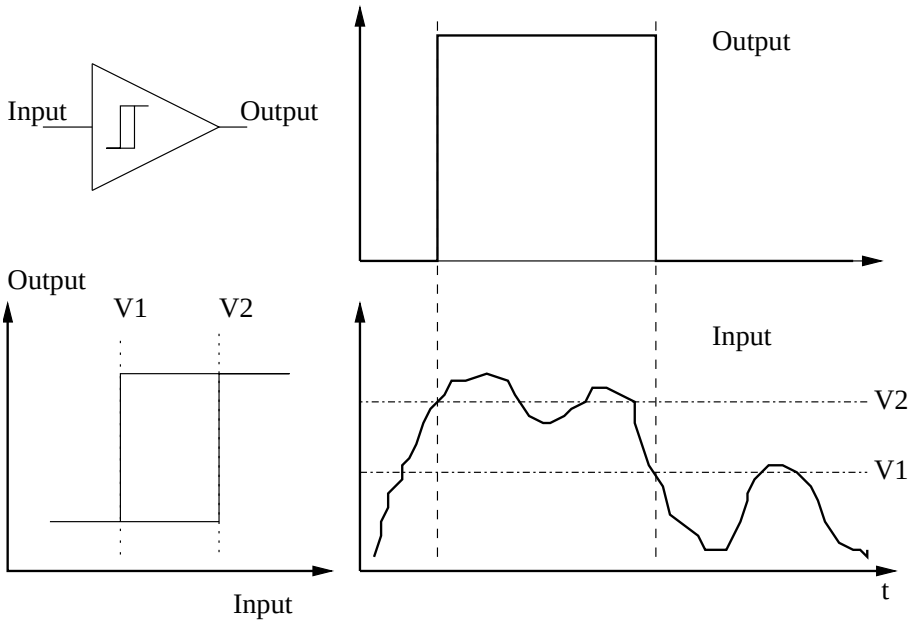
  En la figura podemos observar el diagrama de una patilla que implementa la resistencia de pull-up programable mediante un transistor MOS.



- Entrada con disparador Schmitt.

  Consiste en un búfer de entrada con ciclo de histéresis. Se usa para reducir el ruido cuando la señal es ruidosa y para acelerar el cambio de estado en las señales lentas.

  En la figura podemos ver el esquema de funcionamiento de la histéresis:



- Multifuncionalidad. Significa que los pines están compartidos con otros periféricos internos.

2.3.3. Entrada/salida serie

Función: Acceso digital en serie. Típicamente para comunicación con otros periféricos externos u otros uC.

Características típicas:

- Es importante saber si se dispone del hardware específico para facilitar la transferencia serie o no. En caso negativo es posible que se pueda resolver la transferencia implementando el algoritmo de transmisión/recepción por *software* (cambiando el estado de los pines y temporizando : *bit-banging*).
- Tipo de comunicación:
  - Serie síncrona: bus I2C, bus SPI, bus MicroWire, etc.
  - Serie asíncrona RS-232C, USB, etc.
- Velocidad de transferencia.
- Comunicación punto a punto o en bus.

Comunicación serie asíncrona

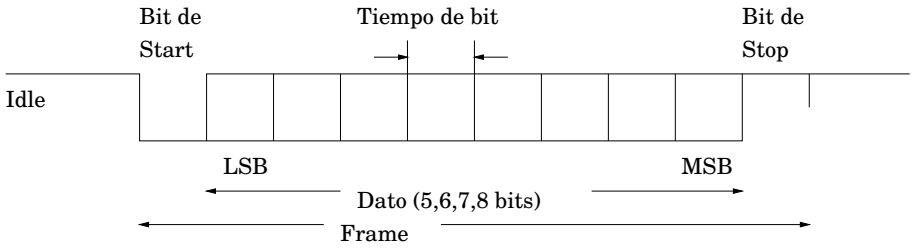
Consiste en la transmisión serie de palabras de datos multi-bit en intervalos indeterminados donde cada palabra tiene un número fijo de bits y un tiempo de bit constante.

Tanto emisor como receptor disponen de un reloj para medir el tiempo de bit (conocido por ambos), pero pueden no coincidir exactamente o estar desfasados. Es necesaria una resincronización periódica. El primer bit de cada dato será un bit de sincronización.

Para finalizar la transferencia de un dato se señaliza con un bit de stop. Puede haber más de un bit de stop (1,1$\frac{1}{2}$,2)

Se puede añadir un bit de paridad entre el MSB y el bit de stop para comprobar si la transmisión fue correcta[1]:

- Paridad par (nº de unos transmitidos es par, incluida la paridad)
- Paridad impar (nº de unos transmitidos es impar).



La comunicación es bidireccional (dos líneas: TxD y RxD, y GROUND).

- Simultanea (full duplex).Control de flujo hardware (2 líneas extra) o software (Xon/Xoff).

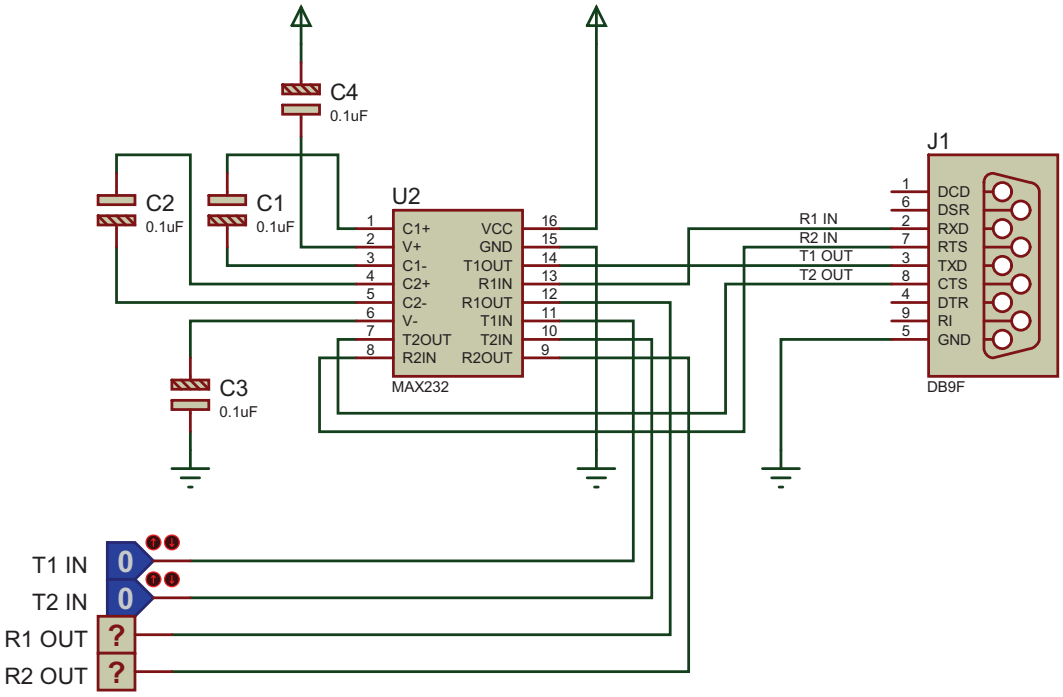
[1]En realidad comprueba si ha habido un número impar de errores en la transmisión. Un número par de errores haría que el bit de paridad fuera correcto.

- No simultanea (half duplex). Control de flujo hardware para decidir quien transmite, o configuración maestro-esclavo.

El estándar RS-232C define los siguientes niveles eléctricos:

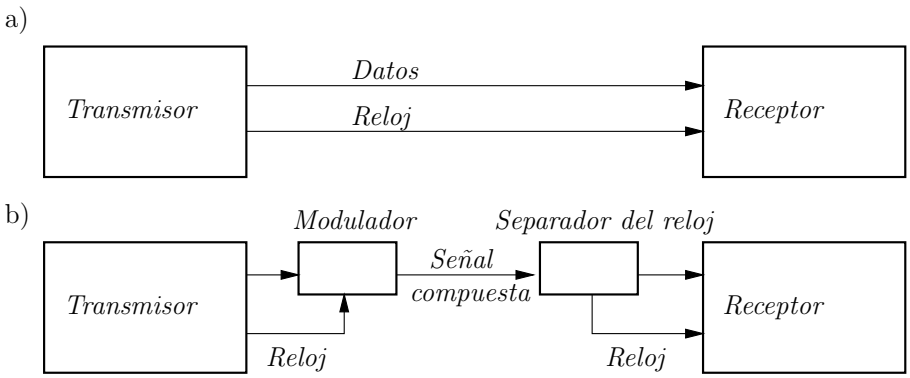
- Espacio, 0 lógico: +3 .. +25 V
- Marca, 1 lógico: -3.. -25 V

El montaje típico para convertir los niveles eléctricos es:



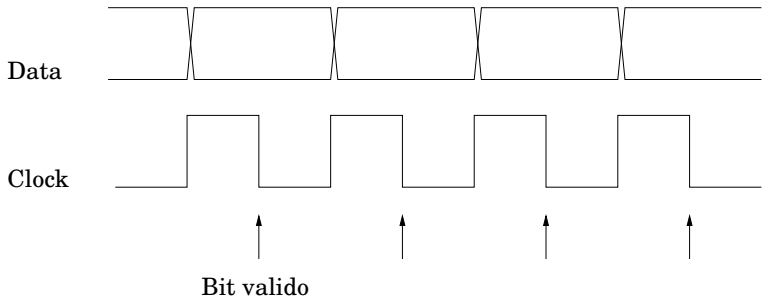
Comunicación síncrona

Consiste en la transmisión serie de datos multi-bit, donde cada bit transmitido está asociado con un pulso de reloj transmitido por una línea de datos separada o compuesto con la misma señal de datos.



Dentro de la transmisión puede ocurrir que existan bits de marcaje de datos o no, y estos bits pueden o no estar asociados al reloj.

La transmisión de bits señalará cuando un bit de datos es válido:



En la transmisión de datos los bytes se reconocen contando bits (bus SPI, longitud fija) o pueden necesitar bits de marcaje especiales (bus I2C).

Algunas características de estos métodos de comunicación son:

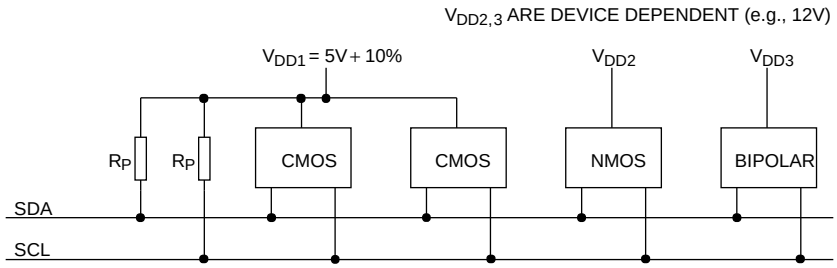
- Teóricamente puede ser más rápida que la transmisión asíncrona.
- La comunicación puede ser punto a punto o en bus.
- Si es una comunicación en bus entonces existe un **protocolo de bus** donde:
  - Hay un dispositivo maestro (uC) y los demás son esclavos (perif. o uCs).
  - El maestro inicia y finaliza la comunicación (controla el bus).
  - El maestro direcciona a los esclavos y genera la señal de reloj.
  - Algunos protocolos permiten cambiar quién es el maestro (**arbitraje del bus**).
- Periféricos.soportados: Memorias serie EEPROM, RAM, etc. ADC y DAC. Displays, RTC (Real Time Clocks).

Entre los diversos estándares de comunicación serie síncrona nos encontramos con:

- **Bus I2C** (*Inter Integrated Circuit, Phillips*). Dos líneas (reloj: SCL y datos: SDA). Además de Vdd y GND. Half duplex. Direccionamiento software de los esclavos.
- **Bus SPI** (*Serial Peripheral Interface, Motorola*). Tres líneas (entrada: MOSI, salida: MISO, reloj: SCK). Además de Vdd, GND y líneas de selección del esclavo. Full duplex. Longitud fija de datos.
- **Bus MicroWire** (*National Semiconductor*). Tres líneas y la selección del esclavo además de Vdd y GND. Full duplex. Longitud fija de datos.

Comunicación síncrona I2C

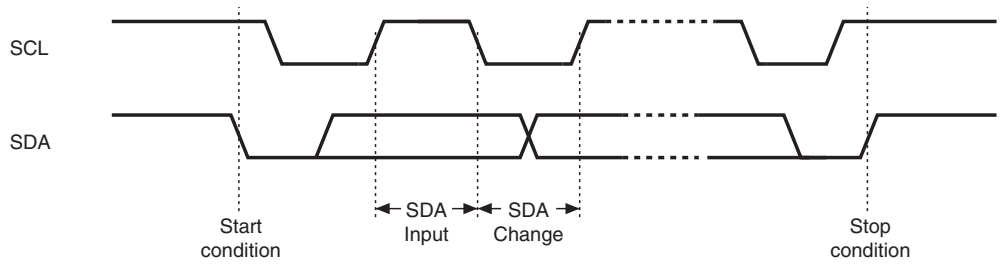
Es un protocolo de bus multi-maestro. Dispone de un mecanismo de detección de colisiones.



- La velocidad de transferencia máxima es de 100 kbit/seg (estándar ampliado a 400 kbit/seg.)
- El bus necesita unas resistencias de pull-up (10k a 100kHz, 2k a 400 kHz) en las líneas de reloj y datos.
- Los transistores de salida de las líneas conectadas al bus deben ser de drenador o colector abierto (conexión AND).
- La capacidad máxima soportada del bus es de 400 pF (que limitará el máximo nº de dispositivos que se podrá conectar).
- El control de flujo se realiza enviando bits de reconocimiento ACK (SDA=0) por parte del maestro o del esclavo.
- La tensión del bus no es fija. Los periféricos ponen un 0 o sueltan el bus que va a 1 gracias a las resistencias de pull-up.

Condición de inicio y de parada. El bus se considera ocupado después de la condición de inicio y se considerará libre un cierto tiempo después de la condición de parada.

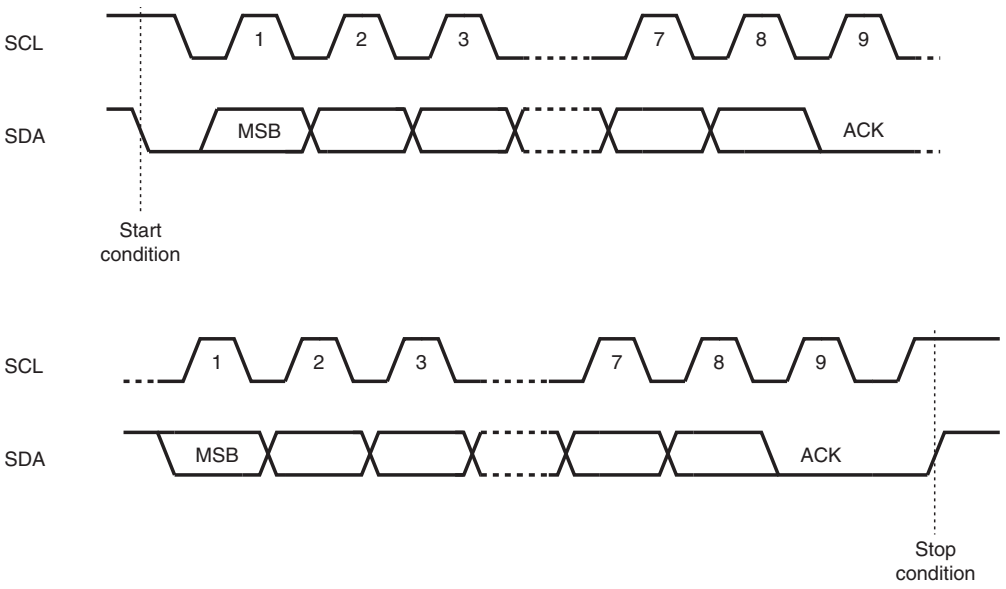
- Condición de inicio: SCL en alta y el dato cambia de 1 a 0.
- Condición de parada: SCL en alta y el dato cambia de 0 a 1.



Transferencia de un bit.

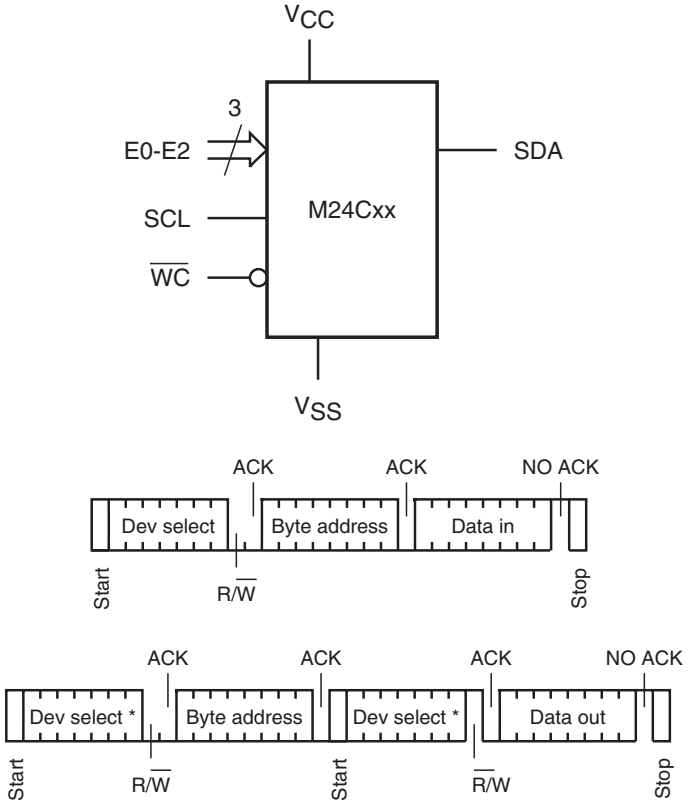
- Cuando SCL está en alta el dato es válido.
- Cuando SCL está en baja el dato puede cambiar.

Transferencia de datos. Los bytes son de 8 bits y tienen una señal ACK que genera el receptor.

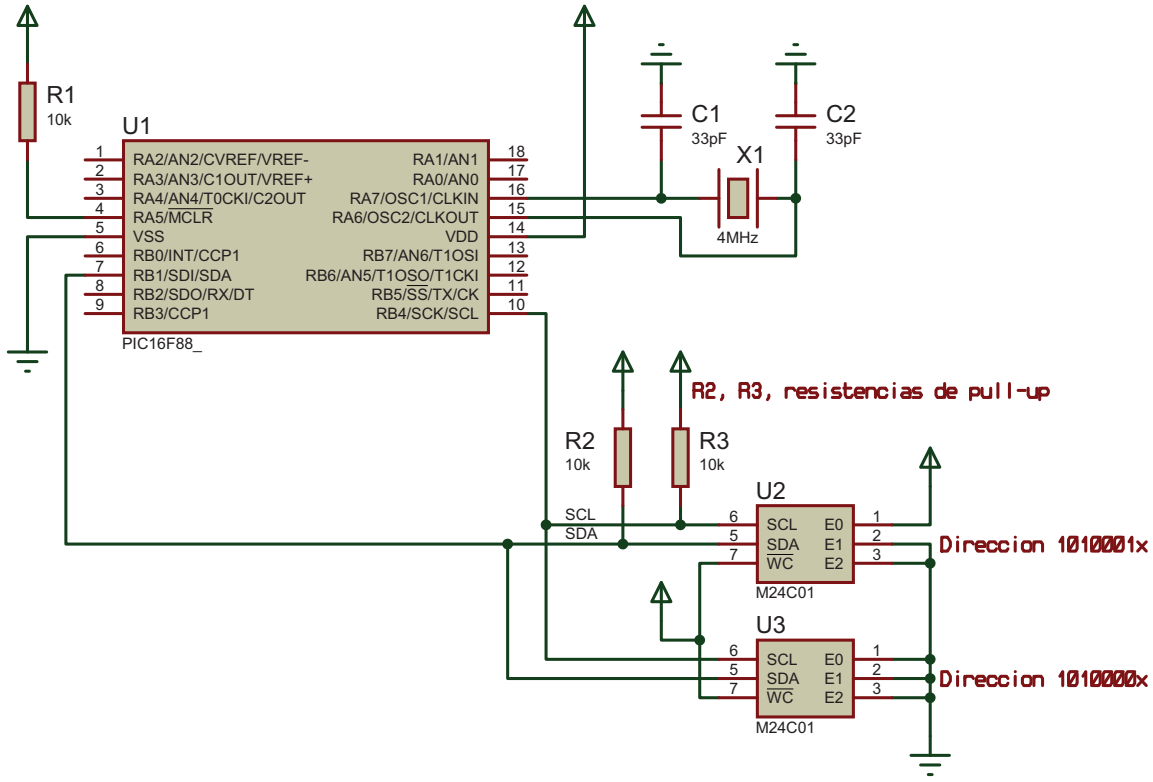


Esquema típico: La trama que se envía comienza con la dirección del dispositivo esclavo al que va dirigida. Al enviar el maestro la dirección, todos los esclavos la leen y la comparan con la suya propia y responden con ACK si coincide.

A modo de ejemplo se presentan las tramas de escritura y lectura en una memoria EEPROM M24C01.



En la siguiente figura se muestra el conexionado de dos memorias con un microcontrolador. Las direcciones de acceso de cada memoria serán las mostradas en la figura.

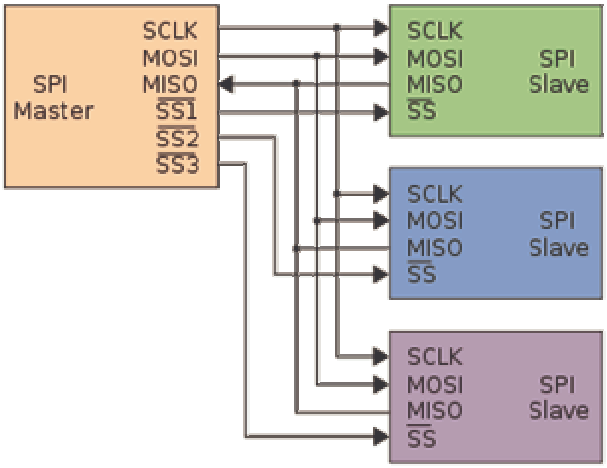


Comunicación síncrona SPI

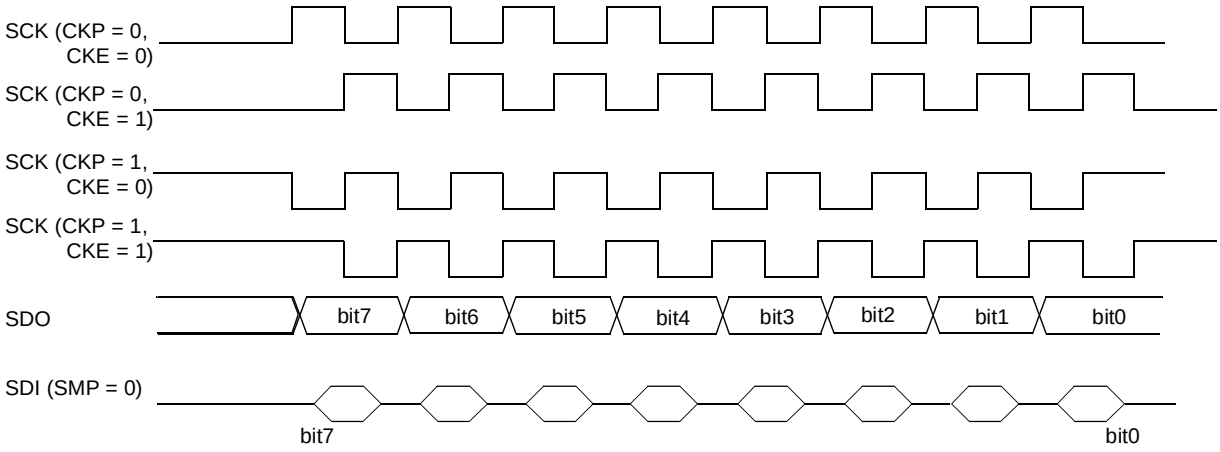
Características:

- Bidireccional maestro/esclavo (full-duplex).
- Maestro: inicia comunicación, genera reloj y líneas selección esclavo.
- Más de un maestro: arbitraje de bus.
- Velocidad de transferencia hasta 1Mb/s.
- Se transmiten 8 bits comenzando por el bit más significativo.

El esquema general de conexión se puede ver en la siguiente figura:



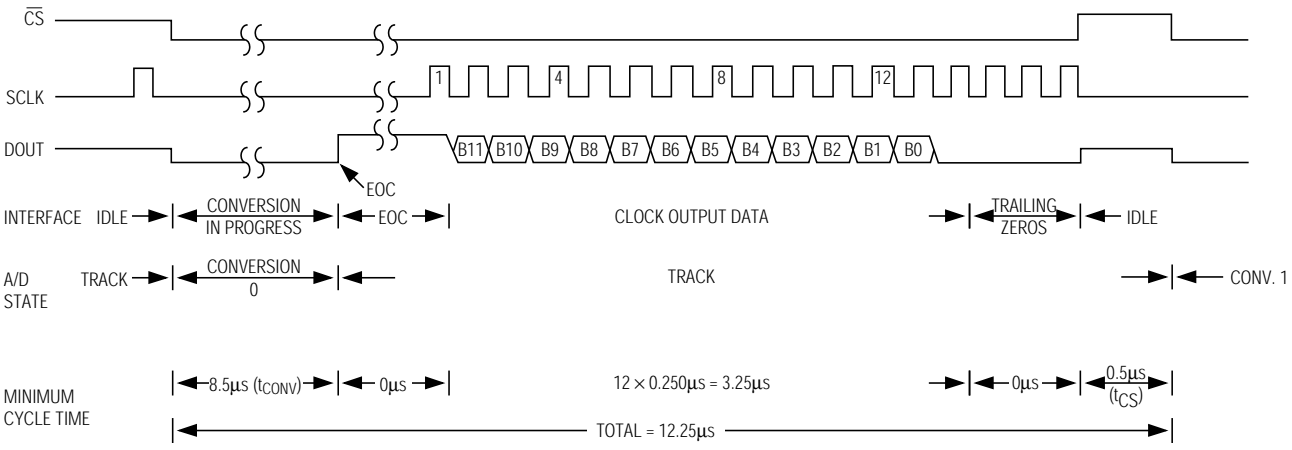
Tiene cuatro modos de funcionamiento dependiendo de la fase (CKP o CPHA) y el flanco (CKE o CPOL) de la señal de reloj (SCK):



Algunos ejemplos de dispositivos SPI:

- Conversor A/D SPI 12 bits (MAX187)
- Memoria EEPROM serie SPI (93C46) 64x16
- Memoria EEPROM serie MicroWire (25C80)
- Display driver MC14489

Ejemplo de cronograma del conversor MAX187:



2.3.4. Conversores A/D

Función: Acceso digital a señales analógicas. Para aplicaciones de instrumentación y captura de datos.

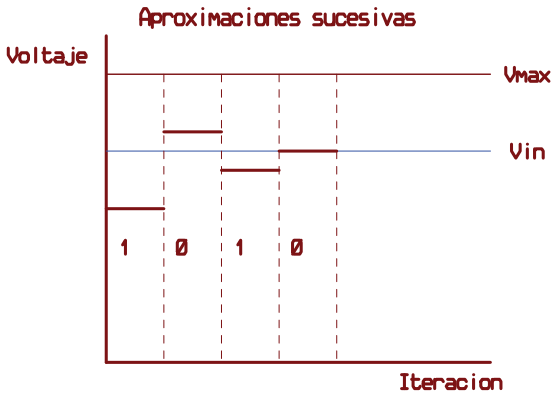
Características típicas:

- Hardware específico interno o no.

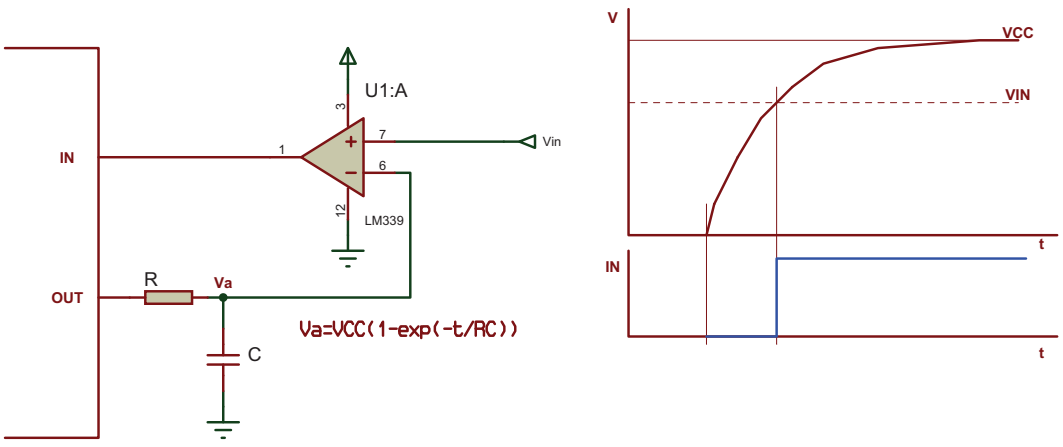
- Se convierten tensiones.
- Tiempo de conversión.
- Resolución (8, 10, 12, 14, 16 bits)
- Varias entradas multiplexadas.

Métodos:

- Conversión por **aproximaciones sucesivas**.



- **Conversor RC**. Se basa en contar el tiempo de carga de un condensador.



El procedimiento a seguir sería:

- Se mantiene OUT a cero hasta asegurarnos de que el condensador esté completamente descargado.
- Se pone OUT=VCC y se cuenta el tiempo que tarda en cambiar IN.
  El valor de tensión se corresponderá con: $V_a = V_{cc}(1 - \exp(-\frac{t}{RC})$. Despejando t hay que hallar un logaritmo neperiano para lo cual será necesario disponer una tabla e interpolar en ella.

- **Conversor Sigma Delta** ($\Sigma\Delta$). Se basa en el mismo esquema hardware anterior. Consiste en:

  - Poner en la salida (OUT) unos y ceros lógicos para conseguir mantener $V_a = V_{in}$.
  - La secuencia durará N ciclos (es un sistema sobremuestreado).
  - Se enviará por OUT el valor de IN complementado.

  El valor medido será: $V_{in} = \frac{N_1}{N}$ donde N_1 es el número de veces que hemos tenido que sacar un uno lógico.

  Con esto podemos conseguir mucha resolución (12, 16 bits, etc) empleando solo dos patillas y un comparador. Este convertidor es lineal en la respuesta pero hay que tener cuidado con el error de offset y la ganancia.

2.3.5. Conversores D/A

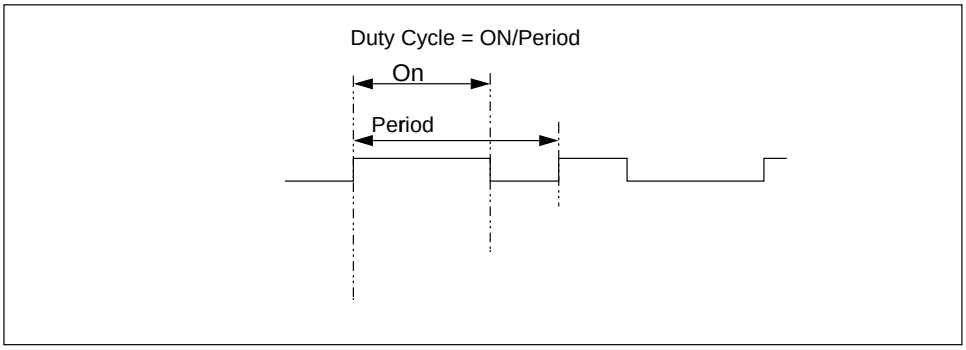
Función: Salida analógica.
Características típicas:

- Hardware específico interno o no.
- Tiempo de conversión.
- Resolución (8, 10, 12, 14, 16 bits).
- No es común que dispongan de este tipo de conversores.

2.3.6. Conversores PWM

Función: Salida digital periódica. Es una forma de conversor D/A.
Características típicas:

- Hardware específico interno o no.
- Generación de frecuencia.
- El ciclo de trabajo (*duty cycle*) se selecciona para que la señal una vez filtrada (pasa-baja) tenga como tensión efectiva la que se desea.



2.3.7. Controlador DMA

Función: Movimiento de datos de memoria a memoria, E/S a memoria, memoria a E/S y E/S a E/S sin intervención de la CPU del uC.

Características típicas:

- Espacio direccionable.
- Número de canales.
- No es muy común.
- Sólo cuando se puede poner memoria externa

2.4. Otras características

2.4.1. Temporizador *watch-dog*

Características:

- El *Watch dog* o perro guardián es un temporizador interno del uC que dispone de su propio oscilador (RC) interno.
- Cuando se activa comienza una cuenta, predefinida por el programador, de forma que cuando finaliza se produce un reset en el uC.
- El programa debe refrescar el contaje del perro guardián para evitar el reset.
- Es un mecanismo pensado para evitar que el uC se encuentre en un estado indeterminado como consecuencia de un error de programación o de un fallo hardware (alimentación, etc.).
- Al rearrancar el sistema se vuelve a un estado conocido y seguro.

2.4.2. *Boot-loader*

El cargador de arranque o *bootloader* es una prestación *hardware* que incorporan algunos microcontroladores y que consiste en que al activarse unos pines del micro al tiempo que se activa el reset del mismo se entra en un estado en el que se carga un programa en el microcontrolador.

Los datos que se cargan pueden proceder de una zona de la memoria externa (carga en paralelo) seleccionable via la activación de algunos pines.

También puede ocurrir que el programa a ejecutar se cargue en la memoria del sistema a través del puerto serie.

Este mecanismo (es un programa) puede venir incorporado de fabrica en una zona de memoria protegida o se puede implementar en aquellos microcontroladores que permitan la escritura de la memoria de programa en tiempo de ejecución.

2.4.3. Programación *In System*

Algunas familias de microcontroladores se pueden programar en forma serie y otros se programan con un protocolo paralelo. Cada fabricante tiene su protocolo bien definido y explicado en sus respectivos *datasheets*.

Sin embargo, y para reducir el tiempo de desarrollo algunos fabricantes incorporan la posibilidad de programar el microcontrolador en el mismo sistema en que va a ser montado el micro. Esto es lo que se llama programación en el sistema o programación en circuito (*In-Circuit Serial Programming*).

Se puede realizar en forma serie utilizando:

- Una señal de reloj (transferencia serie síncrona),
- Una señal de datos,
- La masa,
- La alimentación Vcc
- Y la señal de tensión de programación Vpp.

Los fabricantes de sistemas electrónicos pueden fabricar sus placas utilizando microcontroladores sin programar, y programarlos cuando se vaya a vender el producto, permitiendo utilizar el software:*firmware* más reciente, o incluso software hecho a medida para una determinada aplicación con la misma placa PCB.

2.4.4. Protección del software

En algunos microcontroladores toda la información grabada en él puede protegerse via *hardware*. Existen unos bits de configuración que activan la protección de la información cuando se intenta leer después de programado.

Con este mecanismo se puede proteger por separado la información de la EEPROM y de la memoria FLASH, según la familia del microcontrolador.

La protección puede consistir en que se lea siempre el mismo valor independientemente de la posición de memoria leída, con lo que no es un camino reversible, o puede usar un algoritmo de encriptación reversible (poco frecuente).

Cuando los bits de protección no se han activado, la memoria de programa interna se puede leer para verificar su contenido[2].

2.4.5. Reducción del consumo eléctrico

Los microcontroladores pueden funcionar en sistemas autónomos o alimentados por baterías. Es por tanto muy interesante algún mecanismo para alargar en lo posible la duración de las baterías.

Para ello disponen de modos de funcionamiento de bajo consumo que pueden consistir en:

[2]Es un fallo de principiante el activar la protección y pensar que el programador no funciona cuando se hace la verificación.

- El sistema reduce su frecuencia de funcionamiento.
- El sistema entra en un estado de reposo (SLEEP) pero manteniendo las salidas. Congela el reloj (arquitectura estática).
- El sistema entra en un estado de muy bajo consumo sin mantener las salidas (HALT).

Al sistema se le puede despertar mediante interrupciones externas.

Capítulo 3

Microcontrolador Microchip PIC16F87/16F88

3.1. Introducción

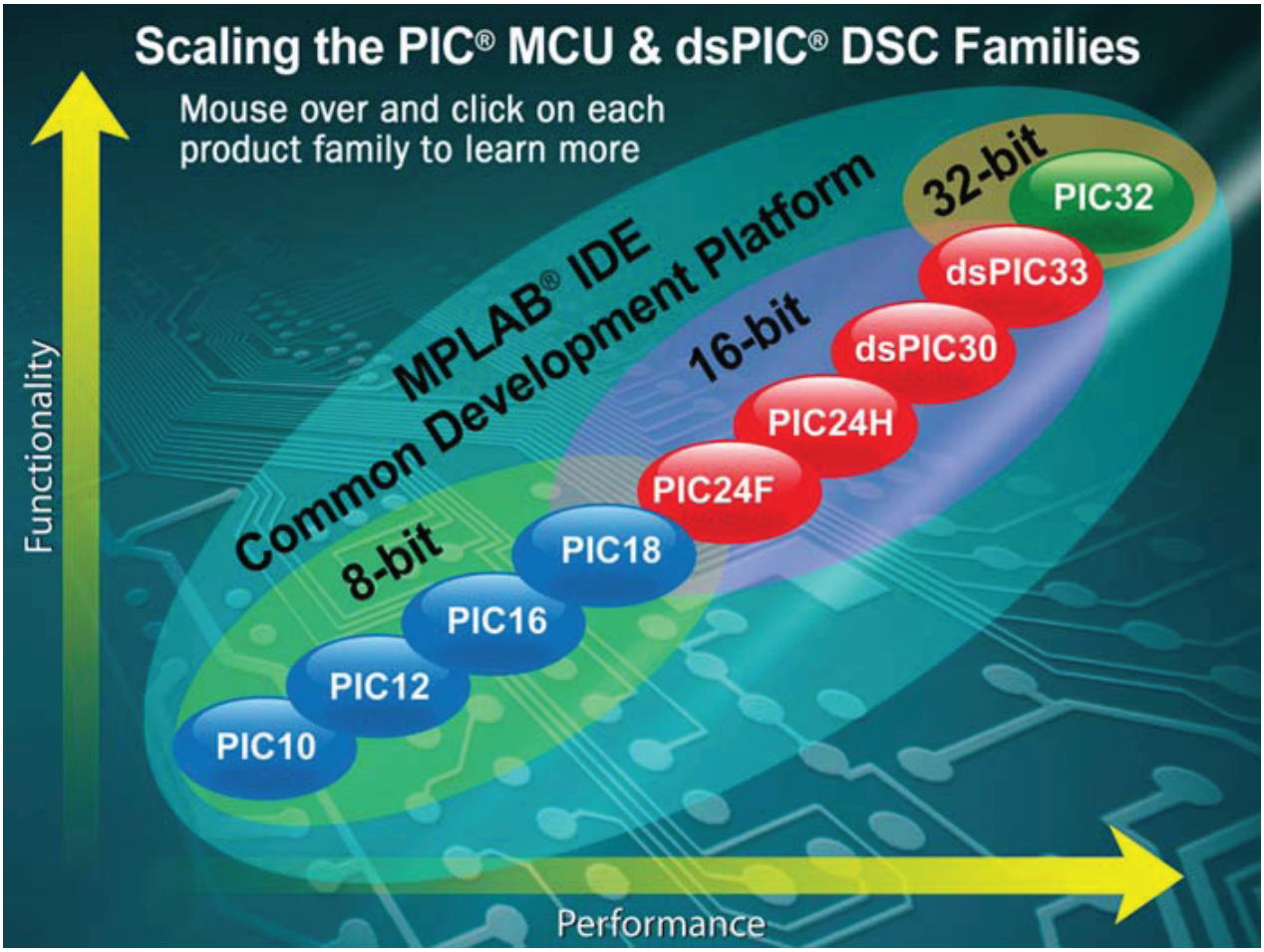
3.1.1. Historia

En 1975, la división de Microelectrónica de GI (*General Instruments*) decidió que era necesario para algunas aplicaciones muy específicas un buen circuito para el manejo de la entrada/salida, y diseñó el *Peripheral Interface Controller* (PIC). Fue diseñado para ser muy rápido ya que iba a controlar la entrada/salida de un micro de 16 bits, y como no necesitaba mucha funcionalidad su conjunto de instrucciones era muy reducido. La arquitectura diseñada en 1975 es básicamente la que hoy tienen los PIC de gama media, solo que ahora se fabrican con tecnología CMOS.

3.1.2. Familias

Dentro del fabricante Microchip[1] tenemos unas cuantas familias a elegir según la complejidad de la aplicación a desarrollar y el coste del producto:

[1] `http://www.microchip.com`



En este momento vamos a estudiar un circuito de la familia PIC16 que es el PIC16F88.

3.1.3. Características generales

Entre otras características tenemos:

- Juego de 35 instrucciones de 14 bits compatibles con la familia PIC16. Todas las instrucciones ocupan una palabra.
- Hasta 20MHz (ciclo de instr. 200 ns, 5 MIPS). Cada ciclo de instrucción emplea 4 ciclos de reloj. Todas las instrucciones duran un ciclo de instrucción excepto las de salto que emplean dos.
- Memoria de programa FLASH de 4096 palabras (14 bits). Borrable/escribible hasta 100.000 veces.
- Memoria de datos RAM de 368 bytes.
- Memoria de datos EEPROM de 256 bytes. Borrable/escribible hasta 1.000.000 de veces. Retención de datos $>$ 40 años.
- Hasta 16 pines de E/S con control individual de sentido.

- Periféricos integrados:
  - 3 Temporizadores (2 de 8 bits/ 1 de 16 bits).
  - Módulo de Captura, Comparación y PWM (*Pulse Width Modulation*).
  - Convertidor Analógico Digital de 10 bits con 7 canales (sólo 16F88).
  - Módulo Puerto Serie Síncrono (SPI e I2C).
  - Módulo AUSART (*Addressable Universal Synchronous Asynchronous Receiver Transmitter*).
  - Comparador analógico doble.
- Programación serie *In Circuit* via 2 pines.
- Reset fiable: *Power-on Reset* (POR), *Power-up Timer* (PWRT), *Oscillator Start-up Timer* (OST).
- *Watch Dog Timer* (WDT). Variable de 1 ms a 268 segundos.
- Protección del código.
- Modo de ahorro de energía (SLEEP).
- Múltiples opciones de oscilador seleccionables. Oscilador interno (8MHz).
- Consumo típico < 2 mA @ 5V, 4MHz
- Tensión de alimentación de 2.0 a 5.5 V

Tenemos dos modelos casi idénticos que solo se diferencian en el convertidor AD:

Device	Program Memory		Data Memory		I/O Pins	10-bit A/D (ch)	CCP (PWM)	USART	Comparators	SSP	Timers 8/16-bit
	FLASH (bytes)	# Single Word Instructions	SRAM (bytes)	EEPROM (bytes)							
PIC16F87	7168	4096	368	256	16	n/a	1	Y	2	Y	2/1
PIC16F88	7168	4096	368	256	16	1	1	Y	2	Y	2/1

Patillaje

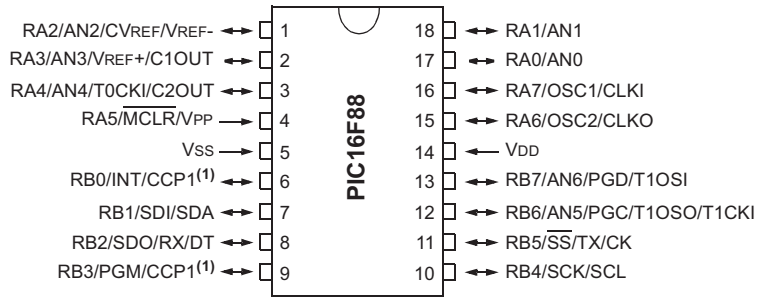
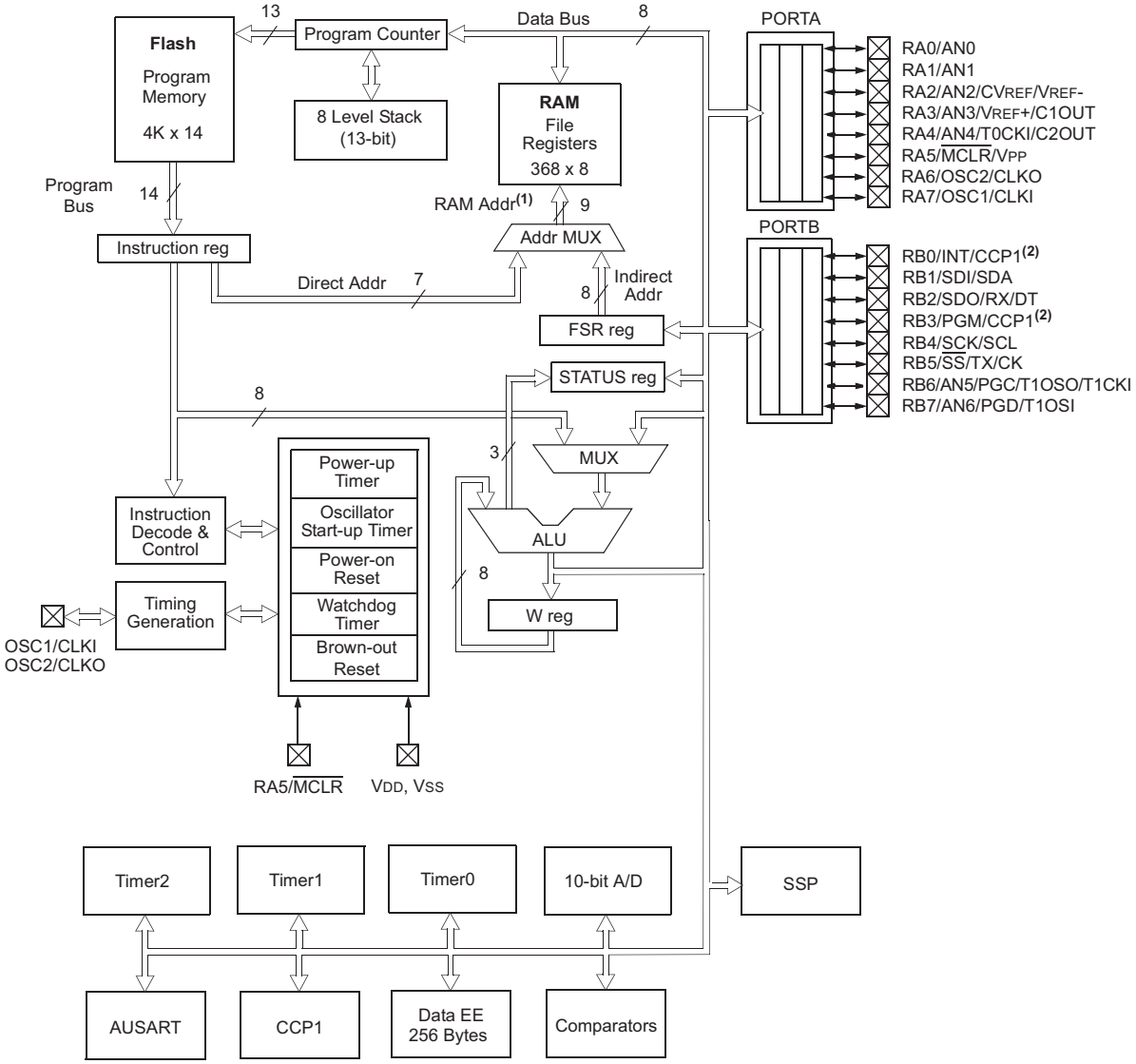


Diagrama de bloques



3.2. Organización de la memoria

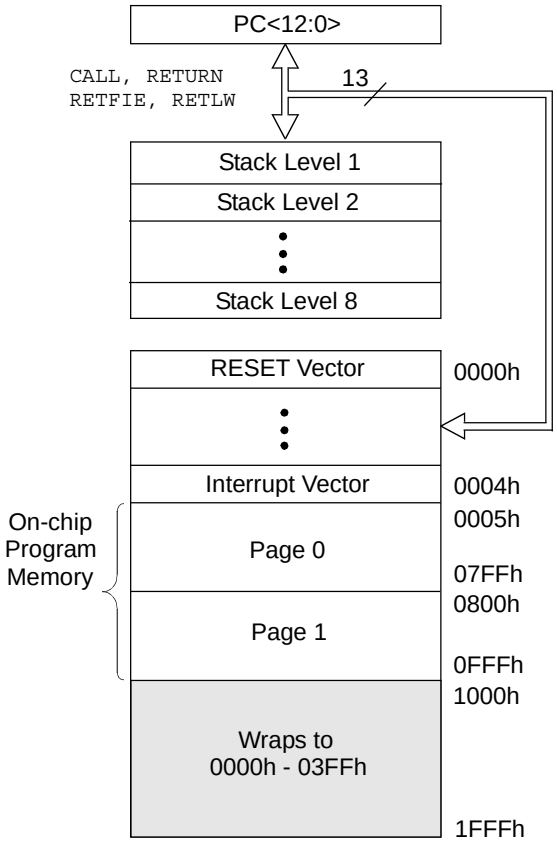
La memoria se divide en memoria de programa y memoria de datos ya que esta familia de microcontroladores implementa una arquitectura Harvard. Además presenta memoria de datos no volátil tipo EEPROM.

3.2.1. Memoria de programa

Esta memoria es de tipo FLASH.

- Se direcciona con el contador de programa que tiene 13 bits.
- En los microcontroladores PIC16F87/88 es de 4k x 14 bit.
- El microcontrolador comienza la ejecución en la posición 0 de la memoria de programa.

- Cuando se produce una interrupción se salta a la posición 4 de la memoria de programa.
- Dispone de una pila hardware de 8 niveles para los retornos de las subrutinas ó interrupciones exclusivamente.
- La memoria de programa está dividida en bloques ó páginas de 2k x14 bits. Esto será importante cuando usemos las instrucciones de salto (**GOTO** ó **CALL**).



3.2.2. Memoria de datos

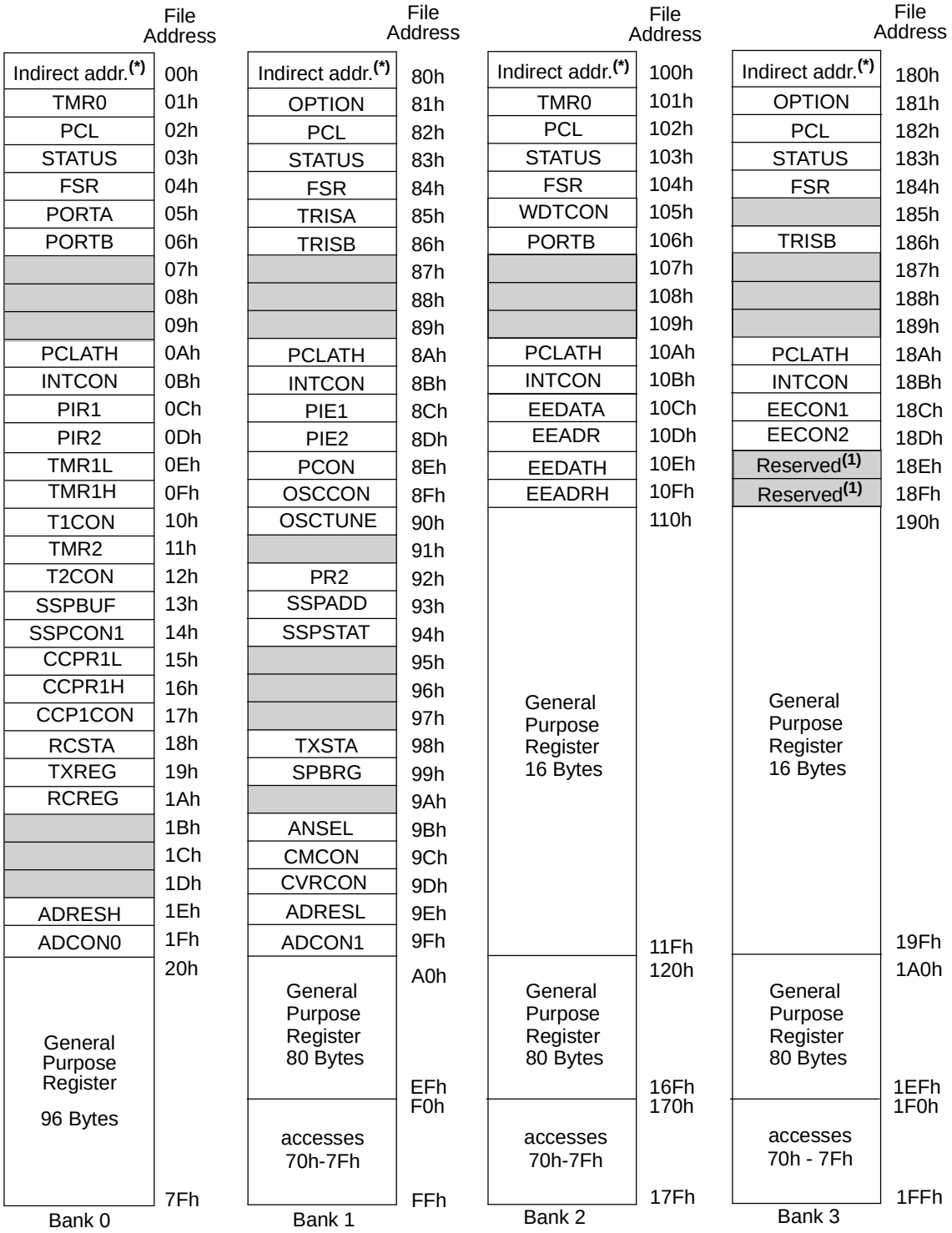
La memoria de datos está dividida en dos bloques: uno implementado en RAM y que describimos a continuación y otro bloque independiente implementado en memoria EEPROM que se verá más adelante. La memoria RAM de datos tiene las siguientes características:

- En los microcontroladores PIC16F87/88 la RAM es de 368 bytes.
- Está dividida en 4 bancos de 128 posiciones (el máximo es de 512 bytes eb otros dispositivos de la misma familia).
- Hay que seleccionar el banco accesible mediante los bits **STATUS.RP1:RP0**.
- En las primeras 32 posiciones de la RAM se encuentran mapeados los registros de propósito específico (SFR: *Special Function Registers*).

- El resto serán posiciones para los registros de propósito general (GPR: *General Purpose Registers*).

3.2.3. Registros de propósito específico

A continuación se muestra un esquema general:



Descripción de algunos registros

- **STATUS**. Presente en todos los bancos. Registro de estado y control.

STATUS

Bit 7	6	5	4	3	2	1	Bit 0
IRP	RP1	RP0	/TO	/PD	Z	DC	C

IRP Selección de banco para direccionamiento indirecto.

0: Selecciona bancos 0 y 1.
1: Selecciona bancos 2 y 3.

RP1:RP0 Selección de banco con direccionamiento directo.

00: Banco 0.
01: Banco 1.
10: Banco 2.
11: Banco 3.

/TO Indica un desbordamiento del temporizador del perro guardián.

0: Desbordamiento (*timeout*) del perro guardián.
1: Después de un arranque normal, o la ejecución de las instrucciones **CLRWDT** ó **SLEEP**.

/PD Indica el paso al modo de bajo consumo (*power down*).

0: Ejecutada una instrucción **SLEEP**.
1: Después de un arranque o la ejecución de **CLRWDT**.

Z Cero.

0: El resultado de la última instrucción aritmética o lógica no fue cero.
1: El resultado de la última instrucción aritmética o lógica fue cero.

DC Acarreo de dígito o semiacarreo.

0: No hubo acarreo.
1: Acarreo hacia el bit 4.

C Acarreo o la que me llevo después de la última operación aritmética o lógica.

0: No hubo acarreo.
1: Acarreo hacia el bit 8.

- **OPTION_REG**. Presente en los bancos 1 y 3. Registro de opciones.

OPTION_REG

Bit 7	6	5	4	3	2	1	Bit 0
/RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

/RBPU Bit de habilitación de las resistencias internas de *pull-up* del PORTB.

0: Activado.
1: Desactivado.

INTEDG Selección del flanco del pin **RB0/INT**.

0: Flanco de bajada.
1: Flanco de subida.

T0CS Selección de la fuente de reloj para el temporizador 0.

0: Ciclo de instrucción interno. Equivale a 4 ciclos de reloj del microcontrolador.
1: Transición en la patilla **RA4/T0CKI**.

T0SE Selección de flanco para la fuente externa del temporizador 0.

0: Se incrementará en el flanco de subida de **RA4/T0CKI**.
1: Se incrementará en el flanco de bajada de **RA4/T0CKI**.

PSA Asignación del divisor de frecuencia.

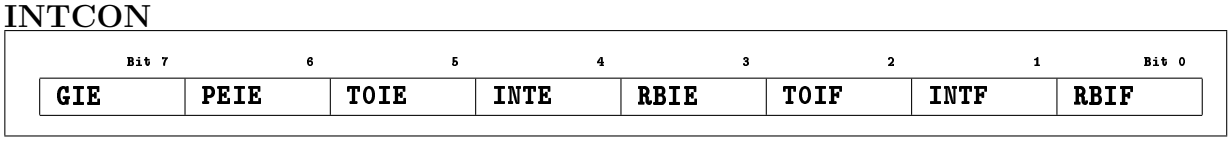
0: Pre-escala asignada al temporizador TMR0.
1: Divisor asignado al perro guardián (WDT) como post-escala.

PS2:PS0 Divisor de frecuencia.

Bits	PSA=0 (TMR0)	PSA=1 (WDT)
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

- **INTCON**. Presente en todos los bancos. Registro de control y notificación de interrupciones.

INTCON



GIE Habilitación global de las interrupciones.

0: Deshabilita.
1: Habilita.

PEIE Habilitación de las interrupciones de los periféricos, a través de los registros **PIR1** y **PIR2**.

0: Deshabilita.
1: Habilita.

T0IE Habilitación de la interrupción debida al temporizador TMR0.

0: Deshabilita.
1: Habilita.

INTE Habilitación de la interrupción debida a la patilla **RB0/INT**.

0: Deshabilita.
1: Habilita.

RBIE Habilitación de la interrupción debida al cambio en la entrada en las patillas **RB4**, **RB5**, **RB6**, **RB7**.

0: Deshabilita.
1: Habilita.

T0IF Notifica el desbordamiento del temporizador TMR0.

0: No pasó nada.
1: Se desbordó TMR0. Hay que borrarlo por software.

INTF Notifica que ocurrió el flanco seleccionado en la patilla **RB0/INT**.

0: No pasó nada.

1: Detectado flanco programado en **RB0/INT**. Hay que borrarlo por software.

RBIF Detectado un cambio en la entrada en las patillas **RB4**, **RB5**, **RB6**, **RB7**.

0: No pasó nada.

1: Cambio detectado. Hay que borrarlo por software. Para que sea efectivo antes hay que leer el puerto B.

- **PIE1**. Presente en el banco 1. Registro de interrupciones de los periféricos.

PIE1

Bit 7	6	5	4	3	2	1	Bit 0
-	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE

ADIE Habilitación de las interrupción del conversor AD.

0: Deshabilita.

1: Habilita.

RCIE Habilitación de la interrupción debida a la recepción del módulo USART.

0: Deshabilita.

1: Habilita.

TXIE Habilitación de la interrupción debida a la transmisión del módulo USART.

0: Deshabilita.

1: Habilita.

SSPIE Habilitación de la interrupción debida al módulo MSSP.

0: Deshabilita.

1: Habilita.

CCP1IE Habilitación de la interrupción debida al módulo CCP1.

0: Deshabilita.

1: Habilita.

TMR2IE Habilitación de la interrupción debida al alcance del periodo (**PR2**) por el temporizador TMR2.

0: Deshabilita.

1: Habilita.

TMR1IE Habilitación de la interrupción debida al desbordamiento del temporizador TMR1.

0: Deshabilita.

1: Habilita.

- **PIR1**. Presente en el banco 0. Registro de notificación de las interrupciones de los periféricos. Son bits pegajosos: deben ponerse de nuevo a cero por software.

PIR1

Bit 7	6	5	4	3	2	1	Bit 0
-	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF

ADIF Notificación de la finalización de una conversión del módulo AD.

0: No pasó nada.
1: Conversión completada.

RCIF Notificación de la recepción de un dato por el módulo USART.

0: No pasó nada.
1: El búfer de recepción está lleno en el módulo USART.

TXIF Notificación de una transmisión realizada en el módulo USART.

0: No pasó nada.
1: El búfer de transmisión está vacío.

SSPIF Notificación de evento ocurrido en el módulo MSSP.

0: No pasó nada.
1: Ocurrió algo:

SPI	Transmisión o recepción finalizada.
I2C Esclavo	Transmisión o recepción finalizada.
I2C Maestro	Transmisión o recepción finalizada.
I2C Maestro	Condición *start* realizada.
I2C Maestro	Condición *stop* realizada.
I2C Maestro	Condición *restart* realizada.
I2C Maestro	Secuencia ACK realizada.
I2C Maestro	Ocurrió una condición *start* cuando el módulo estaba ocioso (arbitraje de bus multimaestro).
I2C Maestro	Ocurrió una condición *stop* cuando el módulo estaba ocioso (arbitraje de bus multimaestro).

CCP1IF Notificación de suceso del módulo CCP1.

0: No ocurrió nada.
1: Ocurrió una captura o una comparación en el módulo CCP1.

TMR2IF Notificación de que **TMR2** alcanzó **PR2**.

0: No pasó nada.
1: Fin de cuenta.

TMR1IF Notificación del desbordamiento de TMR1.

0: No pasó nada.
1: Desbordamiento de TMR1.

- **PIE2**. Presente en el banco 1. Registro de interrupciones de los periféricos.

PIE2

Bit 7	6	5	4	3	2	1	Bit 0
OSFIE	CMIE	-	EEIE	-	-	-	-

OSFIE Habilita la interrupción debida a la detección de fallo del oscilador principal.

0: Interrupción no permitida.
1: Interrupción permitida.

CMIE Habilita la interrupción debida al cambio en la salida del módulo comparador.

0: Interrupción no permitida.

1: Interrupción permitida.

EEIE Habilita la interrupción debida a la finalización de la escritura en la memoria EEPROM (de datos o de programa).

0: Interrupción no permitida.

1: Interrupción permitida.

- `PIR2`. Presente en el banco 0. Registro de notificación de las interrupciones de los periféricos.

PIR2

Bit 7	6	5	4	3	2	1	Bit 0
OSFIF	CMIF	-	EEIF		-	-	

OSFIF Notifica el fallo del oscilador principal.

0: No hay problemas.

1: Fallo.

CMIF Notifica el cambio de la salida del módulo comparador.

0: No cambió.

1: Cambió.

EEIF Notificación de que la escritura en la memoria EEPROM (de datos o de programa) ha finalizado.

0: Operación incompleta. En curso.

1: Operación finalizada.

- `PCON`. Presente en el banco 1. Registro de estado del arranque del microcontrolador.

PCON

Bit 7	6	5	4	3	2	1	Bit 0
-	-	-	-	-	-	/POR	/BOR

/POR Estado del reset de arranque.

0: Ocurrió un arranque del sistema (POR: *Power-on reset*). Se debe poner a uno por *software* para detectarlo.

1: No ocurrió nada.

/BOR Señaliza un rearranque del micro por caída de la tensión de alimentación (BOR: *Brown-Out Reset*).

0: Ocurrió el reset.

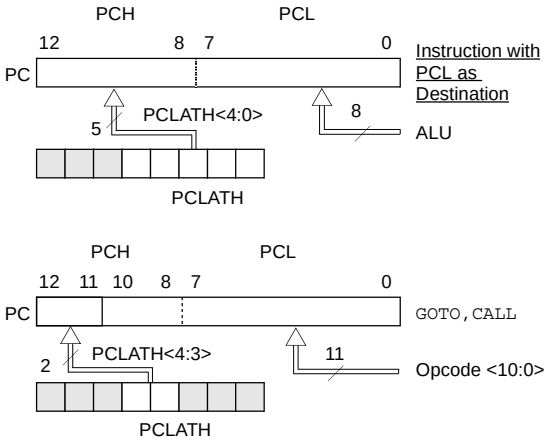
1: No ocurrió.

Contador de programa

El contador de programa `PC` es de 13 bits y no es accesible directamente. Está compuesto de:

- Registro `PCL`: parte baja de `PC`.
- Registro `PCH`: parte alta de `PC`. No es accesible directamente, si no que se puede modificar a través del registro `PCLATH` que se copia en `PCH` al escribir en `PCL` ó cuando se ejecuta una instrucción `GOTO` o `CALL`.

Esquema



Ejemplo

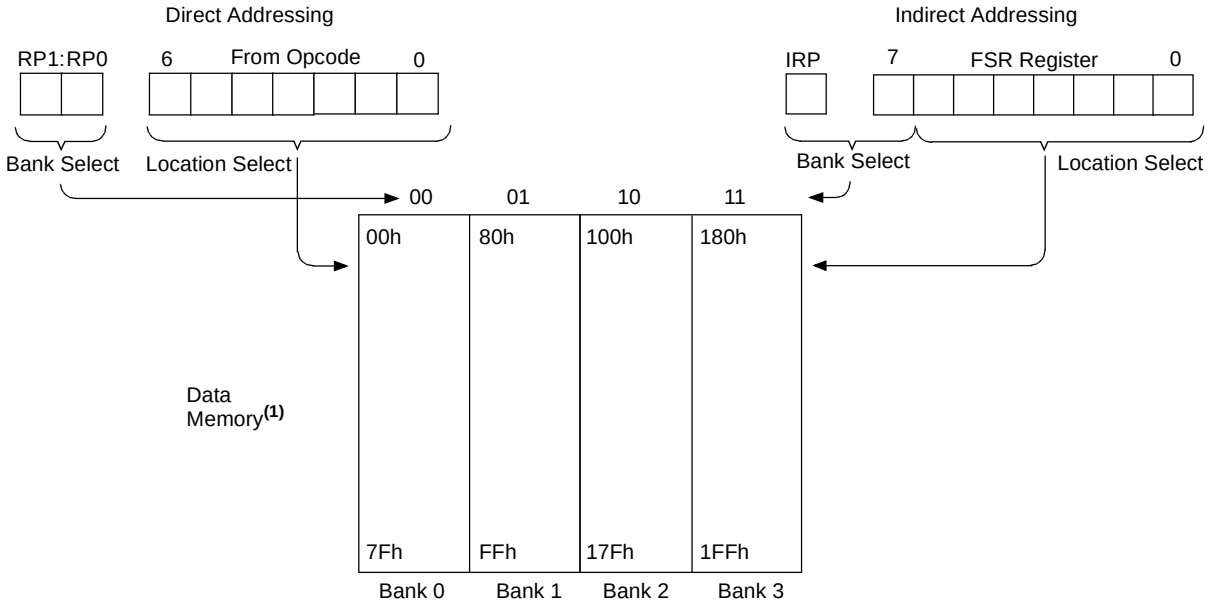
```
1     ORG       0x500
2     BSF       PCLATH,4
3     BCF       PCLATH,3      ; Selecciona la pagina 2 de FLASH (800h-BFFh)
4     CALL      SUB1_P1       ; Llama a la subrutina
5
6     :                       ; pagina 2 (800h-BFFh)
7     :
8     ORG 0x900               ; pagina 2 (800h-BFFFh)
9  SUB1_P1
10    :                       ; subrutina llamada
11    :                       ; pagina 2 (800h-BFFh)
12    :
13    RETURN                  ; retorno de la subrutina
14                            ; hacia la pagina 0 (000h-3FFh)
```

Direccionamiento directo/ indirecto

El registro **FSR** contiene la dirección a la que se accede a través del registro **INDF**. Esto es **INDF**=[**FSR**].

Esquema

El funcionamiento es diferente cuando el direccionamiento es directo de cuando el direccionamiento es indirecto:



Ejemplo de direccionamiento indirecto

El ejemplo muestra como poner a cero un bloque de datos en RAM desde la posición 300 hasta la 400:

```
1          bsf     STATUS,IRP
2          movlw   300             ;
3          movwf   FSR             ; inicializa el puntero FSR = 300
4  NEXT    clrf    INDF            ; [FSR] = INDF = 0
5          incf    FSR,F           ; incrementa FSR
6          movlw   400
7          subwf   FSR,W           ; W = FSR-400
8          btfss   STATUS,Z        ; fin?
9          goto    NEXT            ; si no continua borrando
10 FIN     bcf     STATUS,IRP      ;
```

3.2.4. Acceso a las Memorias EEPROM y FLASH

Características

Las memorias EEPROM de datos y FLASH de programa son accesibles a través de 6 registros: **EECON1**, **EECON2**, **EEDATA**, **EEDATH**, **EEADR** y **EEADRH**.

La memoria de datos EEPROM

- Se accede byte a byte (máximo 256 posiciones) con los registros **EEADR** y **EEDATA** activando el proceso con los registros de control **EECON1** y **EECON2**.
- El tiempo de escritura se controla con un oscilador RC interno. El tiempo de escritura dependerá del voltaje y de la temperatura y puede variar en cada chip.

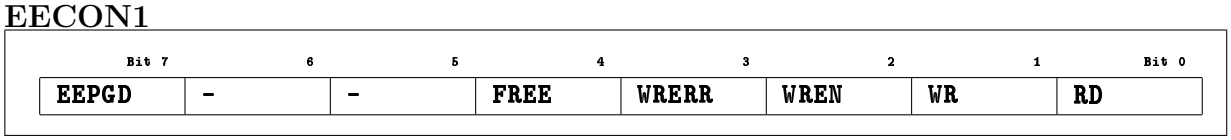
La memoria de programa FLASH

- Es un proceso borra-escribe y mientras dura la escritura no se ejecutan instrucciones aunque los periféricos continúan funcionando. Las interrupciones se atienden al finalizar la escritura.
- Los registros **EEDATH**:**EEDATA** almacenan los 14 bits de cada posición de memoria.
- **EEADRH**:**EEADR** indican la dirección de 13 bits accedida para lectura o escritura.
- Los bits no usados permanecen a cero.

Registros

- **EEADRH** y **EEADR**. Banco 2. Registros de dirección, partes alta y baja respectivamente.
- **EEDATH** y **EEDATA**. Banco 2. Registros de datos, partes alta y baja respectivamente.
- **EECON1**, situado en el banco 3:

EECON1



Bit 7	6	5	4	3	2	1	Bit 0
EEPGD	-	-	FREE	WRERR	WREN	WR	RD

EEPGD Bit de selección de EEPROM de datos o Flash de programa.

0: Accede a la memoria de datos.
1: Accede a la memoria de programa.

FREE *EEPROM Forced Row Erase bit*

0: Solo realiza la escritura.
1: Borra la fila de memoria de programa direccionada por **EEADRH**:**EEADR** en la próxima escritura.

WRERR Bit de error en la escritura.

0: La escritura terminó correctamente.
1: La operación de escritura terminó prematuramente.

WREN Habilitación de escritura.

0: Prohíbe la escritura.
1: Permite la escritura.

WR Bit de control de la escritura.

0: Ciclo de escritura completo.
1: Inicia un ciclo de escritura. Vuelve a cero al finalizar la escritura.

RD Bit de control de la lectura.

0: No inicia el proceso de lectura.
1: Inicia el proceso de lectura. Vuelve a cero él sólo.

- **EECON2**, situado en el banco 3. No es un registro físico. Se usa en la secuencia de escritura.

Registros asociados

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
10Ch	EEDATA	EEPROM/FLASH Data Register Low Byte								xxxx xxxx	uuuu uuuu
10Dh	EEADR	EEPROM/FLASH Address Register Low Byte								xxxx xxxx	uuuu uuuu
10Eh	EEDATH	—	—	EEPROM/FLASH Data Register High Byte						--xx xxxx	--uu uuuu
10Fh	EEADRH	—	—	—	—	—	EEPROM/FLASH Address Register High Byte			---- -xxx	---- -uuu
18Ch	EECON1	EEPGD	—	—	FREE	WRERR	WREN	WR	RD	x--x x000	x--x q000
18Dh	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	---- ----
0Dh	PIR2	OSFIF	CMIF	—	EEIF	—	—	—	—	00-0 ----	00-0 ----
8Dh	PIE2	OSFIE	CMIE	—	EEIE	—	—	—	—	00-0 ----	00-0 ----

Escritura de la memoria EEPROM de datos

En lenguaje ensamblador sería:

```
1     BANKSEL EEADR             ; Macro que conmuta de banco
2     MOVLW   DATA_EE_ADDR      ; Direccion de memoria de datos
3     MOVWF   EEADR             ;
4     MOVLW   DATA_EE_DATA      ; Dato a escribir
5     MOVWF   EEDATA            ;
6     BANKSEL EECON1            ; Macro que conmuta de banco
7     BCF     EECON1, EEPGD     ; Apunta a la EEPROM de datos
8     BSF     EECON1, WREN      ; Habilita la escritura
9     BCF     INTCON, GIE       ; Desactiva las interrupciones
10    MOVLW   55h               ;
11    MOVWF   EECON2            ; Secuencia requerida
12    MOVLW   AAh               ;
13    MOVWF   EECON2            ; Secuencia requerida
14    BSF     EECON1, WR        ; Ordena la escritura
15    BSF     INTCON, GIE       ; Activa las interrupciones
16    SLEEP                     ; Se duerme y espera a que termine
17    BCF     EECON1, WREN      ; Desactiva la escritura
```

En lenguaje C sería:

```
1  #include <pic.h>
2
3  // ...
4   eeprom_wite(addr, value);  // Espera a que termine
5
6  // O como macro, que no espera a que termine
7   EEPROM_WRITE(addr, value);
8   while(WR) continue;        // Espera a que termine
```

Lectura de la memoria EEPROM de datos

En lenguaje ensamblador sería:

```
1     BANKSEL EEADR             ; Macro que conmuta de banco
2     MOVLW   DATA_EE_ADDR      ;
```

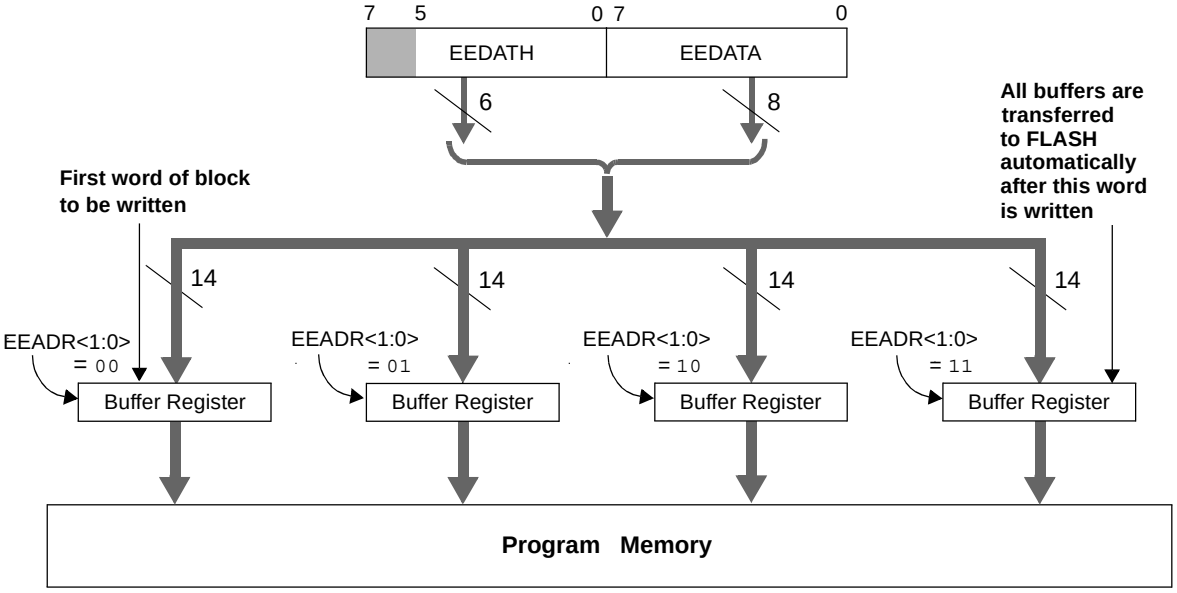
```
3       MOVWF   EEADR           ; Direccion a leer
4       BANKSEL EECON1          ; Macro que conmuta de banco
5       BCF     EECON1, EEPGD   ; Apunta a la EEPROM
6       BSF     EECON1, RD      ; Ordena la lectura
7       BANKSEL EEDATA          ; Macro que conmuta de banco
8       MOVF    EEDATA, W       ; W = EEDATA
```

En lenguaje C sería:

```
1  #include <pic.h>
2
3  // ...
4   value = eeprom_read(addr);
5
6  // O como macro, que no comprueba si se ha finalizado una escritura
7   value = EEPROM_READ(addr);
```

3.2.5. Escritura de la memoria Flash de programa

Es necesario escribir 4 datos en direcciones consecutivas empezando por una múltiplo de cuatro. La escritura real se hará tras la cuarta escritura.



En lenguaje ensamblador sería para una dirección:

```
1       BANKSEL EEADDRH         ; Macro que conmuta de banco
2       MOVLW   ADDRH           ;
3       MOVWF   EEADRH          ; Parte alta de la direccion
4       MOVLW   ADDRL           ;
5       MOVWF   EEADR           ; Parte baja de la direccion
6       MOVLW   DATAH           ;
7       MOVWF   EEDATH          ; Parte alta del dato a escribir
8       MOVLW   DATAL           ;
9       MOVWF   EEDATA          ; Parte baja del dato a escribir
10      BANKSEL EECON1          ; Macro que conmuta de banco
11      BSF     EECON1, EEPGD   ; Apunta a la memoria de programa
12      BSF     EECON1, WREN    ; Habilita la escritura
13      BCF     INTCON, GIE     ; Desactiva las interrupciones
```

```
14      MOVLW   55h             ;
15      MOVWF   EECON2          ; Escribe 55h en EECON2
16      MOVLW   AAh             ;
17      MOVWF   EECON2          ; Escribe AAh en EECON2
18      BSF     EECON1, WR      ; Ordena la escritura
19      NOP                     ; Estas dos instrucciones son ignoradas
20      NOP                     ;  por el microcontrolador
21                              ; Detiene la ejecucion
22                              ;  y espera a que se termine la escritura
23                              ; Despues continua con la tercera
24                              ;  instruccion
25      BSF     INTCON, GIE     ; Habilita las interrupciones
26      BCF     EECON1, WREN    ; Desactiva la escritura
```

En lenguaje C sería:

```
1  #include <pic.h>
2
3  // ...
4   // addr multiplo de 4, como macros
5   FLASH_WRITE(addr  ,data0);
6   FLASH_WRITE(addr+1,data1);
7   FLASH_WRITE(addr+2,data2);
8   FLASH_WRITE(addr+3,data3);
9
10 // Tambien existen:
11 // flash_write(addr,value);
12 // flash_erase(...);
13 // flash_copy(...);
```

Solo puede ser escrita si es un bloque de memoria no protegido y el bit de configuración WRT = 1.

3.2.6. Lectura de la memoria Flash de programa

En lenguaje ensamblador sería:

```
1       BANKSEL EEADRH          ; Macro que conmuta de banco
2       MOVLW   ADDRH           ;
3       MOVWF   EEADRH          ; Parte alta  de la direccion a leer
4       MOVLW   ADDRL           ;
5       MOVWF   EEADR           ; Parte baja de la direccion
6       BANKSEL EECON1          ; Macro que conmuta de banco
7       BSF     EECON1, EEPGD   ; Apunta a la memoria de programa
8       BSF     EECON1, RD      ; Ordena la lectura
9       NOP                     ;  y tarda dos ciclos
10      NOP                     ;
11      BANKSEL EEDATA          ; Macro que conmuta de banco
12      MOVF    EEDATA, W       ; W = parte baja
13      MOVF    EEDATH, W       ; W = parte alta del dato leido
```

En lenguaje C sería:

```
1  #include <pic.h>
2
3  // ...
4   value = FLASH_READ(addr);  // Como macro
```

Protección de la memoria EEPROM de datos

Se protege con `EECON1.WREN = 0` (durante *PowerUp* y *Brown-out*)

Protección de la memoria Flash de programa

El bit de configuración WRT = 0 evita escrituras no deseadas debido a un malfuncionamiento del micro.

Mediante los bits de configuración WRT1:0 y CP se puede seleccionar el modo de protección como se puede apreciar en la siguiente tabla:

WRT1:0	Protegido	Modificable con `EECON`
11	0000h ⇒ 00FFh	0100h ⇒ 0FFFh
10	0000h ⇒ 07FFh	0800h ⇒ 0FFFh
01	0000h ⇒ 0FFFh	Ninguna
00	Ninguna	Ninguna

Para definir el contenido de la memoria EEPROM de datos desde el propio programa en lenguaje C se emplea la siguiente macro:

```
1 #include <pic.h>
2
3 // ...
4
5 // La macro define el contenido de la EEPROM de 8 en 8 bytes
6 __EEPROM_DATA(0,1,2,3,4,5,6,7); // La primera a partir de la dir. 0
7 __EEPROM_DATA(0,1,2,3,4,5,6,7); // Estos iran a partir de la dir. 8
8                                 //  .. etc.
```

3.3. Puertos de E/S

3.3.1. Puerto A

El puerto A está compuesto por las patillas **RA0/AN0**, **RA1/AN1**, **RA2/AN2/CVref/Vref-**, **RA3/AN3/Vref+/C1OUT**, **RA4/AN4/T0CKI/C2OUT**, **RA5//MCLR/VPP**, **RA6/OSC2/CLKO** y **RA7/OSC1/CLKI**.

Características

- Puede funcionar como **entrada/salida digital**. Para ello:
  - **TRISA** indicará el sentido de cada patilla, siendo salida cuando el bit correspondiente vale 0 y entrada cuando vale 1. Excepto con **RA5//MCLR/VPP**que es solo entrada.
  - **ANSEL** = 00000000b para configurar todos los pines como de entrada/salida digital.
  - La patilla **RA4/AN4/T0CKI/C2OUT**tiene un comportamiento especial ya que:
    - Cuando funciona como entrada, es de tipo *Schmitt Trigger* y puede asociarse al temporizador TMR0 como entrada externa.
- Puede funcionar como **entrada analógica**: **RA0/AN0**, **RA1/AN1**, **RA2/AN2/CVref/Vref-**, **RA3/AN3/Vref+/C1OUT**, **RB6/AN5/PGC/T1OSO/T1CKI**. Se verá en el tema dedicado al conversor AD.

Registros asociados

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
05h	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxx 0000(1) xxx0 0000(2)	uuuu 0000(1) uuu0 0000(2)
85h	TRISA	TRISA7	TRISA6	TRISA5(3)	PORTA Data Direction Register					1111 1111	1111 1111
9Fh	ADCON1	ADFM	ADCS2	VCFG1	VCFG0	—	—	—	—	0000 ----	0000 ----
9Bh	ANSEL(4)	—	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	-111 1111	-111 1111

Pines asociados

RA0/AN0 Entrada/salida digital ó entrada analógica. Búfer tipo TTL.

RA1/AN1 Entrada/salida digital ó entrada analógica. Búfer tipo TTL.

RA2/AN2/CVref/Vref- Entrada/salida digital ó entrada analógica ó tensión de referencia inferior para el conversor AD ó salida VREF del comparador. Búfer tipo TTL.

RA3/AN3/Vref+/C1OUT Entrada/salida digital ó entrada analógica ó tensión de referencia superior para el conversor AD ó salida del comparador. Búfer tipo TTL.

RA4/AN4/T0CKI/C2OUT Entrada/salida digital ó entrada analógica ó entrada de reloj externa para el temporizador TMR0 ó salida del comparador 2. Búfer tipo ST.

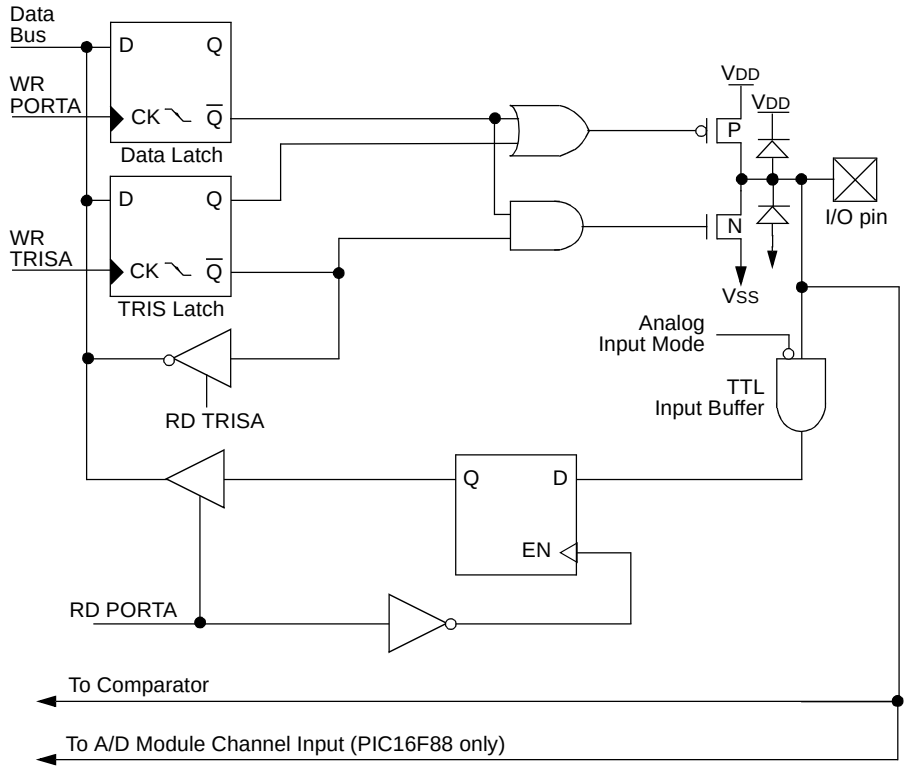
RA5//MCLR/VPP Entrada digital ó reset ó entrada de tensión de programación. Búfer tipo ST.

RA6/OSC2/CLKO Entrada/salida digital ó conexión del cristal ó salida de la frecuencia de instrucción. Búfer tipo ST.

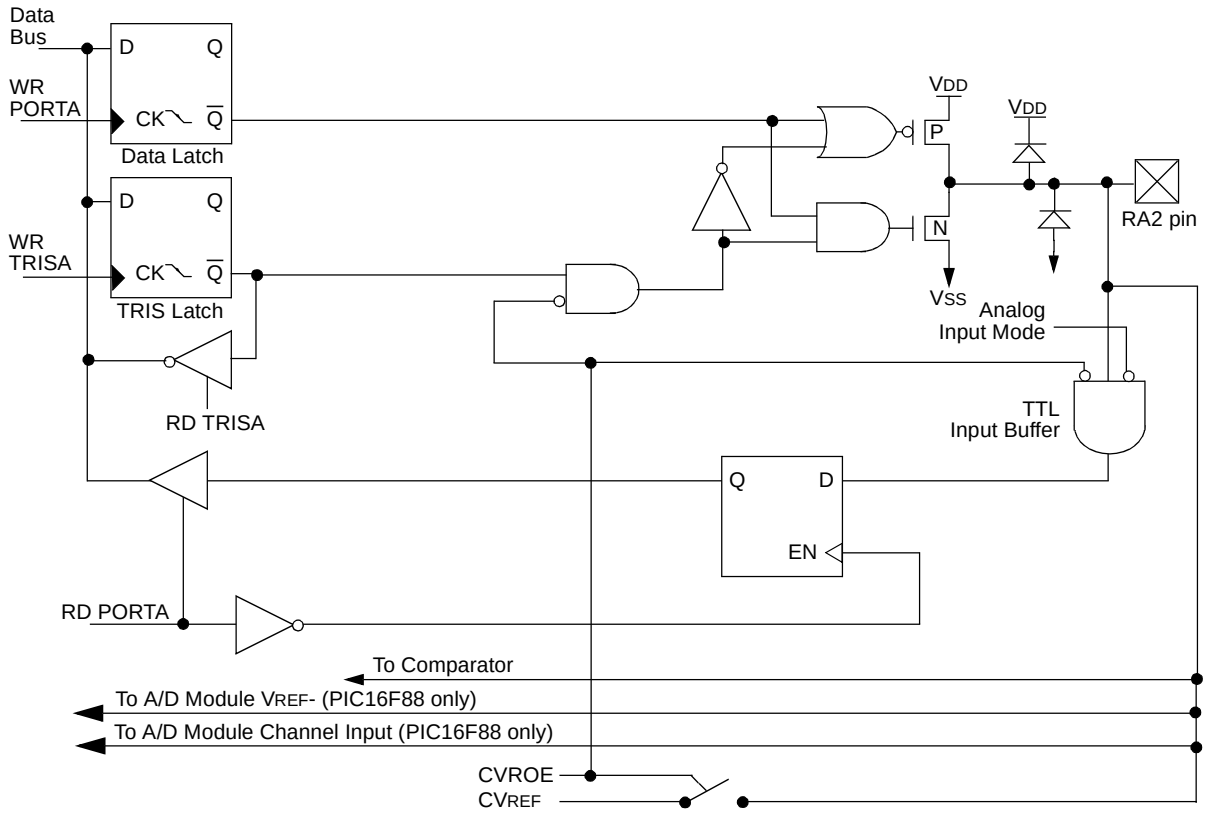
RA7/OSC1/CLKI Entrada/salida digital ó conexión del cristal ó entrada de la frecuencia de reloj. Búfer tipo ST.

Esquemas hardware

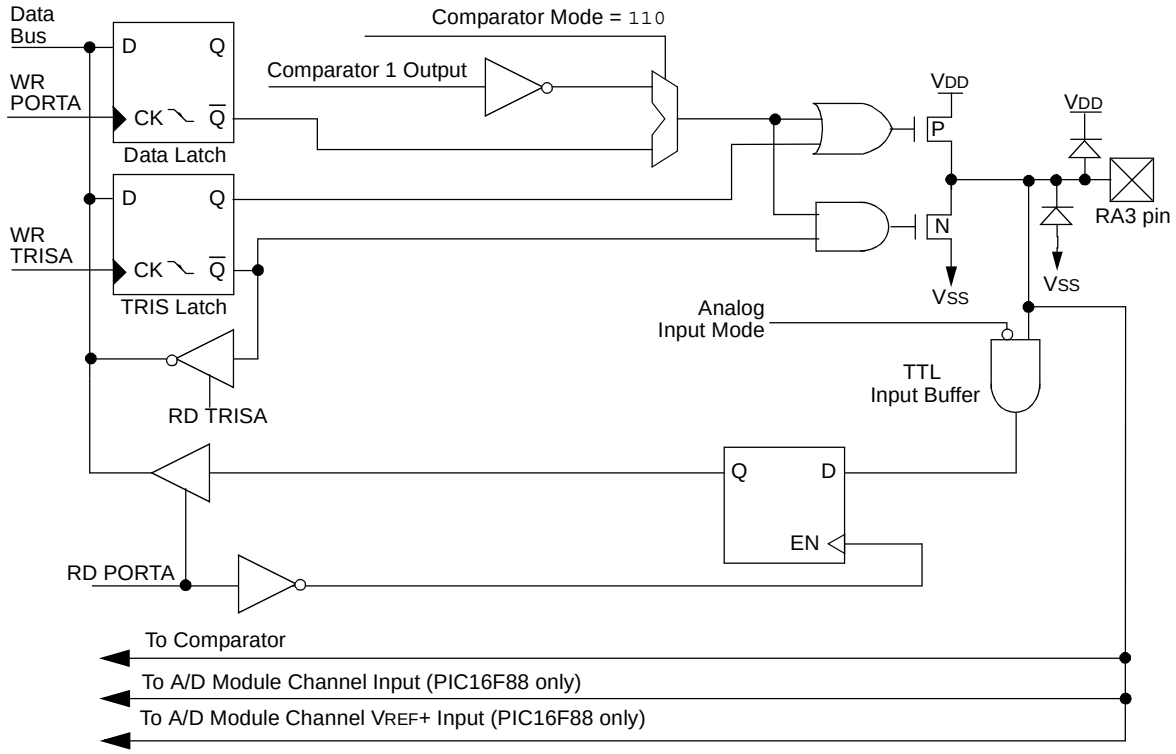
Patillas RA0,RA1



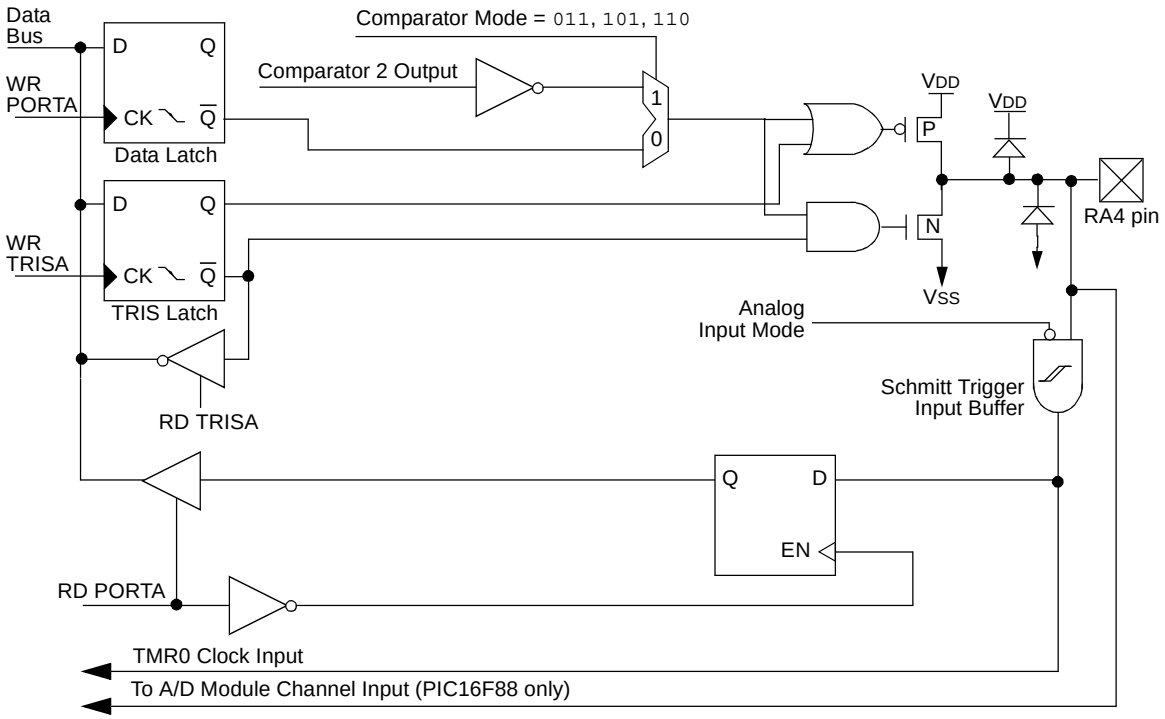
Patilla RA2



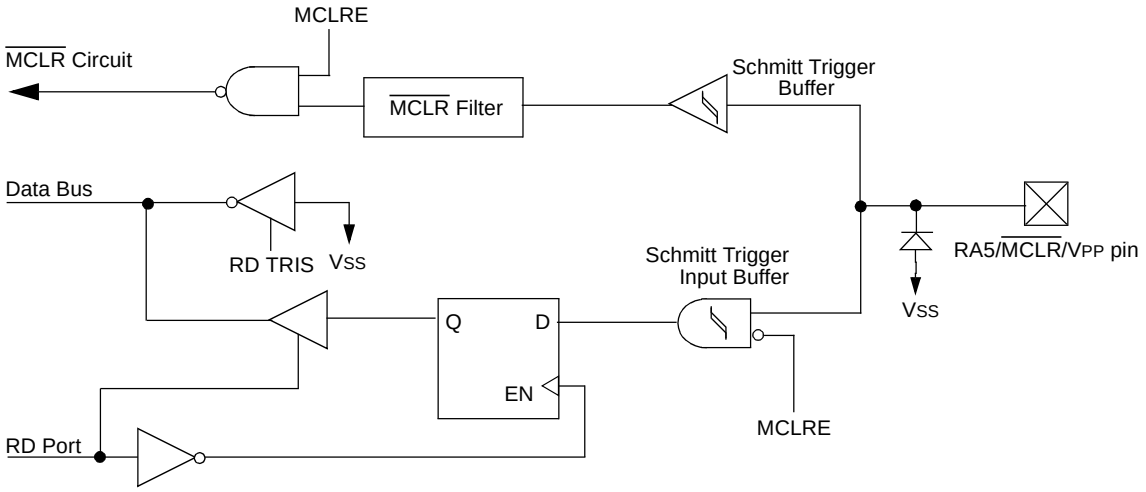
Patilla RA3



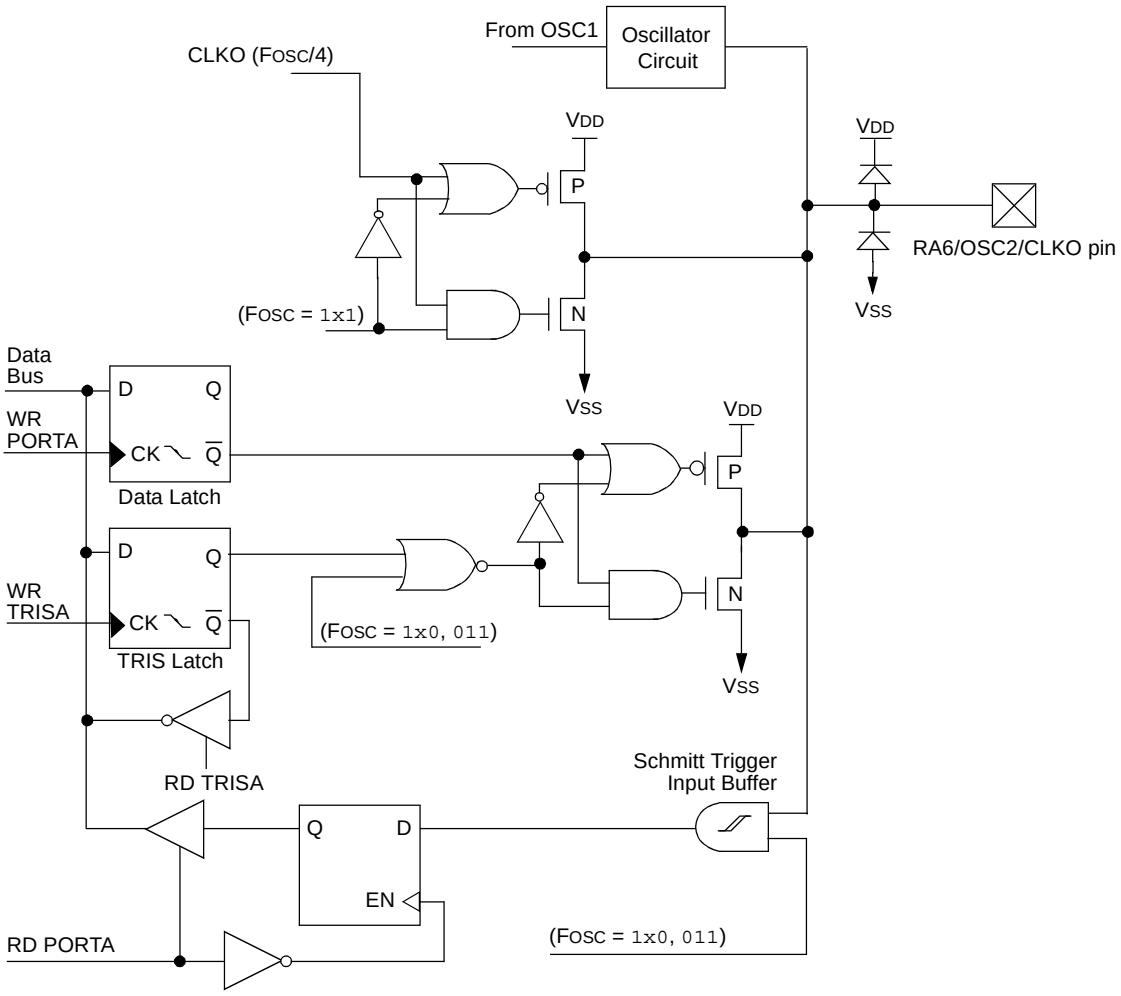
Patilla RA4



Patilla RA5

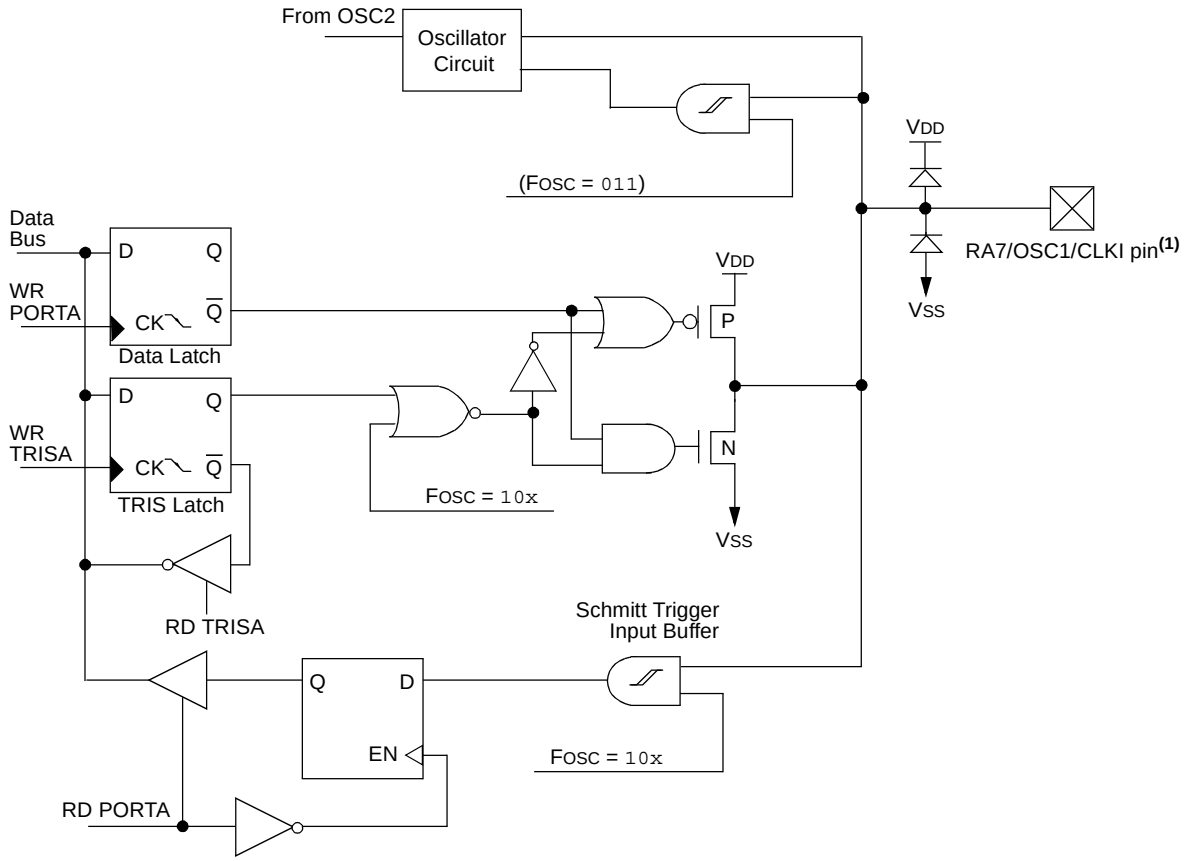


Patilla RA6



Note 1: I/O pins have protection diodes to VDD and VSS.
2: CLKO signal is 1/4 of the FOSC frequency.

Patilla RA7



Note 1: I/O pins have protection diodes to VDD and VSS.

Ejemplo en ensamblador

```
1     BCF     STATUS, RP0 ;
2     BCF     STATUS, RP1 ; Banco 0 de registros
3     CLRF    PORTA       ; Inicializa el PORTA borrandolo
4     BSF     STATUS, RP0 ; Banco 1
5     MOVLW   0x00        ; Configura los pines
6     MOVWF   ANSEL       ;  como entradas digitales
7     MOVLW   0xCF        ;
8     MOVWF   TRISA       ; Pone RA<3:0> como entradas
9                         ; RA<5:4> como salidas
10                        ; RA<7:6> como entradas
```

Ejemplo en lenguaje C

```
1     PORTA   = 0;          // Borra PORTA
2     ANSEL   = 0x00;       // PORTA digital
3     TRISA   = 0xCF;       // RA<3:0> entradas
4                           // RA<5:4> salidas
5                           // RA<7:6> entradas
```

3.3.2. Puerto B

El puerto B está compuesto por las patillas **RB0**, **RB1**, **RB2**, **RB3**, **RB4**, **RB5**, **RB6** y **RB7**.

Características

- Puede funcionar como **entrada/salida digital**. Para ello:
  - **TRISB** indicará el sentido de cada patilla, siendo salida cuando el bit correspondiente vale 0 y entrada cuando vale 1.
  - Dispone de un *pull-up*[2] de entrada programable en cada pin. Se activa globalmente para todo el PORTB (para las patillas que sean entradas) mediante `OPTION_REG.NOT_RBPU = 0`.
  - Los pines **RB3/PGM**, **RB6/PGC** y **RB7/PGD** permiten la reprogramación del dispositivo: modo LVP (*Low Voltage Programming*) que se comentará más adelante.

Registros asociados

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
9Bh	ANSEL(1)	—	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	-111 1111	-111 1111

Pines asociados

RB0/INT/CCP1 Entrada/salida digital ó entrada de interrupciones externas ó entrada/salida módulo CCP. Tiene una resistencia de *pull-up* de entrada programable. Búfer tipo TTL ó ST (entrada de interrupciones).

RB1/SDI/SDA Entrada/salida digital ó entrada/salida módulo SSP. Tiene una resistencia de *pull-up* de entrada programable. Búfer tipo TTL.

RB2/SDO/RX/DT Entrada/salida digital ó salida módulo SSP ó entrada módulo AUSART. Tiene una resistencia de *pull-up* de entrada programable. Búfer tipo TTL.

RB3/PGM/CCP1 Entrada/salida digital o patilla de activación de la programación en modo LVP ó entrada/salida módulo CCP. Tiene una resistencia de *pull-up* de entrada programable. Búfer tipo TTL.

RB4/SCK/SCL Entrada/salida digital (genera interrupción al cambiar la entrada) ó entrada/salida módulo SSP ó entrada/salida módulo AUSART. Tiene una resistencia de *pull-up* de entrada programable. Búfer tipo TTL.

[2] El equivalente a 20k típicamente.

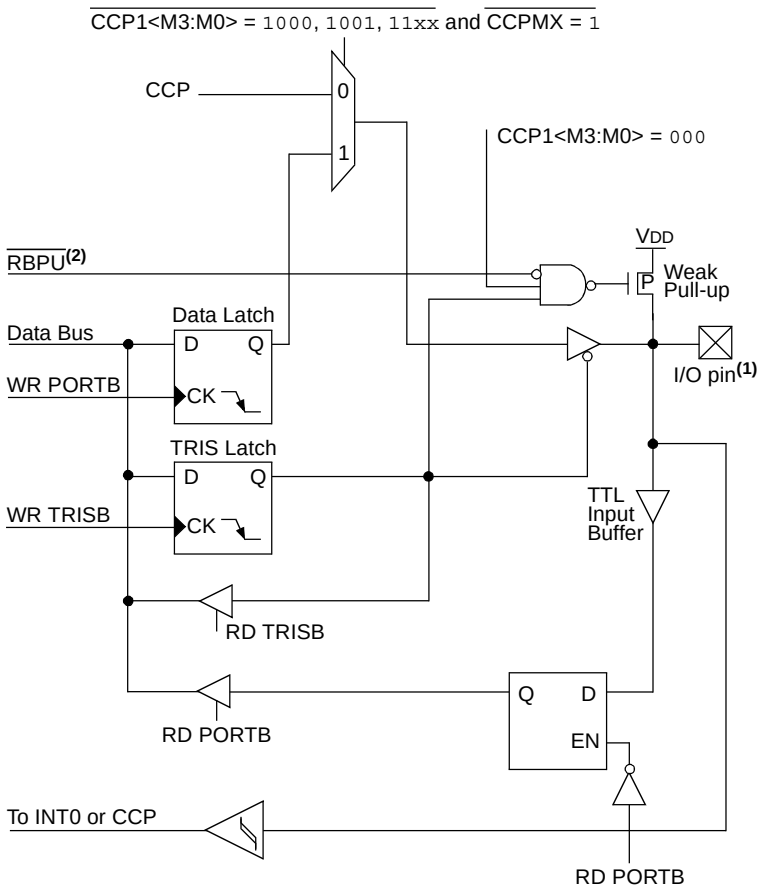
RB5//SS/TX/CK Entrada/salida digital (genera interrupción al cambiar la entrada) ó entrada módulo SSP ó entrada/salida módulo AUSART. Tiene una resistencia de *pull-up* de entrada programable. Búfer tipo TTL.

RB6/AN5/PGC/T1OSO/T1CKI Entrada/salida digital (genera interrupción al cambiar la entrada) ó señal de reloj cuando se programa el dispositivo (ya sea modo LVP o no) ó entrada analógica ó entrada temporizador TMR1 ó patilla para cristal externo con TMR1. Tiene una resistencia de *pull-up* de entrada programable. Búfer tipo TTL ó ST (en programación).

RB7/AN6/PGD/T1OSI Entrada/salida digital (genera interrupción al cambiar la entrada) ó señal de datos cuando se programa el dispositivo (ya sea modo LVP o no) ó entrada analógica ó patilla para cristal externo con TMR1. Tiene una resistencia de *pull-up* de entrada programable. Búfer tipo TTL ó ST (en programación).

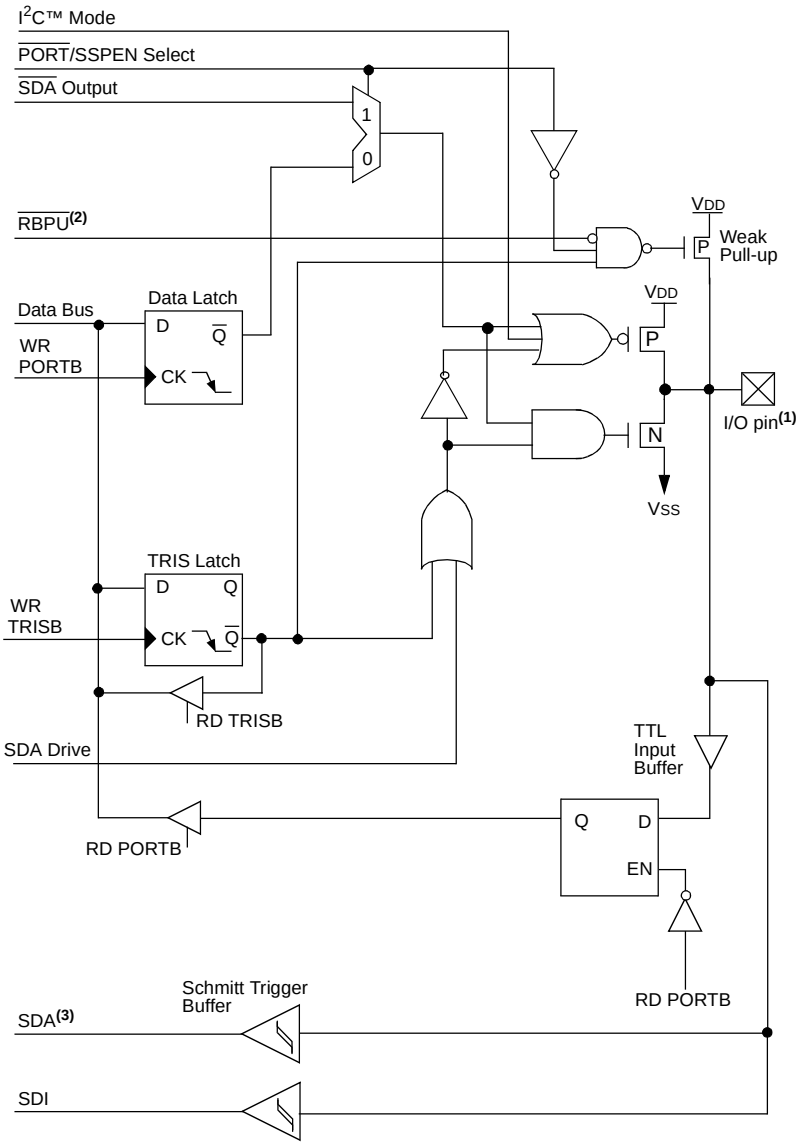
Esquemas hardware

Patilla RB0



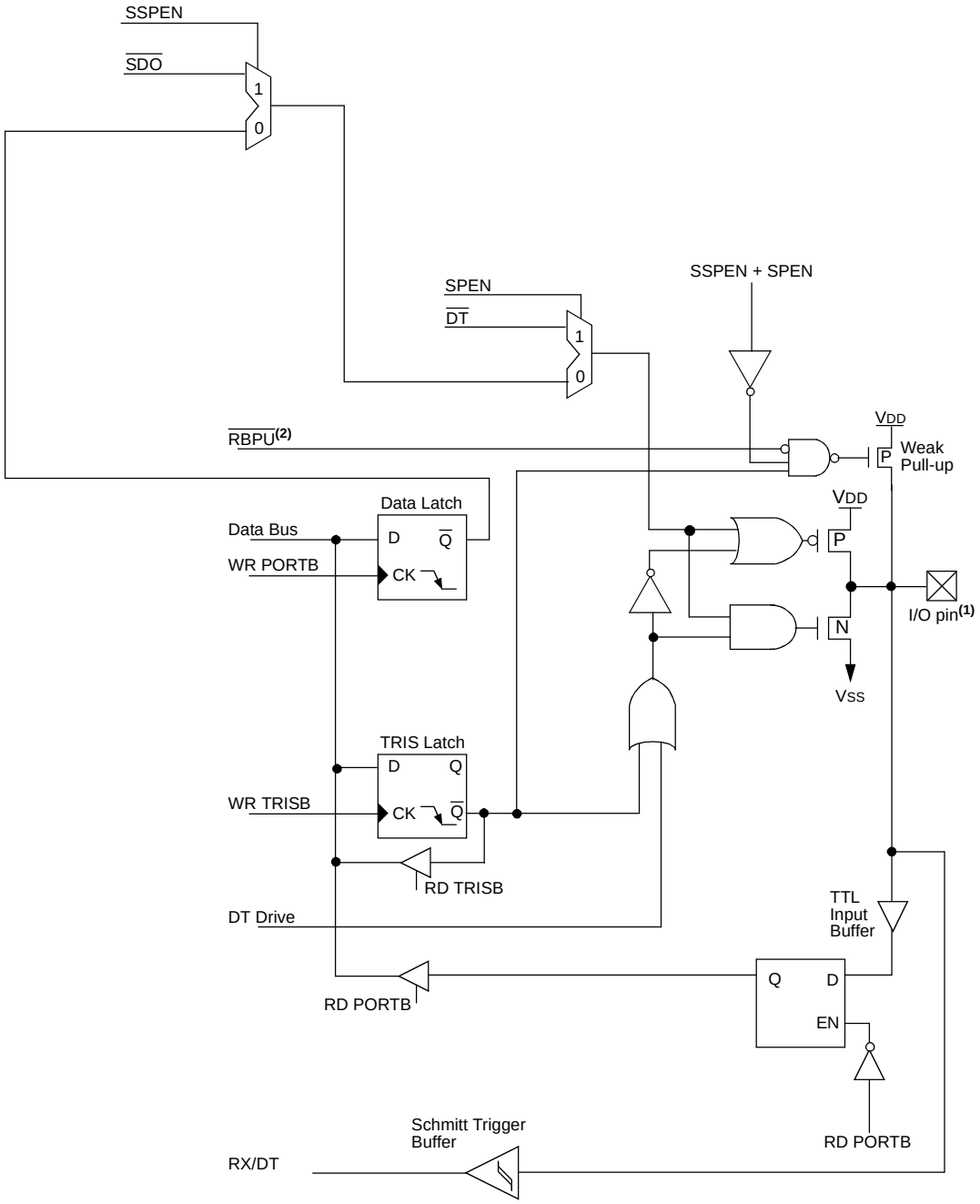
Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit.

Patilla RB1



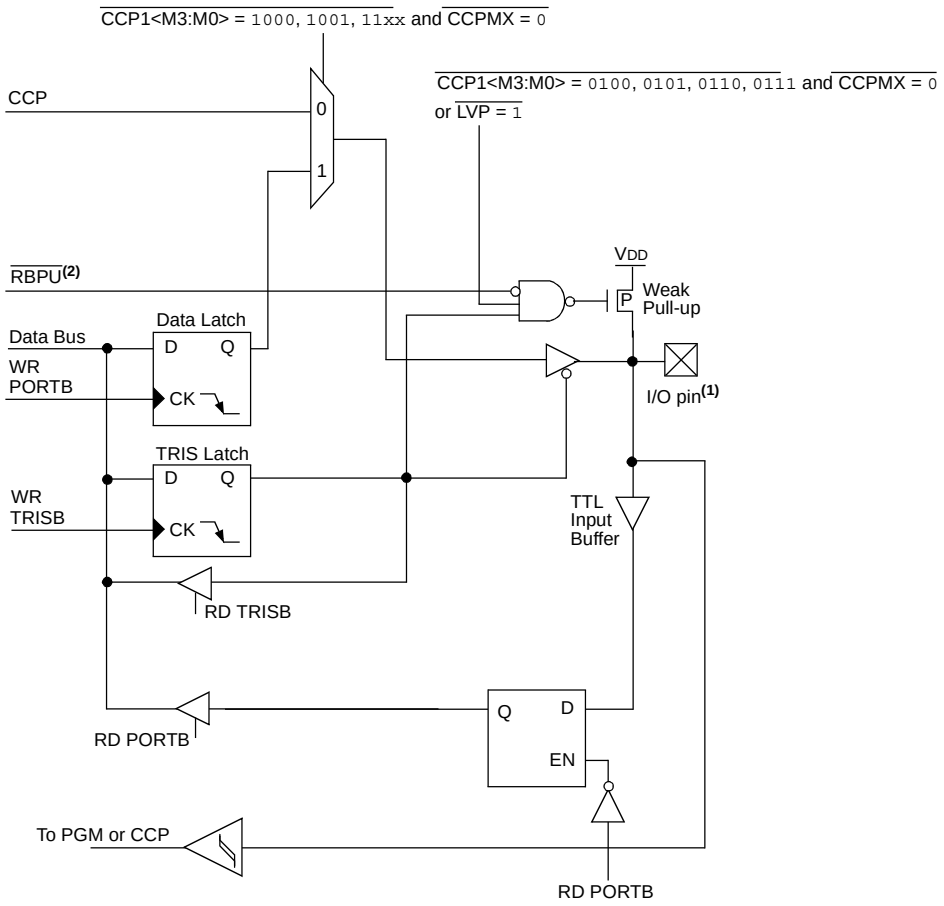
Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit.
3: The SDA Schmitt conforms to the I^2C specification.

Patilla RB2



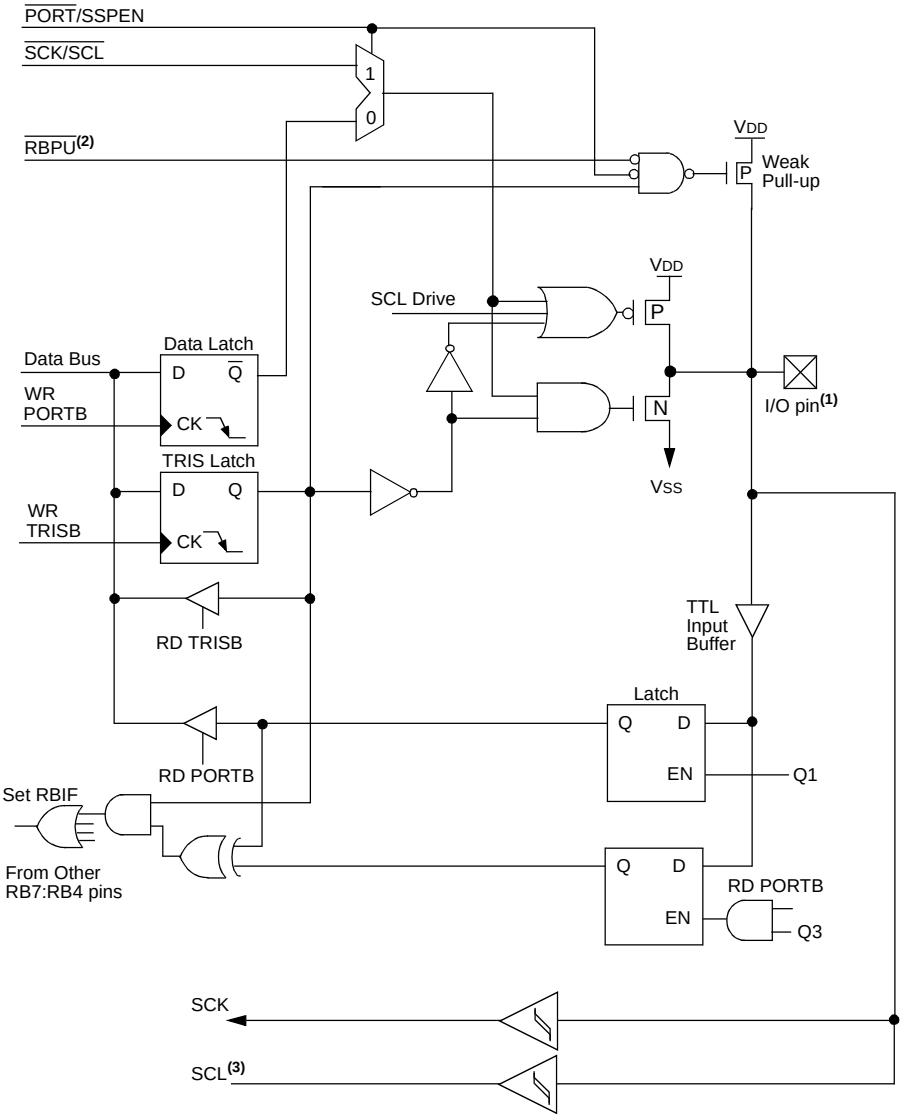
Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit.

Patilla RB3



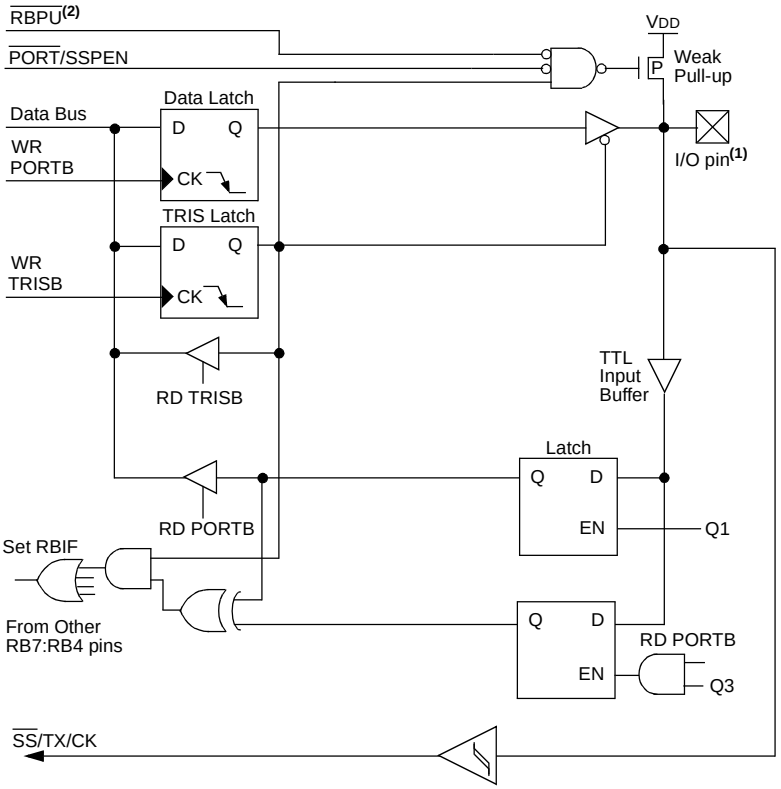
Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit.

Patilla RB4



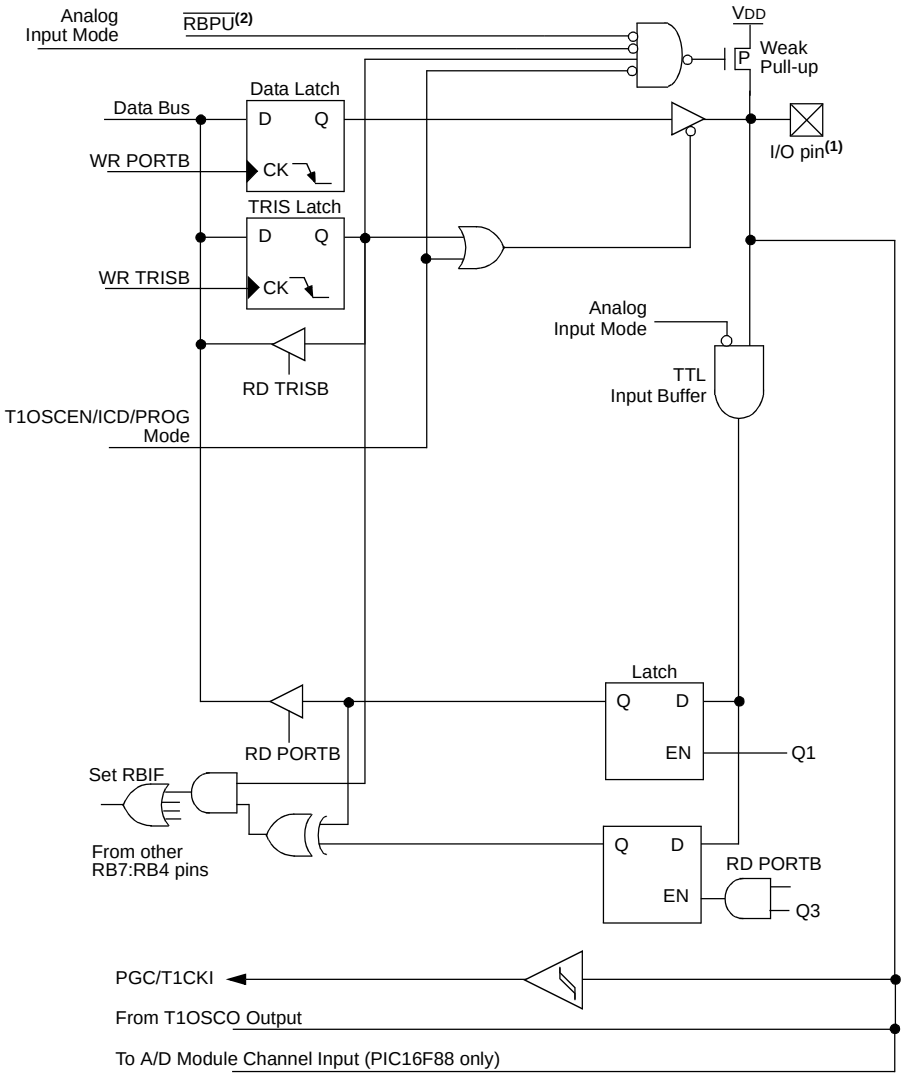
Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit.
3: The SCL Schmitt conforms to the I^2C™ specification.

Patilla RB5



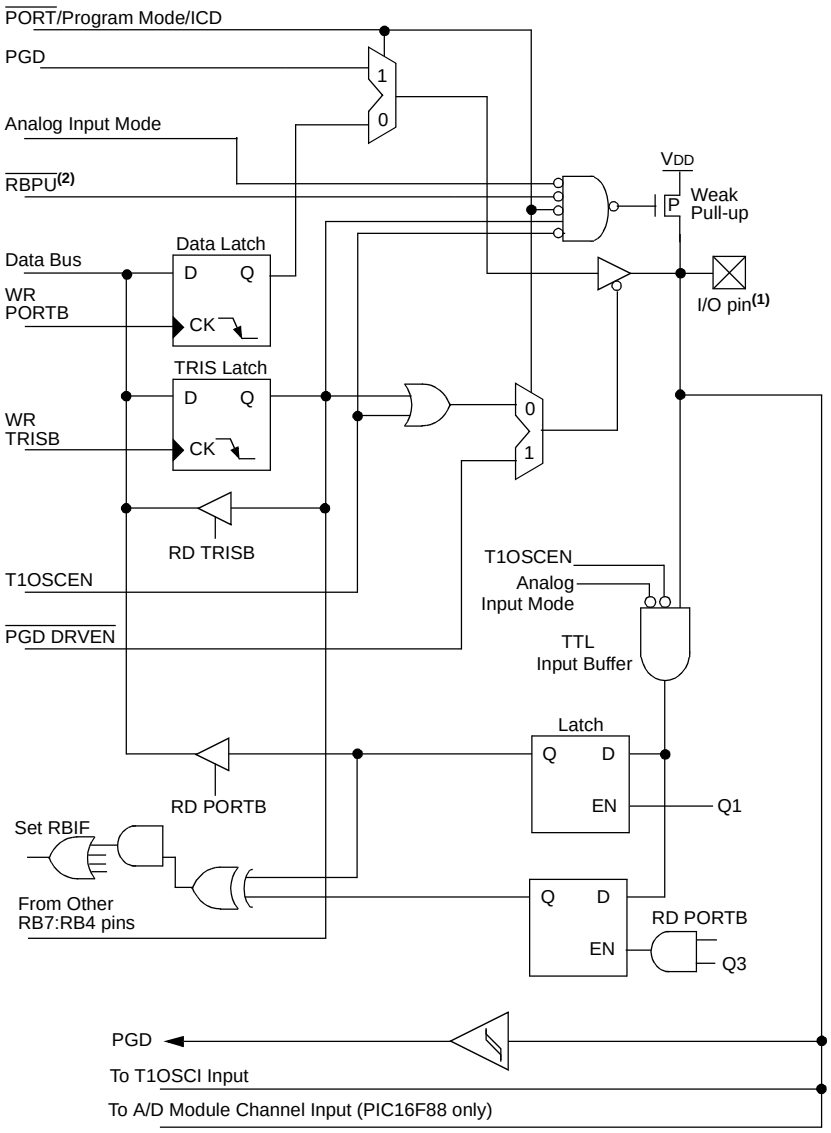
Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit.

Patilla RB6



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit.

Patilla RB7



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit.

3.4. Temporizadores

3.4.1. Temporizador TMR0

- Se trata de un temporizador/contador de 8 bits.
- La fuente de eventos puede ser interna o externa:
  - Si `OPTION_REG.TOCS = 0` actuaría como temporizador.
  - Si `OPTION_REG.TOCS = 1` actuaría como contador de eventos externos recibidos a través de la patilla **RA4/T0CKI**.

    En este caso se puede seleccionar el flanco, de subida `OPTION_REG.TOSE = 0` o de bajada `OPTION_REG.TOSE = 1`.
- Dispone de una pre-escala programable de 8 bits que se selecciona mediante `OPTION_REG.PSA = 0`. De lo contrario estaría asociada al perro guardián.

  Con `OPTION_REG.PS2:PS0` se selecciona el divisor de frecuencia variando desde 1:2 cuando valen 000b a 1:256 cuando valen 111b (ver registro `OPTION_REG` más adelante).
- Produce una interrupción al desbordarse (de 255 a 0). Para ello es necesario que estas estén activadas:
  - `INTCON.GIE = 1`, habilitación global de interrupciones.
  - `INTCON.TOIE = 1`, habilitación de la interrupción asociada a TMR0.

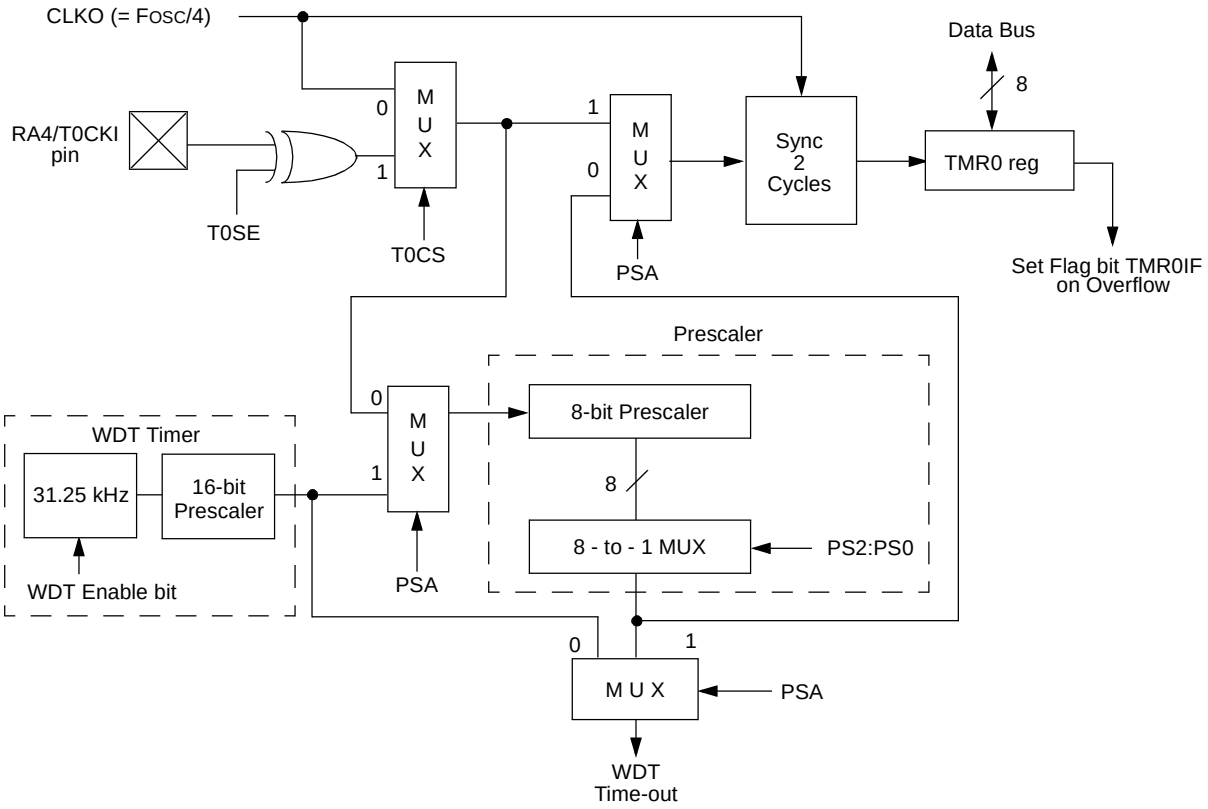
  El flag `INTCON.TOIF`, señalizará (valiendo 1) que se ha producido el desbordamiento y si están activados los flags correspondientes se producirá la interrupción. Hay que ponerlo a cero después de atender la interrupción, o si queremos detectar un nuevo desbordamiento.
- La escritura en el registro `TMR0` que lleva la cuenta, añade un retraso de 2 ciclos de instrucción y borra la cuenta del divisor de frecuencia.

  La temporización se conseguirá aplicando la ecuación:

$$T = 4 * T_{osc} * (Cuenta * Escala + 2)$$

  donde el valor $256 - Cuenta$ se escribirá en el registro `TMR0` y el valor $Escala$ se pondrá con los bits `OPTION_REG.PS2:PS0`.

Esquema hardware



Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION<5:0>).

Registros

- **TMR0**, banco 0. (*Timer 0 Register*) Registro Temporizador número 0.
- **OPTION_REG**, banco 1. Registro de opciones, algunas del temporizador 0

OPTION_REG

Bit 7	6	5	4	3	2	1	Bit 0
/RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

/RBPU Bit de habilitación de las resistencias internas de *pull-up* del PORTB.

0: Activadas.

1: Desactivadas.

INTEDG Selección del flanco del pin **RB0/INT**

0: Flanco de bajada.

1: Flanco de subida.

T0CS Selección de la fuente de reloj para el temporizador TMR0.

0: Ciclo de instrucción interno. Equivale a 4 ciclos de reloj del microcontrolador.

1: Transición en la patilla **RA4/T0CKI**.

T0SE Selección de flanco para la fuente externa del temporizador TMR0.

0: Se incrementará en el flanco de subida de **RA4/T0CKI**.

1: Se incrementará en el flanco de bajada de **RA4/T0CKI**.

PSA Asignación del divisor de frecuencia.

0: Pre-escala asignada al temporizador TMR0.

1: Divisor asignado al perro guardián (WDT) como post-escala.

PS2:PS0 División de frecuencia.

Bits	PSA=0 (TMR0)	PSA=1 (WDT)
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

Resumen de los registros asociados

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
01h,101h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Ejemplos

Si por ejemplo nos dicen que el microcontrolador funciona a 20 MHz y que queremos temporizar 0,5 milisegundos, tendremos que:

$$0,5 * 10^{-3} = 4 * \frac{1}{20 * 10^6}(Cuenta * Escala + 2)$$

Poniendo la $Escala = 64$ y despejando saldrá $Cuenta = 39$. En realidad se temporizará $T = 0{,}4996 * 10^{-3}$ segundos, que para según qué aplicaciones será suficientemente preciso o no.

```
1     bcf     STATUS,RP1
2     bsf     STATUS,RP0      ; Banco 1
3     movlw   10000101b
4     movwf   OPTION_REG      ; PSA = 0       Preescala para el TMR0
5                             ; PS2:PS0 = 101 Preescala = 64
6                             ; T0CS = 0      Reloj interno
7     bcf     STATUS,RP0      ; Banco 0
8     movlw   256-39
9     movwf   TMR0            ; TMR0 = 256-39
10
11 _espera
12    btfss   INTCON,T0IF
13    goto    _espera         ; Espera a que pasen los 0.5 milisegundos
```

Mismo ejemplo escrito en lenguaje C.

```
1  #include <pic.h>
2
```

```
3  void main()
4  {
5      PSA = 0;
6      PS2 = 1;
7      PS1 = 0;
8      PS0 = 1;                  // Preescala = 64
9      T0CS = 0;
10     TMR0 = 256-39;
11
12     while(!T0IF);             // Espera
13 }
```

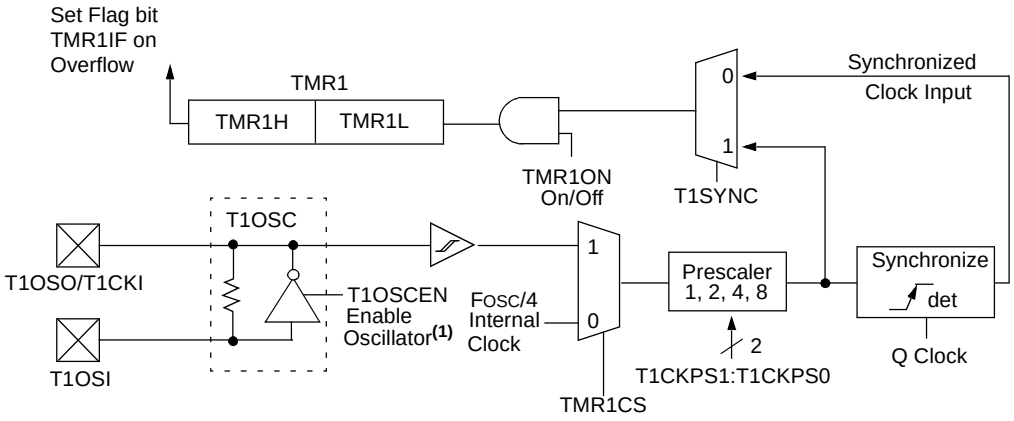
3.4.2. Temporizador TMR1

Características

- Es un temporizador/contador de 16 bits formado por los registros `TMR1H`:`TMR1L` de 8 bits.
- Se activa con `T1CON.TMR1ON = 1`.
- Puede seleccionar una fuente de reloj interna (`T1CON.TMR1CS = 0`) o externa (`T1CON.TMR1CS = 1`)
- Permite la generación de interrupciones al desbordarse el contador, es decir, al pasar de 65535⇒0. Para ello es necesario que:
  - `INTCON.GIE = 1`, habilitación global de interrupciones.
  - `INTCON.PEIE = 1` y `PIE1.TMR1IE = 1`, habilitación local.

  Cuando se produzca el desbordamiento el flag `PIR1.TMR1IF = 1` para señalizarlo. Este bit hay que borrarlo después para detectar el siguiente desbordamiento. Si se atendió una interrupción hay que borrarlo para que no se vuelva a producir.

Esquema hardware



Note 1: When the T1OSCEN bit is cleared, the inverter is turned off. This eliminates power drain.

Registros

- `TMR1H`, banco 0 (*Timer 1 High Part Register*) Parte alta del registro temporizador número 1.
- `TMR1L`, banco 0 (*Timer 1 Low Part Register*) Parte baja del registro temporizador número 1.
- `T1CON` Registro de control del temporizador 1

T1CON

Bit 7	6	5	4	3	2	1	Bit 0
-	-	T1CKPS1	T1CKPS0	T1OSCEN	/T1SYNC	TMR1CS	TMR1ON

T1CKPS1:T1CKPS0 Bits de selección de pre-escala del reloj del temporizador 1.

00: Divisor de frecuencia a 1:1
01: Divisor de frecuencia a 1:2
10: Divisor de frecuencia a 1:4
11: Divisor de frecuencia a 1:8

T1OSCEN Bit de Control de Habilitación del Oscilador Externo de TMR1.

0: Oscilador desactivado.
1: Oscilador activado.

/T1SYNC Control de la sincronización de la entrada de reloj externa de TMR1.

* Si `TMR1CS=1` (reloj externo).
  0: Sincroniza la entrada de reloj externa.
  1: No sincroniza la entrada de reloj externa.
* Si `TMR1CS=0` (reloj interno). Se ignora.

TMR1CS Selección de la fuente de reloj del TMR1.

0: Reloj interno ($F_{osc}/4$).
1: Reloj externo desde **T1OSO/T1CKI** con flanco de subida.

TMR1ON Activación del TMR1.

0: Para el temporizador 1.
1: Habilita el temporizador 1.

Resumen de registros

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	-000 0000	-uuu uuuu

Funcionamiento como temporizador

Para ello es necesario que:

- `T1CON.TMR1CS = 0`, con lo que se contarán ciclos de instrucción.
- En este caso el bit `T1CON.NOT_T1SYNC` de control de sincronización no es tenido en cuenta.

Funcionamiento como contador

Para ello es necesario que el bit `T1CON.TMR1CS = 1`, con lo que se incrementará con el flanco de subida, aunque primero es necesario al menos un flanco de bajada.

Contador sincronizado

Esto se consigue haciendo que `T1CON.NOT_T1SYNC = 0`, con lo que el contador estará sincronizado con el reloj interno después de la escala.

Dependiendo del valor de `T1CON.T1OSCEN` tenemos dos posibilidades:

- `T1CON.T1OSCEN = 0`, con lo que contará pulsos de la patilla **T1OSO/T1CKI**.
- `T1CON.T1OSCEN = 1`, con lo que contará pulsos de la patilla **T1OSI/CCP2**.

Además en modo **SLEEP** no se incrementa TMR1 aunque sí la pre-escala.

Contador asíncrono

Este caso se dará si `T1CON.NOT_T1SYNC = 1`. En modo **SLEEP** también se incrementará TMR1.

Las lecturas correctas no están garantizadas en este modo. Se realizan dos lecturas secuenciales de 8 bits y entre una y otra se puede haber modificado el valor de la cuenta.

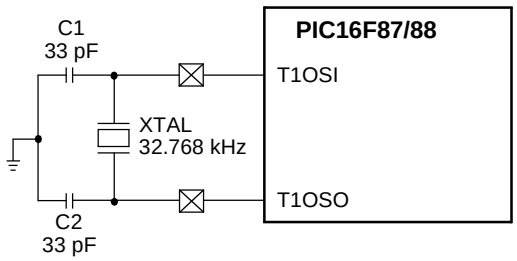
En las escrituras se recomienda parar el contador, de lo contrario el resultado es impredecible.

No se debe usar, en modo asíncrono, como base de tiempos para el módulo de captura/comparación y PWM (CCP).

Funcionamiento como contador con oscilador externo

Se necesita que:

- `T1CON.T1OSCEN = 1` lo active, con lo que las patillas **T1OSI** y **T1OSO** esperan un cristal externo.
- Continua oscilando en modo SLEEP.
- Admite cristales en modo LP (bajo consumo) de hasta 200 kHz. Está pensado para poner osciladores de 32 kHz y contar segundos.
- Es necesario asegurar, por software, el retardo de arranque del oscilador.



Reinicio del Temporizador 1

Dispone de un reset interno que:

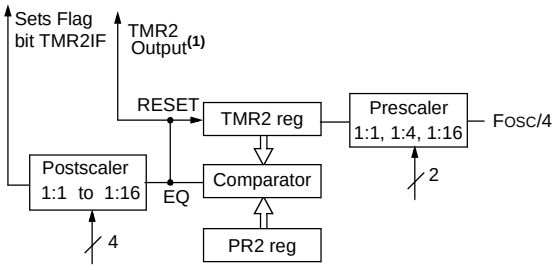
- Se genera desde alguno de los módulos CCP.
  Los bits `CCP1CON.CCP1M3:CCP1M0 = 1011b`.
- TMR1 debe funcionar como temporizador o contador síncrono.
- Los registros `CCPR1H:CCPR1L` pasan a ser registros periodo del temporizador.

3.4.3. Temporizador TMR2

Características

- Temporizador de 8 bits con pre-escala y post-escala.
- Cuenta ciclos de instrucción ($F_{osc}/4$). Se activa con `T2CON.TMR2ON = 1`
- Pre-escala seleccionable entre 1:1, 1:4 ó 1:16 mediante `T2CON.T2CKPS1:T2CKPS0`.
- Dispone de un registro periodo `PR2`. El registro `TMR2` se incrementa partiendo de cero hasta alcanzar `PR2`.
- La salida del temporizador TMR2 pasa por una post-escala programable: 1:1 ... 1:16 (`T2CON.TOUTPS3:TOUTPS0`) antes de generar una interrupción.
- Genera interrupciones si:
  - `INTCON.GIE = 1`, habilitación global.
  - `INTCON.PEIE=1` y `PIE1.TMR2IE = 1`, habilitación local.
- Cuando el campo `PIR1.TMR2IF = 1`, indica que se alcanzó el periodo. Hay que borrarlo después.
- Pre-escala y post-escala. Se borran si:
  - Ocurre una escritura en el registro `TMR2`.
  - Ocurre una escritura en `T2CON`
  - Se ha iniciado como consecuencia de un reset al arrancar (POR), una activación del reset (**`RA5//MCLR/VPP`**), un reset por desbordamiento del perro guardián (WDT) o un reset por caída de tensión de alimentación (BOR).
- La salida de `TMR2` alimenta el módulo SSP (*Synchronous Serial Port*) y puede opcionalmente usarse como generador de baudios.

Esquema hardware



Note 1: TMR2 register output can be software selected by the SSP module as a baud clock.

Registros

- **TMR2**, banco 0 (*Timer 2 Register*). Registro contador.
- **PR2**, banco 1 (*Period Register for Timer 2*). Registro periodo.
- **T2CON**, banco 0. Registro de control del temporizador 2

T2CON

Bit 7	6	5	4	3	2	1	Bit 0
-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

TOUTPS3:TOUTPS0 Bits de selección de la post-escala de salida del temporizador 2.

0000: Divisor de frecuencia a 1:1
0001: Divisor de frecuencia a 1:2
0010: Divisor de frecuencia a 1:3
....:
1111: Divisor de frecuencia a 1:16

TMR2ON Bit de control de habilitación del temporizador 2.

0: Temporizador desactivado.
1: Temporizador activado.

T2CKPS1:T2CKPS0 Bits de selección de la pre-escala del temporizador 2.

00: Divisor de frecuencia a 1:1
01: Divisor de frecuencia a 1:4
1x: Divisor de frecuencia a 1:16

Resumen de registros

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
11h	TMR2	Timer2 Module Register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
92h	PR2	Timer2 Period Register								1111 1111	1111 1111

3.5. Módulos CCP

3.5.1. Características

La familia de microcontroladores PIC16 dispone de módulos multifuncionales de Captura/Comparación y PWM (*Pulse Width Modulation*).

Cada módulo CCP contiene un registro de 16 bits que puede operar como:

- Registro de Captura.
- Registro de Comparación.
- Registro ciclo de trabajo para PWM (Modulación por Anchura de Pulsos) maestro/esclavo.

asociándose al temporizador TMR1 o al temporizador TMR2 según se ve en la figura:

CCP Mode	Timer Resource
Capture	Timer1
Compare	Timer1
PWM	Timer2

- Está compuesto de dos registros de 8 bits llamados **CCPR1H** y **CCPR1L**.
- Se controla con el registro **CCP1CON**.
- Es capaz de reiniciar TMR1 al comparar y coincidir.

3.5.2. Registros

- **CCPR1H** y **CCPR1L**, banco 0. Registros de cuenta parte alta y parte baja del módulo CCP1.
- **CCP1CON**, banco 0. Registro de control del módulo CCP1

CCP1CON

Bit 7	6	5	4	3	2	1	Bit 0
-	-	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0

CCP1X:CCP1Y Bits menos significativos en modo PWM.

Son los dos bits menos significativos del ciclo de trabajo en la modulación PWM. Los ocho bits más significativos estarán en **CCPR1L**.

CCP1M3:CCP1M0 Bits de selección del modo de funcionamiento del módulo CCP1.

0000: Desactiva el módulo CCP1.
0100: Modo captura cada flanco de bajada.
0101: Modo captura cada flanco de subida.
0110: Modo captura cada cuarto flanco de subida.
0111: Modo captura cada decimosexto flanco de subida.

1000: Modo comparación, salida a uno al coincidir. Se hace (**CCP1IF** = 1).

1001: Modo comparación, salida a cero al coincidir. Se hace (**CCP1IF** = 1).

1010: Modo comparación, genera interrupción software al coincidir. Se hace (**CCP1IF** = 1). La patilla **CCP1** no cambia.

1011: Modo comparación, dispara un evento especial. Se hace (**CCP1IF** = 1). La patilla **CCP1** no cambia.

* CCP1 reinicia TMR1 y arranca la conversión A/D (si está habilitado el módulo conversor A/D).

11xx: Modo PWM.

Registros asociados a la captura, comparación y al temporizador TMR1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
0Bh,8Bh 10BH,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
86h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	-000 0000	-uuu uuuu
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000

Registros asociados a la modulación por anchura de pulsos (PWM) y al temporizador TMR2

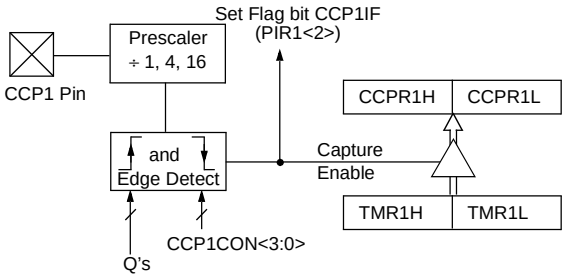
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
0Bh,8Bh 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
86h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
11h	TMR2	Timer2 Module Register								0000 0000	0000 0000
92h	PR2	Timer2 Module Period Register								1111 1111	1111 1111
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000

3.5.3. Modo Captura

- Consiste en capturar el valor del temporizador TMR1 en **CCPR1H**:**CCPR1L** cuando ocurre en la patilla **CCP1** un evento:

  - Cada flanco de bajada.
  - Cada flanco de subida.
  - Cada 4 flancos de subida.
  - Cada 16 flancos de subida.

- La patilla **CCP1** debe ser configurada como entrada.



- TMR1 debe funcionar en modo temporizador o contador síncrono.
- Produce interrupciones si:
  - `INTCON.GIE = 1`, habilitación global de interrupciones.
  - `INTCON.PEIE = 1`, habilitación de interrupciones para periféricos.
  - `PIE1.CCP1IE = 1`, habilitación local

  El campo `PIR1.CCP1IF = 1` señalizará el suceso. Hay que borrarlo después tanto si las interrupciones están activadas (para indicar que fue atendida) como si no.
- Para cambiar la pre-escala es importante parar el modo captura. Esto se puede hacer con `CCP1CON = 00000000b`.

Ejemplo

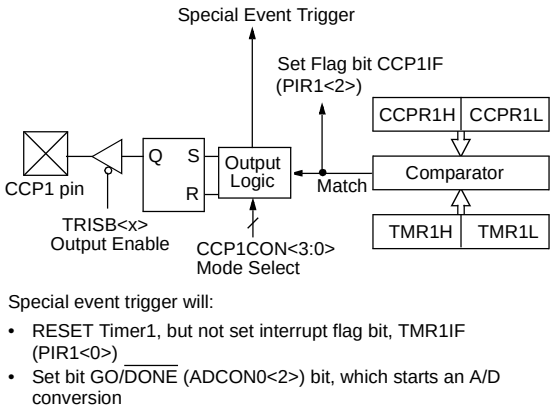
Listado 3.1: Codigo/PIC16F88_Ejemplos/EX_CAPTURA/CAPTURA.C

```
#include <pic.h>
#include "binario.h"

void interrupt ISR(void)
{
 TMR1H = 0;
 TMR1L = 0;      // TMR1 = 0
 CCP1IF = 0;     // Borro la notificación
}

void main()
{
 TRISB0 = 1;     // RB0/CCP1 entrada
 CCP1IE = 1;
 PEIE   = 1;
 GIE    = 1;     // Activo las interrupciones

 T1CON  = 0;     // Desactivo TMR1, div. frec=1:1
```

```
                    //  osc. ext. desactivado, reloj interno
20                  //  Cada paso es 1 microsegundo
    CCP1CON = B00000101;
                    // Captura cada flanco de subida
    TMR1H   = 0;
    TMR1L   = 0;    // TMR1 = 0
25  TMR1ON  = 1;    // Activa el TMR1

    while(1);       // Bucle sin fin
                    // Visualizaremos CCPR1H:CCPR1L y deberá
                    //  coincidir con el periodo de la señal
30                  //  del generador
}
```

3.5.4. Modo Comparación

- Consiste en comparar **TMR1H:TMR1L** con **CCPR1H:CCPR1L**. Cuando haya coincidencia la patilla **CCP1** pasara a:
  - Valer 1.
  - Valer 0.
  - No cambia, pero se genera una interrupción.
- La acción se programará con **CCP1CON.CCP1M3:CCP1M0**.
- **CCP1** debe ser configurado como salida.



- TMR1 debe funcionar en modo temporizador o contador síncrono.
- Produce interrupciones si se selecciona **CCP1CON.CCP1M3:CCP1M0** = 1010b y:
  - **INTCON.GIE** = 1, habilitación global de interrupciones,
  - **INTCON.PEIE** = 1, habilitación de interrupciones de los periféricos,
  - **PIE1.CCP1IE** = 1, habilitación local.

  El campo **PIR1.CCP1IF** = 1, señalizará el evento. Hay que borrarlo después.

- El modo de evento especial se selecciona con: **CCP1CON.CCP1M3:CCP1M0** = 1011b y consiste en:
  - Reinicia TMR1 (**CCPR1H:CCPR1L** funcionarán como un registro periodo) e iniciará la conversión A/D (si está activada).

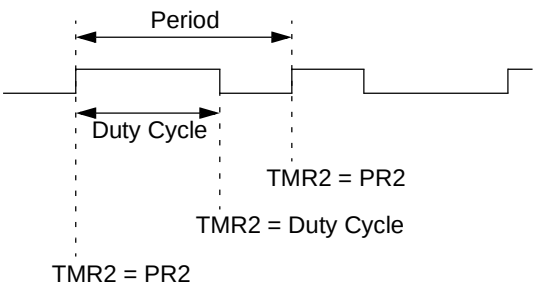
Ejemplo

Listado 3.2: Codigo/PIC16F88_Ejemplos/EX_COMPARACION/COMPARA.C

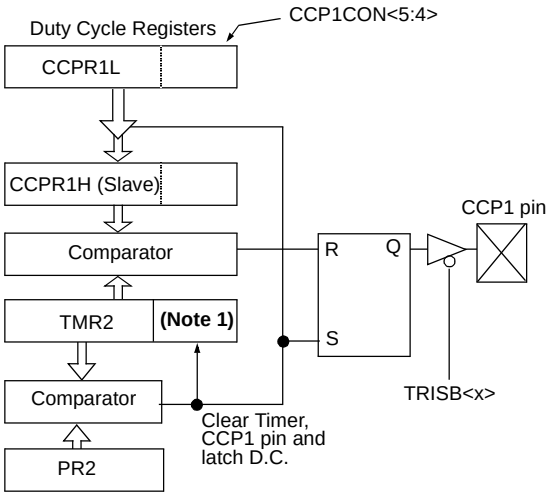
```
#include <pic.h>
#include "binario.h"

void interrupt ISR(void)
{
 if(RB0) RB0 = 0;
 else    RB0 = 1;
 TMR1H = 0;
 TMR1L = 0;      // TMR1 = 0
 CCP1IF = 0;     // Borro la notificación
}

void main()
{
 TRISB0 = 0;     // RB0/CCP1 salida
 CCP1IE = 1;
 PEIE   = 1;
 GIE    = 1;     // Activo las interrupciones

 T1CON  = 0;     // Desactivo TMR1, div. frec=1:1
                 //  osc. ext. desactivado, reloj interno
                 //  Cada paso es 1 microsegundo (Fosc=4 MHz)
 CCP1CON = B00001010;
                 // Comparación. CCP1 no cambia
 CCPR1H = (3000>>8);
 CCPR1L = (3000&0xFF);  // 3000 microsegundos
 TMR1ON = 1;     // Activa el TMR1

 while(1);       // Bucle sin fin
}
```

3.5.5. Modo PWM (Modulación por Anchura de Pulsos)

- Produce una señal PWM con 10 bits de resolución.



- La patilla **CCP1** debe configurarse como salida. El esquema hardware es como sigue:



Note 1: 8-bit timer is concatenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time base.

- El periodo de la señal PWM se calcula:

$$T_{PWM} = [PR2 + 1] * 4 * T_{osc} * PreescalaTimer2$$

  Ejemplo:

  * Sea $F_{osc} = 4$ MHz, luego $4T_{osc} = 1$ microsegundo.
  * Sea $F_{PWM} = 1$ kHz, luego $T_{PWM} = 1$ milisegundo.
  * Aplicando la ecuación:

$$10^{-3} = (PR2 + 1) * 10^{-6} * 16$$

    sale PR2= 61,5. Si hacemos PR2= 62 tendremos un periodo de 1,008 ms lo que dará una frecuencia de 992,06 Hz. Más que aceptable.

- Al cumplir el periodo
  - **TMR2** es borrado.
  - La patilla **CCP1** se pone a 1 (si el ciclo trabajo es distinto de 0).
  - El ciclo de trabajo **CCPR1H** (solo lectura)= **CCPR1L**.
- El ciclo de trabajo (hasta 10 bits) se especifica en **CCPR1L** (8 bits más significativos) y **CCP1CON**.[**CCP1X** y **CCP1Y**] (2 bits menos significativos)

$$CicloTrabajo = CCPR1L : CCP1CON < 5 : 4 > *T_{osc} * PreescalaTimer2$$

  medido en segundos, siendo $CicloTrabajo < PR2$.

  Cuando se cumple este intervalo, la patilla **CCP1** pasa a 0.

- La resolución será de $\log_2(\frac{F_{osc}}{F_{PWM}})$ bits.

- Protocolo a seguir:
  - Se escribe el periodo en **PR2** (banco 1) y se define la pre-escala con (**T2CON.T2CKPS1:T2CKPS0**).
  - Se escribe el ciclo de trabajo en **CCPR1L** y **CCP1CON.CCP1X:CCP1Y**
  - Se configura **CCP1** como pin de salida.
  - Se activa **TMR2** con **T2CON** y se inicia la pre-escala.
  - Se configura el módulo CCP1 para modo PWM.

Ejemplo

Listado 3.3: Codigo/PIC16F88_Ejemplos/EX_PWM/PWM.C

```
#include <pic.h>
#include "binario.h"

void main()
{
 TRISB0 = 0;     // RB0/CCP1 salida

// Queremos una señal PWM que tenga un periodo de 1 KHz
//  y el microcontrolador funciona a 4 MHz
//  luego aplicando la ecuacion sale que PR2 = 62 y que la
//  pre-escala = 16

 PR2 = 62;
 T2CKPS1 = 1;    // Preescala = 1:16

 CCPR1L = 15;    // 15/62 => 25% ciclo de trabajo
 CCP1X = CCP1Y = 0; // Dos bits menos signif. a cero
 CCP1M3 = 1;
 CCP1M2 = 1;     // Activo modo PWM
 TMR2ON = 1;     // Activo TMR2
 while(1);       // Bucle sin fin
}
```

3.6. AUSART

3.6.1. Introducción

El módulo AUSART (*Addressable Universal Synchronous Asynchronous Receiver Transmitter*) es un interfaz de comunicaciones serie, también conocido como SCI (*Serial Communication Interface*).

- Puede ser configurado como:
  - Modo asíncrono (*full duplex*).
  - Modo síncrono maestro (*half duplex*).
  - Modo síncrono esclavo (*half duplex*).
- Independientemente del modo debemos:
  - Habilitarlo con `RCSTA.SPEN = 1`.
- Y dependiendo del modo (TX salida, RX entrada):
  - Configurar `TRISB.5 = 0` (pin **RB5/TX/CK** como salida).
  - `TRISB.2 = 1` (pin **RB2/RX/DT** como entrada).

3.6.2. Registros

- `TXREG`. En el banco 0. Registro de transmisión.
- `RCREG`. En el banco 0. Registro de recepción.
- `TXSTA`. En el banco 1. Registro de estado y control para la transmisión.

TXSTA

Bit 7	6	5	4	3	2	1	Bit 0
CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D

CSRC Selección de la fuente de reloj. En modo asíncrono no importa. En modo síncrono:
- 0: Modo esclavo. El reloj es externo.
- 1: Modo maestro. El reloj proviene del generador de baudios.

TX9 Activa la transmisión del noveno bit.
- 0: Transmisión de 8 bits.
- 1: Transmisión de 9 bits.

TXEN Activa la transmisión.
- 0: Transmisión desactivada.
- 1: Transmisión activada.

SYNC Selección del modo de funcionamiento.
- 0: Modo asíncrono.
- 1: Modo síncrono.

BRGH Selecciona el rango de velocidades altas. Empleado en modo asíncrono.

0: Velocidad baja.
1: Velocidad alta.

TRMT Estado del registro de desplazamiento de la transmisión.

0: Registro lleno/ocupado.
1: Registro vacío.

TX9D Noveno bit a transmitir si está activada la transmisión de 9 bits. Representaría el bit de paridad.

- **RCSTA**. En el banco 0. Registro de estado y control de la recepción.

RCSTA

Bit 7	6	5	4	3	2	1	Bit 0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

SPEN Habilita el módulo USART.

0: Desactivado.
1: Activado.

RX9 Activa la recepción del noveno bit.

0: Recepción de 8 bits.
1: Recepción de 9 bits.

SREN Activa la recepción de un único dato. Solamente en modo síncrono maestro.

0: Desactivada
1: Activada.

CREN Activa la recepción continua de datos.

* Modo asíncrono:
  0: Desactivado.
  1: Activado.
* Modo síncrono:
  0: Desactivado.
  1: Activado hasta que se ponga de nuevo a cero (más prioridad que SREN)

ADDEN Activa la detección de dirección. Sólo en modo asíncrono con 9 bits.

0: Desactivado. Todos lo recibido son datos y el noveno bit es la paridad.
1: Activado. Habilita la interrupción y carga del búfer de recepción cuando el noveno bit vale 1.

FERR Error de marcaje.

0: No hay error.
1: El bit de *stop* no valió 1.

OERR Error de sobreescritura.

0: No hay error.
1: Se recibió otro dato antes de leer el anterior.

RX9D Noveno bit recibido si está activada la recepción de 9 bits. Representaría el bit de paridad.

- **SPBRG**. En el banco 1. Es el valor base para el funcionamiento del generador de baudios.

3.6.3. Generador de baudios

- Funciona en modo asíncrono y síncrono.
- Es un temporizador de 8 bits.
- El registro **SPBRG** controla el periodo de la temporización del generador de pulsos, aunque condicionado por **TXSTA.BRGH** y **TXSTA.SYNC** (modo asíncrono o síncrono) según se indica en la figura:

SYNC	BRGH = 0 (Low-speed)	BRGH = 1 (High-speed)
0	(Asynchronous) Baud Rate = Fosc/(64(X + 1))	Baud Rate = Fosc/(16(X + 1))
1	(Synchronous) Baud Rate = Fosc/(4(X + 1))	N/A

Legend: X = value in SPBRG (0 to 255)

- La escritura en **SPBRG** reinicia la cuenta.
- La señal en **RB2/RX/DT** se muestrea 3 veces durante el ciclo dado por **SPBRG** y se decide su valor por mayoría.

Así en el caso de **BRGH**=1 asíncrono (alta velocidad) tendremos:

$$X = \frac{F_{OSC}}{16 * Baudios} - 1$$

Generador de baudios según la frecuencia. Modo asíncrono lento (BRGH=0)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—
1.2	1.221	+1.75	255	1.202	+0.17	207	1.202	+0.17	129
2.4	2.404	+0.17	129	2.404	+0.17	103	2.404	+0.17	64
9.6	9.766	+1.73	31	9.615	+0.16	25	9.766	+1.73	15
19.2	19.531	+ 1.72	15	19.231	+0.16	12	19.531	+1.72	7
28.8	31.250	+8.51	9	27.778	-3.55	8	31.250	+8.51	4
33.6	34.722	+3.34	8	35.714	+6.29	6	31.250	-6.99	4
57.6	62.500	+8.51	4	62.500	+8.51	3	52.083	-9.58	2
HIGH	1.221	—	255	0.977	—	255	0.610	—	255
LOW	312.500	—	0	250.000	—	0	156.250	—	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.300	0	207	0.3	0	191
1.2	1.202	+0.17	51	1.2	0	47
2.4	2.404	+0.17	25	2.4	0	23
9.6	8.929	+6.99	6	9.6	0	5
19.2	20.833	+8.51	2	19.2	0	2
28.8	31.250	+8.51	1	28.8	0	1
33.6	—	—	—	—	—	—
57.6	62.500	+8.51	0	57.6	0	0
HIGH	0.244	—	255	0.225	—	255
LOW	62.500	—	0	57.6	—	0

Generador de baudios según la frecuencia. Modo asíncrono rápido (BRGH=1)

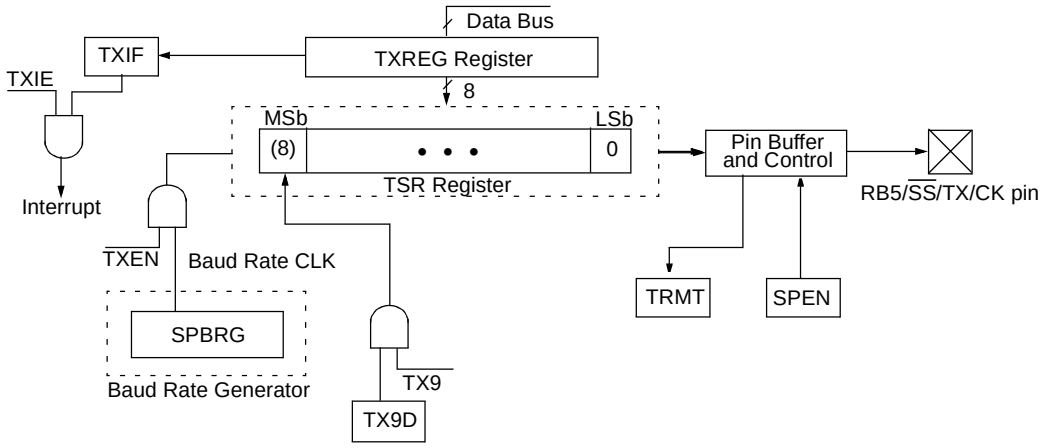
BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	+1.71	255
9.6	9.615	+0.16	129	9.615	+0.16	103	9.615	+0.16	64
19.2	19.231	+0.16	64	19.231	+0.16	51	19.531	+1.72	31
28.8	29.070	+0.94	42	29.412	+2.13	33	28.409	-1.36	21
33.6	33.784	+0.55	36	33.333	-0.79	29	32.895	-2.10	18
57.6	59.524	+3.34	20	58.824	+2.13	16	56.818	-1.36	10
HIGH	4.883	—	255	3.906	—	255	2.441	—	255
LOW	1250.000	—	0	1000.000	—	0	625.000	—	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	—	—	—	—	—	—
1.2	1.202	+0.17	207	1.2	0	191
2.4	2.404	+0.17	103	2.4	0	95
9.6	9.615	+0.16	25	9.6	0	23
19.2	19.231	+0.16	12	19.2	0	11
28.8	27.798	-3.55	8	28.8	0	7
33.6	35.714	+6.29	6	32.9	-2.04	6
57.6	62.500	+8.51	3	57.6	0	3
HIGH	0.977	—	255	0.9	—	255
LOW	250.000	—	0	230.4	—	0

3.6.4. Modo asíncrono: transmisión

Funcionamiento

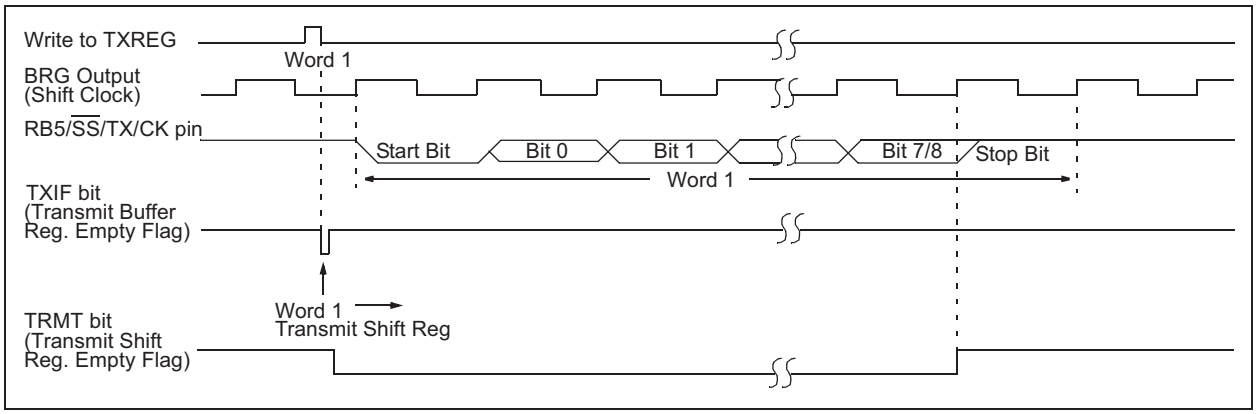
- El dato a transmitir se deposita en **TXREG**.
- Cuando se termina de transmitir el dato anterior se traslada automáticamente al registro de desplazamiento TSR y se indica con **PIR1.TXIF** (**TXREG** vacío). Si **PIE1.TXIE** = 1 se produce una interrupción para recargarlo.
- Se extraen de TSR los bits (LSBit..MSBit) al ritmo indicado por el generador de baudios a través de **TX**.
- El dato se precede de un bit de inicio (0) y al final se añade un bit de parada (1).
- El bit **TXSTA.TRMT** indica cuando se ha terminado de transmitir el contenido de TSR.



Secuencia de operaciones

- Se debe configurar **TX/CK** como salida y **RX/DT** como entrada.
- `RCSTA.SPEN = 1` habilitará el módulo AUSART.
- `TXSTA.SYNC = 0` activará el modo asíncrono.
- Si queremos interrupciones (`PIE1.TXIE = 1`, `INTCON.GIE = 1` y `INTCON.PEIE = 1`).
- Si queremos trasmitir datos de 9 bits, `TXSTA.TX9 = 1`. El noveno bit deberá estar en `TXSTA.TX9D`.
- Se inicia `SPBRG` y se configura `TXSTA.BRGH` para la velocidad deseada.
- Se activa la transmisión con `TXSTA.TXEN = 1`.
- El microcontrolador pondrá `PIR1.TXIF = 1` para indicar que hay que poner un dato.
- Pondremos el dato en `TXREG` (y `TXSTA.TX9D`, si 9 bits).
- Se transmite el dato y se espera otro (`PIR1.TXIF = 1`). Se producirá una interrupción si están habilitadas.

Cronograma de transmisión



Registros asociados a la transmisión

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	R0IF	-000 000x	-000 000u
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Ejemplo

Listado 3.4: Codigo/PIC16F88_Ejemplos/EX_TX_ASINC/TX_ASINC.C

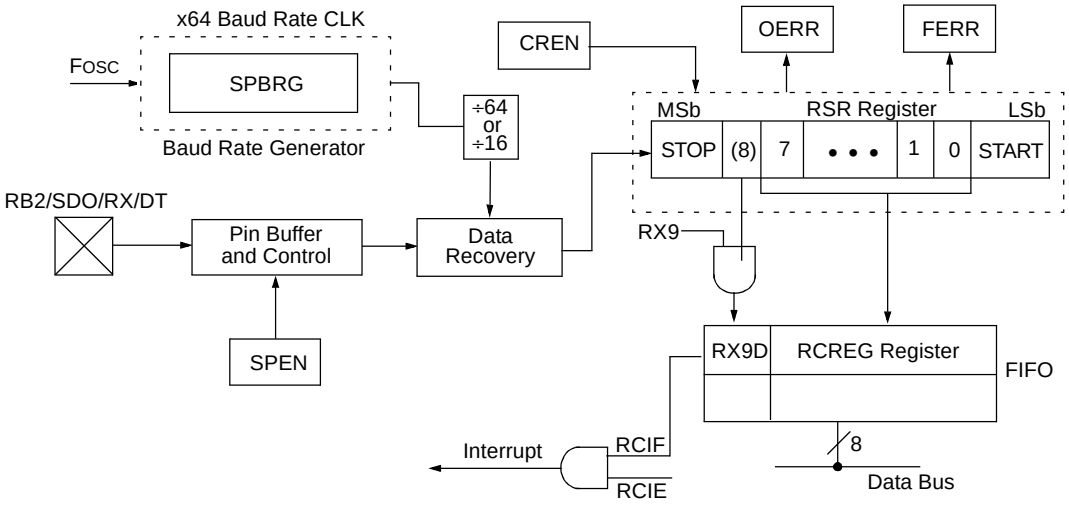
```
#include <pic.h>

void main()
{
 unsigned char i=0;

 TRISB5 = 0;          // RB5/TX salida
 TRISB2 = 1;          // RB2/RX entrada
 TXSTA  = 0x24;       // 8 bits, asincrono, alta velocidad,
                      //  transmision activada
 SPBRG  = 20;         // El micro a 20 MHz, 57600 baudios
 RCSTA  = 0x90;       // Activo USART, 8 bits, recepcion
                      //  continua, no direccion

 while(1)
  {
    while(TRMT==0); // Espera a terminar de enviar un dato
    TXREG = i++;    // Envia un dato
  }
}
```

3.6.5. Modo asíncrono: recepción

Funcionamiento

- Los bits recibidos en **RX/DT** se muestrean (a un ritmo de `SPBRG x 16`) y son introducidos en el registro de desplazamiento RSR al ritmo indicado por `SPBRG`.
- Después de recibir el bit de *stop* el dato es copiado en `RCREG` si está vacío y se indica (`PIR1.RCIF = 1`). Si están activadas (`PIE1.RCIE = 1`, `INTCON.PEIE = 1` y `INTCON.GIE = 1`) se producirá una interrupción.
- `PIR1.RCIF` se pondrá a cero cuando se lea el dato de `RCREG`.
- `RCREG` es un registro doble que funciona como un pila FIFO (*First In, First Out*: primero en entrar, primero en salir).

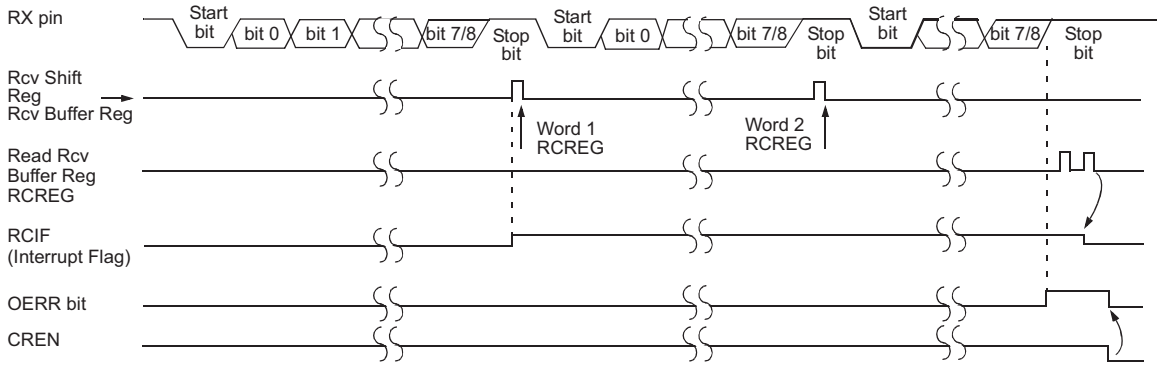
- Si se recibe un tercer dato sin haber leído los anteriores se producirá un error de desbordamiento (`RCSTA.OERR = 1`) y se inhibirán futuras recepciones hasta que se borre.
- Si el dato es de 9 bits, se debe leer primero `RCSTA.RX9D` y después `RCREG` para no perder información.
- Si el dato se recibe incompleto (bit de *stop* incompleto) se indica con `RCSTA.FERR`.



Secuencia

- Se debe configurar el generador de baudios (`SPBRG` y `TXSTA.BRGH`).
- Se debe habilitar el periférico (`RCSTA.SPEN = 1` y `TXSTA.SYNC = 0`).
- Si queremos interrupciones, `PIE1.RCIE = 1`, `INTCON.GIE = 1` y `INTCON.PEIE = 1`.
- Si queremos una recepción de 9 bits, `RCSTA.RX9 = 1` (el noveno bit se recibirá en `RCSTA.RX9D`).
- Se debe activar la recepción continua (`RCSTA.CREN = 1`).
- `PIR1.RCIF = 1` cuando se reciba un dato y se producirá una interrupción si están activadas.
- En `RCSTA.FERR` y `RCSTA.OERR` se comprueba la correcta recepción. Para comenzar de nuevo eliminando los errores, poner `RCSTA.CREN = 0` y de nuevo a uno.
- En `RCSTA.RX9D` estará el noveno bit (si así lo configuramos).
- Se lee de `RCREG` el dato.

Cronograma de recepción asíncrona



Note: This timing diagram shows three words appearing on the RX input. The RCREG (Receive Buffer) is read after the third word, causing the OERR (Overrun) bit to be set.

Registros asociados a la recepción

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Ejemplo

Listado 3.5: Codigo/PIC16F88_Ejemplos/EX_RX_ASINC/RX_ASINC.c

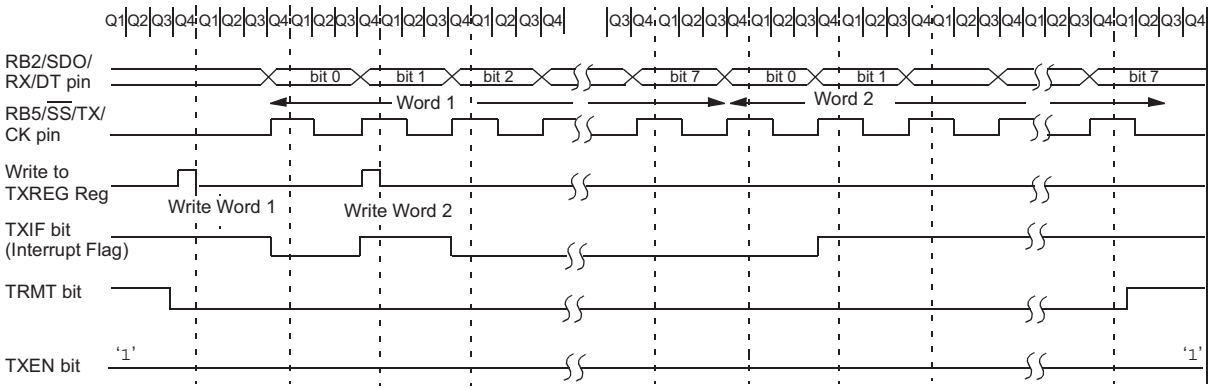
```
#include <pic.h>

void main()
{
 TRISB2 = 1;          // RC7/RX entrada
 TXSTA  = 0x24;       // 8 bits, asincrono, alta velocidad,
                      //  transmision activada
 SPBRG  = 20;         // El micro a 20 MHz, 57600 baudios
 RCSTA  = 0x90;       // Activo USART, 8 bits, recepcion
                      //  continua, no direccion

 while(1)
  {
    while(RCIF==0); // Espera a recibir un dato
    TXREG = RCREG;  // Reenvia el dato que le llega
  }
}
```

En el caso de que `F=4MHz`, y la velocidad hubiera sido de 9600 baudios habríamos usado `BRGH=1` y `SPBRG=25`.

3.6.6. Modo síncrono maestro: transmisión

- Es una transmisión *half duplex*.
- Se activa con `TXSTA.SYNC = 1`.
- Se transmite el reloj maestro (**CK**) y los datos en serie (**DT**).
- El modo maestro o esclavo se selecciona con `TXSTA.CSRC`.
- La secuencia de operaciones es la misma que en modo asíncrono.

Cronograma de transmisión en modo síncrono

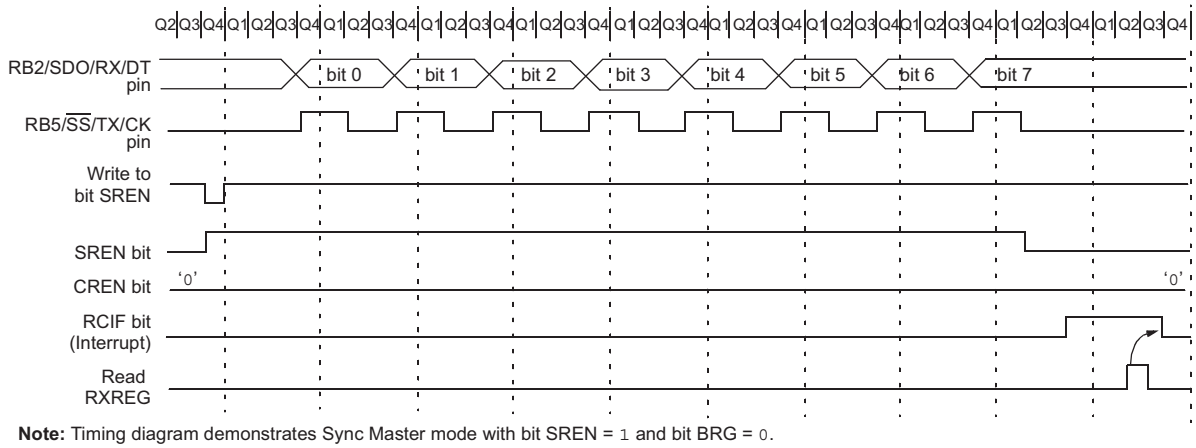


Note: Sync Master mode; SPBRG = 0. Continuous transmission of two 8-bit words.

3.6.7. Modo síncrono maestro: recepción

La secuencia de operaciones es la misma que en modo asíncrono.

Cronograma de recepción en modo síncrono



Note: Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRG = 0.

3.7. Módulo SSP (*Synchronous Serial Port*)

El módulo SSP (*Synchronous Serial Port*) es un interfaz de comunicaciones serie. Puede trabajar en dos modos:

- **Modo SPI** (*Serial Peripheral Interface*). Las patillas asociadas serán **RB4/SCK**, **RB1/SDI**, **RB2/SDO** y **RB5/SS**.
- **Modo I2C** (*Inter-Integrated Circuit*). Las patillas asociadas a este modo serán **RB4/SCL** y **RB1/SDA**.

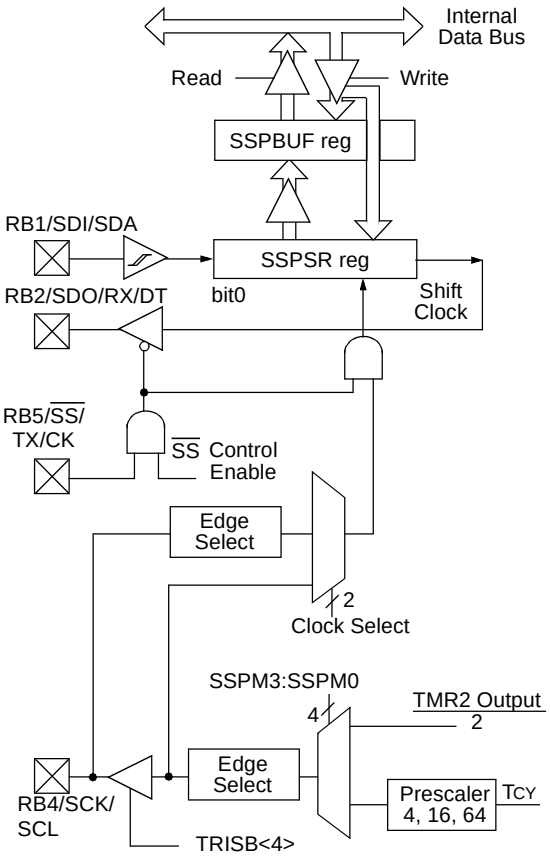
3.7.1. Módulo SSP: protocolo SPI

Características

- Transmisión/recepción de 8 bits *full-duplex*.
- Soporta los cuatro modos de funcionamiento del protocolo SPI (ver cronograma más adelante).
- En modo maestro se emplean tres pines:
  - *Serial Data Out* (**SDO**). Salida de datos serie.
  - *Serial Data In* (**SDI**). Entrada de datos serie.
  - *Serial Clock* (**SCK**). Reloj serie. Será una salida.
- En modo esclavo se usa además *Slave Select* (**/SS**) como entrada (activo en baja). En este caso la señal (**SCK**) será una entrada.

Esquema hardware

El esquema hardware del módulo SSP cuando está funcionando en modo SPI es el siguiente:



Configuración

- Se hace a través de los registros **SSPCON**, **SSPSTAT** y **SSPBUF**.
- Hay que elegir entre modo maestro (**SCK** de salida) o esclavo (**SCK** de entrada).
- Hay que elegir la polaridad del reloj o lo que es lo mismo el valor del estado ocioso de **SCK**.
- También tenemos que seleccionar la fase de los datos y el flanco del reloj.
- Por supuesto hay que definir la velocidad del reloj (en modo maestro).
- Y el modo de selección en caso de funcionar como esclavo.
- Habrá que habilitar el SSP con `SSPCON.SSPEN = 1`
- Y el sentido de las patillas debe programarse con **TRISB**.
  - En modo maestro: **SDI** de entrada; **SDO** y **SCK** salidas.
  - En modo esclavo: **SDI**, **SCK**, y **/SS** entradas; **SDO** salida.

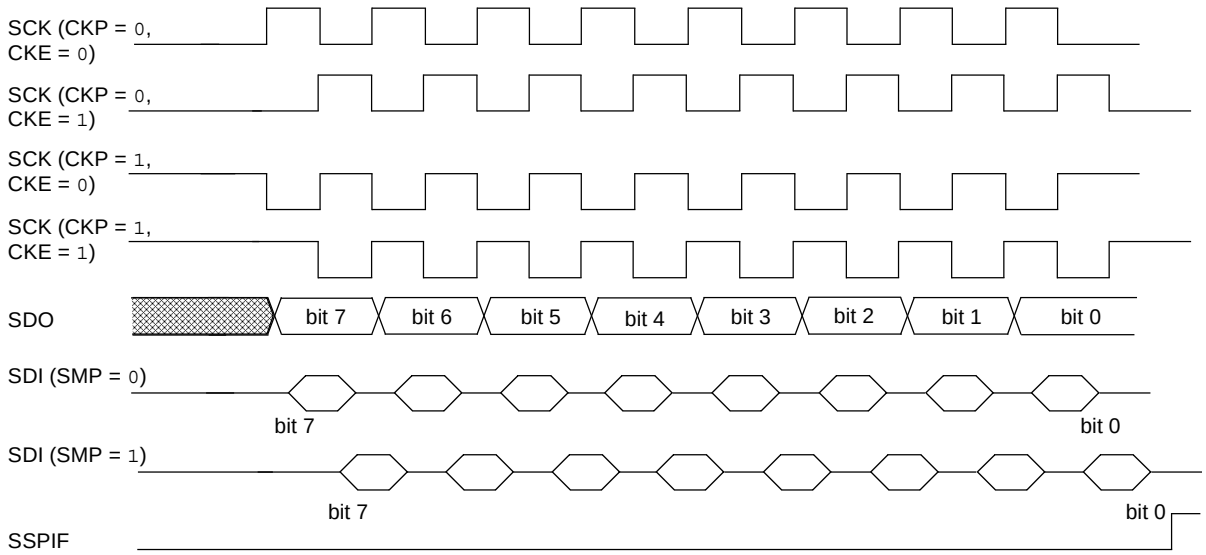
Modo SPI maestro

Características

- El microcontrolador controla la transmisión/recepción y el reloj.

- Se transmite tan pronto como el registro **SSPBUF** es escrito.
- La recepción automáticamente rellena el registro **SSPBUF**.
- Si no se necesita la recepción se puede aprovechar la patilla como entrada (se continuará monitorizando y almacenando en **SSPBUF**) o como salida.
- La polaridad se selecciona con **SSPCON**.**CKP**.
- La velocidad de transferencia se puede programar como $F_{osc}/4$, $F_{osc}/16$, $F_{osc}/64$ ó con la salida **TMR2**/2.

Cronograma



La señal de entrada **SDI** se muestrea en el punto medio (**SMP**=0) o al final del ciclo (**SMP**=1).

Ejemplo

Se muestra un ejemplo escrito en lenguaje C[3].

Listado 3.6: Codigo/PIC16F88_Ejemplos/EX_TX_SPI/TX_SPI.c

```
#include <pic.h>

void main()
{
 unsigned char i;

 TRISB4 = 0;          // RB4/SCK salida. Modo maestro
 TRISB1 = 1;          // RB1/SDI entrada
 TRISB2 = 0;          // RB2/SDO salida

 SSPEN  = 1;          // Activa el modulo MSSP
 CKP    = 1;          // SPI, estado ocioso = 1
 SSPCON = 0x32;       // Modo SPI maestro, reloj = Fosc/64
                      //  Micro a 1 MHz => 15625 bits por segundo
```

[3]Se hace notar que en lenguaje C algunos nombres de los campos de los registros pueden variar con respecto al ensamblador. Se puede ver en uno de los apéndices cómo están definidos.

```
                        //              => 1953.125 bytes por segundo
    STAT_CKE   = 0;     // Dato valido en flanco de subida del reloj

    i = 'A';
    SSPIF = 0;
    while(1)
    {
     SSPBUF = i++;      // Envio dato
     while(!SSPIF);     // Espero hasta que termine
     SSPIF = 0;         // Lo borro
    }
}
```

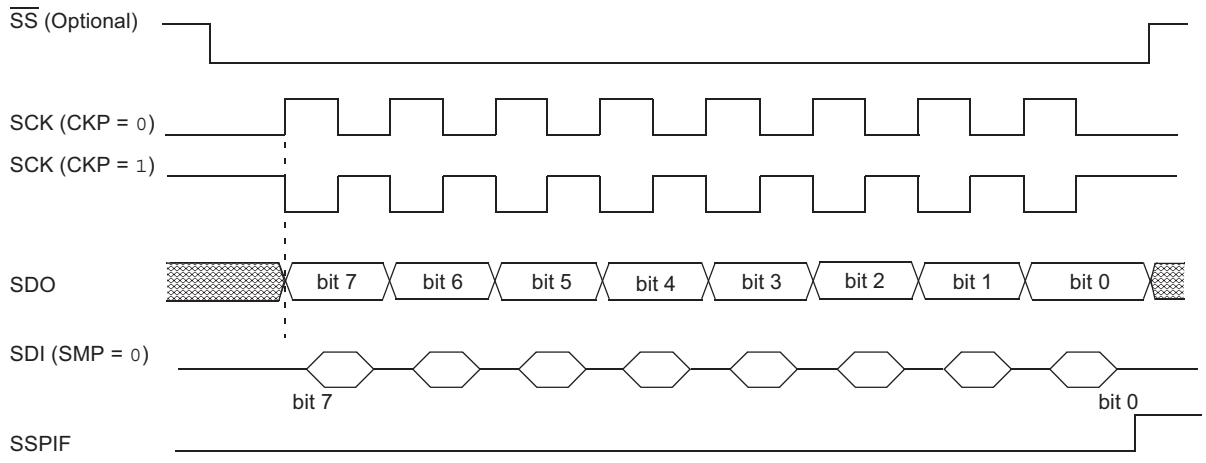
Modo SPI esclavo

Características

- El ritmo de transferencia o recepción viene dado externamente a través de **SCK**. Al recibir el último bit se activa `PIR1.SSPIF`.
- Funciona incluso en modo bajo consumo. Cuando se recibe un dato el microcontrolador es despertado. Opcionalmente cuando **/SS**=0

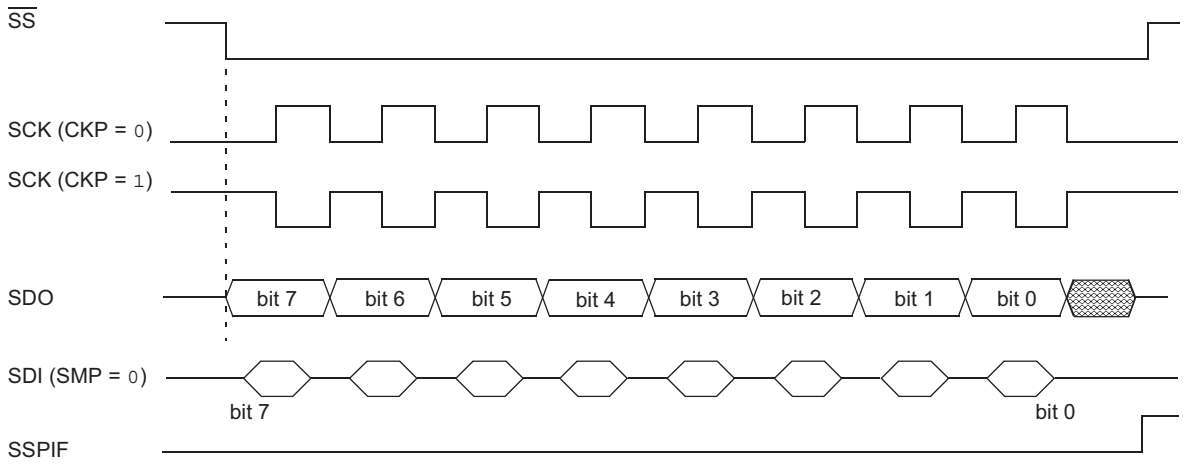
Cronograma: modo esclavo (CKE=0)

Vemos el cronograma en el que se distinguen dos casos dependiendo del valor de `SSPCON.CKP` para el caso de `SSPSTAT.CKE = 0`.

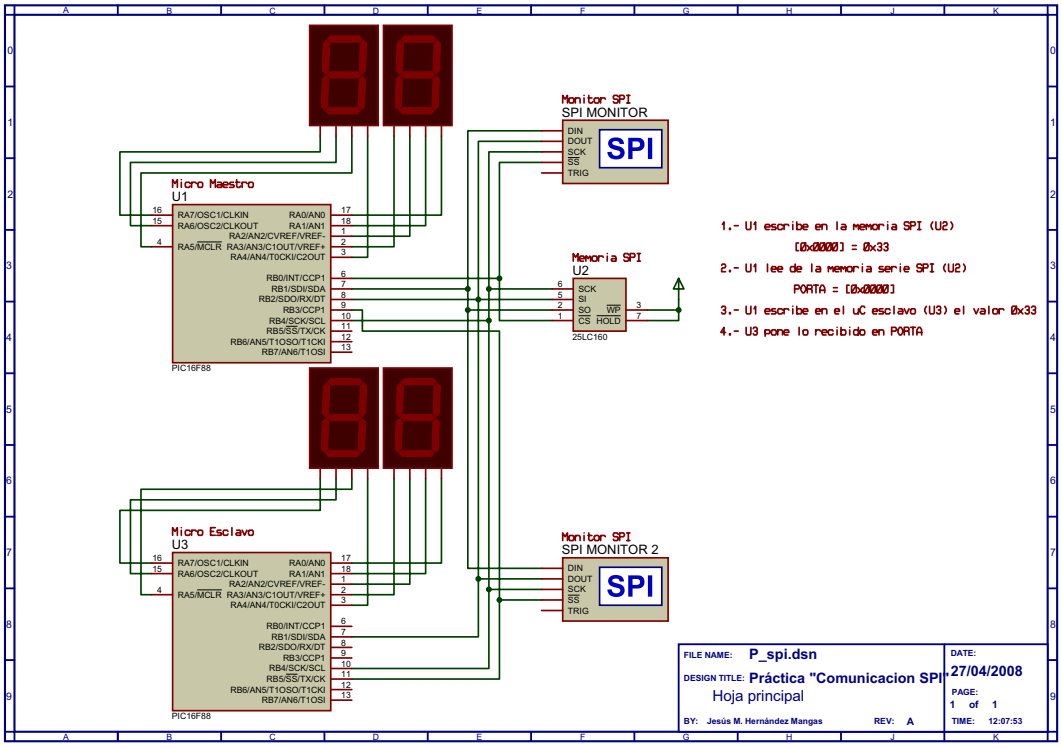


Cronograma: modo esclavo (CKE=1)

Cronograma si `SSPSTAT.CKE = 1`.



Ejemplo transmisión/recepción en SPI



Listado 3.7: Codigo/PIC16F88_Ejemplos/EX_TXRX_SPI/P_SPI.c

```
#include <pic.h>
#include "binario.h"

// Para el PIC16F88
__CONFIG( WDTDIS & INTIO & UNPROTECT & PWRTDIS & CCPRB0 & DEBUGDIS & LVPDIS
        & BORDIS & MCLRDIS );  // Palabra de configuración 0x2007
__CONFIG( FCMDIS & IESODIS );  // Palabra de configuración 0x2008


unsigned char SPI_EnviaRecibe(unsigned char c)
{
 SSPBUF = c;                   // Envía
```

```
 while(!BF);                    // Espera a que se transmita
 return SSPBUF;
}

unsigned char Lee_25LC16(unsigned int dir)
{
 unsigned char c;

 do
 {
  RB0 = 0;                      // Selecciono la memoria
  SPI_EnviaRecibe(B00000101);   // Comando READ STATUS
  c = SPI_EnviaRecibe(0);       // Lee el dato
  RB0 = 1;                      // Deselecciono la memoria
 } while(c&0x01);               // Mientras valga 1 el bit WIP (WriteInProgress)

 RB0 = 0;                       // Selecciono la memoria
 SPI_EnviaRecibe(B00000011);    // Comando READ
 SPI_EnviaRecibe(dir>>8  );     // Parte alta de la dirección
 SPI_EnviaRecibe(dir&0xFF);     // Parte baja
 c = SPI_EnviaRecibe(0);        // Lee el dato
 RB0 = 1;                       // Deselecciono la memoria
 return c;
}

void Escribe_25LC16(unsigned int dir, unsigned char d)
{
 RB0 = 0;                       // Selecciono la memoria
 SPI_EnviaRecibe(B00000110);    // Comando WRITE ENABLE
 RB0 = 1;                       // Para que lo ejecute

 RB0 = 0;                       // Selecciono la memoria
 SPI_EnviaRecibe(B00000010);    // Comando WRITE
 SPI_EnviaRecibe(dir>>8  );     // Parte alta de la dirección
 SPI_EnviaRecibe(dir&0xFF);     // Parte baja
 SPI_EnviaRecibe(d);            // Escribe el dato
 //SPI_EnviaRecibe(B00001000);    // Comando WRITE DISABLE
 RB0 = 1;                       // Deselecciono la memoria
}

void Escribe_uC_Slave(unsigned char c)
{
 RB3 = 0;                       // Selecciona uC_Slave
 SPI_EnviaRecibe( c );          // Envía dato, descarta lo recibido
 RB3 = 1;                       // Deselecciona uC_Slave
}

void main()
{
 unsigned char t;

 ANSEL  = 0;              // Configuro el PORTA como digital
 TRISB0 = 0;              // RB0 control memoria salida
 TRISB1 = 1;              // RB1/SDI input
 TRISB2 = 0;              // RB2/SDO output
 TRISB4 = 0;              // RB4/SCK output
 TRISB3 = 0;              // RB3 control uC slave salida
 TRISA = 0;               // Todo el puerto A es de salida
 RB4 = 0;                 // CKP = 0
 RB0 = 1;                 // Deselecciona la memoria
 RB3 = 1;                 // Deselecciona el uC slave
```

```
75  SSPCON  = B00100010;    // Control del modulo SSP
                            // WCOL 0   No collision
                            // SSPOV 0  No overflow
                            // SSPEN 1  Enables SSP
                            // CKP  0   Transmit on rising edge of SCK
80                          // SSPM<3:0>    0010    SPI Master, clk = OSC/64
                            //                      => (4 MHz)/64 =
    SSPSTAT = B00000000;    // Estado del módulo SSP
                            // SMP  0   Input data sampled at middle
                            // CKE  0   Data on falling edge of SCK (CKP=0)
85                          // D/A  0   -i2c only-
                            // P    0   -i2c only-
                            // S    0   -i2c only-
                            // R/W  0   -i2c only-
                            // UA   0   -i2c only-
90                          // BF   0   Ocupado -read only-

    Escribe_25LC16( 0, 0x33 );
    t = Lee_25LC16( 0 );
    PORTA = t;
95
    Escribe_uC_Slave( 0x44 );

    while(1);
}
```

Listado 3.8: Codigo/PIC16F88_Ejemplos/EX_TXRX_SPI/P_SPI_Slave.c

```
1   #include <pic.h>
    #include "binario.h"

    // Para el PIC16F88
    __CONFIG( WDTDIS & INTIO & UNPROTECT & PWRTDIS & CCPRB0 & DEBUGDIS & LVPDIS
6           & BORDIS & MCLRDIS );  // Palabra de configuración 0x2007
    __CONFIG( FCMDIS & IESODIS );  // Palabra de configuración 0x2008

    void main()
    {
11   ANSEL  = 0;             // Configuro el PORTA como digital
     //        76543210
     TRISB  = B00111011;
     //TRISB1 = 1;           // RB1/SDI input
     //TRISB2 = 0;           // RB2/SDO output
16   //TRISB4 = 1;           // RB4/SCK input
     //TRISB5 = 1;           // RB5//SS input (slave mode)
     RBPU = 1;               // Pull-ups desactivados
     TRISA  = 0;             // Puerto A son salidas

21   SSPBUF = 0x22;
     SSPCON  = B00100101;    // Control del modulo SSP
                             // WCOL 0   No collision
                             // SSPOV 0  No overflow
                             // SSPEN 1  Disables SSP
26                           // CKP  0   Transmit on rising edge of SCK
                             // SSPM<3:0>    0101    SPI Slave, SCK active
                             //
     SSPSTAT = B00000000;    // Estado del módulo SSP
                             // SMP  0   Input data sampled at middle
31                           // CKE  0   Data on falling edge of SCK (CKP=0)
                             // D/A  0   -i2c only-
                             // P    0   -i2c only-
                             // S    0   -i2c only-
```

```
                                // R/W   0    -i2c only-
36                              // UA    0    -i2c only-
                                // BF    0    Ocupado -read only-

    while(1)
     {
41     if(BF)                   // Dato recibido
        {
          PORTA = SSPBUF;       // Lo saco fuera
        }
     }
46 }
```

Registros

- **SSPBUF**. En el banco 0. Registro de transmisión y recepción.
- **SSPCON**. En el banco 0. Registro de control de la comunicación.

SSPCON

Bit 7	6	5	4	3	2	1	Bit 0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

WCOL Detección de colisión en la escritura en modo I2C.

* Modo maestro
  - 0: No hay colisión.
  - 1: Se ha intentado una escritura en **SSPBUF** y las condiciones I2C no eran válidas.
* Modo esclavo
  - 0: No hay colisión.
  - 1: **SSPBUF** se ha escrito y el dato anterior no se había transmitido.

SSPOV Indicador de desbordamiento en la recepción.

* Modo SPI
  - 0: No hubo sobreesccritura (*overflow*).
  - 1: Se recibió un nuevo dato que sobreescribió un dato anterior. Se debe borrar por software.
* Modo I2C
  - 0: No hubo *overflow*.
  - 1: Se recibió un nuevo dato que sobreescribió un dato anterior. Se debe borrar por software.

SSPEN Activa el módulo MSSP.

- 0: Módulo desactivado.
- 1: Módulo activado.

CKP Selección de la polaridad del reloj.

* Modo SPI
  - 0: El estado ocioso del reloj es un uno lógico.
  - 1: El estado ocioso del reloj es un cero lógico.
* Modo I2C esclavo. No usado en I2C maestro.
  - 0: Mantiene la señal de reloj externa a cero. Obliga al maestro a esperar.
  - 1: Deja libre el reloj externo.

SSPM3:SSPM0 Selecciona el modo de funcionamiento del módulo MSSP.

- 0000: Modo SPI maestro, reloj = $F_{osc}/4$.

0001: Modo SPI maestro, reloj = $F_{osc}/16$.
0010: Modo SPI maestro, reloj = $F_{osc}/64$.
0011: Modo SPI maestro, reloj = (salida del temporizador TMR2)/2.
0100: Modo SPI esclavo, reloj = **SCK**. Control de selección con patilla **/SS** activado.
0101: Modo SPI esclavo, reloj = **SCK**. Control de selección con patilla **/SS** desactivado. La patilla se puede usar como de propósito general.
0110: Modo I2C esclavo, dirección de 7 bits.
0111: Modo I2C esclavo, dirección de 10 bits.
1000,1001,1010: Reservados.
1011: Modo I2C master, controlado por tu propio *firmware*. Es un modo *Slave idle* que permite implementar el protocolo maestro por *bit-banging*.
1100, 1101: Reservados.
1110: Modo I2C esclavo. Direcciones de 7 bit con habilitación de interrupciones en los bit de *start* y *stop*.
1111: Modo I2C esclavo. Direcciones de 10 bit con habilitación de interrupciones en los bit de *start* y *stop*.

- **SSPSTAT**. En el banco 1. Registro de estado del módulo MSSP.

SSPSTAT

Bit 7	6	5	4	3	2	1	Bit 0
SMP	CKE	D/A	P	S	R/W	UA	BF

SMP Control del modo de muestreo de los bits recibidos.
* Modo SPI maestro. En modo SPI esclavo debe permanecer a cero.
  0: Dato muestreado en el punto medio.
  1: Dato muestreado al final.
* Modo I2C.
  0: Habilitación del control del ritmo de subida (*slew rate*) para el modo de 400 kHz.
  1: Deshabilitación del control de ritmo de subida para los modos de 100 kHz y 1MHz.

CKE Selección del flanco de reloj.
* Modo SPI maestro. En modo SPI esclavo debe permanecer a cero.

CKP=0 .
  0: La transmisión se produce al cambiar el reloj del estado ocioso al estado activo.
  1: La transmisión se produce al cambiar el reloj del estado activo al estado ocioso.

CKP=1 .
  0: Transmisión en el flanco de subida de **SCK**.
  1: Transmisión en el flanco de bajada de **SCK**.

* Modo I2C.
  0: Niveles de entrada compatibles con el protocolo SMBUS[4].
  1: Niveles de entrada compatibles con el protocolo I2C.

D/A Solo para modo I2C. Indica dirección o dato recibido.
  0: El último byte recibido fue una dirección.
  1: El último byte recibido fue un dato.

P Bit de Stop. Solo modo I2C.
  0: No se detectó la condición de *stop*.
  1: Se ha detectado la condición de *stop*.

[4]Protocolo muy similar al I2C, pero de Intel y otros del año 1995. Trabaja con voltaje fijo. Tiene una frecuencia de funcionamiento mínima de 10 kHz. Dispone de un tiempo máximo para la selección de los esclavos. Incluye otras características menores. Más información: `http://www.smbus.org/specs/`

S Bit de Start. Solo modo I2C.

0: No se detectó la condición de *start*.
1: Se ha detectado la condición de *start*.

R/W Escritura o lectura. Solo modo I2C.

* Modo esclavo.
  0: Escritura.
  1: Lectura.
* Modo maestro.
  0: No se transmite nada.
  1: Transmisión en progreso.

UA Actualiza dirección. Solo modo I2C de 10 bits.

0: La dirección no necesita ser actualizada.
1: El usuario debe actualizar la dirección en el registro **SSPADD**.

BF Búfer lleno.

* Recepción (modos SPI e I2C).
  0: Recepción no completa. **SSPBUF** vacío.
  1: Recepción completa. **SSPBUF** lleno.
* Transmisión (sólo modo I2C).
  0: Transmisión completa (no incluye /ACK ni la condición de *stop*). **SSPBUF** vacío.
  1: Transmisión en curso. **SSPBUF** lleno.

Registros asociados

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
0Bh,8Bh 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
86h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000

3.7.2. Módulo SSP: protocolo I2C

Características

- Funciona en modo maestro ó esclavo.
- Permite el direccionamiento de 7 bits o de 10 bits.
- No atiende a la llamada general (`dir = 00000000b`). A esta contesta todos los dispositivos conectados al bus.
- Funciona a 100 kHz, a 400 kHz y a 1 MHz.
- La patilla **SCL** es el reloj (bidireccional).
- La patilla **SDA** son los datos serie (bidireccional).

- Se activa el modulo con `SSPCON.SSPEN = 1`.
- `TRISB` debe configurar **SCL** y **SDA** como entradas.
- Se controla con 5 registros:
  - `SSPCON`, registro de control del módulo SSP.
  - `SSPSTAT`, registro de estado del módulo SSP.
  - `SSPBUF`, búfer de recepción/envío.
  - `SSPADD`, registro de dirección.
- Modos de operación:
  - I2C esclavo (dirección de 7 bits).
  - I2C esclavo (dirección de 10 bits).
  - I2C maestro controlador por firmware.

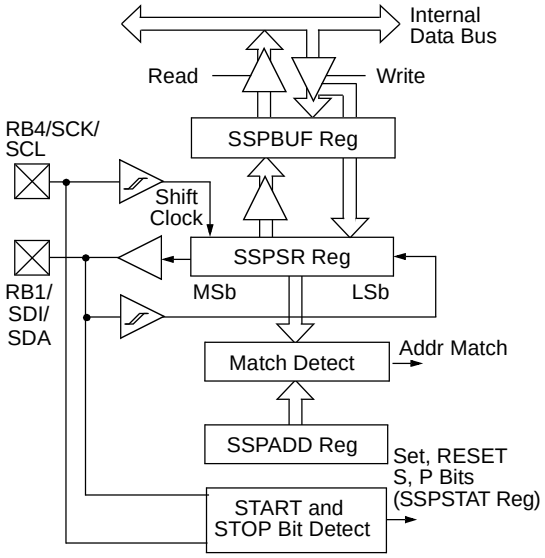
Funcionamiento

- `SSPSTAT.CKE = 0`, configurará niveles compatibles I2C.
- `SSPSTAT.CKE = 1`, configurará niveles eléctricos compatibles SMBus (Intel 1995).
- `SSPSTAT.S` indicará la condición de *start*.
- `SSPSTAT.P` indicará la condición de *stop*.
- `SSPSTAT.D_A` indicará si se recibió dato o dirección.
- `SSPSTAT.UA` indicará si el dato es la extensión a 10 bits de la dirección.

Modo I2C esclavo

- Se detectará la condición *start*.
- Se leerá el primer dato y se comparará con la dirección contenida en `SSPADD`.
- Si hay coincidencia se generará la señal ACK automáticamente excepto si
  - `SSPSTAT.BF = 1` que indicará ocupado.
  - `SSPCON.SSPOV = 1` que indicará `SSPBUF` desbordado.
- El dato se recibe bit a bit en el registro `SSPSR` y una vez completo se transfiere a `SSPBUF` y se indica con `PIR1.SSPIF`.

Esquema hardware

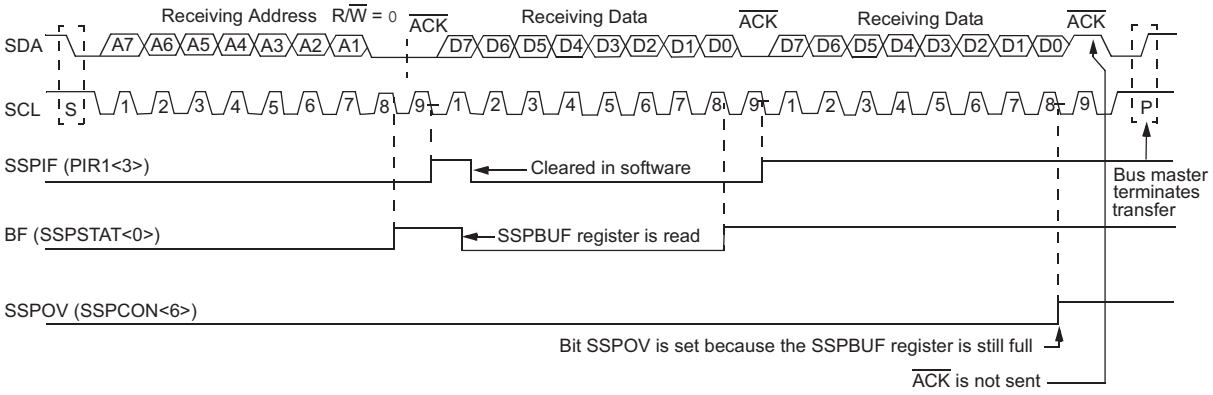


Bits de estado

Status Bits as Data Transfer is Received		SSPSR → SSPBUF	Generate $\overline{\text{ACK}}$ Pulse	Set SSPIF Bit (SSP Interrupt Occurs if Enabled)
BF	SSPOV			
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes
0	1	No	No	Yes

Note 1: Shaded cells show the conditions where the user software did not properly clear the overflow condition.

Cronograma de recepción (dirección de 7 bits)



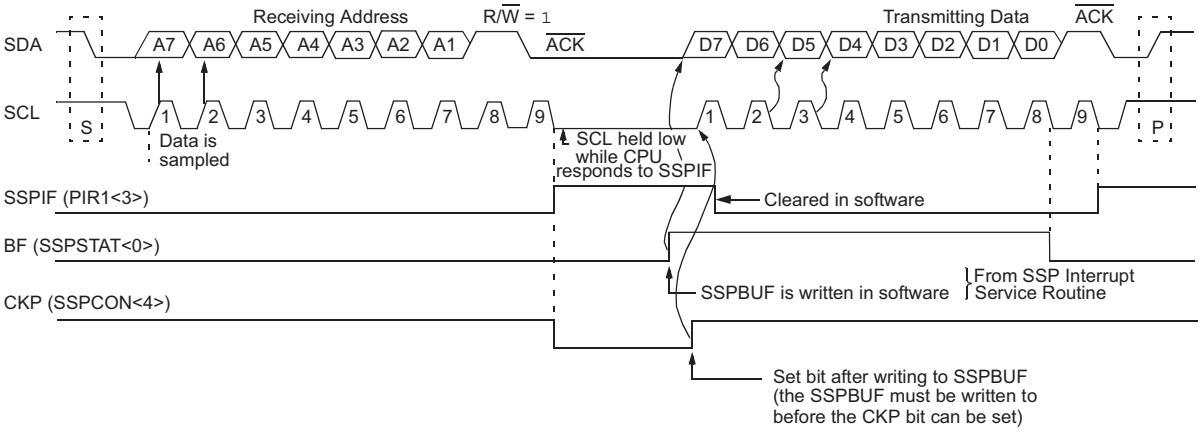
Registros asociados

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	0000 0000
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
94h	SSPSTAT	SMP(1)	CKE(1)	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000
86h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111

Modo I2C maestro

Hay que implementarlo mediante *bit-banging*, esto es, moviendo las señales por programa y temporizando adecuadamente. Se deberá hacer que cuando se quiera poner un 1 en **SDA** o **SCL** hay que poner como entrada el bit correspondiente. Cuando se quiera poner un cero hay que poner el cero y la patilla como salida.

Cronograma de transmisión (dirección de 7 bits)



3.8. Convertidor Analógico Digital

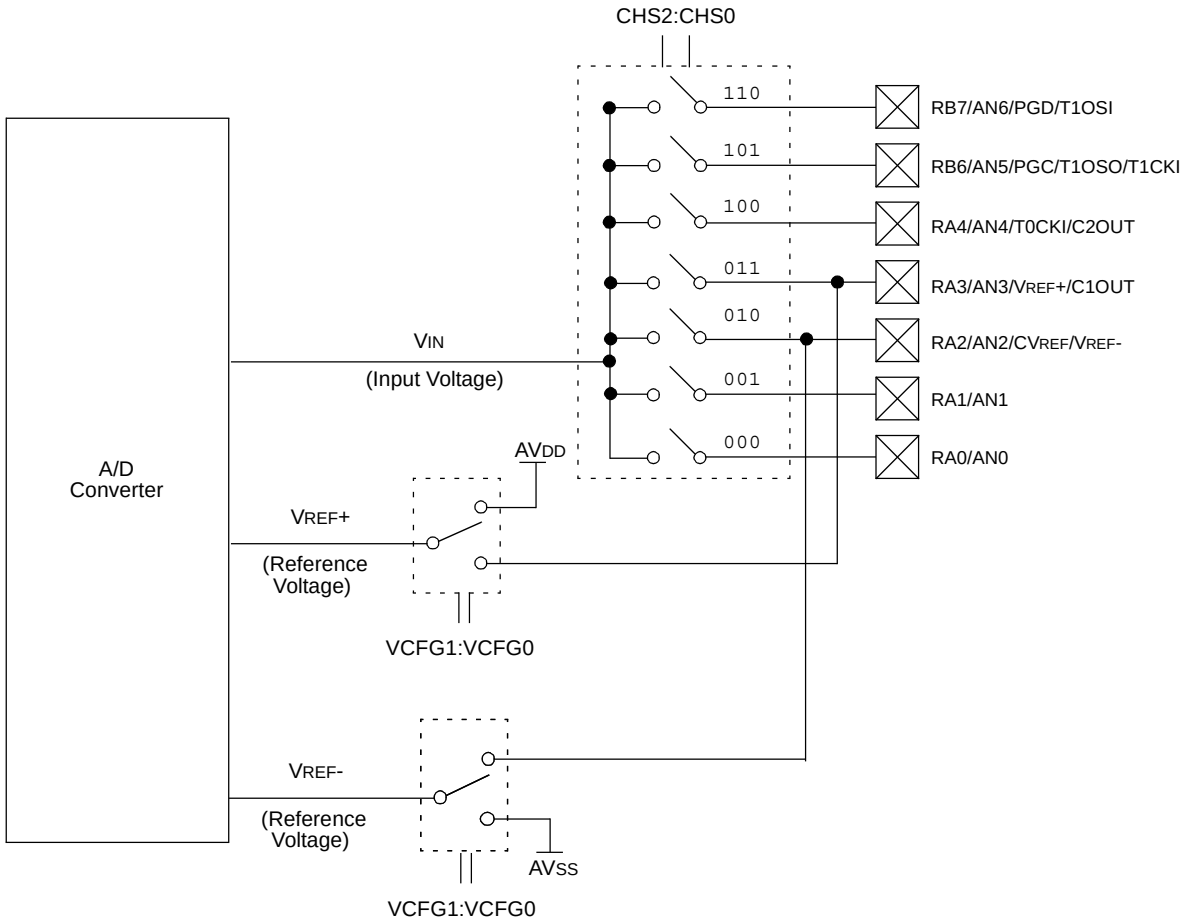
3.8.1. Introducción

El conversor analógico/digital presente en este microcontrolador presenta las siguientes características:

- Entradas analógicas multiplexadas. Internamente dispone de un único conversor A/D pero incorpora un multiplexor analógico con un número de entradas dependiente del modelo (solo está presente en el 16F88).
  - 16F88: tiene 7 entradas (patillas **RA0/AN0**,**RA1/AN1**,**RA2/AN2/CVref/Vref-**, **RA3/AN3/Vref+/C1OUT**,**RA4/AN4/T0CKI/C2OUT**,**RB6/AN5/PGC/T1OSO/T1CKI** y **RB7/AN6/PGD/T1OSI**).
- Conversor A/D de 10 bits.
- Referencias de tensión. Son seleccionables por software:
  - Tensión máxima (**VDD** ó **RA3/AN3/Vref+/C1OUT**).
  - Tensión mínima (**VSS** ó **RA2/AN2/CVref/Vref-**).
- Es capaz de funcionar en modo bajo consumo (SLEEP) gracias a un oscilador RC interno.

3.8.2. Esquema

El esquema hardware asociado junto con los campos que controlan su funcionamiento puede verse en la figura siguiente:

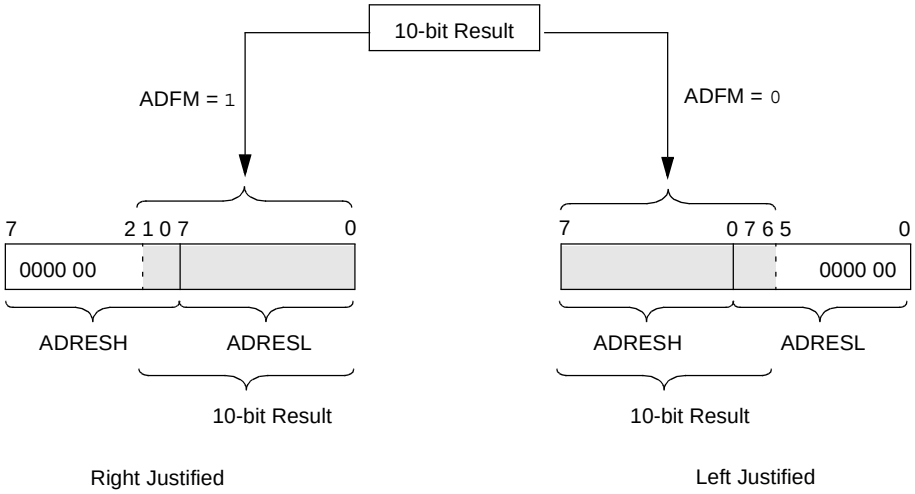


3.8.3. Registros asociados

Para el control del conversor estos microcontroladores disponen de cuatro registros, que son:

- **ADRESH**, banco 0 (*A/D Result High*). Es la parte alta de la conversión A/D.
- **ADRESL**, banco 1 (*A/D Result Low*). Es la parte baja de la conversión A/D.

  Estos dos registros contienen los 10 bits resultado de la conversión. La justificación a la izquierda o a la derecha se selecciona con el campo **ADCON1**.**ADFM** según se ve en la figura:



- **ANSEL**, banco 1 (*Analog Select Register*). Selecciona que patillas se comportarán como entradas analógicas.

ANSEL

Bit 7	6	5	4	3	2	1	Bit 0
-	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0

ANS6:ANS0 Bits de selección de las entradas analógicas.

0: Entrada/salida digital.
1: Entrada analógica.

- **ADCON0**, banco 0 (*A/D Control Register 0*) Registro de control AD número 0.

ADCON0

Bit 7	6	5	4	3	2	1	Bit 0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	-	ADON

ADCS1:ADCS0 Bits de selección del reloj para la conversión AD.

- Si **ADCON1.ADCS2**=0
  00: $F_{osc}/2$
  01: $F_{osc}/8$
  10: $F_{osc}/32$
  11: F_{RC}. Reloj derivado de un oscilador RC.
- Si **ADCON1.ADCS2**=1
  00: $F_{osc}/4$
  01: $F_{osc}/16$
  10: $F_{osc}/64$
  11: F_{RC}. Reloj derivado de un oscilador RC.

CHS2:CHS0 Bits de selección del canal analógico a convertir.

000: Canal 0, (**RA0/AN0**)
001: Canal 1, (**RA1/AN1**)
010: Canal 2, (**RA2/AN2**)
011: Canal 3, (**RA3/AN3**)
100: Canal 4, (**RA4/AN4**)
101: Canal 5, (**RB6/AN5**)

110: Canal 6, (**RB7/AN6**)

GO/DONE Estado de la conversión. Siempre que **ADON**=1:

0: Conversión finalizada. Se pone automáticamente a cero al terminar la conversión.
1: Conversión en curso. Poner a uno este bit inicia la conversión.

ADON Activación del conversor AD.

0: Módulo desconectado. No consume corriente.
1: Módulo operativo.

- **ADCON1**, banco 1 (*A/D Control Register 1*) Registro de control AD número 1.

ADCON1

Bit 7	6	5	4	3	2	1	Bit 0
ADFM	ADCS2	VCFG1	VCFG0	-	-	-	-

ADFM Formato de justificación del valor de 10 bis.

0: Justificación a la izquierda. Los 6 bits menos significativos de **ADRESL** valdrán 0.
1: Justificación a la derecha. Los 6 bits más significativos de **ADRESH** valdrán 0.

ADCS2 Divide por dos el reloj de conversión A/D cuando está activo.

0: Desactivado.
1: Activado.

VCFG1:0 Bits de configuración de la referencia de voltaje.

00: Vref+ = AVdd, Vref- = AVss.
01: Vref+ = AVdd, Vref- = Vref-.
10: Vref+ = Vref+, Vref- = AVss.
11: Vref+ = Vref+, Vref- = Vref-.

3.8.4. Configuración y adquisición de datos

Para la correcta configuración del módulo conversor AD es necesario que:

- Las patillas a utilizar estén configuradas como entradas. Para ello deberemos modificar los bits necesarios de los registros **TRISA** y **TRISB**.
- Definamos qué pines serán analógicos (**ANSEL**) y cuales digitales a la vez que debemos definir las tensiones de referencia a emplear (**ADCON1**).
- Debemos activar el módulo y seleccionar el canal de entrada (**ADCON0**).
- Si pretendemos emplear las interrupciones deberemos configurarlas como sigue:
  - Borrando **PIR1.ADIF**.
  - Activando **INTCON.GIE**, **INTCON.PEIE** y **PIE1.ADIE**.
- Ordenar la adquisición mediante **ADCON0**.GO_DONE = 1.
- Esperar a que se complete la adquisición que tarda $10 * T_{AD}$ (el tiempo de conversion de cada bit depende de la frecuencia de reloj de conversion programada y de la frecuencia de funcionamiento del microcontrolador).
  - O bien, comprobando si **ADCON0**.GO_DONE = 0

  - O bien, esperando la notificación de finalización con **PIR1.ADIF**.

- Finalmente tendremos que leer el resultado de la pareja **ADRESH:ADRESL**.
- Y borrar **PIR1.ADIF** si fuera necesario.
- Antes de una nueva adquisición es preciso esperar un mínimo de $2 * T_{AD}$.

Además habrá que tener en cuenta que:

- La impedancia máxima de salida de la fuente analógica será de $10k\Omega$.
- La selección del reloj de conversión (**ADCON0.ADCS1:ADCS0**). Se requiere un mínimo de 1.6 μseg.

AD Clock Source (TAD)			Maximum Device Frequency
Operation	**ADCS<2>**	**ADCS<1:0>**	**Max.**
2 TOSC	0	00	1.25 MHz
4 TOSC	1	00	2.5 MHz
8 TOSC	0	01	5 MHz
16 TOSC	1	01	10 MHz
32 TOSC	0	10	20 MHz
64 TOSC	1	10	20 MHz
RC(1,2,3)	X	11	**(Note 1)**

Note 1: The RC source has a typical TAD time of 4 µs, but can vary between 2-6 µs.

- Es capaz de funcionar en modo SLEEP si la fuente de reloj usada es RC. Esto evita el ruido de conmutación digital. Se despertará al terminar la conversión (interrupciones activadas).

Resumen de registros

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
0Bh, 8Bh 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
1Eh	ADRESH(1)	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu
9Eh	ADRESL(1)	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
1Fh	ADCON0(1)	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON	0000 00-0	0000 00-0
9Fh	ADCON1(1)	ADFM	ADCS2	VCFG1	VCFG0	—	—	—	—	0000 ----	0000 ----
9Bh	ANSEL(1)	—	AN6	AN5	AN4	AN3	AN2	AN1	AN0	-111 1111	-111 1111
05h	PORTA (PIC16F87) (PIC16F88)	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxx 0000 xxx0 0000	uuuu 0000 uuu0 0000
05h, 106h	PORTB (PIC16F87) (PIC16F88)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx 00xx xxxx	uuuu uuuu 00uu uuuu
85h	TRISA	TRISA7	TRISA6	TRISA5(2)	PORTA Data Direction Register					1111 1111	1111 1111
86h, 186h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111

Ejemplo

En lenguaje C:

Listado 3.9: Codigo/PIC16F88_Ejemplos/AD.c

```
#include <pic.h>

void main()
{
    TRISA = 0xFF;    // Todo entradas
    TRISB = 0x00;    // Todo salidas
    ANSEL = 0x7F;    // Todo entradas analogicas
    ADCON1= 0x00;    // Justificado a la izquierda. Referencias = alim.

    ADCON0 =0xC1;    // 11000001b Configura Conversor AD
                     // ADCS[1:0] = 11  Oscilador interno RC
                     //                   de 2 a 6 useg.
                     // CHS[2:0]  = 000 Canal 0 RA0/AN0
                     // GO/DONE   = 0   Fin conversion
                     //           = 0   No usado
                     // ADON      = 1   Conversor AD habilitado
    while(1)
    {
      ADIF = 0;          // Borro indicador de fin de conversion
      ADGO = 1;          // Inicia la conversion
                         //  (!!nombre, es diferente al ensambl.)

      while(ADIF == 0); // Espera a convertir

      PORTB = ADRESL;
    };
}
```

3.9. Comparador

3.9.1. Introducción

El módulo comparador incluye dos comparadores analógicos. Las entradas a estos comparadores están multiplexadas con las patillas **RA0**, **RA1**, **RA2** y **RA3** y las salidas están multiplexadas con las patillas **RA3** y **RA4**. La referencia de voltaje integrada también puede ser entrada de los comparadores. La máxima impedancia de la fuente conectada al comparador deberá ser de 10 kΩ.

3.9.2. Registros

El registro **CMCON** controla las entradas y salidas de los comparadores:

CMCON

Bit 7	6	5	4	3	2	1	Bit 0
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0

C2OUT Bit de salida del comparador 2.

- Si C2INV=0
  - 0: C2 V_{in+} <C2 V_{in-}.
  - 1: C2 V_{in+} >C2 V_{in-}.
- Si C2INV=1
  - 0: C2 V_{in+} >C2 V_{in-}.
  - 1: C2 V_{in+} <C2 V_{in-}.

C1OUT Bit de salida del comparador 1.

- Si C2INV=0
  - 0: C1 V_{in+} <C1 V_{in-}.
  - 1: C1 V_{in+} >C1 V_{in-}.
- Si C1INV=1
  - 0: C1 V_{in+} >C1 V_{in-}.
  - 1: C1 V_{in+} <C1 V_{in-}.

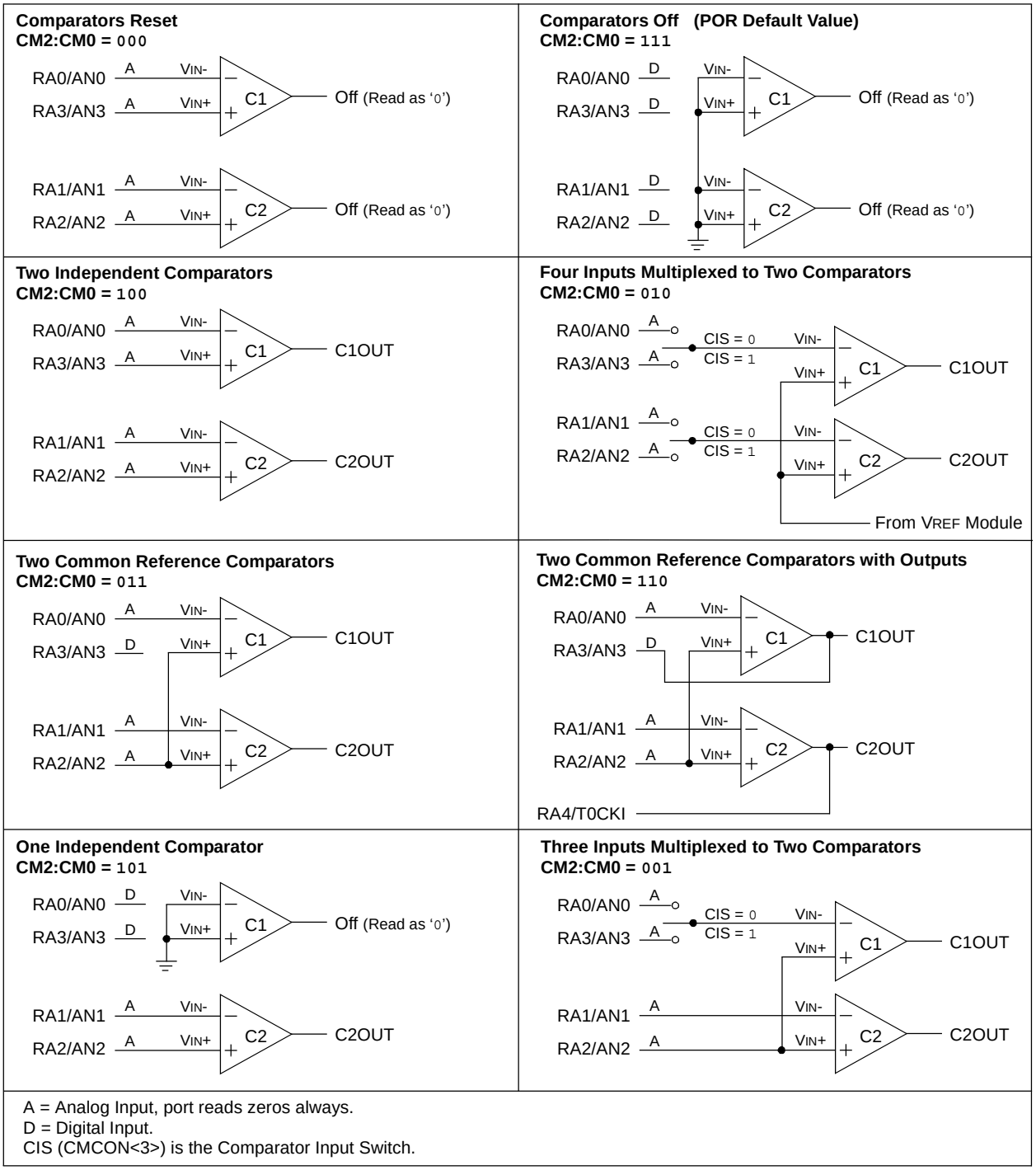
C2INV Invierte la salida del comparador 2.

C1INV Invierte la salida del comparador 1.

CIS *Comparator input switch.*

- Si CM2..CM0=001
  - 0: C1 V_{in-} se conecta con **RA3**
  - 1: C1 V_{in-} se conecta con **RA0**.
- Si CM2..CM0=010
  - 0: C1 V_{in-} se conecta con **RA0** y C2 V_{in-} se conecta con **RA1**
  - 1: C1 V_{in-} se conecta con **RA3** y C2 V_{in-} se conecta con **RA2**

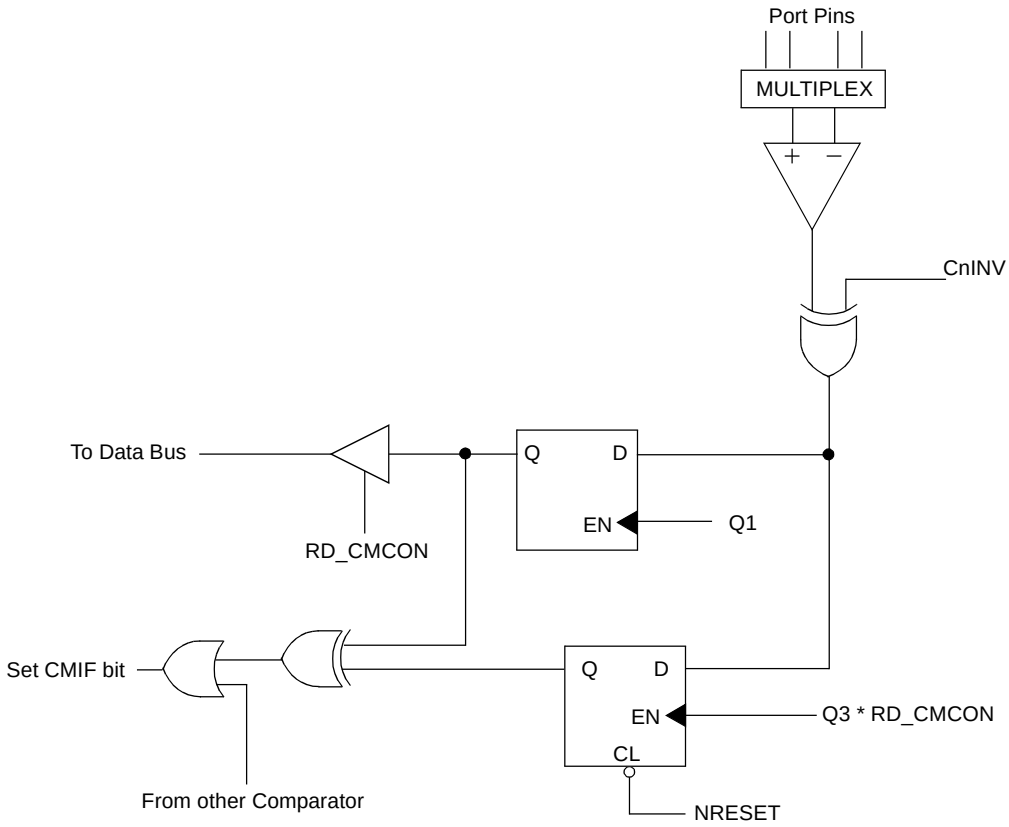
CM2:0 Modo de funcionamiento del comparador:



A = Analog Input, port reads zeros always.
D = Digital Input.
CIS (CMCON<3>) is the Comparator Input Switch.

Interrupciones

Es posible generar interrupciones cuando cambia el estado de un comparador:



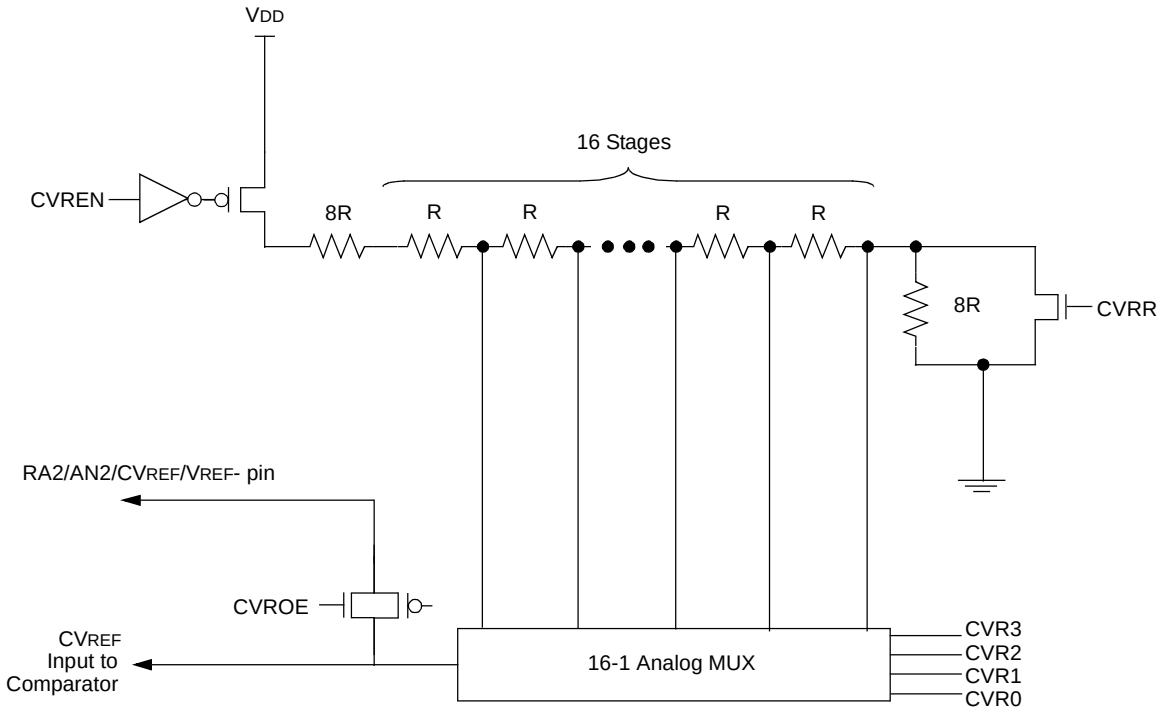
Para ello deberemos activar **GIE**=1, **PEIE**=1, **CMIE**=1.

Resumen de registros

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other RESETS
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	0000 0111
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	000- 0000
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTIE	RBIE	TMR0IF	INTIF	RBIF	0000 000x	0000 000u
0Dh	PIR2	OSFIF	CMIF	—	EEIF	—	—	—	—	00-0 ----	00-0 ----
8Dh	PIE2	OSFIE	CMIE	—	EEIE	—	—	—	—	00-0 ----	00-0 ----
05h	PORTA (PIC16F87) (PIC16F88)	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxx 0000 xxx0 0000	uuuu 0000 uuu0 0000
85h	TRISA	TRISA7	TRISA6	TRISA5(1)	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	1111 1111

3.9.3. Referencia interna de voltaje

El generador interno de la referencia de voltaje es una red de resistencias que proporciona un voltaje fijo:



Para controlarlo se usa el registro **CVRCON**

CVRCON

Bit 7	6	5	4	3	2	1	Bit 0
CVREN	CVROE	CVRR	-	CVR3	CVR2	CVR1	CVR0

CVREN Habilita la referencia de voltaje para el comparador.

0: Desactiva.
1: Activa.

CVROE Habilita la salida externa de la referencia de voltaje.

0: Desactiva.
1: Activa. La salida es por el pin **RA2**.

CVRR Selección del rango para la referencia de voltaje.

0: Entre $(0{,}00V_{dd} \longrightarrow 0{,}625V_{dd})$ con pasos de $V_{dd}/24$.
1: Entre $(0{,}25V_{dd} \longrightarrow 0{,}720V_{dd})$ con pasos de $V_{dd}/32$.

CVR3..CVR0 Valor de la referencia de voltaje.

- Si **CVRR**=0. Referencia $= (CVR/24) * V_{dd}$
- Si **CVRR**=1. Referencia $= 0{,}25 * V_{dd} + (CVR/32) * V_{dd}$

Resumen de registros

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other RESETS
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	000- 0000
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	0000 0111

3.10. Otras características

Existen otras características dentro de esta familia de microcontroladores que resumimos a continuación y que estudiaremos en este capítulo:

- Palabra de configuración del microcontrolador.
- *Reset*. Mecanismos de reinicio:
  - *Power-on reset* (POR). Al conectar la alimentación del circuito.
  - *Power-up timer* (PWRT). Durante un tiempo al principio.
  - *Oscillator start-up timer* (OST). Hasta que el oscilador oscile nominalmente.
  - *Brown-out reset* (BOR). Si la tensión de alimentación cae.
- Interrupciones.
- *Watchdog timer* (WDT). Temporizador perro guardián.
- Modo de bajo consumo: instrucción `SLEEP`.
- Protección del código.
- Identificación.
- Programación serie en el sistema (ICSP: *In Circuit Serial Programming*).
- Programación serie en el sistema de bajo voltaje (LVP: *Low Voltage Programming*)

3.10.1. Configuración del microcontrolador

Algunas de estas prestaciones se configurarán en tiempo de grabación del microcontrolador a través de las palabras de configuración tomando el significado que aparece a continuación. Estas palabras de configuración se encuentran en el espacio de memoria de configuración.

En la dirección `2007h`:

R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
CP	CCPMX	DEBUG	WRT1	WRT0	CPD	LVP	BOREN	MCLRE	FOSC2	$\overline{\text{PWRTEN}}$	WDTEN	FOSC1	FOSC0
bit 13													bit 0

bit 13 **CP:** Flash Program Memory Code Protection bits
1 = Code protection off
0 = 0000h to 0FFFh code-protected (all protected)

bit 12 **CCPMX:** CCP1 Pin Selection bit
1 = CCP1 function on RB0
0 = CCP1 function on RB3

bit 11 **DEBUG:** In-Circuit Debugger Mode bit
1 = In-Circuit Debugger disabled, RB6 and RB7 are general purpose I/O pins
0 = In-Circuit Debugger enabled, RB6 and RB7 are dedicated to the debugger

bit 10-9 **WRT<1:0>:** Flash Program Memory Write Enable bits
11 = Write protection off
10 = 0000h to 00FFh write-protected, 0100h to 0FFFh may be modified by EECON control
01 = 0000h to 07FFh write-protected, 0800h to 0FFFh may be modified by EECON control
00 = 0000h to 0FFFh write-protected

bit 8 **CPD:** Data EE Memory Code Protection bit
1 = Code protection off
0 = Data EE memory code-protected

bit 7 **LVP:** Low-Voltage Programming Enable bit
1 = RB3/PGM pin has PGM function, Low-Voltage Programming enabled
0 = RB3 is digital I/O, HV on $\overline{\text{MCLR}}$ must be used for programming

bit 6 **BOREN:** Brown-out Reset Enable bit
1 = BOR enabled
0 = BOR disabled

bit 5 **MCLRE:** RA5/$\overline{\text{MCLR}}$/V_{PP} Pin Function Select bit
1 = RA5/$\overline{\text{MCLR}}$/V_{PP} pin function is $\overline{\text{MCLR}}$
0 = RA5/$\overline{\text{MCLR}}$/V_{PP} pin function is digital I/O, $\overline{\text{MCLR}}$ internally tied to V_{DD}

bit 3 **$\overline{\text{PWRTEN}}$:** Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

bit 2 **WDTEN:** Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 4, 1-0 **FOSC<2:0>:** Oscillator Selection bits
111 = EXTRC oscillator; CLKO function on RA6/OSC2/CLKO
110 = EXTRC oscillator; port I/O function on RA6/OSC2/CLKO
101 = INTRC oscillator; CLKO function on RA6/OSC2/CLKO pin and port I/O function on RA7/OSC1/CLKI pin
100 = INTRC oscillator; port I/O function on both RA6/OSC2/CLKO pin and RA7/OSC1/CLKI pin
011 = ECIO; port I/O function on RA6/OSC2/CLKO
010 = HS oscillator
001 = XT oscillator
000 = LP oscillator

En la dirección 2008h:

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1	R/P-1	R/P-1
—	—	—	—	—	—	—	—	—	—	—	—	IESO	FCMEN
bit 13													bit 0

bit 13-2 **Unimplemented:** Read as '1'

bit 1 **IESO:** Internal External Switchover bit
1 = Internal External Switchover mode enabled
0 = Internal External Switchover mode disabled

bit 0 **FCMEN:** Fail-Safe Clock Monitor Enable bit
1 = Fail-Safe Clock Monitor enabled
0 = Fail-Safe Clock Monitor disabled

Para definirla desde el propio código fuente en ensamblador haremos:

Listado 3.10:

```
    LIST P=16F88, R=DEC
    INCLUDE "P16F88.INC"    ; Despues de esta inclusion ...

    ;Program Configuration Register 1
    ;(Ojo: todo escrito en la misma linea!)
    __CONFIG    _CONFIG1, _CP_OFF & _CCP1_RB0 & _DEBUG_OFF
                & _WRT_PROTECT_OFF & _CPD_OFF & _LVP_OFF
                & _BODEN_OFF & _MCLR_ON & _PWRTE_OFF
                & _WDT_OFF & _XT_OSC

    ;Program Configuration Register 2
    __CONFIG    _CONFIG2, _IESO_OFF & _FCMEN_OFF
```

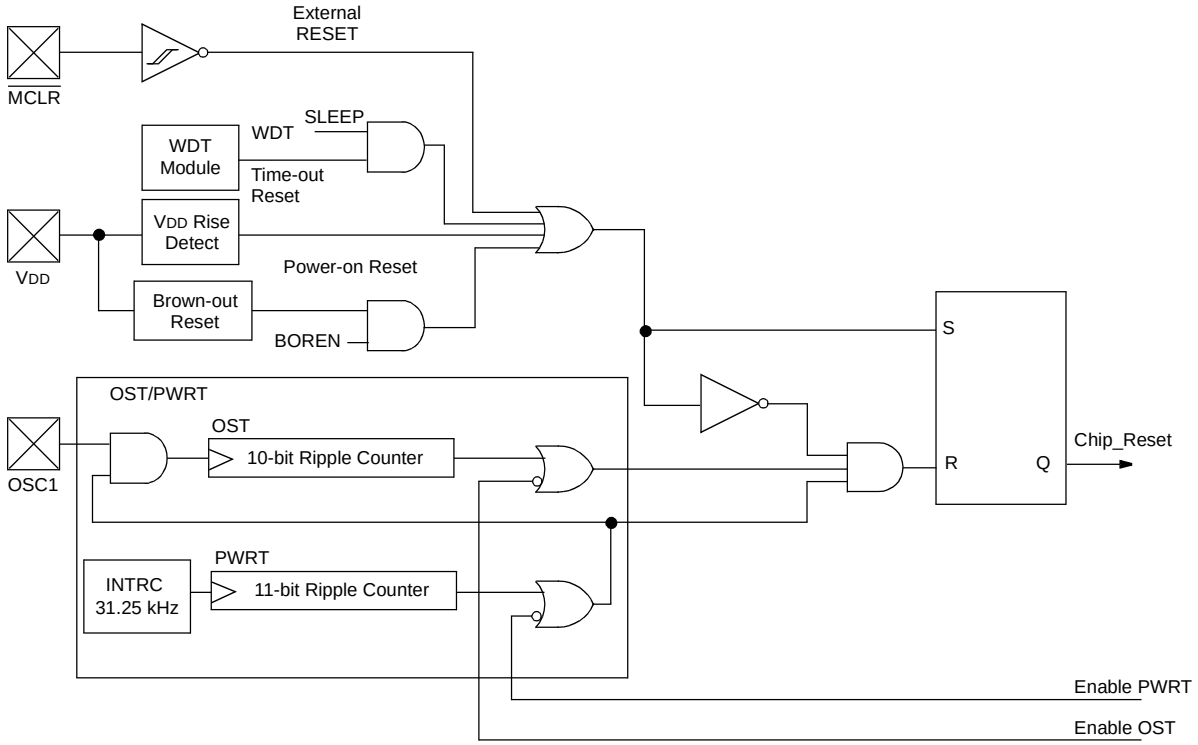
Para definirla desde el propio código fuente en lenguaje C haremos:

Listado 3.11:

```
#include <pic.h>

// Para el PIC16F88
    __CONFIG(WDTDIS & XT & UNPROTECT & PWRTDIS & CCPRB0
             & DEBUGDIS & LVPDIS & BORDIS & MCLREN
             & FCMDIS & IESODIS); // PIC16F88
```

3.10.2. *Reset*

La señal de *reset* que recibe el núcleo del microcontrolador procede de diversas fuentes como se puede apreciar en el esquema siguiente:



Las diferentes fuentes de *reset* se enumeran a continuación:

- *Power-On Reset* (POR). *Reset* generado mientras que V_{dd} esté entre 1,2 y 1,7 V. Evita el uso de redes RC en **RA5//MCLR/VPP**(una resistencia de 10 $k\Omega$ es suficiente).
- *PoWeR-up Timer* (PWRT). Proporciona 72 milisegundos añadidos de *reset* después del POR. Esta temporización depende de la temperatura, la tensión de alimentación y puede variar según el lote de fabricación del chip. Se activa en la palabra de configuración.
- *Oscillator Start-up Timer* (OST). Proporciona 1024 ciclos de reloj extra de señal de *reset* después de PWRT para asegurar que el oscilador funcione correctamente. Se activa automáticamente si se selecciona XT, LP o HS como modos de oscilación.
- *Brown-Out Reset* (BOR). Se activa con el bit de configuración BODEN. Si V_{dd} cae por debajo de 4 V durante más de 100 microsegundos se reinicia el sistema, que no comenzará a ejecutar instrucciones hasta que V_{dd} >4 V y después del PWRT correspondiente.

Resumen de los tiempos de *reset*

Oscillator Configuration	Power-up		Brown-out Reset		Wake-up from Sleep
	$\overline{\text{PWRTE}}$ = 0	$\overline{\text{PWRTE}}$ = 1	$\overline{\text{PWRTE}}$ = 0	$\overline{\text{PWRTE}}$ = 1	
XT, HS, LP	TPWRT + 1024 • TOSC	1024 • TOSC	TPWRT + 1024 • TOSC	1024 • TOSC	1024 • TOSC
EXTRC, INTRC	TPWRT	5-10 µs(1)	TPWRT	5-10 µs(1)	5-10 µs(1)
T1OSC	—	—	—	—	5-10 µs(1)

Note 1: CPU start-up is always invoked on POR, BOR and wake-up from Sleep. The 5-10 µs delay is based on a 1 MHz system clock.

Bits de estado STATUS.TO y STATUS.PD y PCON.POR y PCON.BOR después del *reset*

$\overline{\text{POR}}$	$\overline{\text{BOR}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	
0	x	1	1	Power-on Reset
0	x	0	x	Illegal, $\overline{\text{TO}}$ is set on $\overline{\text{POR}}$
0	x	x	0	Illegal, $\overline{\text{PD}}$ is set on $\overline{\text{POR}}$
1	0	1	1	Brown-out Reset
1	1	0	1	WDT Reset
1	1	0	0	WDT Wake-up
1	1	u	u	$\overline{\text{MCLR}}$ Reset during normal operation
1	1	1	0	$\overline{\text{MCLR}}$ Reset during SLEEP or interrupt wake-up from SLEEP

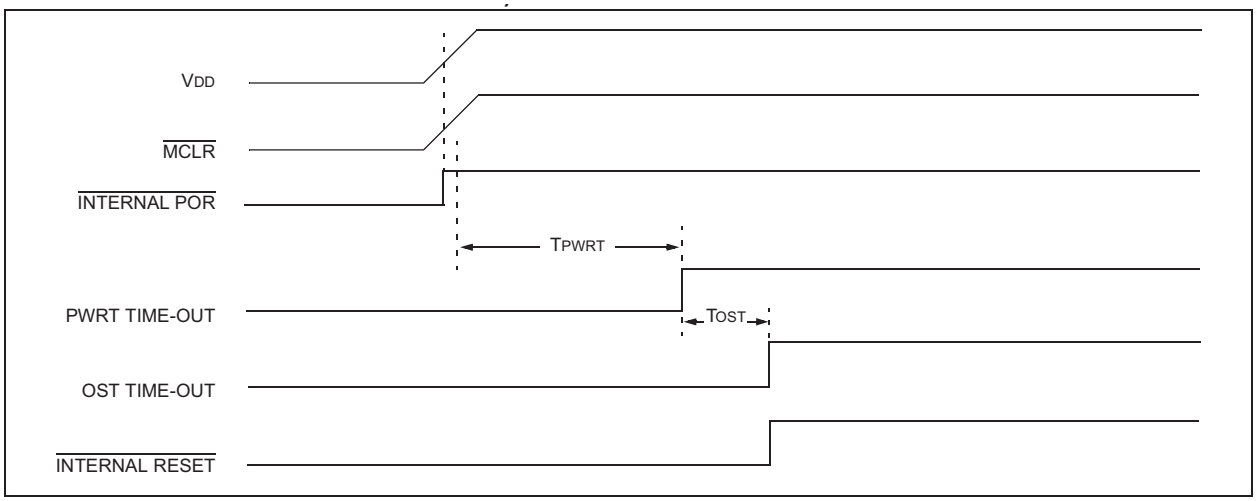
Estado inicial después del *reset*

Condition	Program Counter	STATUS Register	PCON Register
Power-on Reset	000h	0001 1xxx	---- --0x
$\overline{\text{MCLR}}$ Reset during normal operation	000h	000u uuuu	---- --uu
$\overline{\text{MCLR}}$ Reset during Sleep	000h	0001 0uuu	---- --uu
WDT Reset	000h	0000 1uuu	---- --uu
WDT Wake-up	PC + 1	uuu0 0uuu	---- --uu
Brown-out Reset	000h	0001 1uuu	---- --u0
Interrupt Wake-up from Sleep	PC + 1(1)	uuu1 0uuu	---- --uu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0'

Note 1: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

Cronograma típico de arranque



3.10.3. Configuraciones del oscilador

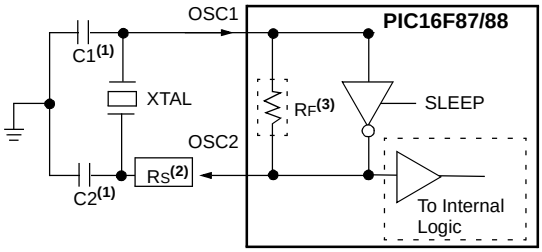
Características

El microcontrolador PIC16F87/88 dispone de ocho modos de funcionamiento del oscilador seleccionables en la palabra de configuración:

- **LP**. Cristal o resonador. Modo de bajo consumo.
- **XT**. Cristal o resonador. Frecuencias intermedias.
- **HS**. Cristal o resonador. Alta velocidad.
- **RC**. Red RC externa. Por **RA6** sale la frecuencia de instrucción.
- **RCIO**. Red RC externa. **RA6** para entrada/salida.
- **INTIO1**. Oscilador interno. Por **RA6** sale la frecuencia de instrucción. **RA7** para propósito general.
- **INTIO2**. Oscilador interno. **RA6** y **RA7** para propósito general.
- **ECIO**. Reloj externo con **RA6** para propósito general.

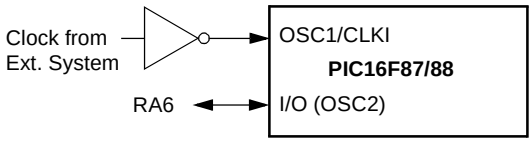
Cristales o resonadores

Se coloca entre las patillas **OSC1/CLKI** y **OSC2/CLKO**.



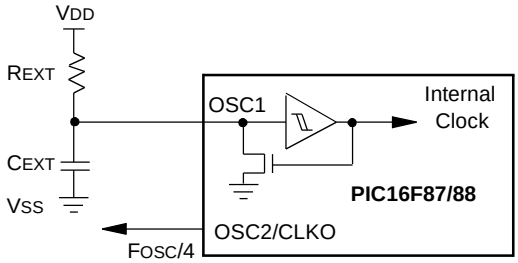
Reloj externo

La configuración se puede ver en la figura.

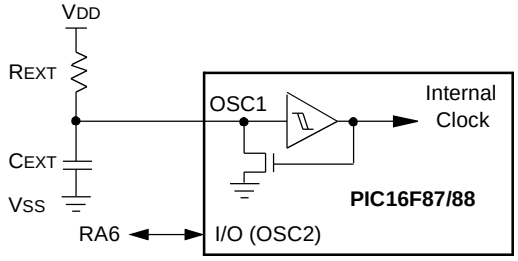


Oscilador RC

Podemos ver en las figuras los dos modos de operación:



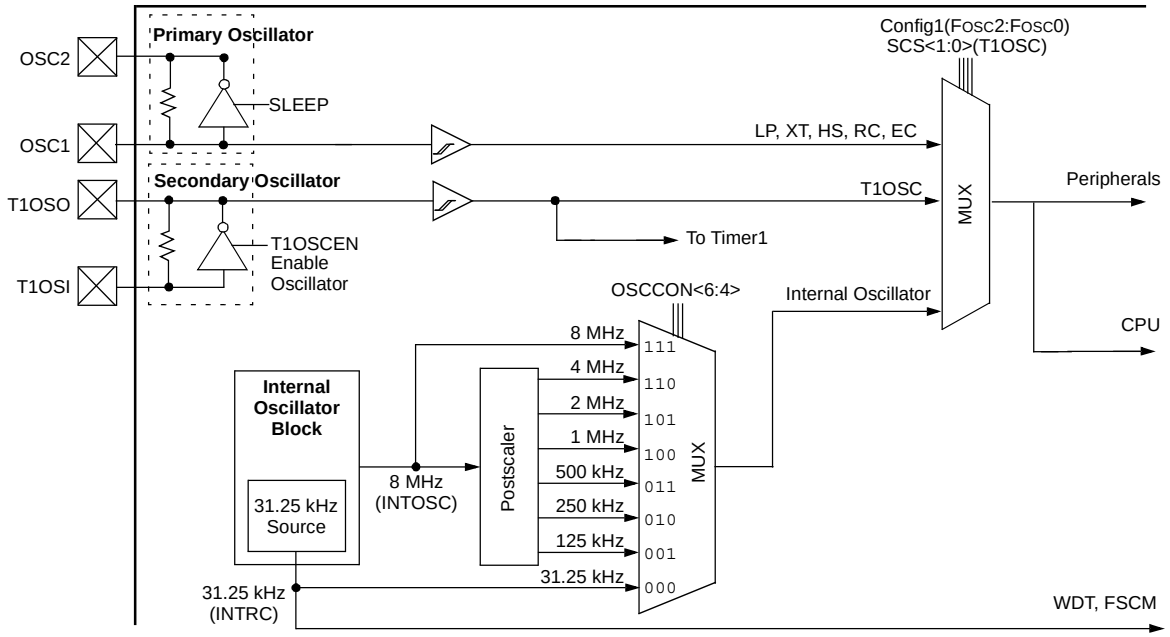
Recommended values: 3 kΩ ≤ REXT ≤ 100 kΩ
CEXT > 20 pF



Recommended values: 3 kΩ ≤ REXT ≤ 100 kΩ
CEXT > 20 pF

Oscilador interno

Este microcontrolador dispone de dos fuentes de reloj internas. Una de 8 MHz que se puede dividir hasta 6 veces (125 kHz) y otra de 31.25 kHz. En la siguiente figura se puede ver un esquema.



Para seleccionar la fuente de reloj interno se emplea el registro **OSCCON**

OSCCON. Presente en el banco 1. Registro de sintonización del oscilador interno.

OSCCON

Bit 7	6	5	4	3	2	1	Bit 0
-	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0

IRCF2:0 Bits de selección de la frecuencia del oscilador RC interno.

Bits	Frecuencia
000	31.25 kHz
001	125 kHz
010	250 kHz
011	500 kHz
100	1 MHz
101	2 MHz
110	4 MHz
111	8 MHz

OSTS Notificación de entrada en funcionamiento del oscilador secundario interno.

0: Funciona el oscilador secundario.

1: Funciona el oscilador primario.

IOFS Notificación de frecuencia estable.

0: No estable.

1: Estable.

SCS1:0 Selección del modo del oscilador.

Bits	Frecuencia
00	Oscilador definido por FOSC2:0
01	T1OSC como oscilador
10	Oscilador RC interno
11	Reservado

Calibración del oscilador interno

El oscilador interno tiene la posibilidad de ser calibrado mediante el registro **OSCTUNE** que permite variaciones de ±12,5 %. El circuito viene calibrado de fábrica pero se puede ajustar por software.

OSCTUNE. Presente en el banco 1. Registro de sintonización del oscilador interno.

OSCTUNE

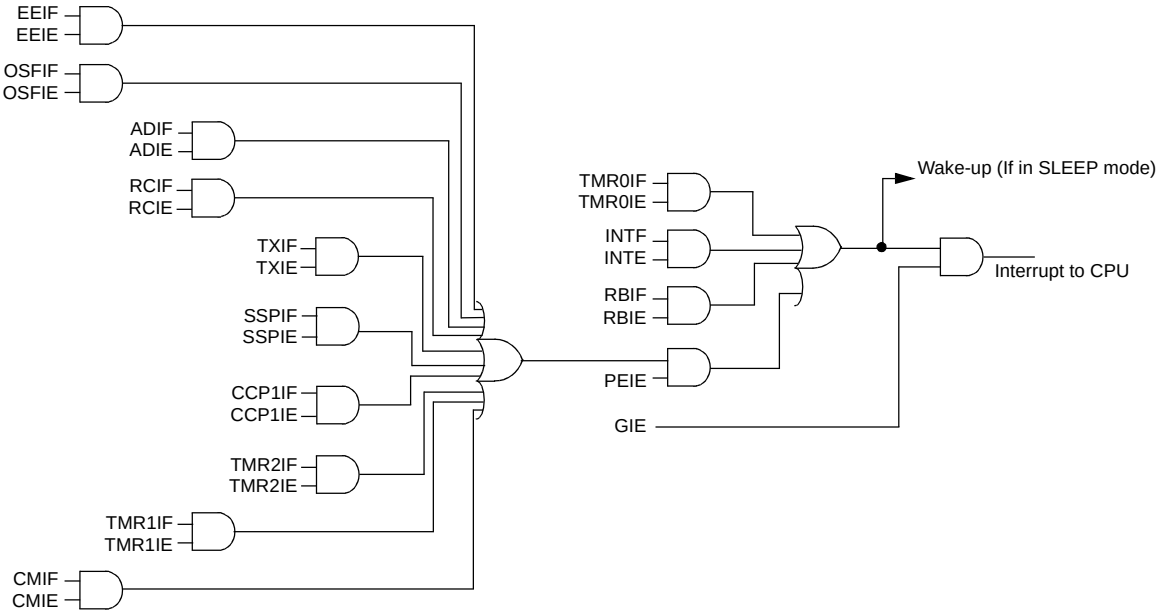
Bit 7	6	5	4	3	2	1	Bit 0
-	-	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0

TUN5:0 Bits de ajuste de la frecuencia.

Bits	Frecuencia
01111	Máxima frecuencia
01110	
01101	
...	
00001	Frecuencia central
11111	
11110	
....	
10000	Mínima frecuencia

3.10.4. Interrupciones

Estos microcontroladores dispone de 12 fuentes de interrupción. Se muestra a continuación el esquema *hardware* de generación de interrupciones en el que se pueden apreciar los bits de habilitación (terminados con la letra E) y los bits de notificación (terminados con la letra F).



Descripción

- **INTCON.GIE** habilita globalmente las interrupciones.
- Al producirse una interrupción la ejecución del programa salta a la posición `4` de la memoria de programa y se inhiben globalmente las interrupciones hasta que es ejecutada la instrucción de retorno de interrupción: `RETFIE`.
- Independientemente de que estén habilitadas global o localmente las interrupciones, los flags que notifican la ocurrencia de la condición se ponen a 1. Es necesario que el programador los vuelva a poner a 0 (la mayoría son bits pegajosos: *sticky bits*). Después de atender una interrupción es necesario borrarlos para evitar que se vuelva a producir la misma interrupción.
- El registro **INTCON** almacena las habilitaciones y señalizaciones de las interrupciones debidas a **RB0/INT**, cambio en el puerto **RB4**..**RB7** y desbordamiento del temporizador TMR0.
- Los demás flags se encuentran en los registros **PIR1** y **PIR2**. Los bits de habilitación están en los registros **PIE1** y **PIE2**. Para habilitar estos hay que activar el bit **INTCON.PEIE**.
- Para las interrupciones externas puede haber un retardo en su atención de 3 o 4 ciclos de instrucción por un tema de sincronización.

- Es necesario salvaguardar el contexto de ejecución.

  Las variables `W_TEMP, STATUS_TEMP y PCLATH_TEMP` deben estar definidas en los 16 bytes últimos de los bancos (ya que son comunes a los 4 bancos y facilita el proceso). Véase el ejemplo:

Listado 3.12:

```
W_TEMP       equ 127
STATUS_TEMP  equ 126
PCLATH_TEMP  equ 125

    MOVWF    W_TEMP           ; W_TEMP = W
    SWAPF    STATUS,W         ; W = swap(STATUS)
    CLRF     STATUS           ; Banco 0 al borrar IRP, RP1 y RP0
    MOVWF    STATUS_TEMP      ; STATUS_TEMP = W = swap(STATUS)
    MOVF     PCLATH, W        ; Solo requerido si usamos las
                              ;  paginas de codigo 1, 2 o 3
    MOVWF    PCLATH_TEMP      ; PCLATH_TEMP = PCLATH
    CLRF     PCLATH           ; Pone pagina 0 de codigo
    :
    :(ISR)                    ; Atencion de la interrupcion
    :
    MOVF     PCLATH_TEMP, W   ; Recupera PCLATH
    MOVWF    PCLATH
    SWAPF    STATUS_TEMP,W    ; Recupera STATUS
    MOVWF    STATUS
    SWAPF    W_TEMP,F         ; Recupera W
    SWAPF    W_TEMP,W
```

En lenguaje C se pondrá:

Listado 3.13:

```
void interrupt ISR(void)
{
 // ...
}
```

donde no habrá que preocuparse de salvaguardar el contexto porque lo hará el propio C y donde debemos procurar: que ninguna función llame a la subrutina de atención a las interrupciones (ISR: *Interrupt Service Routine*, el nombre es opcional), y que desde la ISR no se llame a otras funciones que sean llamadas a su vez desde otros sitios. No existe una pila dinámica para guardar los valores de las variables.

3.10.5. Temporizador perro guardián

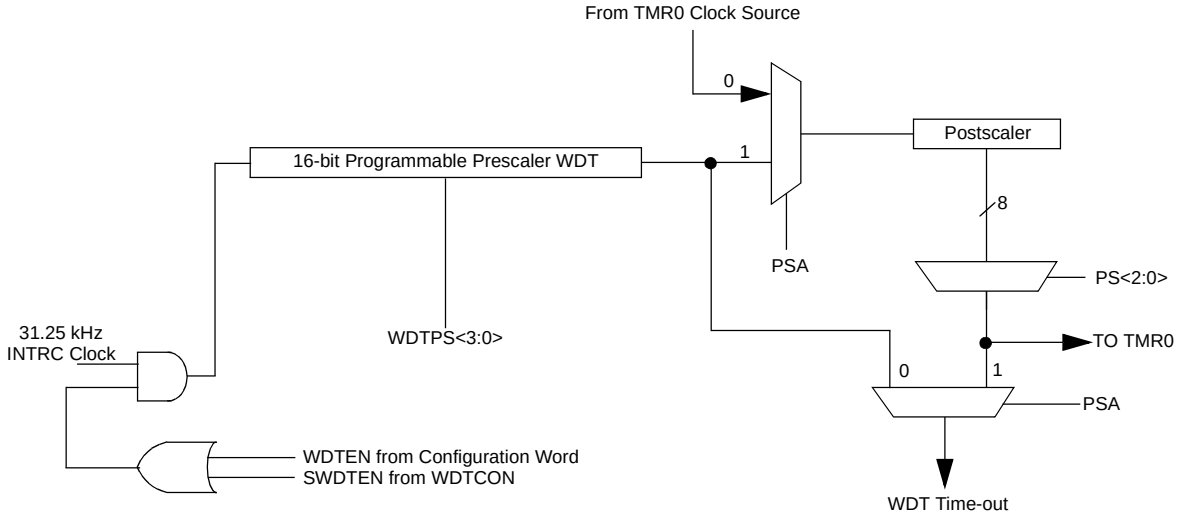
Es un temporizador cuyo oscilador RC se encuentra integrado en el microcontrolador. El perro guardián provoca un *reset* del sistema cuando se desborda. Si el micro se encuentra en modo de bajo consumo (instrucción `SLEEP`) solamente lo despierta. El bit `STATUS.NOT_TO = 0` señalizará el desbordamiento de este temporizador.

Su función es la de que el sistema no entre en un estado descontrolado: el programador deberá, si ha activado el WDT, ponerlo a cero (instrucción `CLRWDT`) con una cierta frecuencia dentro de su programa. Si debido a un mal funcionamiento *hardware* o *software* la ejecución de instrucciones no borra el WDT, el sistema se reiniciará.

El tiempo nominal de desbordamiento es dependiente de la temperatura, la tensión de alimentación y varía con el *chip*. Este micro incorpora una nueva arquitectura para el perro guardián con respecto a otros micros de la misma familia. Permite usar el oscilador interno RC que tiene una frecuencia base de 31.25 kHz y una pre-escala programable de 16 bits, que nos va a permitir elegir el periodo de desbordamiento del perro guardián entre 16.38 ms y 2.097 segundos.

Si además usamos la pre-escala compartida con el temporizador 0 podemos multiplicar hasta por 128 el periodo de desbordamiento del perro guardián: $2{,}096 * 128 = 268{,}416$ segundos.

Esquema hardware



Registros asociados

El control del periodo se hace mediante el registro **WDTCON**

WDTCON. Presente en el banco 2. Registro de control del perro guardián.

WDTCON

Bit 7	6	5	4	3	2	1	Bit 0
-	-	-	WDTPS3	WDTPS2	WDTPS1	WDTPS0	SWDTEN

WDTPS3:0 Valor de la preescala.

Bits	Preescala
0000	1:32
0001	1:64
0010	1:128
0011	1:256
0100	1:512
0101	1:1024
0110	1:2048
0111	1:4096
1000	1:8192
1001	1:16384
1010	1:32768
1011	1:65536
...	reservadas

SWDTEN Habilitación software del perro guardián. Si el perro está activado mediante los bits de configuración este bit no tiene uso. En caso de estar desactivado, es posible, mediante este bit activarlo por software.

0: WDT apagado.

1: WDT en marcha.

Resumen de registros

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
81h,181h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
2007h	Configuration bits	LVP	BOREN	MVCLRE	Fosc2	$\overline{\text{PWRTEN}}$	WDTEN	Fosc1	Fosc0
105h	WDTCON	—	—	—	WDTPS3	$\overline{\text{WDTPS2}}$	WSTPS1	WDTPS0	SWDTEN

3.10.6. Modo de bajo consumo: SLEEP

- Se entra en este modo ejecutando la instrucción `SLEEP`.
- Entonces `STATUS.NOT_PD = 0` y se inicia la cuenta del WDT, que despertará al micro al desbordarse este, siempre que esté activo (bits de configuración)
- En este modo el reloj del sistema deja de oscilar, aunque las salidas se mantendrán en el mismo estado.
- El sistema se puede despertar de las siguientes maneras:
  - Activación del reset externo **RA5//MCLR/VPP**.
  - Desbordamiento del WDT.
  - Interrupción externa (**RB0** o cambio en **RB4**..**RB7**).
  - Interrupciones de otros periféricos:
    - Interrupción de TMR1 (contador asíncrono).
    - Interrupción en modo captura (CCP).
    - Disparo de un evento especial (TMR1 con reloj externo).
    - Interrupción por detección *start* o *stop* en el módulo SSP.
    - Transmisión/recepción en modo esclavo SSP (SPI/I2C).
    - Transmisión/recepción de la USART en modo esclavo.
    - Finalización de la conversión A/D.
    - Finalización escritura en la EEPROM de datos.
    - La salida del comparador cambia de estado.

3.10.7. Protección del código

Para proteger la propiedad intelectual de nuestro trabajo se puede proteger contra lecturas el acceso tanto a la memoria de programa como a la memoria de datos EEPROM de nuestro microcontrolador. Esto se hace a través de la palabra de configuración vista en una sección anterior.

3.10.8. Identificación

Dispone de 4 posiciones de la memoria de configuración (2000h-2003h) para identificar el sistema. Solamente emplea 4 bits por dirección. Tendremos por tanto 4 dígitos hexadecimales. Se puede almacenar información como el *checksum* del programa grabado (es un código para comprobar si el programa se modificó) ó se puede almacenar un código que haga referencia a la versión del programa grabado, etc.

Para definirlo desde el propio código fuente:

Listado 3.14:

```
    LIST P=16F88, R=DEC
    INCLUDE "P16F88.INC"    ; Despues de esta inclusion ...

    __IDLOCS(1234h)
```

Para definirla desde el propio código fuente en lenguaje C haremos:

Listado 3.15:

```
1 #include <pic.h>

  __IDLOC(0x1234);
```

3.10.9. Programación serie en el sistema (ICSP)

Esta prestación nos permite grabar el programa en el microcontrolador. Se necesita un *hardware* especial y un *software* a medida (lo vemos en prácticas).

3.10.10. Programación en circuito de bajo voltaje (LVP)

- El bit de configuración LVP = 1 para activarlo. Esto solo se puede hacer de la forma ordinaria con un voltaje de programación más elevado.
- La patilla **RB3/PGM** pasa a ser la patilla de activación de la programación con bajo voltaje (V_{dd}) quedando condenada a ser una entrada con esa función específica.
- Son necesarias además V_{dd} y V_{ss} para la alimentación del circuito y las patillas **RB6** y **RB7** para el reloj y los datos serie. Idem.

3.10.11. Juego de instrucciones

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected
BYTE-ORIENTED FILE REGISTER OPERATIONS					
ADDWF	**f, d**	Add W and f	1	00 0111 dfff ffff	C,DC,Z
ANDWF	**f, d**	AND W with f	1	00 0101 dfff ffff	Z
CLRF	**f**	Clear f	1	00 0001 lfff ffff	Z
CLRW	**-**	Clear W	1	00 0001 0xxx xxxx	Z
COMF	**f, d**	Complement f	1	00 1001 dfff ffff	Z
DECF	**f, d**	Decrement f	1	00 0011 dfff ffff	Z
DECFSZ	**f, d**	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff	
INCF	**f, d**	Increment f	1	00 1010 dfff ffff	Z
INCFSZ	**f, d**	Increment f, Skip if 0	1(2)	00 1111 dfff ffff	
IORWF	**f, d**	Inclusive OR W with f	1	00 0100 dfff ffff	Z
MOVF	**f, d**	Move f	1	00 1000 dfff ffff	Z
MOVWF	**f**	Move W to f	1	00 0000 lfff ffff	
NOP	**-**	No Operation	1	00 0000 0xx0 0000	
RLF	**f, d**	Rotate Left f through Carry	1	00 1101 dfff ffff	C
RRF	**f, d**	Rotate Right f through Carry	1	00 1100 dfff ffff	C
SUBWF	**f, d**	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z
SWAPF	**f, d**	Swap nibbles in f	1	00 1110 dfff ffff	
XORWF	**f, d**	Exclusive OR W with f	1	00 0110 dfff ffff	Z
BIT-ORIENTED FILE REGISTER OPERATIONS					
BCF	**f, b**	Bit Clear f	1	01 00bb bfff ffff	
BSF	**f, b**	Bit Set f	1	01 01bb bfff ffff	
BTFSC	**f, b**	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff	
BTFSS	**f, b**	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff	
LITERAL AND CONTROL OPERATIONS					
ADDLW	**k**	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z
ANDLW	**k**	AND literal with W	1	11 1001 kkkk kkkk	Z
CALL	**k**	Call subroutine	2	10 0kkk kkkk kkkk	
CLRWDT	**-**	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$
GOTO	**k**	Go to address	2	10 1kkk kkkk kkkk	
IORLW	**k**	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z
MOVLW	**k**	Move literal to W	1	11 00xx kkkk kkkk	
RETFIE	**-**	Return from interrupt	2	00 0000 0000 1001	
RETLW	**k**	Return with literal in W	2	11 01xx kkkk kkkk	
RETURN	**-**	Return from Subroutine	2	00 0000 0000 1000	
SLEEP	**-**	Go into Standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$
SUBLW	**k**	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z
XORLW	**k**	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z

Capítulo 4

FAMILIAS DE MICROCONTROLADORES

Se puede consultar una guía actualizada de los microcontroladores existentes en el mercado en:

`http://www.edn.com/info/1340009098.html`

Parte II

Procesadores de señal digital (DSP)

Capítulo 5

Introducción

5.1. Conceptos básicos

La primera pregunta que nos surge al hablar de los Procesadores de Señal Digitales (DSP: *Digital Signal Processors*) es saber **¿cuál es la diferencia entre un DSP y un microprocesador?**

La respuesta no es sencilla dado que a medida que avanza la tecnología los microprocesadores incorporan prestaciones presentes en los DSPs, y éstos a su vez incorporan características de los microprocesadores de propósito general. Sin embargo podemos decir que los DSP tienen características diseñadas para permitir tareas de **alto rendimiento**, **repetitivas** o de **cálculo intensivo**. Los microprocesadores o los microcontroladores o no están especializados para una tarea concreta (microprocesadores de propósito general) o están diseñados para aplicaciones orientadas al control (microcontroladores).

En general, los DSP disponen de:

- Capacidad de sumar y multiplicar[1] en un único ciclo de reloj (instrucción MAC).
- Modos de direccionamiento especializados:
  - Punteros de dirección con pre- ó post-modificación.
  - Direccionamiento circular.
  - Direccionamiento de bit invertido (*bit-reversed*).
- Varias configuraciones de memoria *on-chip* y periféricos. Incorporan arquitecturas de memoria con acceso múltiple (arquitectura Harvard).
- Tiene un control de ejecución especializado, como por ejemplo los bucles hardware que permiten no tener que leer/decodificar las instrucciones una y otra vez al ejecutar un bucle corto, además de no tener que decrementar ni comprobar el contador.
- Disponen de un conjunto de instrucciones irregular que permite codificar varias operaciones en una misma instrucción. Por ejemplo: 2 sumas, 2 multiplicaciones y 4 movimientos de datos en un único ciclo de instrucción.

Habrá que elegir siempre el DSP, microprocesador o microcontrolador que mejor se adapte a nuestra aplicación concreta u que salga más económico tanto a nivel de fabricación como de diseño.

[1] Multiplicador no microprogramado.

5.2. Campos de aplicación

Se emplean para el tratamiento digital de la información.

Ventajas sobre los sistemas analógicos

- **Reprogramabilidad**. Se pueden variar fácilmente los algoritmos.
- **Estabilidad, repetitividad y comportamiento previsible**. No varía con la temperatura, ni con el envejecimiento de los componentes. No necesita un calibrado.
- **Funciones especiales**. Algunos algoritmos no se pueden implementar de forma analógica, o es muy complicado.
- **Mayor inmunidad al ruido en la transmisión y el almacenamiento**. Lo dicho.

Inconvenientes sobre los sistemas analógicos

- **Más complejos** que la solución analógica. Aunque no son más caros.
- **Dificultad para aplicaciones de alta frecuencia**. Hace falta potencia computacional. En un analógico no.

Operaciones comunes en tratamiento digital de señales

Podemos encontrarnos entre otras:

- Filtrado de señal: $y(n) = \sum_{k=0}^{N} b(k)x(n-k) + \sum_{k=1}^{M} a(k)y(n-k)$
- Análisis espectral: transformadas de Fourier. $X(s) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)e^{-j\frac{2\pi ks}{N}}$
- Cálculo matricial: modulación y demodulación. $C_{ik} = \sum_{j=1}^{m} a_{ij}b_{jk}$

que se resumen en una sola:

$$y(m) = \sum_{k,n} a(k)b(n)$$

que es la operación de multiplicación y suma (MAC).

Características de la aplicación y del sistema

Aplicación	**Sistema**
Tiempo real	Operaciones MAC rápidas
Algoritmos cálculo intensivo	Manejo cómodo de datos: direccionamientos
Gran cantidad de datos a procesar	Programabilidad
Sistema flexible	

Alternativas de Diseño

Software	**Hardware**
Flexibilidad	Circuito integrado a medida para cada algoritmo
Algoritmos complejos	Menor coste
Ejecución secuencial	Mejores prestaciones
	Menos flexible

La mejor solución sería por tanto:

- Microprocesador de propósito general de altas prestaciones. Es un sistema complejo que necesita periféricos aparte y su potencia de cálculo mantenida quizá no sea tan alta como se espera.
- Procesador de señal digital. Muchos son específicos para cada tipo de aplicación y salen más económicos. Es un punto intermedio entre una alternativa software y un hardware.

5.3. Aplicaciones típicas

- Comunicaciones
  - Telefonía móvil digital
  - MODEMs
  - Redes de área local (LAN)
- Industria
  - Control de motores
- Instrumentación
  - Sistemas GPS (*Global Positioning System*)
  - Sistemas de Resonancia Magnética Nuclear
  - Ecógrafos.
- Electrónica de consumo
  - Cámaras fotográficas digitales.
  - Audio/vídeo digital (MPEG, MP3)
  - Radio/Televisión digital.
- Militar/aeroespacial
  - RADAR/SONAR.
  - Guiado de misiles.

Capítulo 6

Características de los DSPs

6.1. Arquitectura de la CPU

Formato de los datos

Es importante conocer la anchura en bits de los datos para saber si cubren el rango numérico de la aplicación, y el formato de los mismos.

Números enteros o de punto fijo . Los números enteros se representan en complemento a 2. Así el valor en binario `01010011` representará el valor decimal

$$(0 * 2^7 + 1 * 2^6 + 0 * 2^5 + 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0) = 64 + 16 + 2 + 1 = 83$$

Cuando el rango de los valores enteros no es suficiente, podemos emplear la técnica del punto fijo, que consiste en la existencia (para nosotros) de un punto fijo (punto decimal) que introduce un factor de escala, de 2^{-BP}, en el valor entero. El valor BP (Binary Point) será la posición del punto fijo. Así en el ejemplo anterior, si decidimos que $BP = 3$, el valor binario será para nosotros `01010.011` que representará el valor decimal

$$(2^6 + 2^4 + 2^1 + 2^0) * 2^{-3} = 2^3 + 2^1 + 2^{-2} + 2^{-3} = 8 + 2 + 1/4 + 1/8 = 10{,}375$$

La precisión, por tanto, será de 2^{-BP}. Se reduce el rango de los valores representables. Es responsabilidad del programador interpretar correctamente la posición del punto binario definido. La programación es, a veces, laboriosa y no muy eficiente.

Números en punto flotante . Estos números se componen de:

- Mantisa. Que es un número en punto fijo.
- Exponente. Valor entero que determina el valor del punto binario, BP, dinámicamente.
- Signo. Suele ser un bit aparte, el que determina el signo.

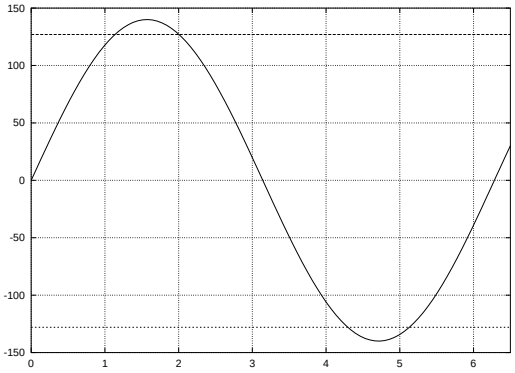
De esta manera el valor decimal representado se formará típicamente (dependerá del estándar) como:

$$Valor = (-1)^{Signo} * Mantisa * 2^{Exponente}$$

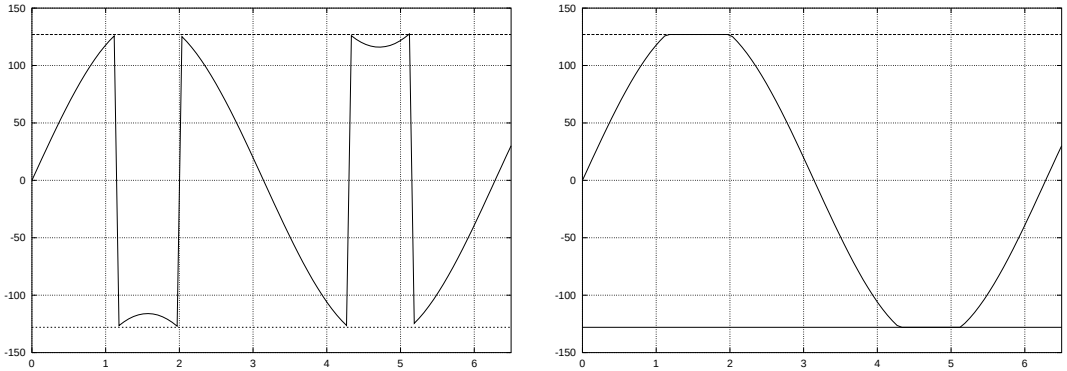
Este formato de números es más flexible que el punto fijo y da mayor precisión, aunque debe ser soportado por el DSP en cuestión.

Aritmética con saturación

Ante situaciones de rebose en una operación, es decir, cuando el resultado de una operación no cabe en el rango de valores representables por el número de bits de un registro o variable en memoria, la aritmética con saturación lo que hace es *saturar* el valor rebosado al valor máximo o mínimo, según corresponda, del rango de valores representables.



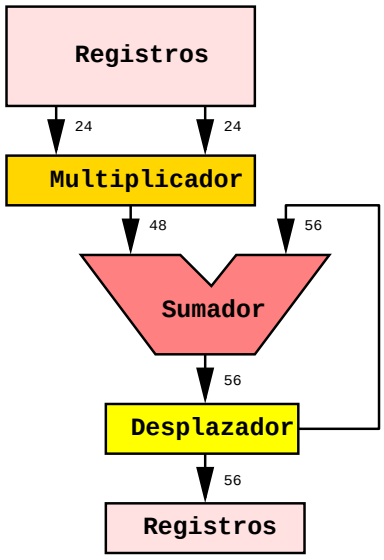
Ejemplo: resultado de una operación aritmética.



A la izquierda el rebose, a la derecha el resultado con saturación.

Multiplicación y acumulación: MAC

Típicamente, los DSPs, implementan en hardware una unidad MAC de multiplicación y acumulación no microprogramada.



El multiplicador hardware está cableado, Es capaz de multiplicar en un solo ciclo máquina.

La acumulación, procedente del ciclo anterior, se realiza en el mismo ciclo máquina, en paralelo.

Es un proceso segmentado (pipelined) en el que se suma el producto anterior mientras se calcula el siguiente producto.

Puede trabajar con números enteros (punto fijo) o con números en punto flotante. Dependerá de cada DSP.

Al repetirse las operaciones de acumulación pueden darse situaciones de rebose obteniéndose resultados erróneos (por que los datos no caben). Para evitar esto se sobredimensiona el número de bits del sumador: son los **bits de guarda**. En la figura tenemos 8 bits de guarda.

Otra estrategia para evitar el rebose consiste en escalar los resultados antes de sumarlos desplazando los valores a la derecha (dividiéndolos por 2^N) mediante un **desplazador** (*barrel shifter*). Este parte del circuito no requiere ciclos de instrucción (es un circuito combinacional muy rápido) y es independiente del número de bits a desplazar. Con esto perdemos resolución, pero será una solución aceptable en algunos casos, dependiendo de la aplicación.

Otra manera de evitar el rebose es emplear la **saturación aritmética** (los bloques se llaman limitadores).

Direccionamientos especiales

Los DSPs suelen disponer de más de una unidad aritmético-lógica (ALU) para generar el direccionamiento de los datos. Esto les confiere una mayor velocidad de procesamiento.

Además incorporan direccionamientos especiales específicos de determinado tipo de aplicaciones.

Disponen de unidades aritméticas específicas (no comparten la unidad lógico-aritmética general) que operan en paralelo con el resto de las unidades funcionales, con un conjunto especializado de registros de dirección.

Algunos de los direccionamientos especiales:

- **Direccionamientos con pre- o post- modificación**. Permiten realizar operaciones aritméticas sencillas (sumar o restar) valores literales a cada registro antes (pre-) o después (post-) de utilizar su valor en la instrucción donde están incluidos. La acción se realiza en paralelo.

- **Direccionamiento circular**. Hay aplicaciones que requieren gestionar un búfer FIFO (*First In, First Out*) manteniendo un puntero a memoria de lectura y un puntero de escritura. Cuando al incrementarlo (o decrementarlo) un puntero llega al final (o al principio) del búfer es necesario comprobarlo y reiniciar su valor apuntando al comienzo (o al final) del búfer. Esto consume tiempo. Para hacer esto algunos DSPs incorporan el direccionamiento circular o modular. El resultado de la unidad de direccionamiento al modificar un puntero (registro que apunta a memoria) está limitado, por hardware, a un rango determinado de direcciones de memoria.

- **Direccionamiento de bits permutados** (*bit-reversed*). En el cálculo de las transformadas de Fourier (FFT) es muy interesante poder acceder a los datos en un orden especial que se obtiene fácilmente invirtiendo el orden de los bits del índice (pasando de MSB..LSB a LSB...MSB). Esta prestación acelera el cálculo de las transformadas rápidas de Fourier.

6.1.1. Arquitectura de la Memoria

Arquitectura Harvard

Para poder ejecutar las instrucciones de multiplicación y suma (MAC) en un ciclo máquina es necesario disponer de al menos tres operandos. La separación de los datos y el programa en memorias separadas (arquitectura Harvard) permite que se lea un operando a la vez que se lee la instrucción.

Para poder leer los otros dos operandos podemos encontrarnos con que se permite almacenar datos en la memoria de programa (arquitectura Harvard modificada).

En la arquitectura Harvard mejorada se dispone de dos o más memorias de datos separadas con sus propios buses de acceso y se permite que la memoria de programa almacene datos. Esto solo es valido con la memoria interna (*on-chip*) del DSP.

Arquitectura	**Mem. Programa**	**Mem. Datos 1**	**Mem. Datos 2**	Ciclos
Harvard	Programa	Datos	-	3
Harvard modificada	Programa y Datos	Datos	-	2
Harvard mejorada	Programa y Datos	Datos	Datos	2 ó 1

Otras estrategias

Otras estrategias no excluyentes del empleo de la arquitectura Harvard son:

- **Memorias de acceso múltiple**. Entre las que tenemos:

  - Memorias rápidas que permiten varios accesos secuenciales por cada ciclo máquina o de instrucción con un único grupo de buses.
  - Memorias multipuerto que permiten varios accesos simultáneos con varios grupos de buses independientes.

Ambos tipos de estrategias se suelen integrar dentro del DSP (*on-chip*) ya que permitir memorias externas de este tipo encarece y complica mucho el diseño del DSP.

Hay que distribuir el programa y los datos entre las diferentes memorias de forma que no se sobrepase el ancho de banda de cada una de ellas.

- **Memoria caché de programa**. Esta estrategia evita tener que acceder a la memoria de programa para leer una instrucción, permitiendo en el mismo ciclo leer un dato de la misma.

6.1.2. Juego de Instrucciones

Bucles hardware

Muchas aplicaciones /algoritmos requieren la ejecución repetitiva de una misma instrucción o de un conjunto de ellas (bucle interno). Las comprobaciones, decrementos y saltos introducen penalizaciones en el tiempo de ejecución.

Los DSP suelen proporcionar los bucles hardware (*zero overhead looping*) que no pierden tiempo puesto que se realizan y comprueban mediante un hardware específico en paralelo.

Listado 6.1:

```
;  Ejemplo
        rpts    16                      ; Repite 16+1 veces la
                                        ;  instruccion siguiente
        mpyf3   *ar0++, *ar1++, r0      ; r0 = [ar0]*[ar1]
                                        ; ar0 = ar0 + 1
                                        ; ar1 = ar1 + 1
```

Paralelismo

Dado que existen muchas secciones hardware que funcionan independientemente, se pueden ejecutar varias ordenes o instrucciones de forma paralela. Éstas se codifican en un único código de operación y tiene que ser el programador el que indique si quiere ejecutarlas o no en paralelo.

Listado 6.2:

```
; Ejemplo
        mpyf3   *ar0++, *ar1++, r0
      || addf3    r0,r2,r2              ; En esta instruccion
                                        ;  hacemos concurrentemente:
                                        ; r0 = [ar0]*[ar1]
                                        ; ar0 = ar0 + 1
                                        ; ar1 = ar1 + 1
                                        ; r2 = r2 + r0 (anterior)
```

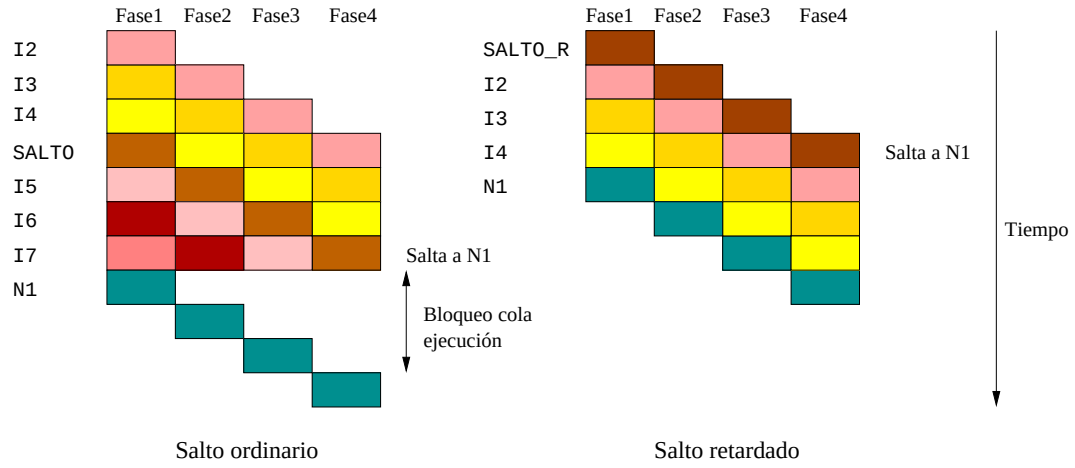
Saltos retardados

La ejecución de instrucciones en los DSP, que tienen arquitecturas internas típicamente RISC, está segmentada. Por defecto la ejecución de una instrucción de salto vacía la cola de ejecución de instrucciones.

En los saltos retardados no se vacía la cola de ejecución de instrucciones. Este tipo de instrucción hay que usarla con precaución ya que es responsabilidad del programador que el flujo de ejecución quede igual que antes de usar el salto retardado.

Suponemos una cola de ejecución (*pipeline*) de 4 etapas. El salto normal necesitará 4 ciclos de instrucción, hasta que se vuelva a llenar la cola de ejecución de instrucciones, mientras que un salto retardado ejecuta las tres instrucciones siguiente tanto si se cumple la condición del salto (suponiendo un salto condicional) como si no.

Hay que planificarlo, es como colocar la instrucción de salto tres instrucciones antes de donde la pondríamos si fuera un salto normal, siempre y cuando no se necesite el resultado de esas tres instrucciones para resolver la condición del salto.



Salto ordinario Salto retardado

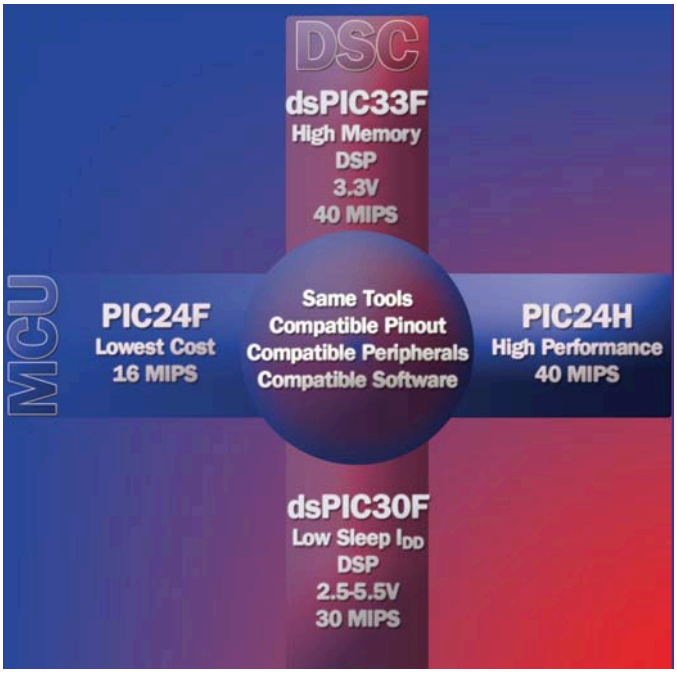
Capítulo 7

Procesador de señal digital dsPIC30F

7.1. Introducción

7.1.1. ¿Qué son los dsPIC?

La familia de microcontroladores de 16 bit de Microchip[1] está compuesta por cuatro subfamilias que comparten patillaje, juego de instrucciones, periféricos base, herramientas de desarrollo, etc. Dos de las subfamilias además añaden un motor DSP para aplicaciones de tratamiento de señal. En la siguiente figura se pueden ver de manera esquemática las diferencias (precio/prestaciones) de las subfamilias.



La subfamilia dsPIC30F es la que vamos a estudiar en detalle y tiene prestaciones de DSP, además de un rango de voltajes de operación compatible con los sistemas a los que estamos más acostumbrados.

[1] Ver en www.microchip.com el documento en012562.pdf

7.1.2. dsPIC30F: Vistazo general

Entre las características más destacadas están:

- CPU de alto rendimiento con núcleo RISC.
- Arquitectura Harvard modificada: bus de memoria de datos de 16 bits; bus de memoria de programa (instrucciones) de 24 bits.
- Repertorio de 84 instrucciones optimizadas para el lenguaje C (76 normales y 8 del motor DSP). Cada instrucción ocupa una dirección de la memoria de programa (24 bits). Incluye instrucciones de multiplicación y división enteras.
- Memoria integrada:
  - Memoria de programa -tipo Flash- de hasta 144 KB (256 KB en otras subfamilias) que se puede borrar/escribir hasta 100000 veces.
  - EEPROM de datos de hasta 4 KB (1 millon de borrados/escrituras).
  - SRAM de datos de hasta 8 KB (hasta 30 KB en otras subfamilias).
- Banco de 16 registros de trabajo de 16 bits.
- Rendimiento de hasta 30 MIPS[2].
  - Hasta 40 MHz de entrada externa de reloj.
  - PLL[3] integrado (x4, x8, x16) (máx. 120 MHz internos).
  - Necesita cuatro ciclos de reloj por instrucción. Dispone de pipeline de ejecución de instrucciones.
- Puede tener hasta 45 fuentes de interrupción (5 externas, 4 excepciones especiales).
- Prioridad de las interrupciones seleccionable entre 8 niveles.
- **Motor DSP**:
  - Búsqueda de datos dual (lectura de dos datos a la vez).
  - Modos de direccionamiento circular y de bit invertido.
  - Dos acumuladores de 40 bit con opciones de redondeo y saturación.
  - Multiplicador entero de 17x17 bits en un único ciclo máquina.
  - Todas las instrucciones DSP se ejecutan en un único ciclo máquina.
  - Desplazador de 16 bits en un único ciclo.
- **Periféricos integrados**
  - Patillas de entrada/salida de alta corriente (proporciona/absorbe 25mA máx.).

[2] Millones de Instrucciones Por Segundo.
[3] *Phase Looked Loop*: Lazo de enganche de fase. Es un multiplicador de frecuencia.

- Módulos temporizadores programables con pre-escala: 5 temporizadores de 16 bits que se pueden emparejar opcionalmente para formar hasta 2 contadores de 32 bits.
- Funciones de Captura (entrada) de 16 bits.
- Funciones de Comparación y PWM (salida) de 16 bits.
- Módulos SPI.
- Módulos I2C.
- Módulos UART con búferes FIFO.
- Módulos CAN 2.0B[4].
- Convertidores A/D de 10 bits (9 canales, hasta 1 Msps[5]).

▪ **Otras prestaciones**

- Reset *Brown-out* con nivel programable.
- La memoria de programa se puede automodificar en tiempo de ejecución (posibilita la implementación de un bootloader).
- Reset de arranque (POR), Temporizador de arranque (PWRT), Temporizador de arranque del oscilador (OST).
- Temporizador perro guardián (WDT) con oscilador RC integrado de baja velocidad.
- Monitor de fallo del oscilador principal.
- Modos de bajo consumo: *Sleep, Idle, Alternate clock*.

▪ Bajo consumo de potencia (en torno a 4 mA a 5V y 1 MIPS; 125 mA a 5V y 30 MIPS).

▪ Funcionamiento entre 2.5V y 5.5V.

7.1.3. Tabla comparativa

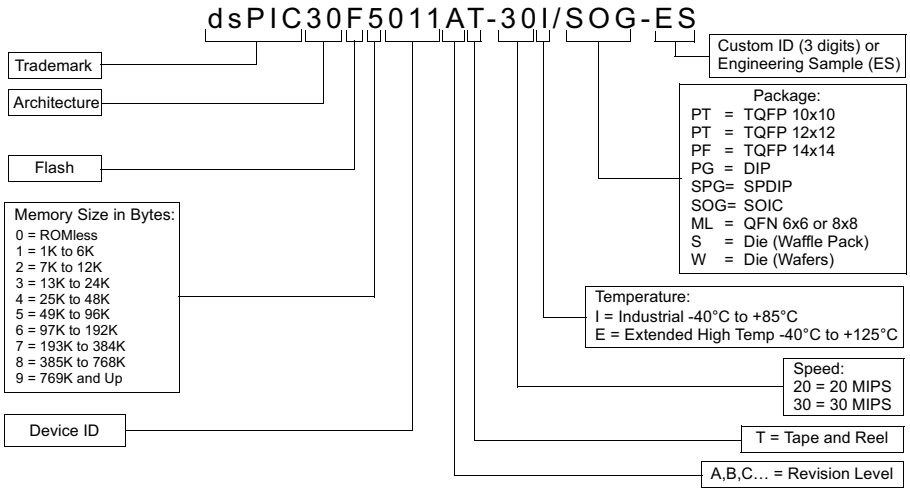
Se muestran en la siguiente tabla algunos de los miembros de esta familia de DSPs.

Device	Pins	Program Mem. Bytes/ Instructions	SRAM Bytes	EEPROM Bytes	Timer 16-bit	Input Cap	Output Comp/Std PWM	Motor Control PWM	10-Bit A/D 1 Msps	Quad Enc	UART	SPI	I^2C^{TM}	CAN
dsPIC30F2010	28	12K/4K	512	1024	3	4	2	6 ch	6 ch	Yes	1	1	1	-
dsPIC30F3010	28	24K/8K	1024	1024	5	4	2	6 ch	6 ch	Yes	1	1	1	-
dsPIC30F4012	28	48K/16K	2048	1024	5	4	2	6 ch	6 ch	Yes	1	1	1	1
dsPIC30F3011	40/44	24K/8K	1024	1024	5	4	4	6 ch	9 ch	Yes	2	1	1	-
dsPIC30F4011	40/44	48K/16K	2048	1024	5	4	4	6 ch	9 ch	Yes	2	1	1	1
dsPIC30F5015	64	66K/22K	2048	1024	5	4	4	8 ch	16 ch	Yes	1	2	1	1
dsPIC30F6010	80	144K/48K	8192	4096	5	8	8	8 ch	16 ch	Yes	2	2	1	2

[4] *Controller Area Network*: Red de área de controladores. Ver http://www.semiconductors.bosch.de/pdf/can2spec.pdf

[5] *Mega sample per second:* Millones de muestras por segundo

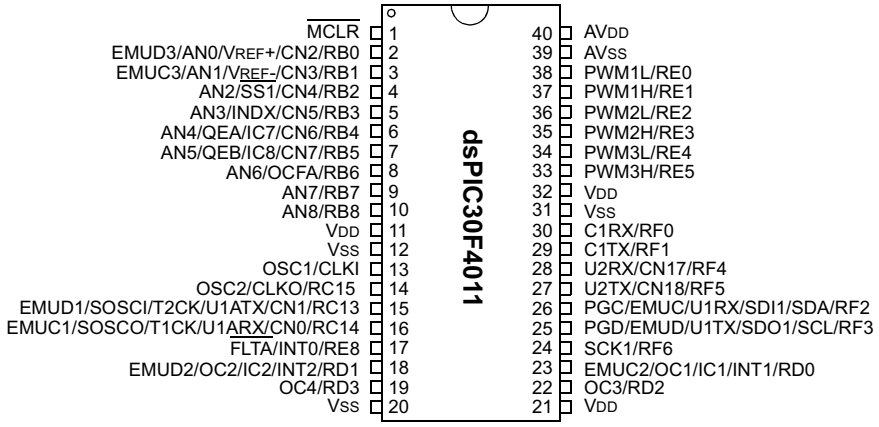
La numeración de los circuitos se puede analizar en la siguiente figura:



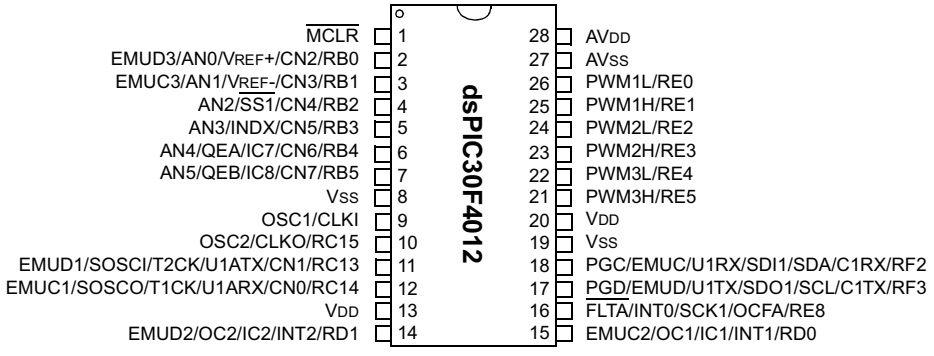
7.1.4. Patillaje de algunos modelos

A continuación se muestra el patillaje de algunos dsPIC:

- dsPIC30F4011

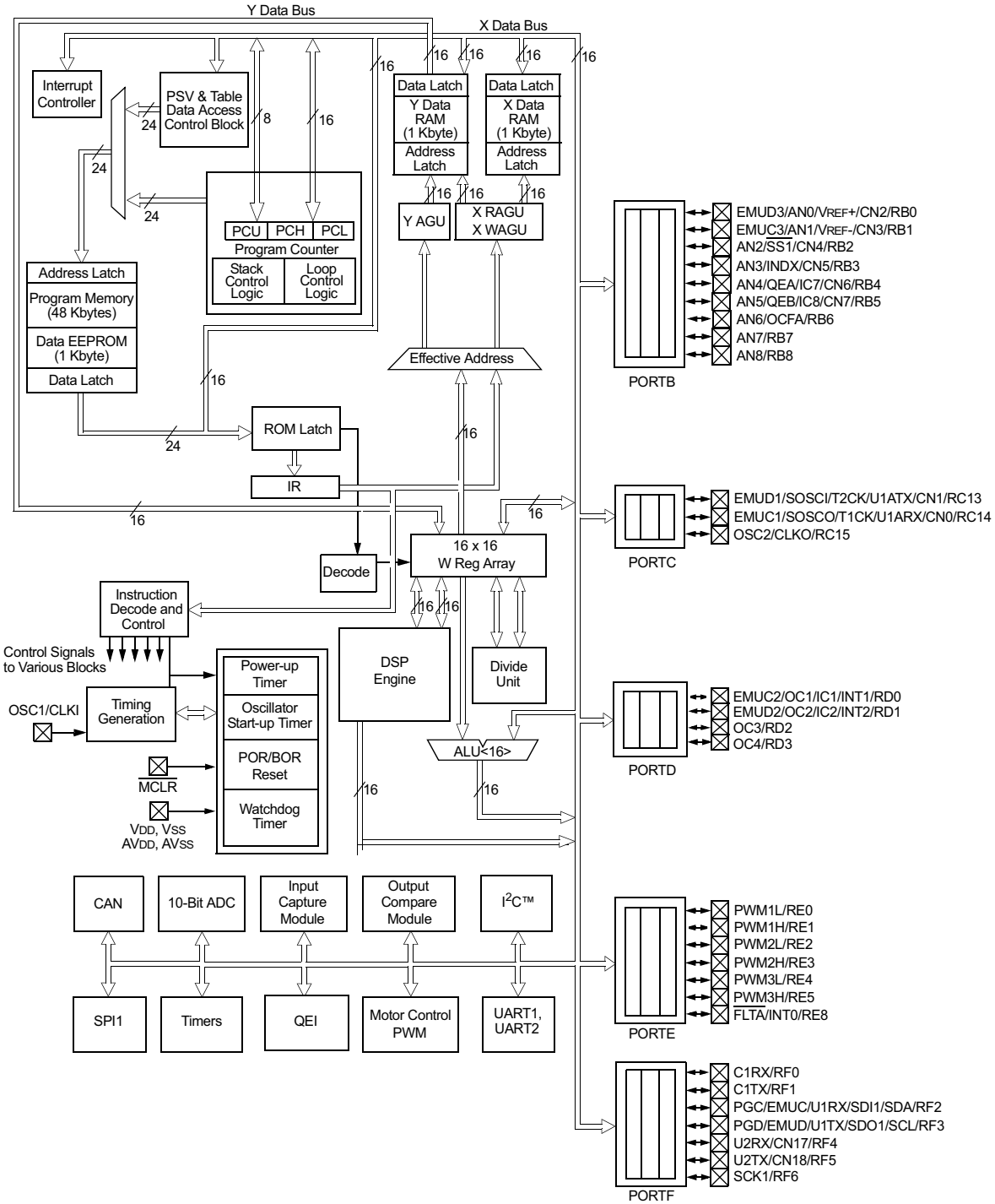


- dsPIC30F4012



7.1.5. Diagrama de bloques

En particular el diagrama de bloques del dsPIC30F4011



La descripción de las patillas la tenemos aquí:

Pin Name	Pin Type	Buffer Type	Description
AN0-AN8	I	Analog	Analog input channels. AN0 and AN1 are also used for device programming data and clock inputs, respectively.
AVDD	P	P	Positive supply for analog module.
AVSS	P	P	Ground reference for analog module.
CLKI CLKO	I O	ST/CMOS —	External clock source input. Always associated with OSC1 pin function. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes. Always associated with OSC2 pin function.
CN0-CN7 CN17-CN18	I	ST	Input change notification inputs. Can be software programmed for internal weak pull-ups on all inputs.
C1RX C1TX	I O	ST —	CAN1 bus receive pin. CAN1 bus transmit pin.
EMUD EMUC EMUD1 EMUC1 EMUD2 EMUC2 EMUD3 EMUC3	I/O I/O I/O I/O I/O I/O I/O I/O	ST ST ST ST ST ST ST ST	ICD Primary Communication Channel data input/output pin. ICD Primary Communication Channel clock input/output pin. ICD Secondary Communication Channel data input/output pin. ICD Secondary Communication Channel clock input/output pin. ICD Tertiary Communication Channel data input/output pin. ICD Tertiary Communication Channel clock input/output pin. ICD Quaternary Communication Channel data input/output pin. ICD Quaternary Communication Channel clock input/output pin.
IC1, IC2, IC7, IC8	I	ST	Capture inputs 1, 2, 7 and 8.
INDX QEA QEB	I I I	ST ST ST	Quadrature Encoder Index Pulse input. Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode. Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode.
INT0 INT1 INT2	I I I	ST ST ST	External interrupt 0. External interrupt 1. External interrupt 2.
$\overline{\text{FLTA}}$ PWM1L PWM1H PWM2L PWM2H PWM3L PWM3H	I O O O O O O	ST — — — — — —	PWM Fault A input. PWM1 low output. PWM1 high output. PWM2 low output. PWM2 high output. PWM3 low output. PWM3 high output.
$\overline{\text{MCLR}}$	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active-low Reset to the device.
OCFA OC1-OC4	I O	ST —	Compare Fault A input (for Compare channels 1, 2, 3 and 4). Compare outputs 1 through 4.

Legend: CMOS = CMOS compatible input or output Analog = Analog input
ST = Schmitt Trigger input with CMOS levels O = Output
I = Input P = Power

Pin Name	Pin Type	Buffer Type	Description
OSC1	I	ST/CMOS	Oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise.
OSC2	I/O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes.
PGD	I/O	ST	In-Circuit Serial Programming™ data input/output pin.
PGC	I	ST	In-Circuit Serial Programming clock input pin.
RB0-RB8	I/O	ST	PORTB is a bidirectional I/O port.
RC13-RC15	I/O	ST	PORTC is a bidirectional I/O port.
RD0-RD3	I/O	ST	PORTD is a bidirectional I/O port.
RE0-RE5, RE8	I/O	ST	PORTE is a bidirectional I/O port.
RF0-RF6	I/O	ST	PORTF is a bidirectional I/O port.
SCK1	I/O	ST	Synchronous serial clock input/output for SPI1.
SDI1	I	ST	SPI1 data in.
SDO1	O	—	SPI1 data out.
$\overline{SS1}$	I	ST	SPI1 slave synchronization.
SCL	I/O	ST	Synchronous serial clock input/output for I^2C™.
SDA	I/O	ST	Synchronous serial data input/output for I^2C.
SOSCO	O	—	32 kHz low-power oscillator crystal output.
SOSCI	I	ST/CMOS	32 kHz low-power oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise.
T1CK	I	ST	Timer1 external clock input.
T2CK	I	ST	Timer2 external clock input.
U1RX	I	ST	UART1 receive.
U1TX	O	—	UART1 transmit.
U1ARX	I	ST	UART1 alternate receive.
U1ATX	O	—	UART1 alternate transmit.
U2RX	I	ST	UART2 receive.
U2TX	O	—	UART2 transmit.
VDD	P	—	Positive supply for logic and I/O pins.
VSS	P	—	Ground reference for logic and I/O pins.
VREF+	I	Analog	Analog voltage reference (high) input.
VREF-	I	Analog	Analog voltage reference (low) input.

Legend: CMOS = CMOS compatible input or output; Analog = Analog input
ST = Schmitt Trigger input with CMOS levels; O = Output
I = Input; P = Power

7.1.6. Resumen de subfamilias

PIC24F Family

16 MIPS, Lowest Cost

The PIC24F family is ideal for cost-sensitive applications or applications migrating from 8-bit designs for a boost in performance or memory.

Product	Pins	Flash Kbytes	RAM Kbytes	Timer	Capture	Output Compare PWM	RTCC	ADC 10-bit 500 ksps	Comparators	UART	SPI	I²C™	JTAG	Package Code
PIC24FJ16GA004	44	16	8	4	5	5	Y	1 ADC, 13 ch	2	2	2	2	Y	ML, PT
PIC24FJ16GA002	28	16	8	4	5	5	Y	1 ADC, 10 ch	2	2	2	2	Y	ML, SO, SP, SS
PIC24FJ32GA002	28	32	8	5	5	5	Y	1 ADC, 10 ch	2	2	2	2	Y	ML, SO, SP, SS
PIC24FJ32GA004	44	32	8	5	5	5	Y	1 ADC, 13 ch	2	2	2	2	Y	ML, PT
PIC24FJ48GA004	44	48	8	5	5	5	Y	1 ADC, 13 ch	2	2	2	2	Y	ML, PT
PIC24FJ48GA002	28	48	8	5	5	5	Y	1 ADC, 10 ch	2	2	2	2	Y	ML, SO, SP, SS
PIC24FJ64GA002	28	64	8	5	5	5	Y	1 ADC, 10 ch	2	2	2	2	Y	ML, SO, SP, SS
PIC24FJ64GA004	44	64	8	5	5	5	Y	1 ADC, 13 ch	2	2	2	2	Y	ML, PT
PIC24FJ64GA006	64	64	8	5	5	5	Y	1 ADC, 16 ch	2	2	2	2	Y	PT
PIC24FJ64GA008	80	64	8	5	5	5	Y	1 ADC, 16 ch	2	2	2	2	Y	PT
PIC24FJ64GA010	100	64	8	5	5	5	Y	1 ADC, 16 ch	2	2	2	2	Y	PT, PF
PIC24FJ96GA006	64	96	8	5	5	5	Y	1 ADC, 16 ch	2	2	2	2	Y	PT
PIC24FJ96GA008	80	96	8	5	5	5	Y	1 ADC, 16 ch	2	2	2	2	Y	PT
PIC24FJ96GA010	100	96	8	5	5	5	Y	1 ADC, 16 ch	2	2	2	2	Y	PT, PF
PIC24FJ128GA006	64	128	8	5	5	5	Y	1 ADC, 16 ch	2	2	2	2	Y	PT
PIC24FJ128GA008	80	128	8	5	5	5	Y	1 ADC, 16 ch	2	2	2	2	Y	PT
PIC24FJ128GA010	100	128	8	5	5	5	Y	1 ADC, 16 ch	2	2	2	2	Y	PT, PF

PIC24H Family

40 MIPS, Highest Performance

The PIC24H family is ideal for applications with greater performance or memory requirements or require extensive data movement.

Product	Pins	Flash Kbytes	RAM Kbytes	DMA # Ch	Timer	Capture	Output Compare PWM	ADC 10-/12-bit* 1.1/0.5 Msps	CodeGuard™ Security Segments	UART	SPI	I²C™	CAN	JTAG	Package Code
PIC24HJ12GP201	18	12	1	–	3	4	2	1 ADC, 6 ch	2	1	1	1	–	Y	P, SO
PIC24HJ12GP202	28	12	1	–	3	4	2	1 ADC, 10 ch	2	1	1	1	–	Y	SP, SO, ML
PIC24HJ32GP202	28	32	2	–	3	4	2	1 ADC, 10 ch	2	1	1	1	–	Y	SP, SO, MM
PIC24HJ16GP304	44	16	2	–	3	4	2	1 ADC 13 ch	2	1	1	1	–	Y	PT, ML
PIC24HJ32GP204	44	32	2	–	3	4	2	1 ADC 13 ch	2	1	1	1	–	Y	PT, ML
PIC24HJ64GP206	64	64	8	8	9	8	8	1 ADC, 18 ch	3	2	2	1	–	Y	PT
PIC24HJ64GP506	64	64	8	8	9	8	8	1 ADC, 18 ch	3	2	2	2	1	Y	PT
PIC24HJ128GP206	64	128	8	8	9	8	8	1 ADC, 18 ch	3	2	2	2	–	Y	PT
PIC24HJ128GP306	64	128	16	8	9	8	8	1 ADC, 18 ch	3	2	2	2	–	Y	PT
PIC24HJ128GP506	64	128	8	8	9	8	8	1 ADC, 18 ch	3	2	2	2	1	Y	PT
PIC24HJ256GP206	64	256	16	8	9	8	8	1 ADC, 18 ch	3	2	2	2	–	Y	PT
PIC24HJ64GP210	100	64	8	8	9	8	8	1 ADC, 32 ch	3	2	2	2	–	Y	PT, PF
PIC24HJ64GP510	100	64	8	8	9	8	8	1 ADC, 32 ch	3	2	2	2	1	Y	PT, PF
PIC24HJ128GP210	100	128	8	8	9	8	8	1 ADC, 32 ch	3	2	2	2	–	Y	PT, PF
PIC24HJ128GP310	100	128	16	8	9	8	8	1 ADC, 32 ch	3	2	2	2	–	Y	PT, PF
PIC24HJ128GP510	100	128	8	8	9	8	8	1 ADC, 32 ch	3	2	2	2	1	Y	PT, PF
PIC24HJ256GP210	100	256	16	8	9	8	8	1 ADC, 32 ch	3	2	2	2	–	Y	PT, PF
PIC24HJ256GP610	100	256	16	8	9	8	8	2 ADC, 32 ch	3	2	2	2	2	Y	PT, PF

*PIC24H devices feature one or two user-selectable 1.1 Msps 10-bit ADC (4 S&H) or 500 ksps 12-bit ADC (1 S&H)

dsPIC33F Product Families

General Purpose Family

The dsPIC33F General Purpose Family is ideal for a wide variety of 16-bit embedded control applications. In addition, the variants with codec interfaces are well suited for speech and audio applications.

Product	Pins	Flash Kbytes	RAM Kbytes	DMA # Ch	Timer 16-bit	Input Capture	Output Compare/ Standard PWM	Codec Interface	ADC 10-12-bit* 1.1/0.5 Msps	CodeGuard™ Security Segments	Codec	UART	SPI	I²C™	CAN	Package Code
dsPIC33FJ12GP201	18	12	1	–	3	4	2	–	1 ADC, 8 ch	2	–	1	1	1	–	P, SO
dsPIC33FJ12GP202	28	12	1	–	3	4	2	–	1 ADC, 10 ch	2	–	1	1	1	–	SO, SP, ML
dsPIC33FJ32GP202	28	32	2	–	3	4	2	–	1 ADC, 10 ch	2	–	1	1	1	–	SO, SP, MM
dsPIC33FJ16GP304	44	16	2	–	3	4	2	–	1 ADC, 13 ch	2	–	1	1	1	–	PT,ML
dsPIC33FJ32GP204	44	32	2	–	3	4	2	–	1 ADC, 13 ch	2	–	1	1	1	–	PT,ML
dsPIC33FJ64GP206	64	64	8	8	9	8	8	1	1 ADC, 18 ch	3	1	2	2	1	–	PT
dsPIC33FJ64GP306	64	64	16	8	9	8	8	1	1 ADC, 18 ch	3	1	2	2	2	–	PT
dsPIC33FJ64GP706	64	64	16	8	9	8	8	1	2 ADC, 18 ch	3	1	2	2	2	2	PT
dsPIC33FJ128GP206	64	128	8	8	9	8	8	1	1 ADC, 18 ch	3	1	2	2	1	–	PT
dsPIC33FJ128GP306	64	128	16	8	9	8	8	1	1 ADC, 18 ch	3	1	2	2	2	–	PT
dsPIC33FJ128GP706	64	128	16	8	9	8	8	1	2 ADC, 18 ch	3	1	2	2	2	2	PT
dsPIC33FJ256GP506	64	256	16	8	9	8	8	1	1 ADC, 18 ch	3	1	2	2	2	1	PT
dsPIC33FJ64GP708	80	64	16	8	9	8	8	1	2 ADC, 24 ch	3	1	2	2	2	2	PT
dsPIC33FJ128GP708	80	128	16	8	9	8	8	1	2 ADC, 24 ch	3	1	2	2	2	2	PT
dsPIC33FJ64GP310	100	64	16	8	9	8	8	1	1 ADC, 32 ch	3	1	2	2	2	–	PT, PF
dsPIC33FJ64GP710	100	64	16	8	9	8	8	1	2 ADC, 32 ch	3	1	2	2	2	2	PT, PF
dsPIC33FJ128GP310	100	128	16	8	9	8	8	1	1 ADC, 32 ch	3	1	2	2	2	–	PT, PF
dsPIC33FJ128GP710	100	128	16	8	9	8	8	1	2 ADC, 32 ch	3	1	2	2	2	2	PT, PF
dsPIC33FJ256GP510	100	256	16	8	9	8	8	1	1 ADC, 32 ch	3	1	2	2	2	1	PT, PF
dsPIC33FJ256GP710	100	256	30	8	9	8	8	1	2 ADC, 32 ch	3	1	2	2	2	2	PT, PF

Motor Control and Power Conversion Family

This dsPIC33F family supports motor control applications, such as brushless DC, single- and 3-phase induction and switched reluctance motors. These are also ideal for UPS, inverter and power factor correction applications.

Product	Pins	Flash Kbytes	RAM Kbytes	DMA # Ch	Timer 16-bit	Input Capture	Output Compare/ Standard PWM	Motor Control PWM	Quadrature Encoder Interface	ADC 10/12-bit* 1.1/0.5 Msps	CodeGuard™ Security Segments	UART	SPI	I²C™	CAN	Package Code
dsPIC33FJ12MC201	20	12	1	–	3	4	2	4+2 ch	1	1 ADC, 4 ch	2	1	1	1	0	SO, P, SS
dsPIC33FJ12MC202	28	12	1	–	3	4	2	6+2 ch	1	1 ADC, 6 ch	2	1	1	1	0	SO, SP, ML
dsPIC33FJ32MC202	28	32	2	–	3	4	2	6+2 ch	1	1 ADC, 6 ch	2	1	1	1	0	SO, SP, MM
dsPIC33FJ16MC304	44	16	2	–	3	4	2	6+2 ch	1	1 ADC, 9 ch	2	1	1	1	0	PT,ML
dsPIC33FJ32MC204	44	32	2	–	3	4	2	6+2 ch	1	1 ADC, 9 ch	2	1	1	1	0	PT,ML
dsPIC33FJ64MC506	64	64	8	8	9	8	8	8 ch	1	1 ADC, 16 ch	3	2	2	2	1	PT
dsPIC33FJ64MC706	64	64	16	8	9	8	8	8 ch	1	2 ADC, 16 ch	3	2	2	2	1	PT
dsPIC33FJ128MC506	64	128	8	8	9	8	8	8 ch	1	1 ADC, 16 ch	3	2	2	2	1	PT
dsPIC33FJ128MC706	64	128	16	8	9	8	8	8 ch	1	2 ADC, 16 ch	3	2	2	2	1	PT
dsPIC33FJ64MC508	80	64	8	8	9	8	8	8 ch	1	1 ADC, 18 ch	3	2	2	2	1	PT
dsPIC33FJ128MC708	80	128	16	8	9	8	8	8 ch	1	2 ADC, 18 ch	3	2	2	2	2	PT
dsPIC33FJ64MC510	100	64	8	8	9	8	8	8 ch	1	1 ADC, 24 ch	3	2	2	2	1	PT, PF
dsPIC33FJ64MC710	100	64	16	8	9	8	8	8 ch	1	2 ADC, 24 ch	3	2	2	2	2	PT, PF
dsPIC33FJ128MC510	100	128	8	8	9	8	8	8 ch	1	1 ADC, 24 ch	3	2	2	2	1	PT, PF
dsPIC33FJ128MC710	100	128	16	8	9	8	8	8 ch	1	2 ADC, 24 ch	3	2	2	2	2	PT, PF
dsPIC33FJ256MC510	100	256	16	8	9	8	8	8 ch	1	1 ADC, 24 ch	3	2	2	2	1	PT, PF
dsPIC33FJ256MC710	100	256	30	8	9	8	8	8 ch	1	2 ADC, 24 ch	3	2	2	2	2	PT, PF

*dsPIC33 devices feature one or two user-selectable 1.1 Msps 10-bit ADC (4 S&H) or 500 ksps 12-bit ADC (1 S&H).

dsPIC30F Product Families

General Purpose Family

The dsPIC30F General Purpose Family is ideal for a wide variety of 16-bit embedded control applications. The variants with codec interfaces are well suited for speech and audio applications.

Product	Pins	Flash Memory Kbytes	RAM Bytes	EEPROM Bytes	Timer 16-bit	Input Capture	Output Compare/ Standard PWM	Codec Interface	ADC 12-bit 200 ksps	CodeGuard™ Security Segments	UART	SPI	I²C™	CAN	Package Code
dsPIC30F3014	40/44	24	2048	1024	3	2	2	–	13 ch, 1 S/H	1	2	1	1	–	P, PT, ML
dsPIC30F4013	40/44	48	2048	1024	5	4	4	AC97, I²S	13 ch, 1 S/H	3	2	1	1	1	P, PT, ML
dsPIC30F5011	64	66	4096	1024	5	8	8	AC97, I²S	16 ch, 1 S/H	3	2	2	1	2	PT
dsPIC30F6011A	64	132	6144	2048	5	8	8	–	16 ch, 1 S/H	3	2	2	1	2	PF, PT
dsPIC30F6012A	64	144	8192	4096	5	8	8	AC97, I²S	16 ch, 1 S/H	3	2	2	1	2	PF, PT
dsPIC30F5013	80	66	4096	1024	5	8	8	AC97, I²S	16 ch, 1 S/H	3	2	2	1	2	PT
dsPIC30F6013A	80	132	6144	2048	5	8	8	–	16 ch, 1 S/H	3	2	2	1	2	PF, PT
dsPIC30F6014A	80	144	8192	4096	5	8	8	AC97, I²S	16 ch, 1 S/H	3	2	2	1	2	PF, PT

Sensor Family

The dsPIC30F Sensor family products have features designed to support high-performance, cost-sensitive and space-constrained applications. Offered in packages as small as 6x6 mm and with pin counts as low as 18 pins.

Product	Pins	Flash Memory Kbytes	RAM Bytes	EEPROM Bytes	Timer 16-bit	Input Capture	Output Compare/ Standard PWM	ADC 12-bit 200 ksps	UART	SPI	I²C™	I/O Pins (Max.)	Package Code
dsPIC30F2011	18	12	1024	–	3	2	2	8 ch, 1 S/H	1	1	1	12	P, SO, 28-pin ML
dsPIC30F3012	18/44	24	2048	1024	3	2	2	8 ch, 1 S/H	1	1	1	12	P, SO, 44-pin ML
dsPIC30F2012	28	12	1024	–	3	2	2	10 ch, 1 S/H	1	1	1	20	SP, SO, 28-pin ML
dsPIC30F3013	28/44	24	2048	1024	3	2	2	10 ch, 1 S/H	2	1	1	20	SP, SO, 44-pin ML

Motor Control and Power Conversion Family

This dsPIC30F family supports motor control applications, such as brushless DC, single- and 3-phase induction and switched reluctance motors. These are also ideal for UPS, inverter and power factor correction applications.

Product	Pins	Flash Memory Kbytes	RAM Bytes	EEPROM Bytes	Timer 16-bit	Input Capture	Output Compare/ Standard PWM	Motor Control PWM	Quadrature Encoder	ADC 10-bit 1 Msps	CodeGuard™ Security Segments	UART	SPI	I²C™	CAN	Package Code
dsPIC30F2010	28	12	512	1024	3	4	2	6 ch	Yes	6 ch, 4 S/H	1	1	1	1	–	SP, SO, MM
dsPIC30F3010	28/44	24	1024	1024	5	4	2	6 ch	Yes	6 ch, 4 S/H	1	1	1	1	–	SP, SO, 44-pin ML
dsPIC30F4012	28/44	48	2048	1024	5	4	2	6 ch	Yes	6 ch, 4 S/H	1	1	1	1	1	SP, SO, 44-pin ML
dsPIC30F3011	40/44	24	1024	1024	5	4	4	6 ch	Yes	9 ch, 4 S/H	1	2	1	1	–	P, PT, ML
dsPIC30F4011	40/44	48	2048	1024	5	4	4	6 ch	Yes	9 ch, 4 S/H	1	2	1	1	1	P, PT, ML
dsPIC30F5015	64	66	2048	1024	5	4	4	8 ch	Yes	16 ch, 4 S/H	1	1	2	1	1	PT
dsPIC30F6015	64	144	8192	4096	5	8	8	8 ch	Yes	16 ch, 4 S/H	3	2	2	1	1	PT
dsPIC30F5016	80	66	2048	1024	5	4	4	8 ch	Yes	16 ch, 4 S/H	1	1	2	1	1	PT
dsPIC30F6010A	80	144	8192	4096	5	8	8	8 ch	Yes	16 ch, 4 S/H	3	2	2	1	2	PF, PT

Digital Power Conversion Family

This dsPIC30F family supports applications such as Switch Mode Power Supplies (SMPS), induction cooking, UPS, inverter, power factor correction and digital control loops. These devices contain 1 nS resolution PWMs coupled with our fastest on-chip ADC and comparators to facilitate a variety of applications and power supply topologies.

Product	Pins	Flash Memory Kbytes	RAM (Bytes)	ADC 10-bit, 2 Msps Ch.	Analog Comparators	High-Speed PWM	Timers	Input Capture	Output Compare/ Standard PWM	UART	SPI	I²C™	Package Code
dsPIC30F1010	28	6	256	6 ch, 2 S&H	2	2 x 2	2	–	1	1	1	1	SO, SP, MM
dsPIC30F2020	28	12	512	8 ch, 4 S&H	4	4 x 2	3	1	2	1	1	1	SO, SP, MM
dsPIC30F2023	44	12	512	12 ch, 4 S&H	4	4 x 2	3	1	2	1	1	1	PT, ML

7.2. Arquitectura

7.2.1. Vistazo general al núcleo

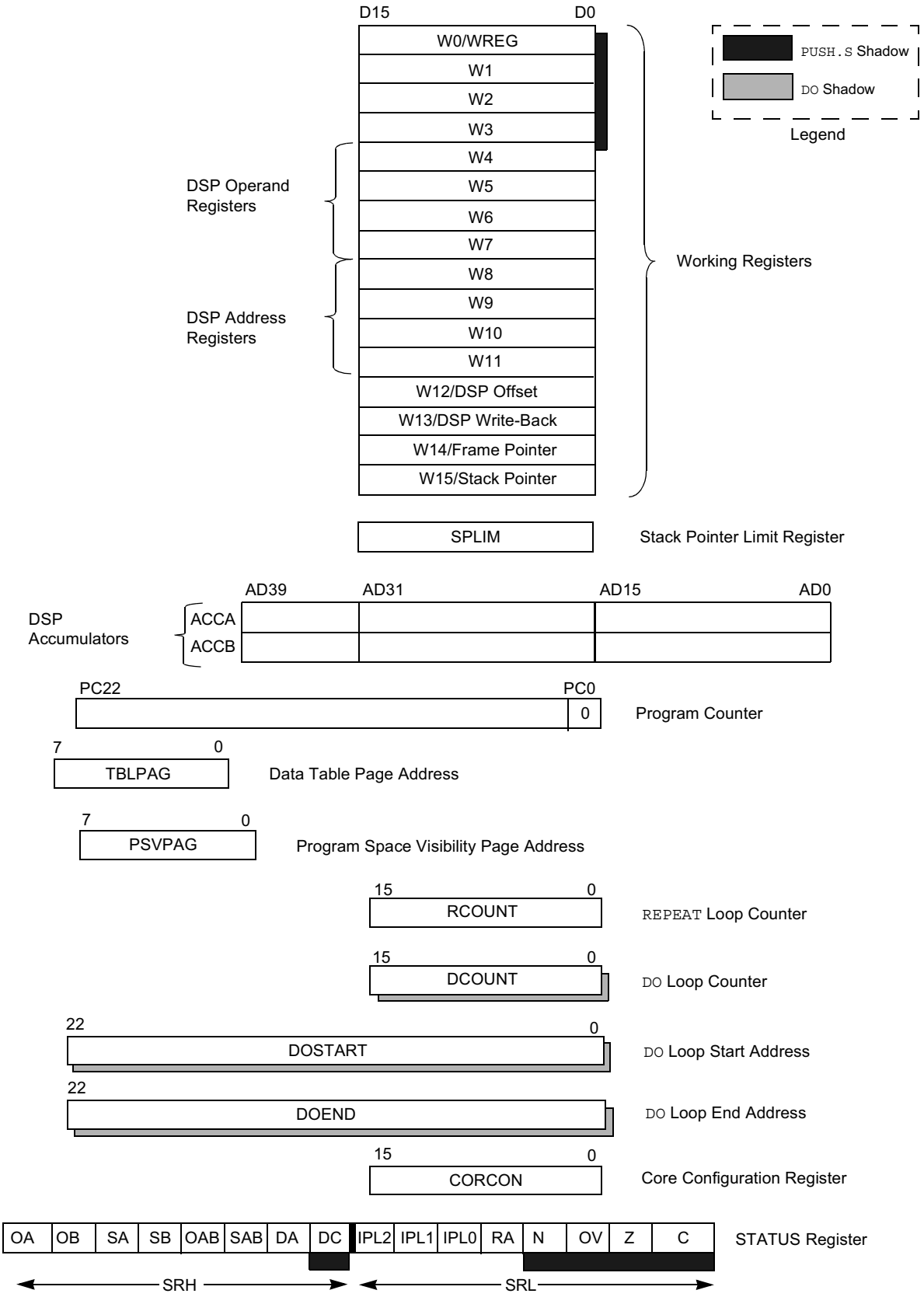
Las instrucciones ocupan 24 bits. El contador de programa es de 23 bits aunque el bit menos significativo vale siempre cero y el bit más significativo se ignora excepto en la ejecución de algunas instrucciones. Por tanto, el espacio de programa de usuario permite 4M instrucciones ($2^{22} = 2^2 * 2^{20} = 4M$).

7.2.2. Modelo de programación

Se muestra en la figura siguiente y consta de 16 registros de trabajo de 16 bits (`W0`..`W15`), 2 acumuladores de 40 bits (`ACCA` y `ACCB`), el registro de estado (`STATUS`), el registro de página de tabla de datos (`TBLPAG`), el registro de página visible del espacio de programa (`PSVPAG`), los registros para bucles hardware (`DOSTART`, `DOEND`, `DCOUNT` y `RCOUNT`) y el contador de programa (`PC`). Los registros de trabajo pueden funcionar como registros de datos o direcciones. Todos los registros están mapeados en la memoria de datos volátil.

Algunos de estos registros disponen de registros sombra asociados a ellos. Estos registros se emplean como almacenamiento temporal de los registros principales que representan cuando ocurre un suceso y no son accesibles directamente:

- Instrucciones `PUSH.S` y `POP.S`. Los registros `W0`, `W1`, `W2`, `W3` y `STATUS` (solo los campos `DC`, `N`, `OV`, `Z` y `C`) son transferidos a los registros sombra.
- Instrucción `DO`. Los registros `DOSTART`, `DOEND`, `DCOUNT` son salvados y recuperados después de bucle de los registros sombra.



Cuando se realiza una operación en tamaño byte solo se ve afectado el byte menos significativo del registro.

Puntero de pila software y puntero marco

El registro **W15** es el puntero de pila software y se modifica automáticamente al procesar llamadas a rutinas, retornos y atención a interrupciones y excepciones. Se inicializa con el valor `0x0800` durante el reset. Tiene un límite superior indicado por el registro **SPLIM** y un límite inferior que es `0x0800`. Si se superan estos límites se genera una excepción.

El registro **W14** funciona como un puntero marco (*frame pointer*) como se indica en las instrucciones **LNK** y **ULNK**.

Registro STATUS

Las partes alta y baja se denominan **SRH** y **SRL** respectivamente. Contiene los flags de estado de la unidad lógico-aritmética así como los bits para indicar el nivel de prioridad de las interrupciones (`IPL[3..0]`) y el bit de repetición activa (**RA**). Durante las excepciones, **SRL** se concatena con el byte más significativo del **PC** para completar una palabra de 16 bits y almacenarlo en la pila.

La parte superior del registro contiene los bits de estado del sumador/restador del motor DSP, el bit **DA** de bucle **do** activo y el semiacarreo **DC**.

Contador de programa

Puede direccionar hasta 4M instrucciones y el bit 0 siempre está a cero.

7.2.3. Soporte de la división

El dsPIC implementa una operación de división fraccionaria[6] con signo de 16/16 bits y también divisiones enteras de 32/16 bits y 16/16 bits con signo o sin signo.

Las instrucciones deben ser ejecutadas dentro de un bucle tipo **repeat** ya que su funcionamiento depende del registro **RCOUNT** que debe ser iniciado para hacer 18 iteraciones.

Instruction	Function
DIVF	Signed fractional divide: Wm/Wn → W0; Rem → W1
DIV.sd	Signed divide: (Wm + 1:Wm)/Wn → W0; Rem → W1
DIV.s	Signed divide: Wm/Wn → W0; Rem → W1
DIV.ud	Unsigned divide: (Wm + 1:Wm)/Wn → W0; Rem → W1
DIV.u	Unsigned divide: Wm/Wn → W0; Rem → W1

7.2.4. Motor DSP

El motor DSP consiste en un multiplicador de 17x17 bits, un desplazador (*barrel shifter*) y un sumador/restador de 40 bits (con dos acumuladores y lógica de redondeo y saturación).

El funcionamiento del motor DSP se puede controlar a través de varios bits de configuración en el registro **CORCON** (ver pag. 187):

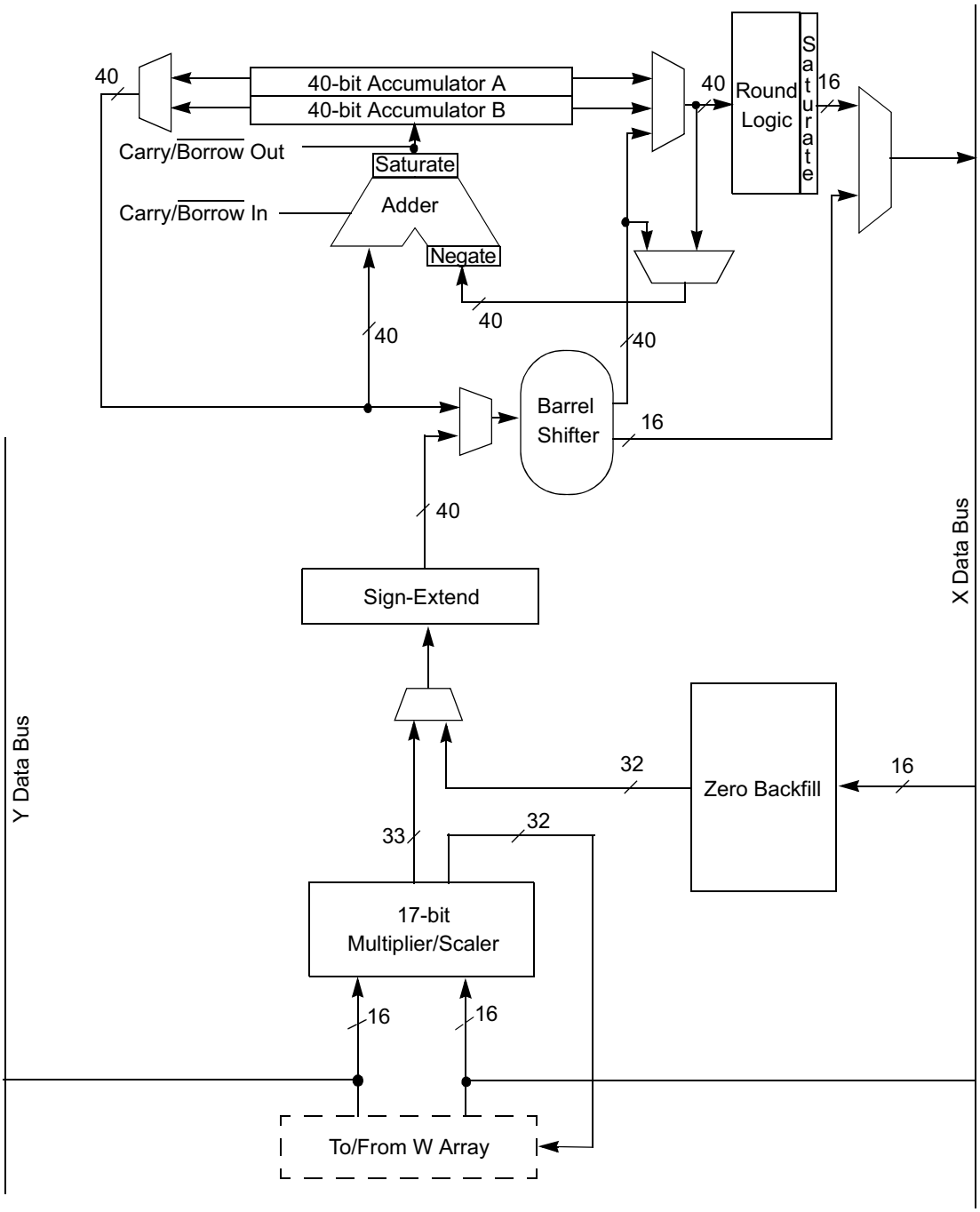
[6] Se emplea un formato de número de punto fijo 1.31 (si operando de 32 bits) ó 1.15 (si operando de 16 bits), con un bit de parte entera y el resto como parte fraccionaria.

- **CORCON.IF** Multiplicación DSP entera o fraccionaria.
- **CORCON.US** Multiplicación DSP sin signo o con signo.
- **CORCON.RND** Redondeo convergente o convencional.
- **CORCON.SATA** Saturación automática para el acumulador **ACCA**.
- **CORCON.SATB** Saturación automática para el acumulador **ACCB**.
- **CORCON.SATDW** Saturación automática para escrituras en la memoria de datos.
- **CORCON.ACCSAT** Selección del modo de saturación.

Las instrucciones que emplean el motor DSP son:

Instruction	Algebraic Operation
CLR	A = 0
ED	A = $(x - y)^2$
EDAC	A = A + $(x - y)^2$
MAC	A = A + (x * y)
MOVSAC	No change in A
MPY	A = x * y
MPY.N	A = – x * y
MSC	A = A – x * y

Diagrama del motor DSP:



Multiplicador

El multiplicador hardware de 17x17 bits permite multiplicaciones con o sin signo. Además a la salida dispone de un multiplexor para seleccionar el formato de salida, pudiendo elegir entre formato entero o formato fraccionario 1.31. Los datos de entrada se extienden a 17 bits con o sin signo dependiendo de la instrucción y la salida de 34 bits se extiende a 40 bits.

Cuando el multiplicador se configura para el formato fraccionario 1.31 los datos se representan en complemento a 2 donde el bit más significativo indicará el signo y el punto decimal se sitúa justo después. El rango de los números representados será desde $-1{,}0$

a $1 - 2^{1-N}$. Si $N = 32$ entonces irá desde $-1{,}0$ hasta 0,999999999 y la precisión será de $4{,}65662 * 10^{-10} = 2^{1-32}$.

Acumuladores de datos y sumador/restador

El acumulador de datos es un sumador/restador de 40 bits con extensión lógica de signo. Se puede seleccionar uno de los dos acumuladores (A ó B) como fuente y destino de una suma. Para las instrucciones **ADD** y **LAC**, los datos que se van a sumar se pueden escalar via el desplazador integrado.

Desbordamiento y saturación

El sumador/restador es de 40 bits y permite opcionalmente por una de las entradas un sumando nulo y por la otra entrada el segundo sumando o su complemento a dos (para restar). Genera cuatro señales que aparecen en el registro de estado **SR**:

- **SA/SB** que indican el desbordamiento del bit 39. Es un desbordamiento catastrófico ya que destruye el bit de signo. **SAB = SA | SB**.
- **OA/OB** que indican el desbordamiento en los bits de guarda (32 a 38). **OAB = OA|OB**. Se puede generar una excepción con **INTCON1.OVATE**=1 o **INTCON1.OVBTE**=1.

El sumador dispone además de un bloque de saturación que controla la saturación de los datos. Emplea el resultado del sumador así como los bits de desbordamiento y los bits de control de la saturación (**CORCON.SATA**, **CORCON.SATB** y **CORCON.ACCSAT**) para determinar cómo y cuándo se satura.

El sumador soporta tres modos de saturación:

- Desbordamiento del bit 39 y saturación. En este caso se satura al máximo valor positivo o negativo en formato 9.31. Se denomina **supersaturación** y proporciona protección frente a errores en los datos o resultados inesperados en los algoritmos.
- Desbordamiento del bit 31 y saturación. Se satura al máximo valor positivo o negativo en formato 1.31. En este modo no se emplean los bits de guarda.
- Desbordamiento catastrófico del bit 39. No se realiza ningún tipo de saturación y el signo se pierde. Se puede generar una excepción (**INTCON1.COVTE**) si se desea.

Acumulador Write-back

Las instrucciones del tipo **MAC** (con la excepción de **MPY**, **MPY.N**, **ED** y **EDAC**) pueden escribir una versión redondeada de la palabra alta del acumulador no usado por la instrucción en el espacio de memoria de datos. Esta escritura se realiza a través del bus X en el espacio combinado de direcciones X e Y. Soporta los siguientes modos de direccionamiento:

- Directo a registro **W13**. El contenido redondeado de la parte alta del acumulador no empleado en la instrucción se escribe en el registro **W13** en formato fraccionario 1.15.
- Indirecto a registro con post-incremento , **[W13]+=2**. El contenido redondeado de la parte alta del acumulador no empleado en la instrucción se escribe en la dirección apuntada por **W13** en formato 1.15 y **W13** se incrementa en dos unidades.

Lógica de redondeo

La lógica de redondeo es un bloque funcional que realiza un redondeo a la hora de escribir el resultado de manera convencional (polarizada) o convergente (no polarizada). El modo se determina con el campo `CORCON.RND`. Si se indica la saturación la parte alta se redondea antes de escribir el resultado. De lo contrario se escribe el valor truncado a la parte baja exclusivamente.

El **redondeo convencional** toma el bit 15 del acumulador y lo suma a `ACCxH`. Si el valor de `ACCxL` está entre `0x8000` y `0xFFFF`, `ACCxH` es incrementado, de lo contrario no se hace nada. Como consecuencia de este redondeo en promedio los redondeos tienden a ser ligeramente positivos.

El **redondeo convergente** opera igual que el anterior a excepción de cuando `ACCxL` vale `0x8000`. En este caso se examina el bit menos significativo de `ACCxH`. Si vale 1, `ACCxH` es incrementado, de lo contrario se deja como está. Este tipo de redondeo no tiene tendencia positiva como el anterior.

Saturación en la escritura en el espacio de datos

Los datos escritos en memoria también se pueden saturar, pero sin afectar al contenido de los acumuladores. El bloque de saturación en la escritura acepta valores de 16 bits en formato fraccionario 1.15.

El campo `CORCON.SATDW` controla este redondeo.

Desplazador

Es capaz de realizar desplazamientos lógicos o aritméticos a derecha o izquierda de hasta 16 bits en un único ciclo. Las fuentes pueden ser cualesquiera de los dos acumuladores o el espacio de datos X.

Un valor positivo en el número de bits a desplazar indicará desplazamiento a la derecha. El valor cero dejará como estaba el valor.

El desplazador es de 40 bits obteniendo valores de 40 bits para operaciones tipo DSP y obteniendo valores de 16 bits para operaciones tipo MCU.

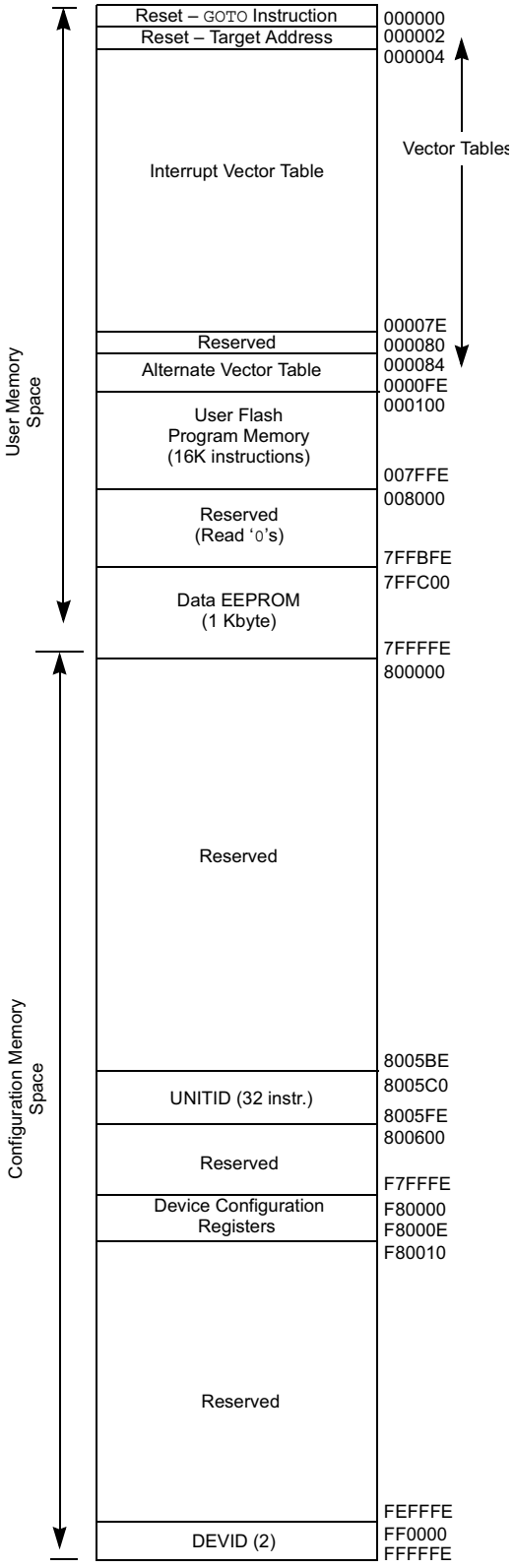
7.3. Organización de la memoria

7.3.1. Espacio de direcciones de programa

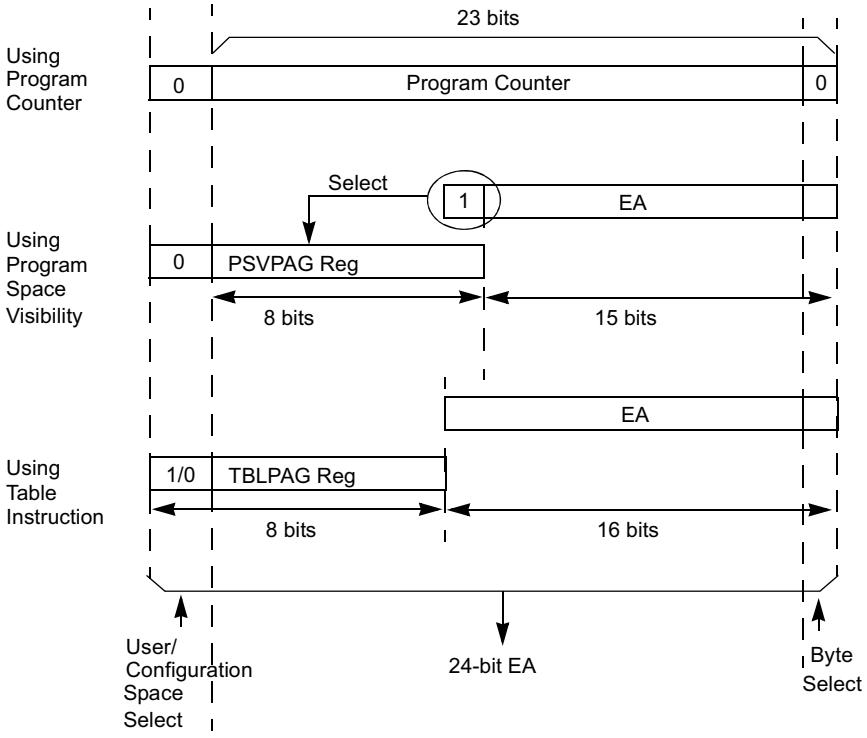
El espacio de direcciones de memoria de programa es de 4M palabras (de 16 bits). Se direccionan mediante el registro `PC` (23 bits), las instrucciones de tablas. El espacio de datos puede verse desde el espacio de programa.

Access Type	Access Space	Program Space Address				
		<23>	<22:16>	<15>	<14:1>	<0>
Instruction Access	User	0	PC<22:1>			0
TBLRD/TBLWT	User (TBLPAG<7> = 0)	TBLPAG<7:0>		Data EA<15:0>		
TBLRD/TBLWT	Configuration (TBLPAG<7> = 1)	TBLPAG<7:0>		Data EA<15:0>		
Program Space Visibility	User	0	PSVPAG<7:0>		Data EA<14:0>	

El acceso al espacio de programa de usuario está restringido a los 2M palabras inferiores con la excepción de las instrucciones de tablas (`TBLRD` , `TBLWT`) que emplean el campo **`TBLPAG`** para determinar el acceso a memoria de usuario o de configuración.



El acceso a datos desde el espacio de memoria de programa:

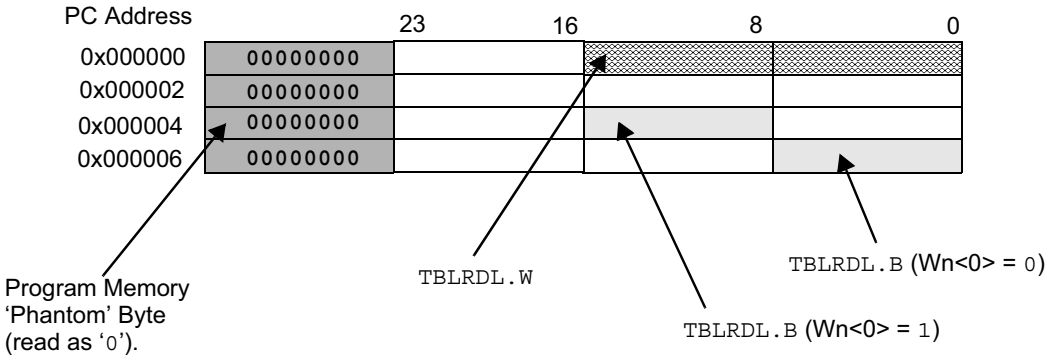


Dado que la arquitectura del DSP es Harvard modificada se puede acceder a datos en la memoria de programa.

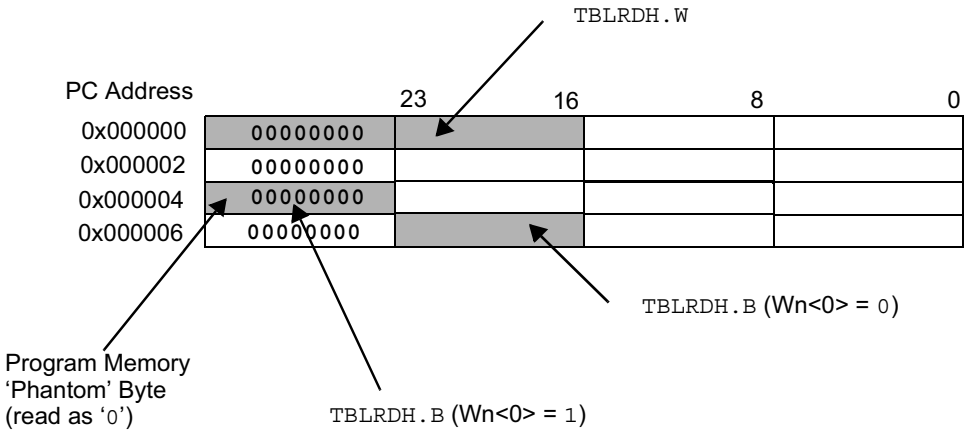
Hay dos métodos para acceder a la memoria de programa como si fuera de datos: a través de instrucciones especiales o remapeando una página de 16k palabras de la memoria de programa en la parte superior de la memoria de datos.

Acceso a datos en la memoria de programa: instrucciones de tabla

Las instrucciones **`TBLRDL`** y **`TBLWTL`** ofrecen un método directo de leer y escribir la palabra menos significativa (16 bits) de cualquier dirección de la memoria de programa.



Las instrucciones **`TBLRDH`** y **`TBLWTH`** permiten acceder a los 8 bits superiores de cada dirección de la memoria de programa.

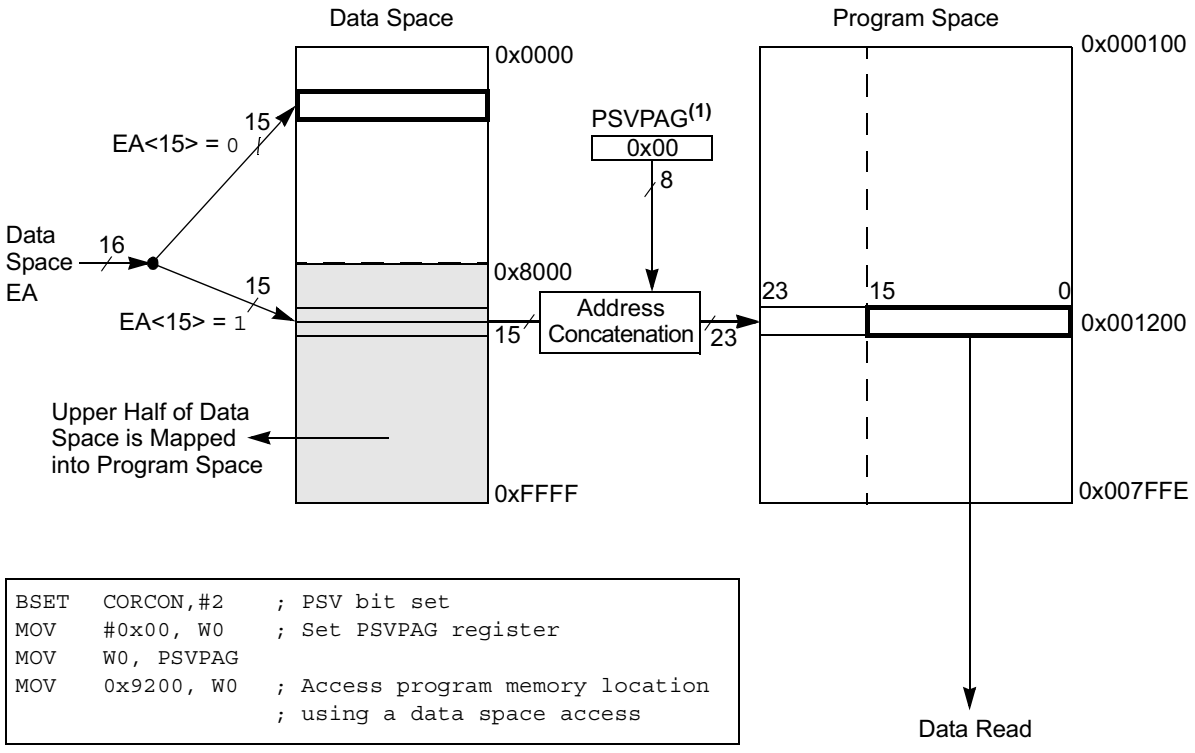


Acceso a datos en la memoria de programa: visibilidad del espacio de programa

Las 32KB superiores de la memoria de datos pueden mapearse en cualquier página de 16K palabras de la memoria de programa. Esto proporciona un acceso transparente a datos almacenados en la memoria de programa a través del espacio de datos X. Esto ocurre si el bit más significativo de la dirección efectiva (EA) vale 1 y el campo **CORCON.PSV** está activado.

Estos accesos añaden un ciclo extra a ejecución de la instrucción, luego se requieren dos ciclos de búsqueda en memoria.

El acceso concatena el registro **PSVPAG** con los 16 bits de la dirección efectiva como se ve en la siguiente figura:




```
BSET   CORCON,#2    ; PSV bit set
MOV    #0x00, W0    ; Set PSVPAG register
MOV    W0, PSVPAG
MOV    0x9200, W0   ; Access program memory location
                    ; using a data space access
```

Note: PSVPAG is an 8-bit register containing bits <22:15> of the program space address (i.e., it defines the page in program space to which the upper half of data space is being mapped).

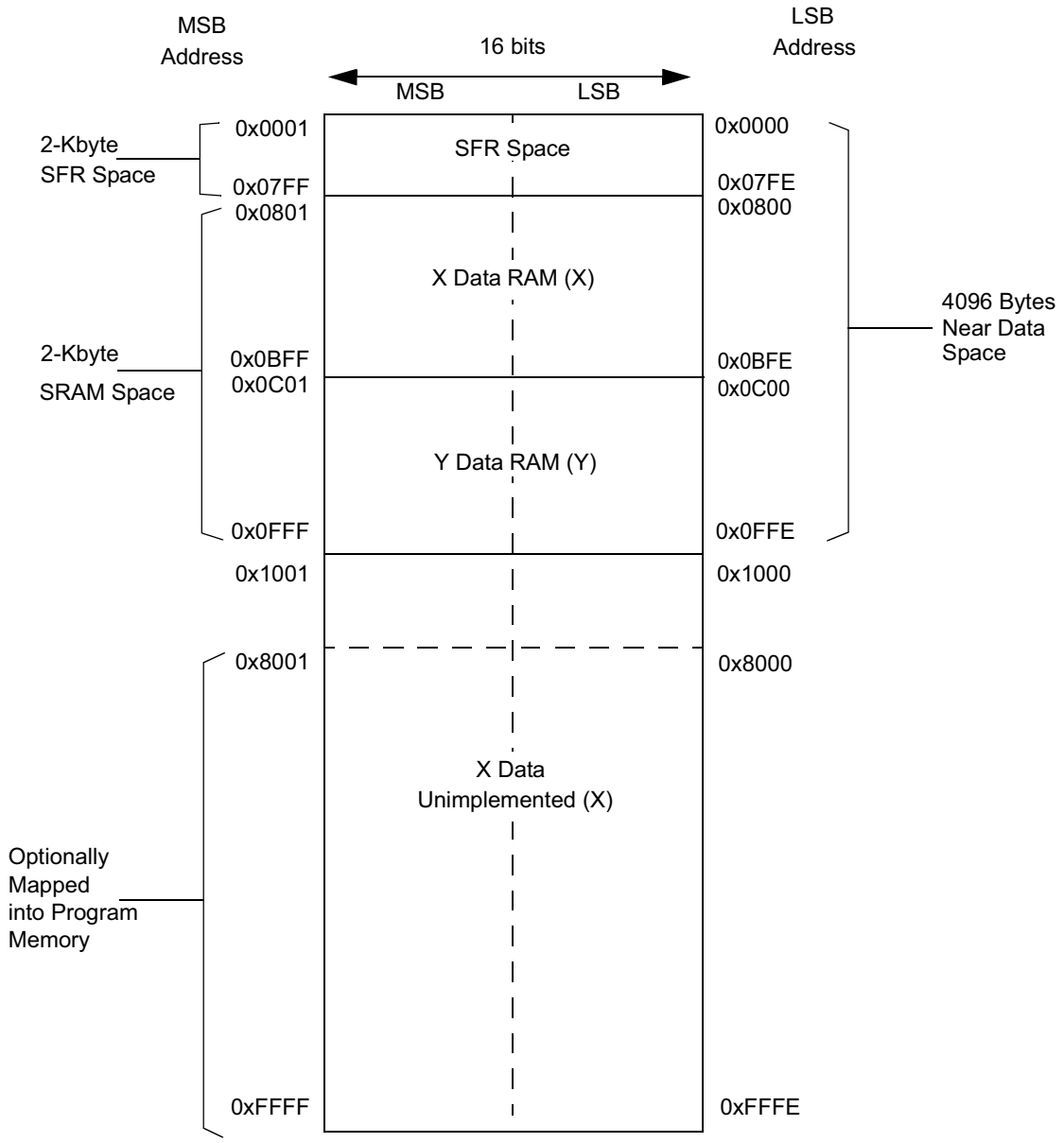
7.3.2. Espacio de direcciones de datos

El núcleo del DSP dispone de dos espacios de direcciones de la memoria de datos. Se consideran espacios separados (para algunas instrucciones tipo DSP) o un rango único de direcciones (para las instrucciones de la MCU). Los espacios de datos se acceden a través de dos unidades generadoras de direcciones (AGUs) con buses separados.

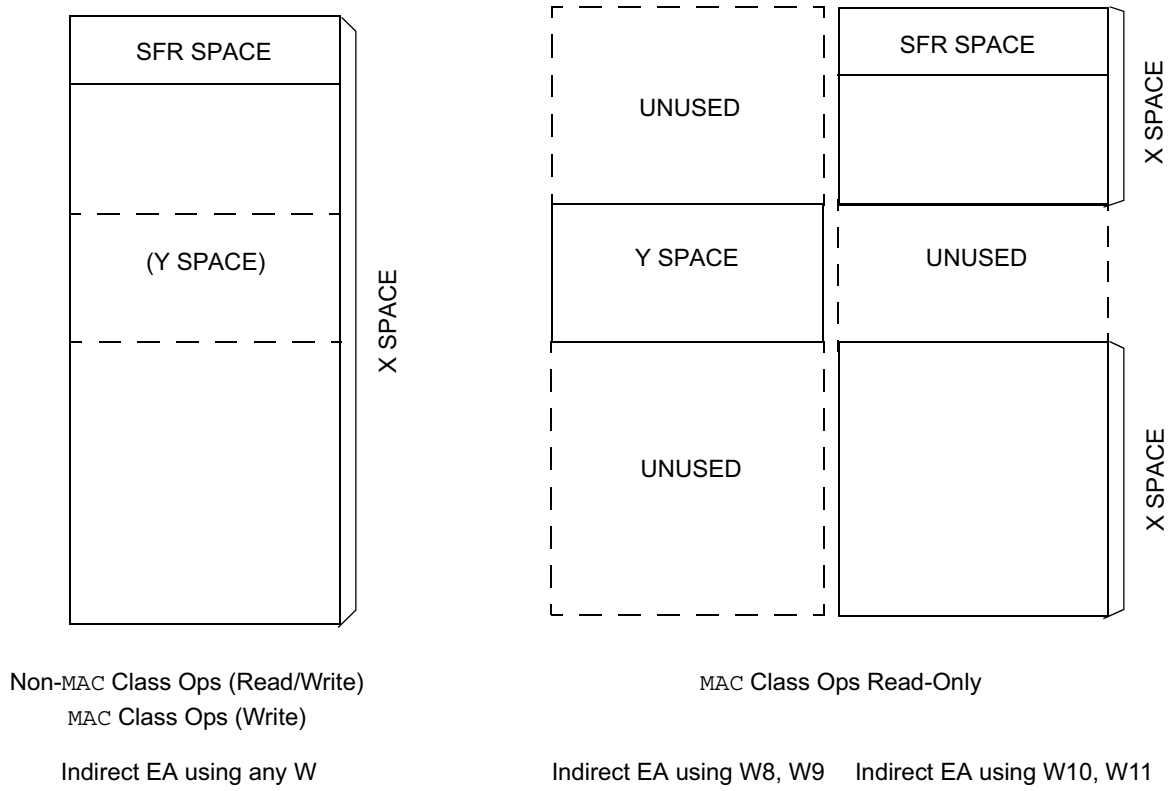
Mapa de memoria del espacio de datos

La memoria de datos se divide en dos bloques denominados X e Y. Un elemento clave de esta arquitectura es que el espacio de direcciones Y es un subconjunto del espacio X y está totalmente contenido en él.

Cuando se ejecuta cualquier instrucción a excepción de las tipo **MAC**, el bloque X consiste en un espacio de direcciones de 64 KB (que incluye el bloque Y). Cuando se ejecutan instrucciones del tipo **MAC** el bloque X consiste en 64 KB excluyendo las direcciones comprendidas en el bloque Y (solo lectura). El bloque Y se direcciona a través de la dirección efectiva formada por los registro `W10` y `W11`, mientras que el bloque X se direcciona mediante los registros `W8` y `W9`. Ambas direcciones se acceden de forma concurrente solo para las instrucciones tipo **MAC**.



Resumen del acceso a los bloques X e Y:



Espacios de datos

El espacio X es empleado por todas las instrucciones y soporta todos los modos de direccionamiento. Dispone de buses para la lectura y la escritura separados.

El bloque X soporta el direccionamiento modular o circular para todas las instrucciones. El modo de direccionamiento de bit invertido solo es soportado en el bloque X.

El espacio de datos Y se usa en concierto con el espacio X en las instrucciones tipo **MAC** (**CLR**, **ED**, **EDAC**, **MAC**, **MOVSAC MPY**, **MPY.N** y **MSC**) para proporcionar dos caminos de lectura concurrentes.

La definición de los espacios X e Y es fijo y no se puede cambiar. Si se intenta acceder fuera de los límites se devuelve un valor nulo (`0x0000`).

Anchura del espacio de datos

La anchura de la memoria de datos es de 16 bits.

Alineamiento de datos

Los dsPIC soportan el acceso a la memoria (*little endian*) de datos tanto de 16 como de 8 bits. Los datos se alinean en memoria como palabras de 16 bits, aunque las direcciones efectivas apuntan a bytes. Los accesos de lecturas siempre leen palabras y se selecciona el byte teniendo en cuenta el bit menos significativo de la dirección. Esto solo es posible en el espacio de direcciones X.

Los direccionamientos de post-modificación incrementarán los registros en una o dos unidades dependiendo del tipo de acceso (byte o word).

Los accesos no alineados a palabras no están soportados, luego hay que tenerlo en cuenta. Si se intenta, se genera una excepción de error en la dirección.

Espacio de datos cercano

Un bloque de 8KB de datos cercanos es directamente accesible con las instrucciones de direccionamiento directo a memoria. El resto de direcciones, incluyendo el espacio Y, se puede acceder de manera indirecta. La instrucción **MOV** soporta el direccionamiento directo a memoria con direcciones de 16 bits. (64 KB).

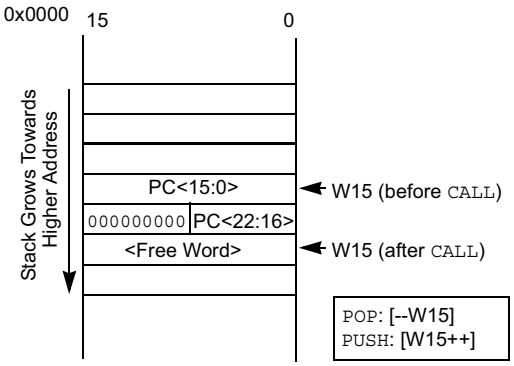
Pila software

El puntero de pila para la implementación de una pila software en los dsPIC se consigue con el registro **W15**. Apunta siempre a la primera palabra libre y crece hacia posiciones de memoria crecientes. El puntero se pre-decrementa con las instrucciones que extraen datos de la pila y se post-incrementa para la introducción de datos en la pila. Al ejecutar instrucciones de llamada a subrutina (**CALL**) se guarda la dirección de retorno en la pila extendiendo la parte más significativa con ceros.

Existe un registro límite del puntero de pila (**SPLIM**) asociado con el puntero de pila. Este valor no está inicializado al arrancar el DSP. Si se sobrepasa (en más de dos unidades) este límite se genera una excepción de error de pila.

Un error de pila se genera también si extraemos datos de la pila por debajo de la dirección `0x800` para evitar meternos en la zona de los registros de función especial mapeados en la memoria de datos.

Una escritura en el registro **SPLIM** no debe ser seguida de una operación con **W15**.



7.3.3. Registros mapeados en el espacio de datos

SFR Name	Address (Home)	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
W0	0000	W0/WREG																0000 0000 0000 0000
W1	0002	W1																0000 0000 0000 0000
W2	0004	W2																0000 0000 0000 0000
W3	0006	W3																0000 0000 0000 0000
W4	0008	W4																0000 0000 0000 0000
W5	000A	W5																0000 0000 0000 0000
W6	000C	W6																0000 0000 0000 0000
W7	000E	W7																0000 0000 0000 0000
W8	0010	W8																0000 0000 0000 0000
W9	0012	W9																0000 0000 0000 0000
W10	0014	W10																0000 0000 0000 0000
W11	0016	W11																0000 0000 0000 0000
W12	0018	W12																0000 0000 0000 0000
W13	001A	W13																0000 0000 0000 0000
W14	001C	W14																0000 0000 0000 0000
W15	001E	W15																0000 1000 0000 0000
SPLIM	0020	SPLIM																0000 0000 0000 0000
ACCAL	0022	ACCAL																0000 0000 0000 0000
ACCAH	0024	ACCAH																0000 0000 0000 0000
ACCAU	0026	Sign Extension (ACCA<39>)								ACCAU								0000 0000 0000 0000
ACCBL	0028	ACCBL																0000 0000 0000 0000
ACCBH	002A	ACCBH																0000 0000 0000 0000
ACCBU	002C	Sign Extension (ACCB<39>)								ACCBU								0000 0000 0000 0000
PCL	002E	PCL																0000 0000 0000 0000
PCH	0030	—	—	—	—	—	—	—	—	—	PCH							0000 0000 0000 0000
TBLPAG	0032	—	—	—	—	—	—	—	—	TBLPAG								0000 0000 0000 0000
PSVPAG	0034	—	—	—	—	—	—	—	—	PSVPAG								0000 0000 0000 0000
RCOUNT	0036	RCOUNT																uuuu uuuu uuuu uuuu
DCOUNT	0038	DCOUNT																uuuu uuuu uuuu uuuu
DOSTARTL	003A	DOSTARTL															0	uuuu uuuu uuuu uuu0
DOSTARTH	003C	—	—	—	—	—	—	—	—	—	DOSTARTH							0000 0000 0uuu uuuu
DOENDL	003E	DOENDL															0	uuuu uuuu uuuu uuu0
DOENDH	0040	—	—	—	—	—	—	—	—	—	DOENDH							0000 0000 0uuu uuuu
SR	0042	OA	OB	SA	SB	OAB	SAB	DA	DC	IPL2	IPL1	IPL0	RA	N	OV	Z	C	0000 0000 0000 0000
CORCON	0044	—	—	—	US	EDT	DL2	DL1	DL0	SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF	0000 0000 0010 0000
MODCON	0046	XMODEN	YMODEN	—	—	BWM<3:0>				YWM<3:0>				XWM<3:0>				0000 0000 0000 0000
XMODSRT	0048	XS<15:1>															0	uuuu uuuu uuuu uuu0

Legend: u = uninitialized bit

Note: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

SFR Name	Address (Home)	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
XMODEND	004A	XE<15:1>															1	uuuu uuuu uuuu uuu1
YMODSRT	004C	YS<15:1>															0	uuuu uuuu uuuu uuu0
YMODEND	004E	YE<15:1>															1	uuuu uuuu uuuu uuu1
XBREV	0050	BREN	XB<14:0>															uuuu uuuu uuuu uuuu
DISICNT	0052	—	—	DISICNT<13:0>														0000 0000 0000 0000

Legend: u = uninitialized bit

Note: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

7.4. Unidades generadoras de direcciones

El DSP contiene dos unidades generadoras de direcciones (AGUs) y soportan tres modos de direccionamiento fundamentales:

- Direccionamientos lineales.
- Direccionamiento modular.
- Direccionamiento de bit invertido.

Los modos lineales y modulares se aplican tanto a la memoria de datos como de programa. El modo de bit invertido solo se aplica a la memoria de datos.

7.4.1. Modos de direccionamiento lineales

Se resumen en la siguiente tabla:

Addressing Mode	Description
File Register Direct	The address of the file register is specified explicitly.
Register Direct	The contents of a register are accessed directly.
Register Indirect	The contents of Wn forms the EA.
Register Indirect Post-Modified	The contents of Wn forms the EA. Wn is post-modified (incremented or decremented) by a constant value.
Register Indirect Pre-Modified	Wn is pre-modified (incremented or decremented) by a signed constant value to form the EA.
Register Indirect with Register Offset	The sum of Wn and Wb forms the EA.
Register Indirect with Literal Offset	The sum of Wn and a literal forms the EA.

Instrucciones con acceso directo al archivo de registros

Acceden directamente a los 8KB primeros de la memoria de datos.

Instrucciones MCU

Las operaciones de tres operandos: Op3 = Op1 (función) Op2. El Op1 siempre es un registro de trabajo. El segundo operando Op2 puede ser un registro de trabajo o un dato literal inmediato de 5 bit. El resultado, Op3, puede ser un registro de trabajo o una dirección de memoria de datos.

Soportan los siguientes modos de direccionamiento:

- Directo a registro.
- Indirecto.
- Indirecto post-modificado.
- Indirecto pre-modificado.
- Literal de 5 o 10 bit.

Instrucciones MOV y con acumulador

Las instrucciones de copia de datos y las de acumulación del motor DSP permiten más modos de direccionamiento que las otras instrucciones.

En resumen:

- Directo a registro.
- Indirecto.
- Indirecto post-modificado.
- Indirecto pre-modificado.

- Indirecto con registro (offset).
- Indirecto con offset.
- Literal de 8 o 16 bit.

Instrucciones MAC

Los dos operandos fuente empleados por las instrucciones del motor DSP deben ser: `W8` y `W9` apuntando al espacio de direcciones X; `W10` y `W11` apuntando al espacio de direcciones Y.

Los modos de direccionamiento permitidos son:

- Indirecto.
- Indirecto post-modificado en 2.
- Indirecto post-modificado en 4.
- Indirecto post-modificado en 6.
- Indirecto con registro offset.

7.4.2. Modo de direccionamiento modular

El objetivo de este modo de direccionamiento es eliminar la necesidad de realizar las comprobaciones de límites en el acceso a un conjunto de datos dentro de un bucle repetitivo.

Este modo funciona tanto en la memoria de programa como en la memoria de datos (uno para el espacio X y otro para el Y). Puede funcionar teniendo como puntero cualquiera de los registros W, sin embargo, no es recomendable emplear los registros `W14` y `W15` dada la función asociada que tienen.

Espacio de datos X

Se necesita:

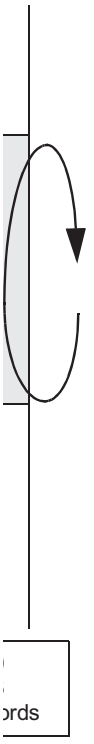
- Dirección de comienzo en el registro `XMODSRT`. Debe cumplir que comience en una dirección que tenga los N bits menos significativos a cero ($N = log_2(S)$), donde S es el tamaño del búfer) siempre que sea un bucle circular donde se incremente el puntero.
- Dirección final del búfer en el registro `XMODEND`. Debe cumplir que sea una dirección que tenga los N bits menos significativos a uno ($N = log_2(S)$), donde S es el tamaño del búfer) siempre que sea un bucle circular donde se decremente el puntero.
- Activación del direccionamiento modular `MODCON.XMODEN = 1`
- Selección del registro W asociado al direccionamiento circular con `MODCON.XWM`. Si vale 15 se desactiva el direccionamiento circular.

Espacio de datos Y

Se necesita:

- Dirección de comienzo en el registro **YMODSRT**. Debe cumplir que comience en una dirección que tenga los N bits menos significativos a cero ($N = log_2(S)$), donde S es el tamaño del búfer) siempre que sea un bucle circular donde se incremente el puntero.
- Dirección final del búfer en el registro **YMODEND**. Debe cumplir que sea una dirección que tenga los N bits menos significativos a uno ($N = log_2(S)$), donde S es el tamaño del búfer) siempre que sea un bucle circular donde se decremente el puntero.
- Activación del direccionamiento modular **MODCON**.**YMODEN** = 1
- Selección del registro W asociado al direccionamiento circular con **MODCON**.YWM. Si vale 15 se desactiva el direccionamiento circular.

Ejemplo




```
MOV     #0x1100,W0
MOV     W0, XMODSRT     ;set modulo start address
MOV     #0x1163,W0
MOV     W0,XMODEND       ;set modulo end address
MOV     #0x8001,W0
MOV     W0,MODCON       ;enable W1, X AGU for modulo
MOV     #0x0000,W0      ;W0 holds buffer fill value
MOV     #0x1110,W1      ;point W1 to buffer
DO      AGAIN,#0x31     ;fill the 50 buffer locations
MOV     W0, [W1++]      ;fill the next location
AGAIN:  INC W0,W0       ;increment the fill value
```

Registro MODCON

MODCON: Modulo and Bit-Reversed Addressing Control Register

Upper Byte:							
R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
XMODEN	YMODEN	—	—	BWM<3:0>			
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YWM<3:0>				XWM<3:0>			
bit 7							bit 0

Cuyo contenido es:

bit 15 **XMODEN:** X RAGU and X WAGU Modulus Addressing Enable bit
`1` = X AGU modulus addressing enabled
`0` = X AGU modulus addressing disabled

bit 14 **YMODEN:** Y AGU Modulus Addressing Enable bit
`1` = Y AGU modulus addressing enabled
`0` = Y AGU modulus addressing disabled

bit 13-12 **Unimplemented:** Read as '0'

bit 11-8 **BWM<3:0>:** X WAGU Register Select for Bit-Reversed Addressing bits
`1111` = Bit-reversed addressing disabled
`1110` = W14 selected for bit-reversed addressing
`1101` = W13 selected for bit-reversed addressing
•
•
`0000` = W0 selected for bit-reversed addressing

bit 7-4 **YWM<3:0>:** Y AGU W Register Select for Modulo Addressing bits
`1111` = Modulo addressing disabled
`1010` = W10 selected for modulo addressing
`1011` = W11 selected for modulo addressing

Note: All other settings of the YWM<3:0> control bits are reserved and should not be used.

bit 3-0 **XWM<3:0>:** X RAGU and X WAGU W Register Select for Modulo Addressing bits
`1111` = Modulo addressing disabled
`1110` = W14 selected for modulo addressing
•
•
`0000` = W0 selected for modulo addressing

Note: A write to the MODCON register should not be followed by an instruction that performs an indirect read operation using a W register. Unexpected results may occur. Some instructions perform an implicit indirect read. These are: `POP`, `RETURN`, `RETFIE`, `RETLW` and `ULNK`.

7.4.3. Modo de direccionamiento de bit invertido

El modo de direccionamiento de bit invertido está pensado para simplificar la reordenación de datos en las transformadas rápidas de Fourier (radix-2). Solo es soportado por la unidad de generación de direcciones para escrituras del espacio X.

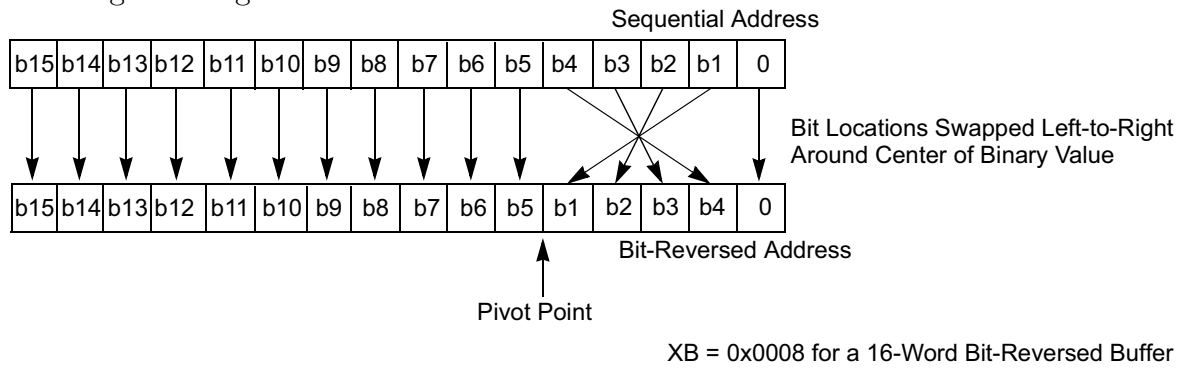
Este modo se activa cuando:

- `MODCON.BWM3:0` contiene un valor diferente de 15.
- `XBREV.BREN = 1`.
- El modo de direccionamiento empleado es el indirecto a registro con pre o post incremento.

Si la longitud del búfer accedido es $M = 2^N$, entonces los N bits menos significativos de la dirección de comienzo del búfer deben ser ceros.

El campo `XBREV.XB14:0` define el punto de pivote. En una FFT este valor debe coincidir con el tamaño mitad del búfer.

En la siguiente figura se observa el funcionamiento:



El contenido del registro `XBREV`:

XBREV: X Write AGU Bit-Reversal Addressing Control Register

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BREN	XB<14:8>						
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XB<7:0>							
bit 7							bit 0

bit 15 **BREN:** Bit-Reversed Addressing (X AGU) Enable bit
1 = Bit-reversed addressing enabled
0 = Bit-reversed addressing disabled

bit 14-0 **XB<14:0>:** X AGU Bit-Reversed Modifier bits
0x4000 = 32768 word buffer
0x2000 = 16384 word buffer
0x1000 = 8192 word buffer
0x0800 = 4096 word buffer
0x0400 = 2048 word buffer
0x0200 = 1024 word buffer
0x0100 = 512 word buffer
0x0080 = 256 word buffer
0x0040 = 128 word buffer
0x0020 = 64 word buffer
0x0010 = 32 word buffer
0x0008 = 16 word buffer
0x0004 = 8 word buffer
0x0002 = 4 word buffer
0x0001 = 2 word buffer

7.5. Interrupciones y Excepciones

7.5.1. Introducción

Las interrupciones y excepciones son acontecimientos externos o internos que provocan la desviación del flujo de programa, que abandona el programa principal y pasa a ejecutar una rutina que atiende la causa que lo ha originado.

Las interrupciones son debidas a sucesos externos y las excepciones son generadas internamente como consecuencia de la ejecución de la alguna instrucción.

Para manejar las interrupciones y las excepciones los dsPIC30F disponen de una Tabla de Vectores de Interrupción llamada IVT[7] que contiene las direcciones de salto de las 62 fuentes diferentes de interrupción (54) ó excepción (8).

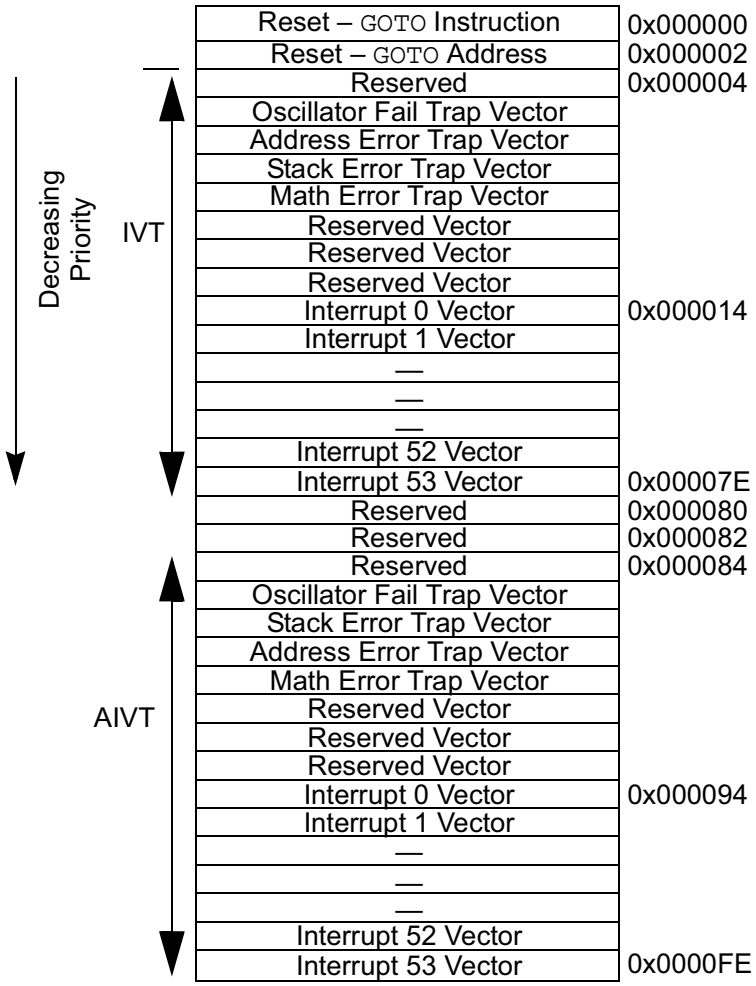
Además de la IVT, se dispone de una tabla alternativa de vectores de interrupción denominada AIVT[8], de estructura similar a la IVT que entra en funcionamiento cuando es seleccionada en vez de la principal (`INTCON2.ALTIVT=1`, típicamente para soporte de emulación y depuración).

Una característica destacable en el control de interrupciones de los dsPIC es el uso de prioridades en el manejo de las interrupciones.

7.5.2. Tabla de vectores de interrupción: IVT

Cada tipo de interrupción está asociado a una de las 62 entradas de la tabla IVT.

[7] *Interrupt Vector Table*: Tabla de Vectores de Interrupción.
[8] *Alternate Interrupt Vector Table*: Tabla de Vectores de Interrupción Alternativa.



Esta tabla reside en la memoria de programa, comenzando en la dirección 0x00000004. Los 8 primeros vectores de la IVT están reservados para excepciones o interrupciones no enmascarables, los 54 restantes para interrupciones enmascarables. Cada vector de interrupción guarda los 24 bits de la dirección de comienzo de la rutina de excepción.

El tratamiento de las interrupciones lo lleva una rutina asociada llamada Rutina de Servicio de Interrupción (ISR), que se pone en marcha cuando la CPU atiende una interrupción.

Vector Number	IVT Address	AIVT Address	Interrupt Source
8	0x000014	0x000094	INT0 – External Interrupt 0
9	0x000016	0x000096	IC1 – Input Compare 1
10	0x000018	0x000098	OC1 – Output Compare 1
11	0x00001A	0x00009A	T1 – Timer 1
12	0x00001C	0x00009C	IC2 – Input Capture 2
13	0x00001E	0x00009E	OC2 – Output Compare 2
14	0x000020	0x0000A0	T2 – Timer 2
15	0x000022	0x0000A2	T3 – Timer 3
16	0x000024	0x0000A4	SPI1
17	0x000026	0x0000A6	U1RX – UART1 Receiver
18	0x000028	0x0000A8	U1TX – UART1 Transmitter
19	0x00002A	0x0000AA	ADC – ADC Convert Done
20	0x00002C	0x0000AC	NVM – NVM Write Complete
21	0x00002E	0x0000AE	I^2C Slave Operation – Message Detect
22	0x000030	0x0000B0	I^2C Master Operation – Message Event Complete
23	0x000032	0x0000B2	Change Notice Interrupt
24	0x000034	0x0000B4	INT1 – External Interrupt 1
25	0x000036	0x0000B6	IC7 – Input Capture 7
26	0x000038	0x0000B8	IC8 – Input Capture 8
27	0x00003A	0x0000BA	OC3 – Output Compare 3
28	0x00003C	0x0000BC	OC4 – Output Compare 4
29	0x00003E	0x0000BE	T4 – Timer 4
30	0x000040	0x0000C0	T5 – Timer 5
31	0x000042	0x0000C2	INT2 – External Interrupt 2
32	0x000044	0x0000C4	U2RX – UART2 Receiver
33	0x000046	0x0000C6	U2TX – UART2 Transmitter
34	0x000048	0x0000C8	SPI2
35	0x00004A	0x0000CA	CAN1
36	0x00004C	0x0000CC	IC3 – Input Capture 3
37	0x00004E	0x0000CE	IC4 – Input Capture 4
38	0x000050	0x0000D0	IC5 – Input Capture 5
39	0x000052	0x0000D2	IC6 – Input Capture 6
40	0x000054	0x0000D4	OC5 – Output Compare 5
41	0x000056	0x0000D6	OC6 – Output Compare 6
42	0x000058	0x0000D8	OC7 – Output Compare 7
43	0x00005A	0x0000DA	OC8 – Output Compare 8
44	0x00005C	0x0000DC	INT3 – External Interrupt 3
45	0x00005E	0x0000DE	INT4 – External Interrupt 4
46	0x000060	0x0000E0	CAN2
47	0x000062	0x0000E2	PWM – PWM Period Match
48	0x000064	0x0000E4	QEI – Position Counter Compare
49	0x000066	0x0000E6	DCI – Codec Transfer Done
50	0x000068	0x0000E8	LVD – Low Voltage Detect
51	0x00006A	0x0000EA	FLTA – MCPWM Fault A
52	0x00006C	0x0000EC	FLTB – MCPWM Fault B
53-61	0x00006E-0x00007E	0x00006E-0x00007E	Reserved

7.5.3. Prioridades

Las excepciones y las interrupciones admiten 16 niveles de prioridad, del nivel 0 al nivel 15.

Para poder iniciar el proceso de atención a interrupciones o excepciones la causa o fuente que llama a la CPU debe tener un nivel de prioridad mayor que el de la CPU en ese instante. Inicialmente el nivel de la CPU será el más bajo de todos, el nivel 0. Este nivel podrá ser modificado de forma manual escribiendo sobre los bits `IPL[3..0]` (formados por `CORCON.IPL3` y `SR.IPL2..IPL0`). La CPU también podrá modificar este nivel automáticamente cuando se está ejecutando una ISR y adopta el nivel de privilegio de la interrupción que se está ejecutando. Al acabar esta se volverá al nivel que tenía anteriormente.

Los 16 niveles se dividen en dos grupos:

- Del 0 al nivel 7: Utilizados para las interrupciones de los periféricos (externos a la CPU). La CPU tiene uno de estos niveles. En caso de coincidencia de nivel de prioridad, cada fuente de interrupción tiene un orden de prioridad natural: se corresponde con qué vector de interrupción se encuentra en una dirección de memoria más baja (tendrá mayor prioridad).

- Del 8 al nivel 15: Excepciones internas, no enmascarables, que se tienen que atender en el momento que se producen.

Las excepciones tienen niveles de prioridad superior a las interrupciones por tratarse de interrupciones no enmascarables.

Todas las interrupciones se pueden deshabilitar poniendo IPL=111, con lo que quedarían deshabilitadas las interrupciones externas que sean inferiores al nivel 7. Las excepciones no se deshabilitan por tener nivel superior a 7.

El bit `CORCON.IPL3` nos indica si la interrupción es enmascarable o no. Si `CORCON.IPL3` está activo, significa que una excepción está en proceso.

7.5.4. Excepciones

Las excepciones se ejecutan siempre y tienen la función de avisar al usuario de que se ha producido una operación errónea y que debe corregirla, tanto a nivel software como hardware.

Las excepciones sólo se detectan cuando ya han ocurrido. Si es una excepción software se termina la instrucción antes de lanzar la excepción. Si es una excepción hardware no se continúa y se salta a la excepción.

Existen cuatro fuentes de excepción, agrupadas en excepciones software y excepciones hardware:

Excepciones software

- **Fallo de pila.** (nivel de prioridad 12). Cuando se escribe el registro límite de pila, `SPLIM`, se activa la detección de sobrepasamiento de la pila y cuando se intenta escribir en una dirección que está fuera del límite se genera una excepción. Este tipo de excepción se detecta por la activación del bit `INTCON1.STKERR`.

  Si la pila decrece por debajo de la dirección `0x0800` también se genera una excepción de pila.

- **Error aritmético.** (nivel de prioridad 11). Son excepciones causadas por:
  - Sobrepasamiento del acumulador A. Se habilita con `INTCON1.OVATE = 1`.
  - Sobrepasamiento del acumulador B. Se habilita con `INTCON1.OVBTE = 1`.
  - Sobrepasamiento catastrófico del acumulador A ó B. Se habilita con `INTCON1.COVTE = 1`.
  - División por cero. No se puede desactivar.
  - El desplazamiento del acumulador (instrucción `SPTAC`) excede en ±16 bits. No puede ser desactivado.

  Estas excepciones se detectan mediante el campo `INTCON1.MATHERR` y después de atenderse debe ser borrada la notificación. Además se borrarán los flags `SR.OA` y/o `SR.OB` si se activaron.

Excepciones hardware

- Error en el direccionamiento (nivel prioridad 13). Se señaliza con `INTCON1.ADDRERR = 1`.
- Fallo del oscilador (nivel prioridad 14). Se señaliza con `INTCON1.OSCFAIL = 1`.

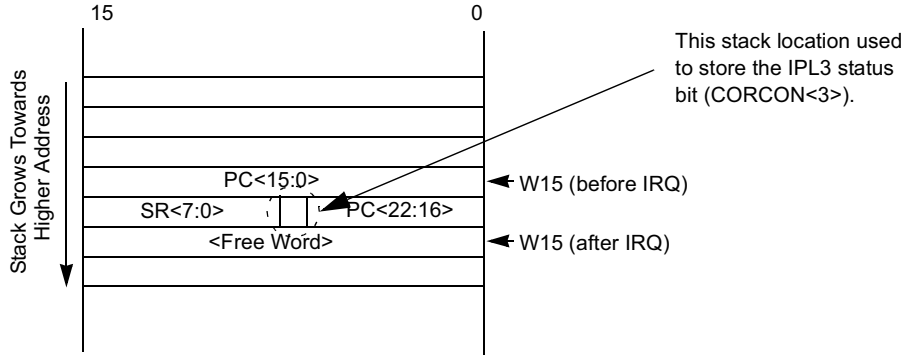
Cuando ocurre un conflicto de excepciones hardware la máquina se reinicia automáticamente y el bit de estado `RCON.TRAPR` se pone a 1.

7.5.5. Gestión de interrupciones

Cuando hay una petición de interrupción pendiente (IRQ[9]) se señaliza poniendo el bit correspondiente a la fuente de interrupción a 1 en el registro `IFSx`. Esta petición será atendida si el bit correspondiente de `IECx` está a 1. Las instrucciones en curso se completan antes de atender la IRQ.

Se evalúan los niveles de prioridad. Si hay una IRQ con un nivel de prioridad mayor que el proceso actual tiene en la CPU, la interrupción se atenderá. Entonces el procesador deberá salvar la siguiente información, que permitirá al procesador el retorno de la instrucción en curso:

- El valor del Contador de Programa, `PC`
- El byte bajo del registro de estado: `SRL`
- El bit de estado `CORCON.IPL3`



[9] *Interrupt ReQuest*: Petición de Interrupción.

Interrupciones anidadas

Las interrupciones son anidables por defecto (al contrario que en la mayoría de microcontroladores o DSPs). Cualquier ISR que esté en proceso puede ser interrumpida por otra fuente de interrupción con mayor nivel de prioridad.

Si se está ejecutando una interrupción de prioridad baja y ocurre una de mayor prioridad, esta se suspenderá y se realizará la de mayor prioridad; pero si la que primero se está ejecutando es la de mayor prioridad y ocurre una de menor prioridad esta no puede ser atendida hasta que la de mayor prioridad termine.

Si el bit `INTCON1.NSTDIS = 1`, las interrupciones anidadas se prohíben y entonces se fuerza a la CPU al nivel 7, `IPL[2:0] = 111b`, descartándose todas las interrupciones de menor nivel al 7.

Despertar la CPU del estado SLEEP o IDLE

Las interrupciones pueden despertar el procesador. Cuando despierta del modo SLEEP o IDLE ocurre una de estas dos acciones:

1. Si el nivel de prioridad de la interrupción es mayor que la prioridad de la CPU, entonces el procesador atenderá la interrupción.

2. Si es menor o igual que el de la CPU entonces continuará la ejecución comenzando con la instrucción siguiente a la que provocó el estado SLEEP o IDLE en la CPU.

7.5.6. Registros de control de interrupciones

Los registros que controlan las interrupciones y su estado se dividen en seis grupos:

Registros INTCON1 e INTCON2

Controlan las funciones globales de las interrupciones.

1. `INTCON1`. Contiene los flags y bits de control de las excepciones y el bit `NSTDIS` que prohíbe el anidamiento de interrupciones.

2. `INTCON2`. Contiene los bits para el control de las interrupciones externas y el bit que activa la AIVT.

INTCON1: Interrupt Control Register 1

Upper Byte:							
R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE
bit 15							bit 8

Lower Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—
bit 7							bit 0

bit 15 **NSTDIS:** Interrupt Nesting Disable bit
1 = Interrupt nesting is disabled
0 = Interrupt nesting is enabled

bit 14-11 **Unimplemented:** Read as '0'

bit 10 **OVATE:** Accumulator A Overflow Trap Enable bit
1 = Trap overflow of Accumulator A
0 = Trap disabled

bit 9 **OVBTE:** Accumulator B Overflow Trap Enable bit
1 = Trap overflow of Accumulator B
0 = Trap disabled

bit 8 **COVTE:** Catastrophic Overflow Trap Enable bit
1 = Trap on catastrophic overflow of Accumulator A or B enabled
0 = Trap disabled

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **MATHERR:** Arithmetic Error Status bit
1 = Overflow trap has occurred
0 = Overflow trap has not occurred

bit 3 **ADDRERR:** Address Error Trap Status bit
1 = Address error trap has occurred
0 = Address error trap has not occurred

bit 2 **STKERR:** Stack Error Trap Status bit
1 = Stack error trap has occurred
0 = Stack error trap has not occurred

bit 1 **OSCFAIL:** Oscillator Failure Trap Status bit
1 = Oscillator failure trap has occurred
0 = Oscillator failure trap has not occurred

bit 0 **Unimplemented:** Read as '0'

INTCON2: Interrupt Control Register 2

Upper Byte:							
R/W-0	R-0	U-0	U-0	U-0	U-0	U-0	U-0
ALTIVT	DISI	—	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
bit 7							bit 0

bit 15	**ALTIVT:** Enable Alternate Interrupt Vector Table bit 1 = Use alternate vector table 0 = Use standard (default) vector table
bit 14	**DISI:** DISI Instruction Status bit 1 = DISI instruction is active 0 = DISI is not active
bit 13-5	**Unimplemented:** Read as '0'
bit 4	**INT4EP:** External Interrupt #4 Edge Detect Polarity Select bit 1 = Interrupt on negative edge 0 = Interrupt on positive edge
bit 3	**INT3EP:** External Interrupt #3 Edge Detect Polarity Select bit 1 = Interrupt on negative edge 0 = Interrupt on positive edge
bit 2	**INT2EP:** External Interrupt #2 Edge Detect Polarity Select bit 1 = Interrupt on negative edge 0 = Interrupt on positive edge
bit 1	**INT1EP:** External Interrupt #1 Edge Detect Polarity Select bit 1 = Interrupt on negative edge 0 = Interrupt on positive edge
bit 0	**INT0EP:** External Interrupt #0 Edge Detect Polarity Select bit 1 = Interrupt on negative edge 0 = Interrupt on positive edge

El campo **INTCON2.DISI** señaliza la ejecución de la instrucción **DISI n** que desactiva temporalmente las interrupciones durante n ciclos.

Interrupciones externas activas por flanco

Dentro del DSP existe la posibilidad en algunas patillas de generar interrupciones causadas por un evento externo (**INT0, INT1,** ..). El control del flanco activo para la generación de las interrupciones se hace mediante los campos **INTCON2.INT0EP, ... INT4EP** pudiendo seleccionar el flanco activo. Ver figura más arriba.

Registro SR

Registro de estado de la CPU. Este registro no es específico para el control de las interrupciones pero tiene un especial interés porque contiene el campo **IPL2..IPL0** que indica el nivel de prioridad de la CPU.

SR, STATUS Register

High Byte (SRH):							
R-0	R-0	R/C-0	R/C-0	R-0	R/C-0	R-0	R/W-0
OA	OB	SA	SB	OAB	SAB	DA	DC
bit 15							bit 8

Low Byte (SRL):							
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL<2:0>			RA	N	OV	Z	C
bit 7							bit 0

bit 15 **OA:** Accumulator A Overflow bit
1 = Accumulator A overflowed
0 = Accumulator A has not overflowed

bit 14 **OB:** Accumulator B Overflow bit
1 = Accumulator B overflowed
0 = Accumulator B has not overflowed

bit 13 **SA:** Accumulator A Saturation bit(1, 2)
1 = Accumulator A is saturated or has been saturated at some time
0 = Accumulator A is not saturated

bit 12 **SB:** Accumulator B Saturation bit(1, 2)
1 = Accumulator B is saturated or has been saturated at some time
0 = Accumulator B is not saturated

bit 11 **OAB:** OA || OB Combined Accumulator Overflow bit
1 = Accumulators A or B have overflowed
0 = Neither Accumulators A or B have overflowed

bit 10 **SAB:** SA || SB Combined Accumulator bit(1, 2, 3)
1 = Accumulators A or B are saturated or have been saturated at some time in the past
0 = Neither Accumulators A or B are saturated

bit 9 **DA:** DO Loop Active bit(4)
1 = DO loop in progress
0 = DO loop not in progress

bit 8 **DC:** MCU ALU Half Carry bit
1 = A carry-out from the Most Significant bit of the lower nibble occurred
0 = No carry-out from the Most Significant bit of the lower nibble occurred

bit 7-5 **IPL<2:0>:** Interrupt Priority Level bits(5)
111 = CPU Interrupt Priority Level is 7 (15). User interrupts disabled
110 = CPU Interrupt Priority Level is 6 (14)
101 = CPU Interrupt Priority Level is 5 (13)
100 = CPU Interrupt Priority Level is 4 (12)
011 = CPU Interrupt Priority Level is 3 (11)
010 = CPU Interrupt Priority Level is 2 (10)
001 = CPU Interrupt Priority Level is 1 (9)
000 = CPU Interrupt Priority Level is 0 (8)

bit 4 **RA:** REPEAT Loop Active bit
1 = REPEAT loop in progress
0 = REPEAT loop not in progress

bit 3 **N:** MCU ALU Negative bit
1 = The result of the operation was negative
0 = The result of the operation was not negative

bit 2 **OV:** MCU ALU Overflow bit
1 = Overflow occurred
0 = No overflow occurred

SR, STATUS Register (Continued)

bit 1 **Z:** MCU ALU Zero bit(6)
1 = The result of the operation was zero
0 = The result of the operation was not zero

bit 0 **C:** MCU ALU Carry/$\overline{\text{Borrow}}$ bit
1 = A carry-out from the Most Significant bit occurred
0 = No carry-out from the Most Significant bit occurred

Note 1: This bit may be read or cleared, but not set.
2: Once this bit is set, it must be cleared manually by software.
3: Clearing this bit will clear SA and SB.
4: This bit is read only.
5: The IPL<2:0> bits are concatenated with the IPL<3> bit (CORCON<3>) to form the CPU Interrupt Priority Level. The value in parentheses indicates the IPL, if IPL<3> = 1.
6: Refer to **4.9 "Z Status Bit"** for operation with ADDC, CPB, SUBB and SUBBR instructions.

Registro CORCON

Contiene el bit de estado **IPL3** con el que se identifica el nivel de prioridad de la CPU.

CORCON: Core Control Register

Upper Byte:

U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-0	R-0
—	—	—	US	EDT	DL<1:0>		
bit 15							bit 8

Lower Byte:

R/W-0	R/W-0	R/W-1	R/W-0	R/C-0	R/W-0	R/W-0	R/W-0
SATA	SATB	SATDW	ACCSAT	**IPL3**	PSV	RND	IF
bit 7							bit 0

bit 3 **IPL3:** CPU Interrupt Priority Level Status bit 3
1 = CPU interrupt priority level is greater than 7
0 = CPU interrupt priority level is 7 or less

Note 1: The IPL3 bit is concatenated with the IPL<2:0> bits (SR<7:5>) to form the CPU interrupt priority level.

Registros IFSx

Registros de notificaciones de interrupción. Cada fuente de interrupción tendrá un bit de estado, que se activará desde la fuente de interrupción y se desactivará por medio de software.

IFS0: Interrupt Flag Status Register 0

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0IF
bit 7							bit 0

bit 15 **CNIF:** Input Change Notification Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 14 **MI2CIF:** I^2C Bus Collision Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 13 **SI2CIF:** I^2C Transfer Complete Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 12 **NVMIF:** Non-Volatile Memory Write Complete Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 11 **ADIF:** A/D Conversion Complete Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10 **U1TXIF:** UART1 Transmitter Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **U1RXIF:** UART1 Receiver Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **SPI1IF:** SPI1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **T3IF:** Timer3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **T2IF:** Timer2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **OC2IF:** Output Compare Channel 2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4 **IC2IF:** Input Capture Channel 2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **T1IF:** Timer1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **OC1IF:** Output Compare Channel 1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

IFS0: Interrupt Flag Status Register 0 (Continued)

bit 1 **IC1IF:** Input Capture Channel 1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **INT0IF:** External Interrupt 0 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

IFS1: Interrupt Flag Status Register 1

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF
bit 7							bit 0

bit 15 **IC6IF:** Input Capture Channel 6 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 14 **IC5IF:** Input Capture Channel 5 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 13 **IC4IF:** Input Capture Channel 4 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 12 **IC3IF:** Input Capture Channel 3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 11 **C1IF:** CAN1 (Combined) Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10 **SPI2IF:** SPI2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **U2TXIF:** UART2 Transmitter Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **U2RXIF:** UART2 Receiver Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **INT2IF:** External Interrupt 2 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **T5IF:** Timer5 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **T4IF:** Timer4 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4 **OC4IF:** Output Compare Channel 4 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **OC3IF:** Output Compare Channel 3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **IC8IF:** Input Capture Channel 8 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

IFS1: Interrupt Flag Status Register 1 (Continued)

bit 1 **IC7IF:** Input Capture Channel 7 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **INT1IF:** External Interrupt 1 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

IFS2: Interrupt Flag Status Register 2

Upper Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIF
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF
bit 7							bit 0

bit 15-13 **Unimplemented:** Read as '0'

bit 12 **FLTBIF:** Fault B Input Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 11 **FLTAIF:** Fault A Input Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10 **LVDIF:** Programmable Low Voltage Detect Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **DCIIF:** Data Converter Interface Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **QEIIF:** Quadrature Encoder Interface Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **PWMIF:** Motor Control Pulse Width Modulation Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **C2IF:** CAN2 (Combined) Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **INT4IF:** External Interrupt 4 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4 **INT3IF:** External Interrupt 3 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **OC8IF:** Output Compare Channel 8 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **OC7IF:** Output Compare Channel 7 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

IFS2: Interrupt Flag Status Register 2 (Continued)

bit 1 **OC6IF:** Output Compare Channel 6 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **OC5IF:** Output Compare Channel 5 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

Registros IECx

Registros de control de permiso de interrupciones. Contienen los bits de habilitación de las interrupciones.

IEC0: Interrupt Enable Control Register 0

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE
bit 7							bit 0

bit 15 **CNIE:** Input Change Notification Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 14 **MI2CIE:** I^2C Bus Collision Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 13 **SI2CIE:** I^2C Transfer Complete Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 12 **NVMIE:** Non-Volatile Memory Write Complete Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 11 **ADIE:** A/D Conversion Complete Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 10 **U1TXIE:** UART1 Transmitter Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 9 **U1RXIE:** UART1 Receiver Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 8 **SPI1IE:** SPI1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 7 **T3IE:** Timer3 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 6 **T2IE:** Timer2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 5 **OC2IE:** Output Compare Channel 2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 4 **IC2IE:** Input Capture Channel 2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 3 **T1IE:** Timer1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 2 **OC1IE:** Output Compare Channel 1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

IEC0: Interrupt Enable Control Register 0 (Continued)

bit 1 **IC1IE:** Input Capture Channel 1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 0 **INT0IE:** External Interrupt 0 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

IEC1: Interrupt Enable Control Register 1

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE
bit 7							bit 0

bit 15 **IC6IE:** Input Capture Channel 6 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 14 **IC5IE:** Input Capture Channel 5 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 13 **IC4IE:** Input Capture Channel 4 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 12 **IC3IE:** Input Capture Channel 3 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 11 **C1IE:** CAN1 (Combined) Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 10 **SPI2IE:** SPI2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 9 **U2TXIE:** UART2 Transmitter Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 8 **U2RXIE:** UART2 Receiver Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 7 **INT2IE:** External Interrupt 2 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 6 **T5IE:** Timer5 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 5 **T4IE:** Timer4 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 4 **OC4IE:** Output Compare Channel 4 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 3 **OC3IE:** Output Compare Channel 3 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 2 **IC8IE:** Input Capture Channel 8 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

IEC1: Interrupt Enable Control Register 1 (Continued)

bit 1 **IC7IE:** Input Capture Channel 7 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 0 **INT1IE:** External Interrupt 1 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

IEC2: Interrupt Enable Control Register 2

Upper Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIE
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE
bit 7							bit 0

bit 15-13 **Unimplemented:** Read as '0'

bit 12 **FLTBIE:** Fault B Input Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 11 **FLTAIE:** Fault A Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 10 **LVDIE:** Programmable Low Voltage Detect Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 9 **DCIIE:** Data Converter Interface Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 8 **QEIIE:** Quadrature Encoder Interface Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 7 **PWMIE:** Motor Control Pulse Width Modulation Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 6 **C2IE:** CAN2 (Combined) Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 5 **INT4IE:** External Interrupt 4 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 4 **INT3IE:** External Interrupt 3 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 3 **OC8IE:** Output Compare Channel 8 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 2 **OC7IE:** Output Compare Channel 7 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

IEC2: Interrupt Enable Control Register 2 (Continued)

bit 1 **OC6IE:** Output Compare Channel 6 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 0 **OC5IE:** Output Compare Channel 5 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

Registros IPCxx

Registros de control de prioridades de interrupciones. Se usan para almacenar el nivel de prioridad de cada fuente de interrupción.

SFR Name	ADR	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
INTCON1	0080	NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000
INTCON2	0082	ALTIVT	DISI	—	—	—	—	—	—	—	—	—	—	—	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0IF	0000 0000 0000 0000
IFS1	0086	—	—	—	—	C1IF	—	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IFS2	0088	—	—	—	—	FLTAIF	—	—	QEIIF	PWMIF	—	—	—	—	—	—	—	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	—	—	—	—	C1IE	—	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IEC2	0090	—	—	—	—	FLTAIE	—	—	QEIIE	PWMIE	—	—	—	—	—	—	—	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	—	T31P<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			0100 0100 0100 0100
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100
IPC3	009A	—	CNIP<2:0>			—	MI2CIP<2:0>			—	SI2CIP<2:0>			—	NVMIP<2:0>			0100 0100 0100 0100
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>			0100 0100 0100 0100
IPC5	009E	—	INT2IP<2:0>			—	T5IP<2:0>			—	T4IP<2:0>			—	OC4IP<2:0>			0100 0100 0100 0100
IPC6	00A0	—	C1IP<2:0>			—	—	—	—	—	U2TXIP<2:0>			—	U2RXIP<2:0>			0100 0000 0100 0100
IPC7	00A2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
IPC8	00A4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
IPC9	00A6	—	PWMIP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	0100 0000 0100 0100
IPC10	00A8	—	FLTAIP<2:0>			—	—	—	—	—	—	—	—	—	QEIIP<2:0>			0100 0000 0000 0100
IPC11	00AA	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000

Legend: u = uninitialized bit
Note: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

7.5.7. Ejemplo

La manera de implementar las rutinas de atención a las interrupciones cuando se trabaja desde lenguaje C es la siguiente:

Listado 7.1:

```
#include <dspic.h>

// Ejemplo para la interrupcion asociada al temporizador 1
void interrupt ISR_T1(void) @ T1_VCTR
{
 T1IF = 0; // Bajo el bit de notificacion (pegajoso)
}
```

Desde el lenguaje Hitech PICC la asociación de las interrupciones con los vectores de interrupción sigue esta tabla:

Listado 7.2:

```
   Macro name  dsPIC30F          Other Devices     Description
   ------------------------------------------------------------------------
3  INT0_VCTR   0x14             0x14         External Interrupt 0
   IC1_VCTR    0x16             0x16         Input Capture 1
   OC1_VCTR    0x18             0x18         Output Compare 1
   T1_VCTR     0x1A             0x1A         Timer 1
   DMA0_VCTR                    0x1C         DMA Channel 0
8  IC2_VCTR    0x1C             0x1E         Input Capture 2
   OC2_VCTR    0x1E             0x20         Output Compare 2
   T2_VCTR     0x20             0x22         Timer 2
   T3_VCTR     0x22             0x24         Timer 3
   SPI1_VCTR   0x24                          Serial Comms 1
13 SPI1E_VCTR                   0x26         Serial Comms 1 Error
   SPI1D_VCTR                   0x28         Serial Comms 1 Transfer Done
   U1RX_VCTR   0x26             0x2A         UART1 Receiver
   U1TX_VCTR   0x28             0x2C         UART1 Transmitter
   ADC1_VCTR                    0x2E         A/D Converter 1
18 DMA1_VCTR                    0x30         DMA Channel 1
   ADC_VCTR    0x2A                         ADC Convert Done
   NVM_VCTR    0x2C                         NVM Write Complete
   SI2C_VCTR   0x2E                         I2C Slave Interrupt
   MI2C_VCTR   0x30                         I2C Master Interrupt
23 SI2C1_VCTR                   0x34         I2C1 Slave Interrupt
   MI2C1_VCTR                   0x36         I2C1 Master Interrupt
   CM_VCTR                      0x38         Comparator Event
   INCH_VCTR   0x32             0x3A         Input Change Interrupt
   INT1_VCTR   0x34             0x3C         External Interrupt 1
28 ADC2_VCTR                    0x3E         A/D Converter 2
   IC7_VCTR    0x36             0x40         Input Capture 7
   IC8_VCTR    0x38             0x42         Input Capture 8
   DMA2_VCTR                    0x44         DMA Channel 2
   OC3_VCTR    0x3A             0x46         Output Compare 3
33 OC4_VCTR    0x3C             0x48         Output Compate 4
   T4_VCTR     0x3E             0x4A         Timer 4
   T5_VCTR     0x40             0x4C         Timer 5
   INT2_VCTR   0x42             0x4E         External Interrupt 2
   U2RX_VCTR   0x44             0x50         UART2 Receiver
38 U2TX_VCTR   0x46             0x52         UART2 Transmitter
   SPI2E_VCTR                   0x54         Serial Comms 2 Error
   SPI2D_VCTR                   0x56         Serial Comms 2 Transfer Done
   C1RX_VCTR                    0x58         ECAN1 Receive Data Ready
   C1E_VCTR                     0x58         CAN1 Error on PS Devices
43 SPI2_VCTR   0x48                          Serial Comms 2
   C1_VCTR     0x4A             0x5A         Combined IRQ for CAN1 or ECAN1
   DMA3_VCTR                    0x5C         DMA Channel 3
   IC3_VCTR    0x4C             0x5E         Input Capture 3
   IC4_VCTR    0x4E             0x60         Input Capture 4
48 IC5_VCTR    0x50             0x62         Input Capture 5
   IC6_VCTR    0x52             0x64         Input Capture 6
   OC5_VCTR    0x54             0x66         Output Compare 5
   OC6_VCTR    0x56             0x68         Output Compare 6
   OC7_VCTR    0x58             0x6A         Output Compare 7
53 OC8_VCTR    0x5A             0x6C         Output Compare 8
   PMP_VCTR                     0x6E         Parallel Port Master
   DMA4_VCTR                    0x70         DMA Channel 4
   T6_VCTR                      0x72         Timer 6
   T7_VCTR                      0x74         Timer 7
58 SI2C2_VCTR                   0x76         I2C2 Slave Interrupt
   MI2C2_VCTR                   0x78         I2C2 Master Interrupt
   INT3_VCTR   0x5C             0x7E         External Interrupt 3
   INT4_VCTR   0x5E             0x80         External Interrupt 4
   C1RX_VCTR                    0x82         ECAN2 Receive Data Ready
63 C1E_VCTR                     0x82         CAN2 Error on PS Devices
   C2_VCTR     0x60             0x84         Combined IRQ for CAN2 or ECAN2
   PWM_VCTR    0x62             0x86         PWM Period Match
   QEI_VCTR    0x64             0x88         QEI Interrupt
   DCIE_VCTR                    0x8A         DCI Error
68 DCID_VCTR                    0x8C         DCI Transfer Done
   DMA5_VCTR                    0x8E         DMA Channel 5
   RTCC_VCTR                    0x90         Real-time Clock/Calendar
   DCI_VCTR    0x66                         Codec Transfer Done
   LVD_VCTR    0x68                         Low Voltage Detect
73 FLTA_VCTR   0x6A             0x92         PWM Fault A
```

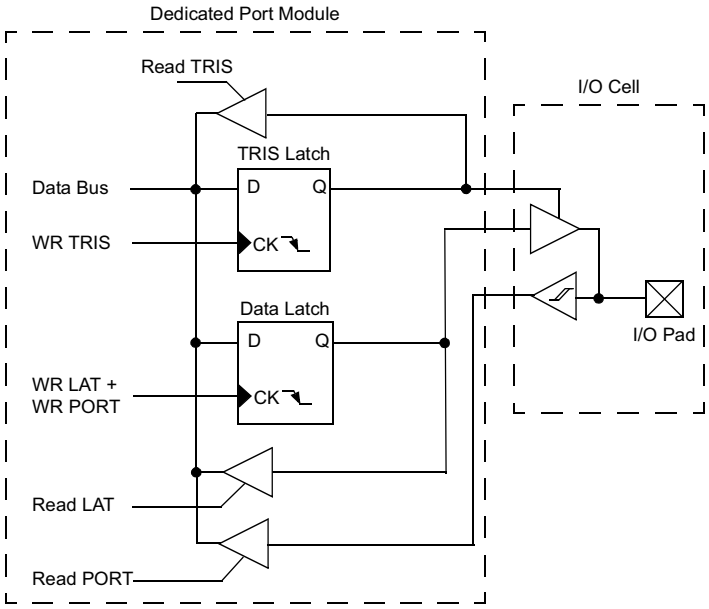
```
    FLTB_VCTR   0x6C            0x94        PWM Fault B
    U1E_VCTR                    0x96        UART1 Error
    U2E_VCTR                    0x98        UART2 Error
    DMA6_VCTR                   0x9C        DMA Channel 6
78  DMA7_VCTR                   0x9E        DMA Channel 7
    C1TX_VCTR                   0xA0        ECAN1 Transmit Data Request
    C2TX_VCTR                   0xA2        ECAN2 Transmit Data Request
```

7.6. Puertos E/S

7.6.1. Introducción

Las patillas de E/S de los dsPIC normalmente están multiplexadas con otras funciones relacionadas con los módulos hardware integrados. Cuando funcionan como patillas asociadas a un módulo, podrían no ser empleadas como de propósito general.

7.6.2. Diagrama de bloques: puerto dedicado



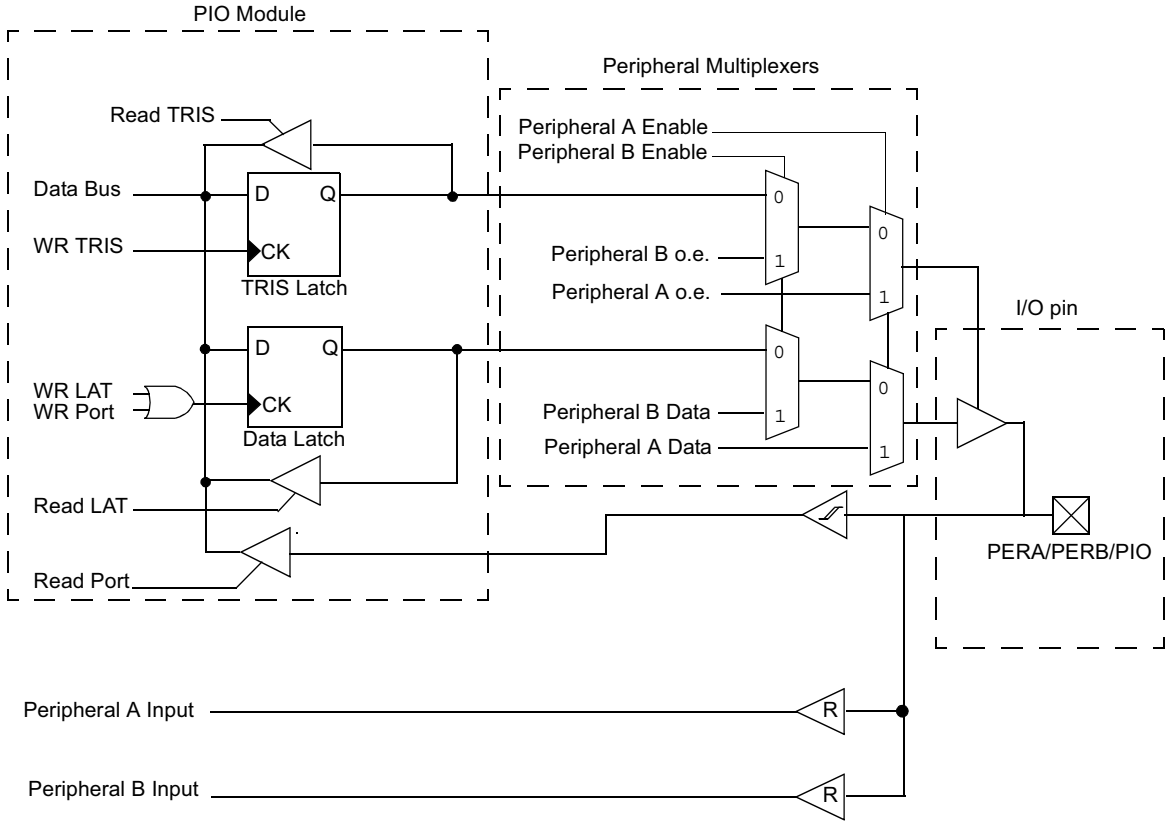
Para el control de las patilla de entrada/salida de propósito general tenemos tres registros por puerto:

- `TRISx`. Registro de dirección de cada patilla del puerto. Un valor de bit a `1` indica que funcionará como una entrada. Un valor `0` indicará salida.

- `PORTx`. Registro que cuando se escribe modifica el valor de salida y que cuando se lee toma el valor de la patilla. Las operaciones son del tipo lee-modifica-escribe. Esto puede dar problemas dado que el valor que se lee coincide con el valor del pin y no el valor del latch que ataca el pin.

- `LATx`. Registro latch. Cuando se escribe modifica la salida y cuando se lee toma el valor del registro que ataca la salida. Elimina los problemas del acceso lee-modifica-escribe dado que lee directamente el valor que hemos puesto anteriormente.

7.6.3. Diagrama de bloques: puerto multiplexado



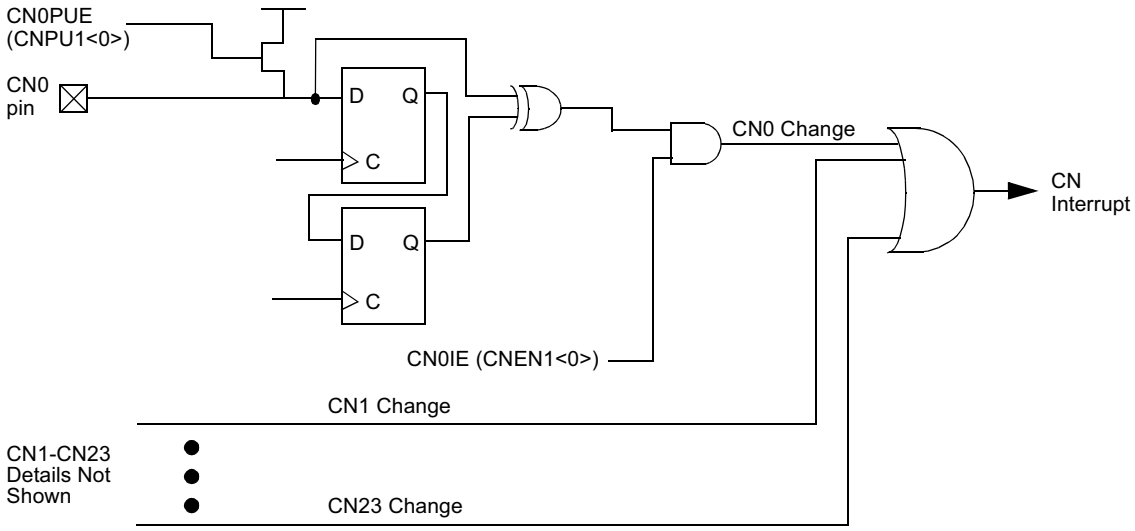
Las patillas de entrada/salida se pueden multiplexar con uno o varios módulos periféricos integrados.

En el caso de que se active el módulo asociado podría no tenerse el control individual de entrada o salida de la patilla ya que pasaría a ser controlada por el módulo.

Si varios módulos asignados a una patilla se activan a la vez existe una prioridad de acceso de los mismos y que tiene que ver con el nombre de la patilla: la primera función nombrada tiene prioridad sobre la segunda, etc.

7.6.4. Patillas de notificación de cambio

Algunas de las patillas de propósito general (hasta 24) pueden tener asociada la función de notificación de cambio cuando están configuradas como entradas.



Los registros que lo controlan son los siguientes:

- **CNEN1** y **CNEN2**. Contienen los bits de control `CNxIE` que activan la notificación de cambio cuando cambia la entrada.
- **CNPU1** y **CNPU2**. Contienen los bits de control `CNxPUE` que habilitan una resistencia de pull-up débil interna cuando las patillas están configuradas como entradas.

Configuración y operación

Para utilizarlo conviene seguir los siguientes pasos:

- Se configuran las patillas deseadas como entradas mediante los correspondientes bits en los registros `TRISx` asociados.
- Se habilita la notificación de cambio en las patillas que se quieran activándolas en los registros **CNEN1** y **CNEN2**.
- Se habilitan los pull-up internos si se necesitan mediante los registros **CNPU1** y **CNPU2**.
- Se borra el flag de notificación **IFS0**.**CNIF**.
- Se selecciona la prioridad de esta fuente de interrupción mediante **IPC3**.**CNIP2**:0.
- Se habilita la interrupción de notificación de cambio (general) con **IEC0**.**CNIE** = 1.

Una vez ocurrida la interrupción se debe leer el puerto asociado para eliminar la condición y que se pueda detectar el siguiente cambio.

Registros asociados

CNEN1: Input Change Notification Interrupt Enable Register1

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	CN10IE	CN9IE	CN8IE
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN7IE	CN6IE	CN5IE	CN4IE	CN3IE	CN2IE	CN1IE	CN0IE
bit 7							bit 0

bit 15-0 **CNxIE:** Input Change Notification Interrupt Enable bits
1 = Enable interrupt on input change
0 = Disable interrupt on input change

CNEN2: Input Change Notification Interrupt Enable Register2

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN23IE	CN22IE	CN21IE	CN20IE	CN19IE	CN18IE	CN17IE	CN16IE
bit 7							bit 0

bit 15-8 **Unimplemented:** Read as '0'

bit 7-0 **CNxIE:** Input Change Notification Interrupt Enable bits
1 = Enable interrupt on input change
0 = Disable interrupt on input change

CNPU1: Input Change Notification Pull-up Enable Register1

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	CN10PUE	CN9PUE	CN8PUE
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE
bit 7							bit 0

bit 15-0 **CNxPUE:** Input Change Notification Pull-up Enable bits
1 = Enable pull-up on input change
0 = Disable pull-up on input change

CNPU2: Input Change Notification Pull-up Enable Register2

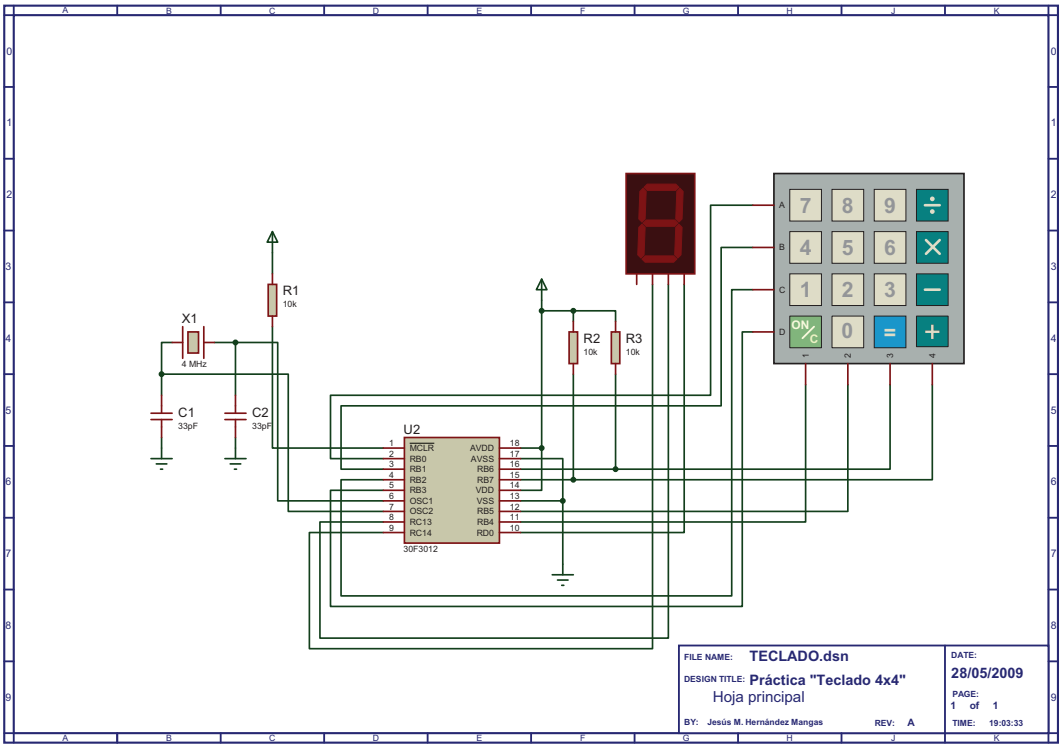
Upper Byte:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN23PUE	CN22PUE	CN21PUE	CN20PUE	CN19PUE	CN18PUE	CN17PUE	CN16PUE
bit 7							bit 0

bit 15-8 **Unimplemented:** Read as '0'

bit 7-0 **CNxPUE:** Input Change Notification Pull-up Enable bits
1 = Enable pull-up on input change
0 = Disable pull-up on input change

Ejemplo : Acceso a un teclado hexadecimal.



Listado 7.3: Codigo/dsPIC_Ejemplos/TECLADO.c

```
#include <dspic.h>
#include "binario.h"

// Para el dsPIC30F3012
__CONFIG( FOSC, XTPLL16 );

const unsigned char Teclas[]={0,
                7,  8,  9,0xF,
```

```
                4,  5,  6,0xE,
                1,  2,   3, 0xD,
          0xC,  0, 0xB, 0xA};

unsigned char TECLADO_Scan()
{
 unsigned char Tecla=1;
//       ; Muestrea el teclado
//       ; 1. Pone la variable Tecla a 1 para el primer código
//       ; 2. Pone todas las filas de salida a 1, excepto la fila
//       ;       a muestrear que tiene un 0.
//       ; 3. Se chequea cada columna de entrada buscando el 0
//       ;       Si no hay ninguna tecla pulsada la columna se leerá
//       ;       como un 1 gracias a las resistencias de pull-up.
//       ;       Si está pulsada se leerá un 0.
//       ; 4. Se continua el proceso con todas las filas.
//       ; Si se ha pulsado una tecla, se sale de la rutina con
//       ;       el valor de la tecla pulsada en la variable Tecla
//       ;   Col.  1    2    3    4
//       ; Fila  +----+----+----+----+
//       ;   1   | 1  | 2  | 3  | 4  |
//       ;       +----+----+----+----+
//       ;   2   | 5  | 6  | 7  | 8  |
//       ;       +----+----+----+----+
//       ;   3   | 9  | 10 | 11 | 12 |
//       ;       +----+----+----+----+
//       ;   4   | 13 | 14 | 15 | 16 |
//       ;       +----+----+----+----+
//       ; Si no se ha pulsado ninguna tecla
//       ;       se sale con el valor 0 en Tecla.
//       ; Si se pulsan dos o más teclas a la vez sólo la
//       ;       primera (siguiendo el orden) se detecta.

 PORTB = B00001110;          // Saco un cero en la fila cero
 CNIF  = 0;                  // Borro la notificación
 while(1)
 {
  if( (PORTB & (1<<4))==0 ) return Tecla; // Comprueba la primera columna
  Tecla++;
  if( (PORTB & (1<<5))==0 ) return Tecla; // Comprueba la segunda columna
  Tecla++;
  if( (PORTB & (1<<6))==0 ) return Tecla; // Comprueba la tercera columna
  Tecla++;
  if( (PORTB & (1<<7))==0 ) return Tecla; // Comprueba la cuarta columna
  Tecla++;

  if(Tecla==17) return 0;    // Si no se pulsó ninguna retorna cero
  PORTB = (PORTB << 1)|1;    // Selecciona otra fila
 }
 return 0;
}

void main()
{
 unsigned char T;

 ADPCFG  = 0xFFFF;           // Configuro todas las lineas como digitales
 TRISB   = 0xF0;             // RB0..RB3 salidas, RB4..RB7 entradas
 TRISC   = 0;
 TRISD   = 0;
 CNEN1   = 0x00F0;           // Activo la notificación de cambio
 CNPU1   = 0x00F0;           // Activo las resistencias de pull-up
```

```
    while(1)
     {
      T      = TECLADO_Scan();   // Comprueba el teclado
75    T      = Teclas[T];        // Traduce
      PORTC  = (T&B00001110)<<12;// RC15,14,13
      PORTD  = (T&B00000001);    // RD0
     }
  }
```

7.7. Timers

7.7.1. Introducción

La familia dsPIC30F dispone de un número variable de temporizadores de 16 bits. El dsPIC30F4011 dispone de 5 temporizadores.

Cada módulo temporizador dispone de los siguientes registros:

- `TMRx`. Registro contador.
- `PRx`. Registro periodo.
- `TxCON`. Registro de control del módulo temporizador.

También dispone de bits en otros registros asociados al control de interrupciones:

- Bits de control para habilitación de interrupciones: `TxIE`.
- Bits de notificación: `TxIF`.
- Bits de control de la prioridad de la interrupción: `TxIP2..0`.

Los módulos temporizadores se pueden clasificar en tres tipos diferentes: tipo A, tipo B y tipo C.

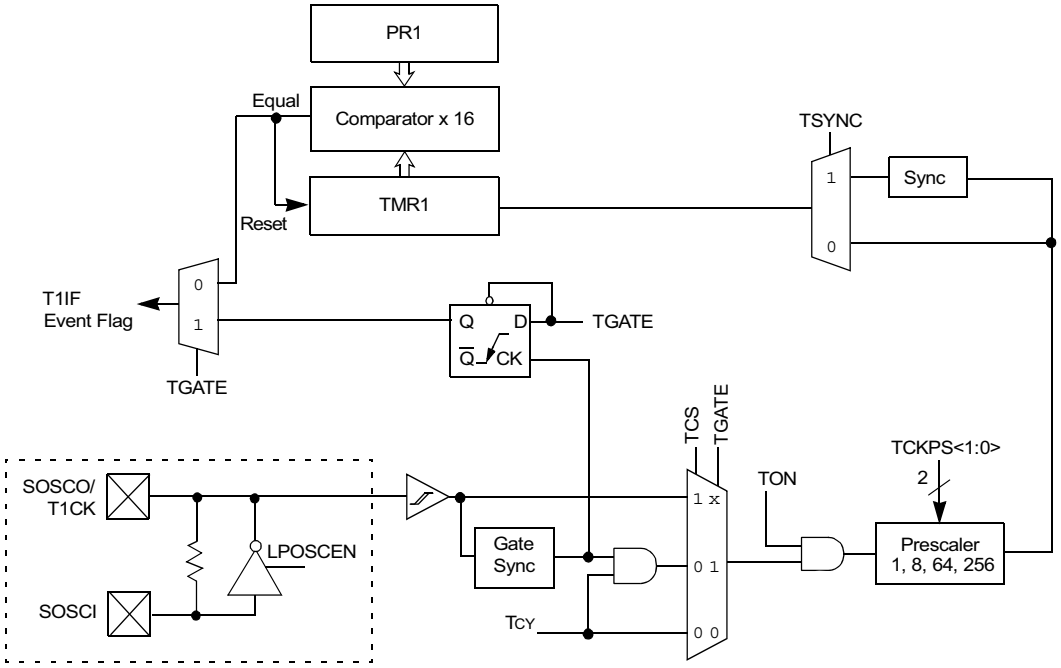
Además los tipos B y C se pueden agrupar para formar un temporizador de 32 bits.

7.7.2. Temporizador tipo A

Al menos uno de los temporizadores será de tipo A (*Timer* 1, típicamente). Características:

- Puede operar con un cristal de bajo consumo de 32 kHz. Esto le permite ser utilizado como un reloj en tiempo real (RTC: *Real Time Clock*).
- Puede funcionar en modo asíncrono desde una fuente de reloj externo.

Podemos ver el diagrama de bloques:


PR1
Equal
Comparator x 16
TMR1
Reset
TSYNC
1
0
Sync
T1IF
Event Flag
0
1
TGATE
Q
D
Q
CK
TGATE
TCS
TGATE
TCKPS<1:0>
2
TON
SOSCO/
T1CK
SOSCI
LPOSCEN
Gate
Sync
TCY
1 x
0 1
0 0
Prescaler
1, 8, 64, 256

TxCON: Type A Time Base Register

Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		—	TSYNC	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer On Control bit
1 = Starts the timer
0 = Stops the timer

bit 14 **Unimplemented:** Read as '0'

bit 13 **TSIDL:** Stop in Idle Mode bit
1 = Discontinue timer operation when device enters Idle mode
0 = Continue timer operation in Idle mode

bit 12-7 **Unimplemented:** Read as '0'

bit 6 **TGATE:** Timer Gated Time Accumulation Enable bit
1 = Gated time accumulation enabled
0 = Gated time accumulation disabled
(TCS must be set to '0' when TGATE = 1. Reads as '0' if TCS = 1)

bit 5-4 **TCKPS<1:0>:** Timer Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3 **Unimplemented:** Read as '0'

bit 2 **TSYNC:** Timer External Clock Input Synchronization Select bit
When TCS = 1:
1 = Synchronize external clock input
0 = Do not synchronize external clock input
When TCS = 0:
This bit is ignored. Read as '0'. Timer1 uses the internal clock when TCS = 0.

bit 1 **TCS:** Timer Clock Source Select bit
1 = External clock from pin TxCK
0 = Internal clock (FOSC/4)

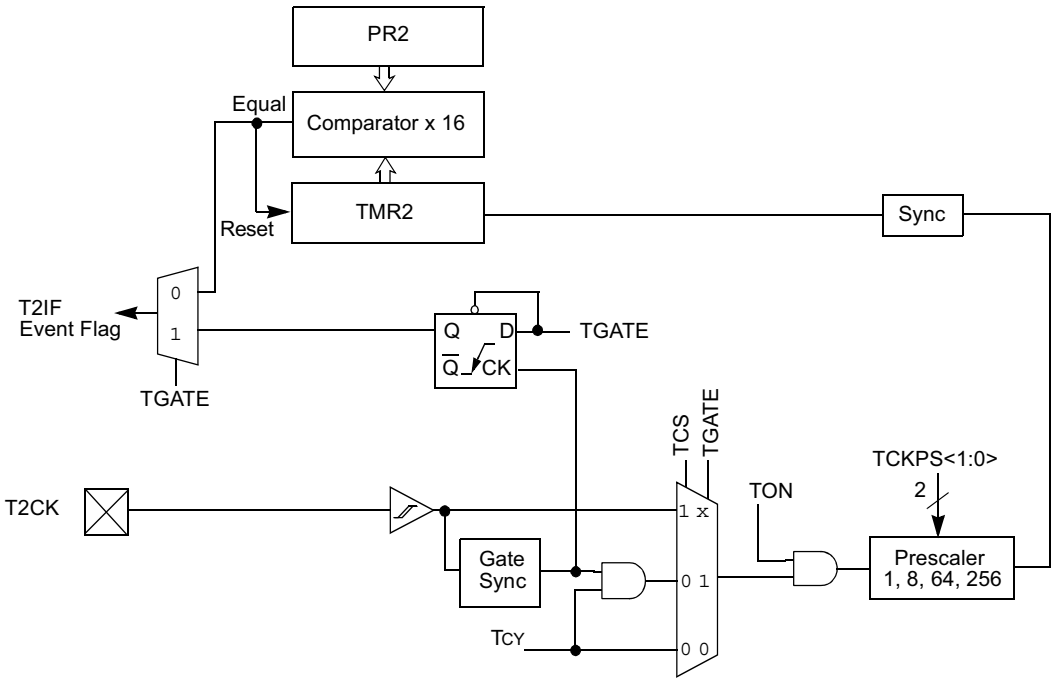
bit 0 **Unimplemented:** Read as '0'

7.7.3. Temporizador tipo B

Los temporizadores 2 y 4 si están presentes son de este tipo. Características únicas:

- Puede concatenarse con un temporizador tipo C para formar un temporizador de 32 bits.
- La sincronización del reloj se realiza después de la lógica de escalado.

Podemos ver el diagrama de bloques:


PR2
Equal
Comparator x 16
TMR2
Reset
Sync
T2IF
Event Flag
0
1
TGATE
Q
D
Q
CK
TGATE
TCS
TGATE
TCKPS<1:0>
2
T2CK
TON
1 x
Gate
Sync
0 1
Prescaler
1, 8, 64, 256
TCY
0 0

TxCON: Type B Time Base Register

Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		T32	—	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer On bit
When T32 = 1 (in 32-bit Timer mode):
1 = Starts 32-bit TMRx:TMRy timer pair
0 = Stops 32-bit TMRx:TMRy timer pair
When T32 = 0 (in 16-bit Timer mode):
1 = Starts 16-bit timer
0 = Stops 16-bit timer

bit 14 **Unimplemented:** Read as '0'

bit 13 **TSIDL:** Stop in Idle Mode bit
1 = Discontinue timer operation when device enters Idle mode
0 = Continue timer operation in Idle mode

bit 12-7 **Unimplemented:** Read as '0'

bit 6 **TGATE:** Timer Gated Time Accumulation Enable bit
1 = Timer gated time accumulation enabled
0 = Timer gated time accumulation disabled
(TCS must be set to logic '0' when TGATE = 1)

bit 5-4 **TCKPS<1:0>:** Timer Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3 **T32:** 32-bit Timer Mode Select bits
1 = TMRx and TMRy form a 32-bit timer
0 = TMRx and TMRy form separate 16-bit timer

bit 2 **Unimplemented:** Read as '0'

bit 1 **TCS:** Timer Clock Source Select bit
1 = External clock from pin TxCK
0 = Internal clock (FOSC/4)

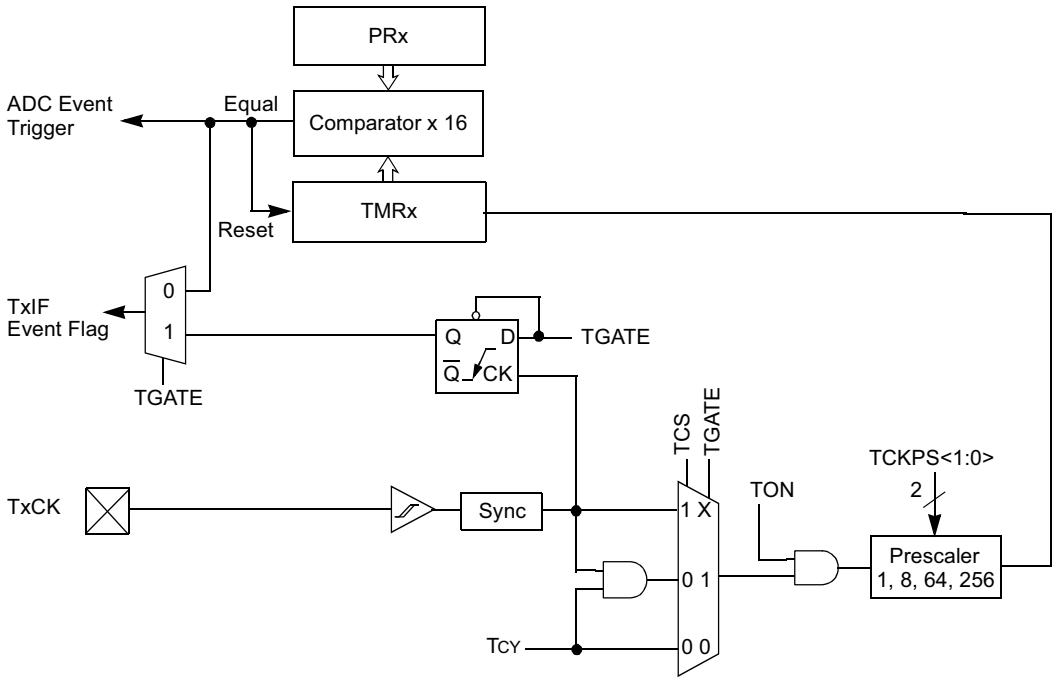
bit 0 **Unimplemented:** Read as '0'

7.7.4. Temporizador tipo C

Los temporizadores 3 y 5 si están presentes son de este tipo. Características únicas:

- Se puede concatenar con un temporizador tipo B para formar un temporizador de 32 bits.
- Al menos un temporizador de este tipo permite disparar el funcionamiento del convertidor A/D.

El diagrama de bloques:



Note: In certain variants of the dsPIC30F family, the TxCK pin may not be available. Refer to the device data sheet for the I/O pin details. In such cases, the timer must use the system clock (FOSC/4) as its input clock, unless it is configured for 32-bit operation.

TxCON: Type C Time Base Register

Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		—	—	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer On bit
1 = Starts 16-bit TMRx
0 = Stops 16-bit TMRx

bit 14 **Unimplemented:** Read as '0'

bit 13 **TSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12-7 **Unimplemented:** Read as '0'

bit 6 **TGATE:** Timer Gated Time Accumulation Enable bit
1 = Timer gated time accumulation enabled
0 = Timer gated time accumulation disabled (Read as '0' if TCS = 1)
(TCS must be set to logic '0' when TGATE = 1)

bit 5-4 **TCKPS<1:0>:** Timer Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3-2 **Unimplemented:** Read as '0'

bit 1 **TCS:** Timer Clock Source Select bit
1 = External clock from pin TxCK
0 = Internal clock (FOSC/4)

bit 0 **Unimplemented:** Read as '0'

7.7.5. Modos de funcionamiento de 16 bits

Cada módulo puede funcionar en alguno de los siguientes modos:

- Temporizador síncrono.
- Contador síncrono.
- Temporizador con puerta.
- Contador asíncrono (solo tipo A).

El modo se selecciona con los bits: `TxCON.TCS`: bit de control de la fuente de reloj, `T1CON.TSYNC`: bit de control de sincronización (solo tipo A), `TxCON.TGATE`: bit de control de puerta para el temporizador.

Cada módulo se puede habilitar con `TxCON.TON`.

Modo temporizador síncrono

Listado 7.4: Codigo/dsPIC_Ejemplos/MTS.c

```
#include <dspic.h>

//__CONFIG(FOSC,XTPLL16);
__CONFIG(FOSC,XT);  // Cristal externo a 7.3728 MHz

void main()
{
 ADPCFG= 0xFFFF;     // Lineas digitales
 TRISB = 0xFFFE;     // RB0 salida

 T1CON = 0;          // Para el temporizador, pre-escala=1:1
 TMR1  = 0;          // Cuenta inicial
 PR1   = 28800;      // Periodo
 T1CONbits.TCKPS=2; // 1:64

 // 64*28800*1/(7372800/4) = 1 segundo

 T1ON = 1;          // Arranca el temporizador

 while(1)
 {
   while(!T1IF);     // Espera a que llegue
   T1IF  = 0;
   LATB0 = ! LATB0; // Conmuta RB0
 }
}
```

Modo contador síncrono

Listado 7.5: Codigo/dsPIC_Ejemplos/MCS.c

```
#include <dspic.h>

//__CONFIG(FOSC,XTPLL16);
__CONFIG(FOSC,XT);  // Cristal externo a 7.3728 MHz

void main()
{
 ADPCFG= 0xFFFF;     // Lineas digitales
 TRISB = 0xFFFE;     // RB0 salida

 T1CON = 0;          // Para el temporizador, pre-escala=1:1
 TMR1  = 0;          // Cuenta inicial
 PR1   = 2;          // Periodo
 T1CONbits.TCKPS=0; // 1:1
 T1CONbits.TCS=1;    // Reloj externo

                     // Cada dos cambios de T1CK/RC14

 T1ON = 1;          // Arranca el temporizador

 while(1)
 {
   while(!T1IF);     // Espera a que llegue
   T1IF  = 0;
   LATB0 = ! LATB0; // Conmuta RB0
```

```
  }
}
```

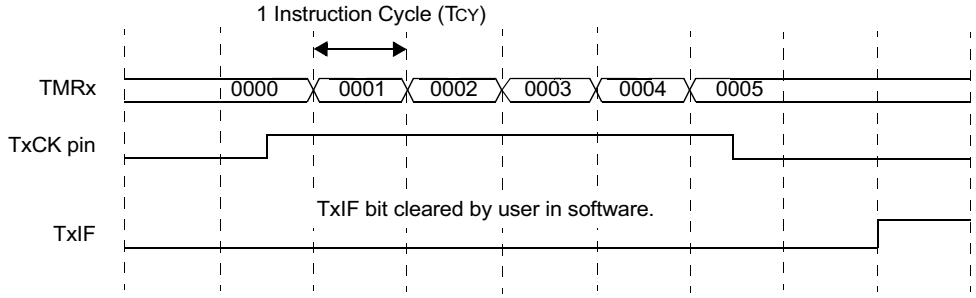
Modo contador asíncrono

El módulo temporizador tipo A permite este modo de funcionamiento asíncrono. Las ventajas de la operación en modo asíncrono son:

- Puede funcionar mientras el dispositivo está dormido y puede generar una interrupción al llegar al final de la cuenta que despierte al procesador.
- La base de tiempos puede funcionar con un cristal externo de 32 kHz en el oscilador de bajo consumo y ser empleado como reloj en tiempo real (RTC). Para esto es necesario activar el oscilador: `OSCCON.LPOSCEN = 1` y un cristal de 32 kHz debe estar conectado entre los pines `SOSCO y SOSCI`.

Modo de acumulación de tiempo con puerta

Este modo de funcionamiento permite incrementar el contador basándose en la cantidad de tiempo que la señal externa de entrada al módulo permanece en alto. Se incrementa el contador tomando como base el ciclo de instrucción del procesador. Contará hasta alcanzar el valor máximo del periodo o hasta que la señal de entrada baje. Cualquiera de los dos casos disparará la interrupción asociada si esta está activada (hay una latencia de unos dos ciclos de instrucción). El flanco de bajada no reinicia el contador.

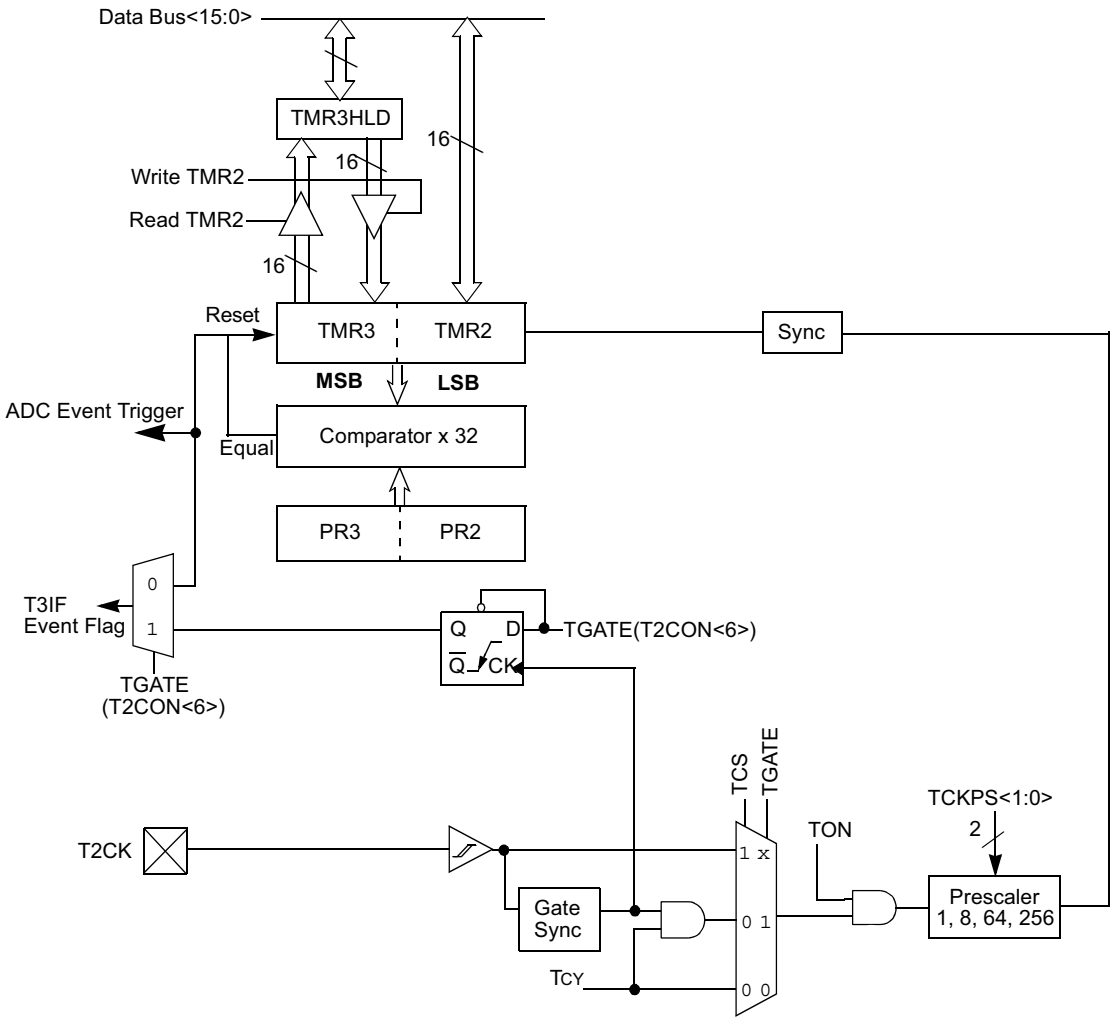


7.7.6. Modo de funcionamiento de 32 bits

Este modo está disponible con la combinación de módulos tipo B y tipo C. El módulo tipo C será la parte más significativa del contador y el de tipo B la parte menos significativa.

El control de la operación de los módulos combinados se hará mediante el registro `TxCON` del módulo tipo B. El otro registro `TxCON` del módulo tipo C no tendrá efecto.

Para el control de las interrupciones generadas por este módulo combinado de 32 bits se usarán los bits de control del módulo tipo C. Los bits de control del módulo B no tendrán efecto.



Note: Timer configuration bit, T32 (T2CON<3>), must be set to '1' for a 32-bit timer/counter operation. All control bits are respective to the T2CON register.

Modo temporizador: 32 bits

Listado 7.6: Codigo/dsPIC_Ejemplos/MT32S.c

```
#include <dspic.h>

__CONFIG(FOSC,XTPLL16); // Frec = 7.3728 * 16 = 117.9648 MHz
//__CONFIG(FOSC,XT);  // Cristal externo a 7.3728 MHz

void main()
{
 TRISB = 0xFFFE;     // RB0 salida

 T2CON = 0;          // Para el temporizador, pre-escala=1:1
 T3CON = 0;          // Para los temporizadores

 TMR2  = 0;          // Cuenta inicial
 TMR3  = 0;

 PR2   = 460800 & 0xFFFF; // Periodo parte baja
 PR3   = 460800 >> 16;    // Periodo parte alta
```

```
    T2CONbits.TCKPS=2; // 1:64
20
    // 64*460800*1/(117964800/4) = 1 segundo

    T2CONbits.T32 = 1; // Activo el temporizador de 32 bits
    T2ON  = 1;         // Arranco el temporizador
25
    while(1)
    {
      while(!T3IF);    // Espera a que llegue
      T3IF  = 0;
30    LATB0 = ! LATB0; // Conmuta RB0
    }
  }
```

7.7.7. Resumen de registros asociados a los módulos temporizadores

Name SFR	Address	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on All Resets
TMR1	0100	Timer1 Register																0000 0000 0000 0000
PR1	0102	Timer1 Period Register																1111 1111 1111 1111
T1CON	0104	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—	0000 0000 0000 0000
TMR2	0106	Timer2 Register																0000 0000 0000 0000
TMR3HLD	0108	Timer3 Holding Register (used in 32-bit mode only)																0000 0000 0000 0000
TMR3	010A	Timer3 Register																0000 0000 0000 0000
PR2	010C	Timer2 Period Register																1111 1111 1111 1111
PR3	010E	Timer3 Period Register																1111 1111 1111 1111
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000
TMR4	0114	Timer4 Register																0000 0000 0000 0000
TMR5HLD	0116	Timer5 Holding Register (used in 32-bit mode only)																0000 0000 0000 0000
TMR5	0118	Timer5 Register																0000 0000 0000 0000
PR4	011A	Timer4 Period Register																1111 1111 1111 1111
PR5	011C	Timer5 Period Register																1111 1111 1111 1111
T4CON	011E	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000
T5CON	0120	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT01F	0000 0000 0000 0000
IFS1	0086	IC61F	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	IC2IE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			INT0IP<2:0>				0100 0100 0100 0100
IPC1	0096	—	T3IP<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			IC2IP<2:0>				0100 0100 0100 0100
IPC5	009E	—	INT2IP<2:0>			—	T5IP<2:0>			—	T4IP<2:0>			OC4IP<2:0>				0100 0100 0100 0100

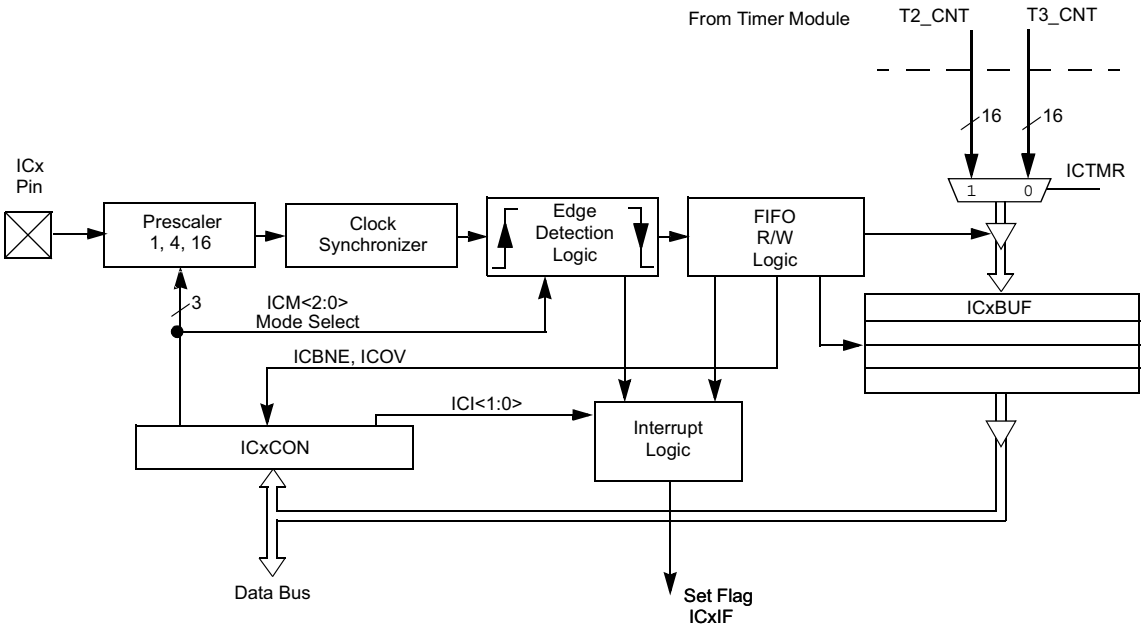
Note: Please refer to the device data sheet for specific memory map details.

7.8. Módulo de captura

7.8.1. Introducción

Estos módulos están pensados para medir pulsos y frecuencias o como entradas adicionales de interrupciones externas. Los dsPIC pueden tener hasta 8 canales de captura. Se controlan con los registros **IC1CON**, **IC2CON**, ..., **IC8CON**.

El diagrama de bloques de funcionamiento lo podemos ver a continuación:



El registro **IC1BUF** se comporta como un búfer FIFO de 4 niveles. Cuando está vacío **IC1CON**.ICBNE=0. Si está lleno y se captura un quinto instante se activará **IC1CON**.ICOV=1. Con cada lectura de **IC1BUF** se elimina del búfer un dato capturado. Para poner a cero el búfer o quitar la condición de desbordamiento del búfer (**IC1CON**.ICOV=1) basta con leer el mismo hasta vaciarlo.

7.8.2. Funcionamiento

Es capaz de capturar el estado del temporizador 2 o el 3:

- Cada flanco de bajada.
- Cada flanco de subida.
- Cada cuarto flanco de subida.
- Cada decimosexto flanco de subida.
- Cada flanco de subida o bajada.

Estos modos se controlan con los bits **IC1CON**.[ICM2..ICM0].

ICxCON: Input Capture x Control Register

Upper Byte:							
U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	ICSIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R-0, HC	R-0, HC	R/W-0	R/W-0	R/W-0
ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>		
bit 7							bit 0

bit 15-14 **Unimplemented:** Read as '0'

bit 13 **ICSIDL:** Input Capture Module Stop in Idle Control bit
1 = Input capture module will halt in CPU Idle mode
0 = Input capture module will continue to operate in CPU Idle mode

bit 12-8 **Unimplemented:** Read as '0'

bit 7 **ICTMR:** Input Capture Timer Select bits
1 = TMR2 contents are captured on capture event
0 = TMR3 contents are captured on capture event

Note: Timer selections may vary. Refer to the device data sheet for details.

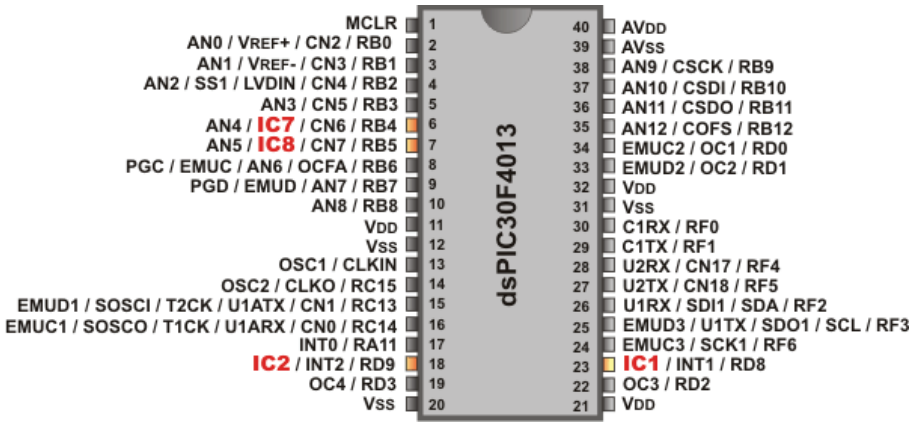
bit 6-5 **ICI<1:0>:** Select Number of Captures per Interrupt bits
11 = Interrupt on every fourth capture event
10 = Interrupt on every third capture event
01 = Interrupt on every second capture event
00 = Interrupt on every capture event

bit 4 **ICOV:** Input Capture Overflow Status Flag (Read Only) bit
1 = Input capture overflow occurred
0 = No input capture overflow occurred

bit 3 **ICBNE:** Input Capture Buffer Empty Status (Read Only) bit
1 = Input capture buffer is not empty, at least one more capture value can be read
0 = Input capture buffer is empty

bit 2-0 **ICM<2:0>:** Input Capture Mode Select bits
111 = Input Capture functions as interrupt pin only, when device is in Sleep or Idle mode
(Rising edge detect only, all other control bits are not applicable.)
110 = Unused (module disabled)
101 = Capture mode, every 16th rising edge
100 = Capture mode, every 4th rising edge
011 = Capture mode, every rising edge
010 = Capture mode, every falling edge
001 = Capture mode, every edge (rising and falling)
(ICI<1:0> does not control interrupt generation for this mode.)
000 = Input capture module turned off

Esquema de las entradas ICx del dsPIC30F4013



Resumen de registros

Memory Map for Input Capture Modules

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0F	0000 0000 0000 0000
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	IR12	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	EI30	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	—	T3IP<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			0100 0100 0100 0100
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>			0100 0100 0100 0100
IPC7	00A2	—	IC6IP<2:0>			—	IC5IP<2:0>			—	IC4IP<2:0>			—	IC3IP<2:0>			0100 0100 0100 0100
IC1BUF	0140	Input 1 Capture Register																uuuu uuuu uuuu uuuu
IC1CON	0142	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC2BUF	0144	Input 2 Capture Register																uuuu uuuu uuuu uuuu
IC2CON	0146	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC3BUF	0148	Input 3 Capture Register																uuuu uuuu uuuu uuuu
IC3CON	014A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC4BUF	014C	Input 4 Capture Register																uuuu uuuu uuuu uuuu
IC4CON	014E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC5BUF	0150	Input 5 Capture Register																uuuu uuuu uuuu uuuu
IC5CON	0152	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC6BUF	0154	Input 6 Capture Register																uuuu uuuu uuuu uuuu
IC6CON	0156	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC7BUF	0158	Input 7 Capture Register																uuuu uuuu uuuu uuuu
IC7CON	015A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC8BUF	015C	Input 8 Capture Register																uuuu uuuu uuuu uuuu
IC8CON	015E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000

Legend: u = uninitialized
Note: Refer to the device data sheet for specific memory map details.

7.9. Módulo de comparación

7.9.1. Introducción

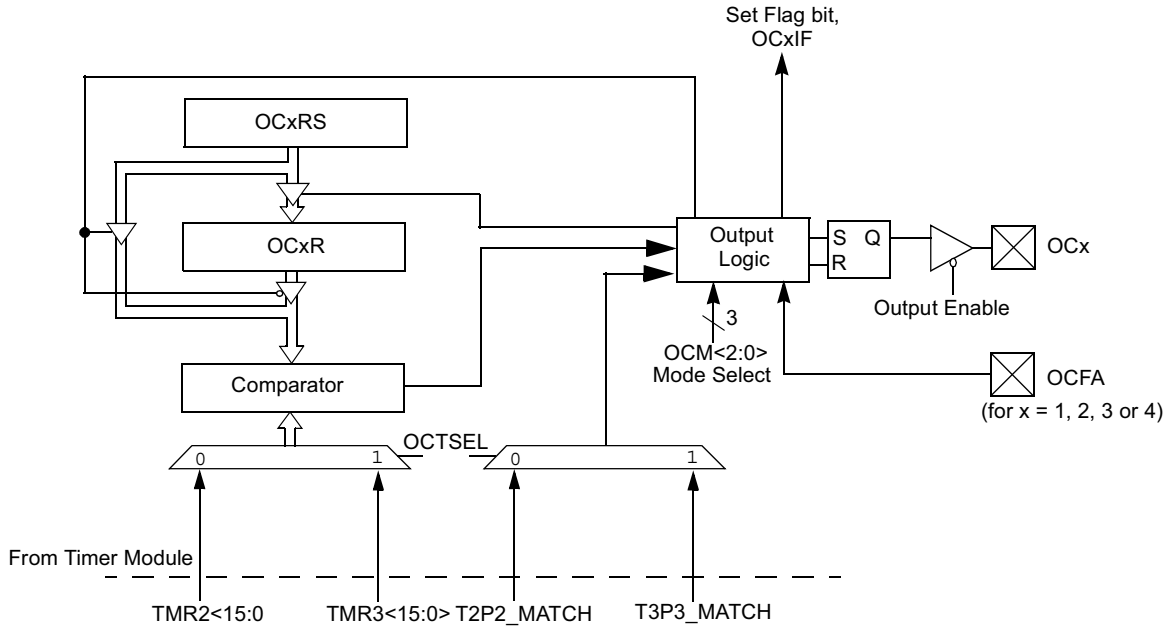
El modulo de comparación permite cambiar una salida o generar una interrupción cuando se alcanza un determinado valor en un temporizador. Podemos encontrarnos hasta 8 módulos de este tipo en los dsPIC. El modelo dsPIC30F4013 dispone de solamente 4 módulos de comparación.

Cada módulo se puede asociar o bien al temporizador TMR2 o al TMR3 mediante el bit `OC1CON.OCTSEL`.

Tiene varios modos de funcionamiento que se controlan con `OC1CON.OCM2..0` y que son:

- Modo de comparación sencillo.
- Modo de comparación doble. Podrá generar un pulso o una secuencia de pulsos.
- Modo PWM (*Pulse Width Modulation*: Modulación por Anchura de Pulsos).

Esquema del módulo de comparación



7.9.2. Registros

OCxCON: Output Compare x Control Register

Upper Byte:							
U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	OCSIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U-0	U-0	U-0	R-0, HC	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OCFLT	OCTSEL	OCM<2:0>		
bit 7							bit 0

bit 15-14 **Unimplemented:** Read as '0'

bit 13 **OCSIDL:** Stop Output Compare in Idle Mode Control bit
1 = Output compare x will halt in CPU Idle mode
0 = Output compare x will continue to operate in CPU Idle mode

bit 12-5 **Unimplemented:** Read as '0'

bit 4 **OCFLT:** PWM Fault Condition Status bit
1 = PWM Fault condition has occurred (cleared in HW only)
0 = No PWM Fault condition has occurred
(This bit is only used when OCM<2:0> = 111.)

bit 3 **OCTSEL:** Output Compare Timer Select bit
1 = Timer3 is the clock source for compare x
0 = Timer2 is the clock source for compare x

Note: Refer to the device data sheet for specific time bases available to the output compare module.

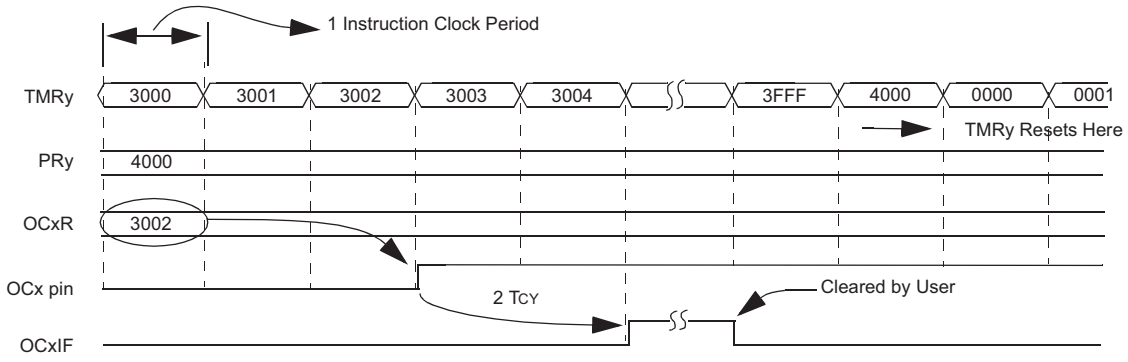
bit 2-0 **OCM<2:0>:** Output Compare Mode Select bits
111 = PWM mode on OCx, Fault pin enabled
110 = PWM mode on OCx, Fault pin disabled
101 = Initialize OCx pin low, generate continuous output pulses on OCx pin
100 = Initialize OCx pin low, generate single output pulse on OCx pin
011 = Compare event toggles OCx pin
010 = Initialize OCx pin high, compare event forces OCx pin low
001 = Initialize OCx pin low, compare event forces OCx pin high
000 = Output compare channel is disabled

Además dispone de los registros de comparación **OC1R**, **OC2R**, ...,**OC8R** y de los registros de comparación secundarios **OC1RS**, **OC2RS**, ..., **OC8RS**.

7.9.3. Modo de comparación sencillo

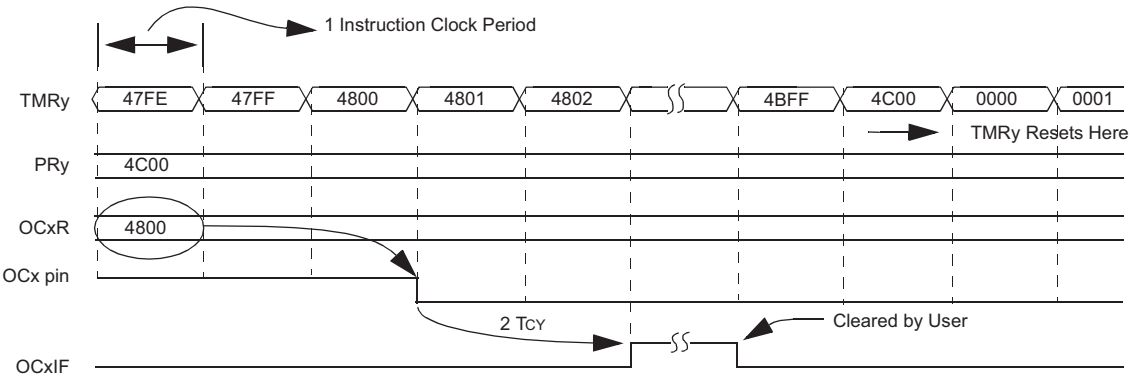
Nos encontraremos en este modo siempre que **OC1CON**.OCM2..0=001,010 o 011. Dependiendo de esto al coincidir el temporizador TMR2 o TMR3 con el registro de comparación (**OC1R**) puede ocurrir:

- La patilla **OC1** se pone en alto y se genera una interrupción (siempre que estén permitidas y tengan la prioridad suficiente).



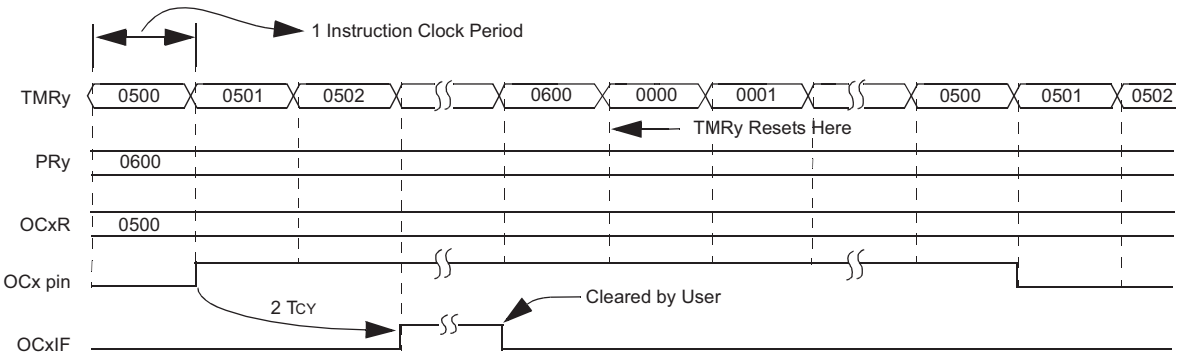
Note: An 'x' represents the output compare channel number. A 'y' represents the time base number.

- La patilla **OC1** se pone en nivel bajo y se genera una interrupción si están permitidas.



Note: An 'x' represents the output compare channel number. A 'y' represents the time base number.

- La patilla **OC1** conmuta su valor.



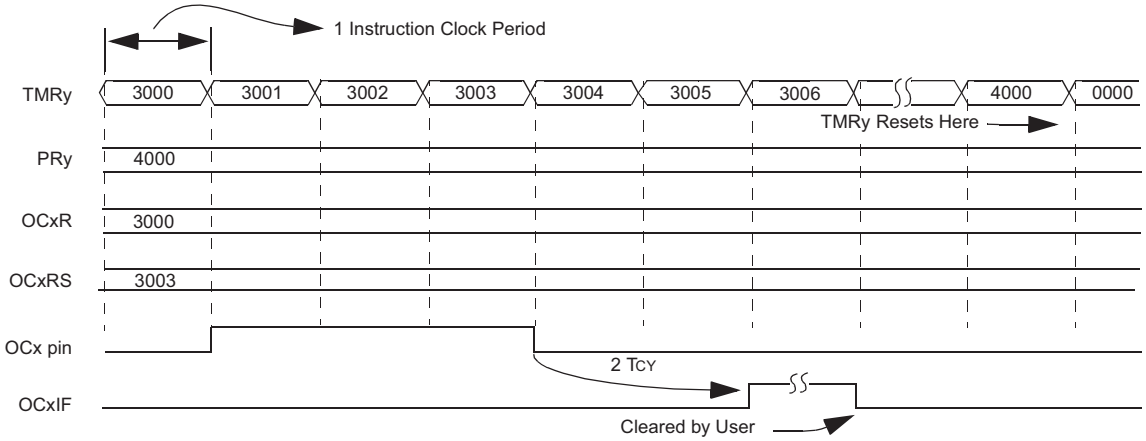
Note: An 'x' represents the output compare channel number. A 'y' represents the time base number.

La interrupción se generará dos ciclos de instrucción después de la coincidencia.

7.9.4. Modo de comparación doble

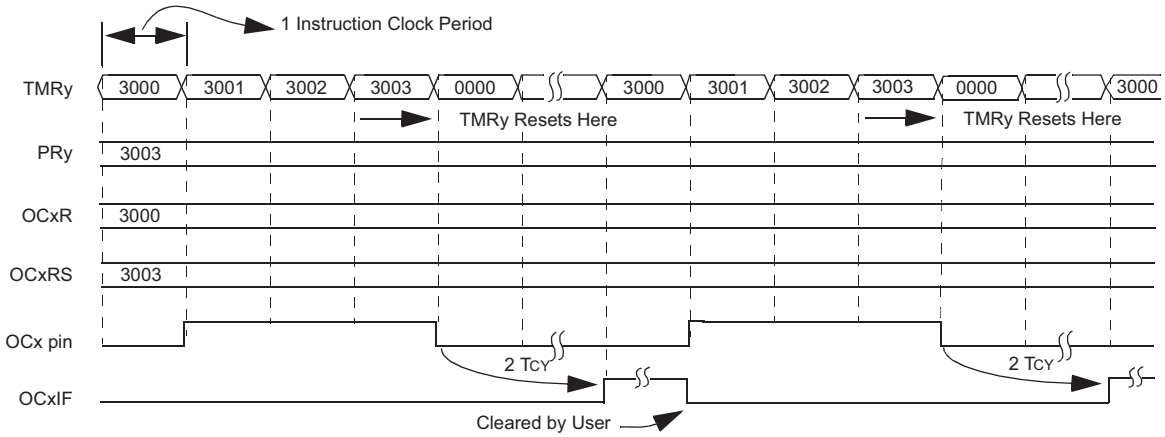
Cuando el campo **OC1CON**.OCM2..0=100 o 101 el módulo de comparación empleará los registros **OC1R** y **OC1RS** para hacer una comparación doble y generar un pulso o una secuencia de pulsos.

- Generación de un pulso. Se deberá cumplir que **OC1R**<**OC1RS**<PRy.



Note 1: An 'x' represents the output compare channel number. A 'y' represents the time base number.
2: OCxR = Compare Register, OCxRS = Secondary Compare Register.

- Generación de una secuencia de pulsos. Se deberá cumplir que **OC1R**<**OC1RS**<PRy.



Note 1: An 'x' represents the output compare channel number. A 'y' represents the time base number.
2: OCxR = Compare Register, OCxRS = Secondary Compare Register.

Si no se cumple la condición **OC1R**<**OC1RS**<PRy nos podemos encontrar con que el pulso no baja.

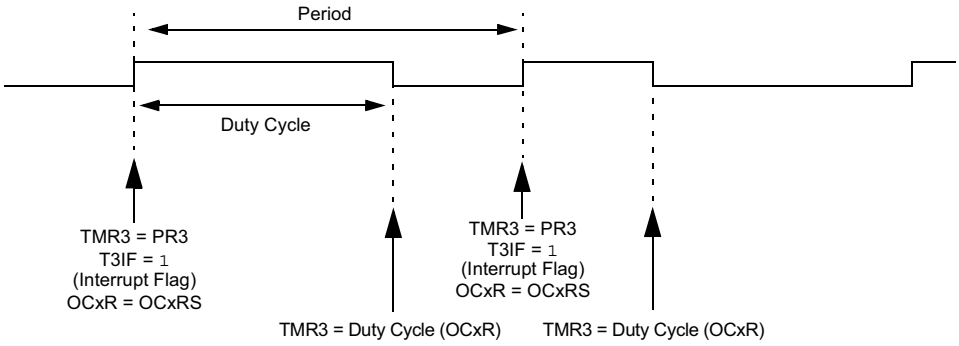
7.9.5. Modo PWM

Cuando **OC1CON**.OCM2..0=110 o 111 el módulo de comparación funciona en modo de modulación de la anchura de pulsos (PWM).

Se emplea el Temporizador TMR2 o TMR3 para definir el periodo de cada pulso.

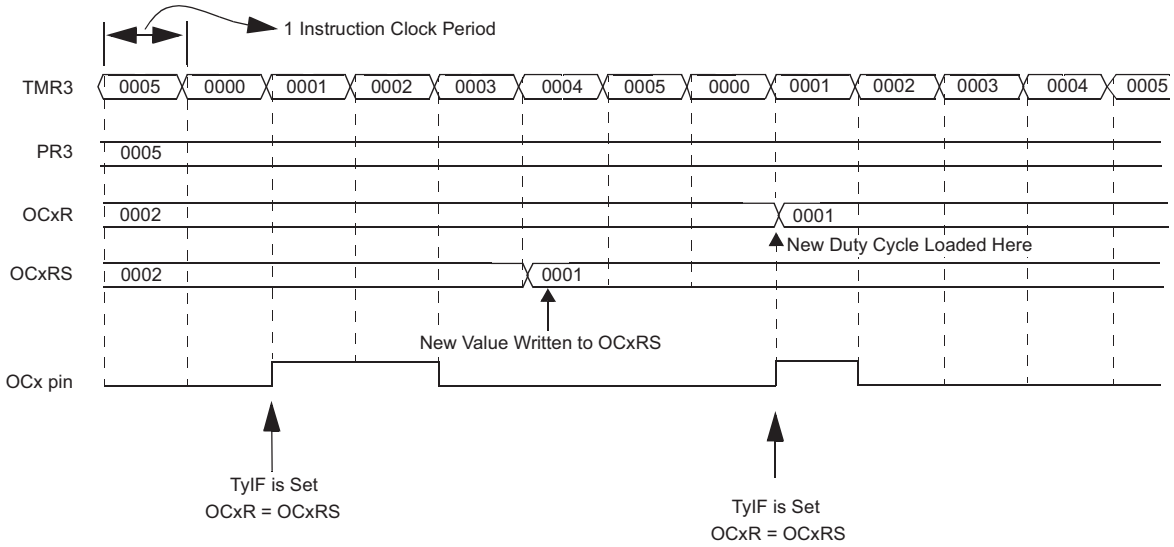
El ciclo de trabajo se define mediante el registro **OC1RS**. El registro **OC1R** será de solo lectura. Al termino de cada periodo el registro **OC1R** = **OC1RS**, por lo que podemos cambiar el ciclo de trabajo después de cada pulso. Al llegar al final de la cuenta del temporizador

la patilla **OC1** pasa a valer cero. Cuando el valor del temporizador es igual al del registro OC1R se pone a uno la patilla **OC1**.



Existen dos modos:

- Modo PWM usando la entrada de protección de fallo. Se usa la patilla **OCFA** si se trata de los módulos de comparación OC1, ... OC4 o la patilla **OCFB** si se trata de los módulos OC5,.. OC8. Este modo interrumpe la generación de la señal PWM cuando por esta entrada llega un valor alto. En este modo si se produce un fallo se puede generar una interrupción para tratar el problema siempre que estén permitidas (OC1CON.OCFLT=1 lo notifica).
- Modo PWM sin usar la entrada de protección de fallos. No se tienen en cuenta las entradas de protección de fallos.



Note 1: An 'x' represents the output compare channel number. A 'y' represents the time base number.
2: OCxR = Compare Register, OCxRS = Secondary Compare Register.

Los pasos para configurar este modo deberían ser:

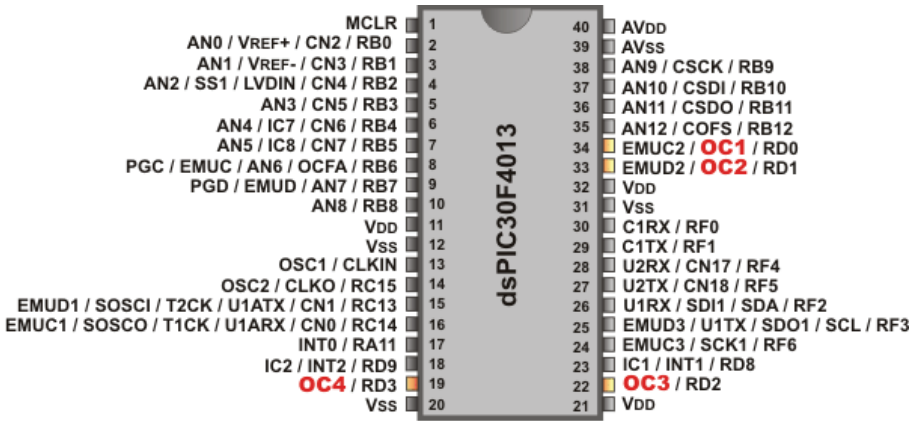
- Definir el periodo de la señal PWM configurando el registro periodo del temporizador seleccionado.

- Definir el ciclo de trabajo mediante el registro **OC1RS** siempre menor que el valor del registro periodo del temporizador.
- Escribir en el registro **OC1R** el valor inicial del primer ciclo de trabajo de la señal.
- Habilitar las interrupciones para el temporizador seleccionado.
- Configurar el módulo de comparación en uno de los dos modos PWM disponibles.
- Terminar de configurar y activar el funcionamiento del temporizador seleccionado.

La resolución del modo PWM dependerá de la siguiente fórmula:

$$Resolution = \frac{\log_{10} T_{PWM} - \log_{10}(T_{CY} * PreescalaTemporizador)}{\log_{10} 2} \tag{7.1}$$

Esquema de las salidas OCx del dsPIC30F4013



7.9.6. Resumen de registros

Table 14-6: Example Register Map Associated with Output Compare Module

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TMR2	0106	Timer2 Register																0000 0000 0000 0000
TMR3	010A	Timer3 Register																0000 0000 0000 0000
PR2	010C	Period Register 2																1111 1111 1111 1111
PR3	010E	Period Register 3																1111 1111 1111 1111
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000
OC1RS	0180	Output Compare 1 Secondary Register																uuuu uuuu uuuu uuuu
OC1R	0182	Output Compare 1 Register																uuuu uuuu uuuu uuuu
OC1CON	0184	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC2RS	0186	Output Compare 2 Secondary Register																uuuu uuuu uuuu uuuu
OC2R	0188	Output Compare 2 Register																uuuu uuuu uuuu uuuu
OC2CON	018A	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC3RS	018C	Output Compare 3 Secondary Register																uuuu uuuu uuuu uuuu
OC3R	018E	Output Compare 3 Register																uuuu uuuu uuuu uuuu
OC3CON	0190	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC4RS	0192	Output Compare 4 Secondary Register																uuuu uuuu uuuu uuuu
OC4R	0194	Output Compare 4 Register																uuuu uuuu uuuu uuuu
OC4CON	0196	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC5RS	0198	Output Compare 5 Secondary Register																uuuu uuuu uuuu uuuu
OC5R	019A	Output Compare 5 Register																uuuu uuuu uuuu uuuu
OC5CON	019C	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC6RS	019E	Output Compare 6 Secondary Register																uuuu uuuu uuuu uuuu
OC6R	01A0	Output Compare 6 Register																uuuu uuuu uuuu uuuu
OC6CON	01A2	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC7RS	01A4	Output Compare 7 Secondary Register																uuuu uuuu uuuu uuuu
OC7R	01A6	Output Compare 7 Register																uuuu uuuu uuuu uuuu
OC7CON	01A8	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC8RS	01AA	Output Compare 8 Secondary Register																uuuu uuuu uuuu uuuu
OC8R	01AC	Output Compare 8 Register																uuuu uuuu uuuu uuuu
OC8CON	01AE	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000

Legend: u = uninitialized

Note: The register map will depend on the number of output compare modules on the device. Please refer to the device data sheet for details.

Table 14-6: Example Register Map Associated with Output Compare Module (Continued)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IFS2	0088	—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IEC2	0090	—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIE	PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>		—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>				0100 0100 0100 0100
IPC1	0096	—	T3IP<2:0>		—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>				0100 0100 0100 0100
IPC4	009C	—	OC3IP<2:0>		—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>				0100 0100 0100 0100
IPC5	009E	—	INT2IP<2:0>		—	T5IP<2:0>			—	T4IP<2:0>			—	OC4IP<2:0>				0100 0100 0100 0100
IPC8	00A4	—	OC8IP<2:0>		—	OC7IP<2:0>			—	OC6IP<2:0>			—	OC5IP<2:0>				0100 0100 0100 0100

Legend: u = uninitialized

Note: The register map will depend on the number of output compare modules on the device. Please refer to the device data sheet for details.

7.10. Módulo ADC

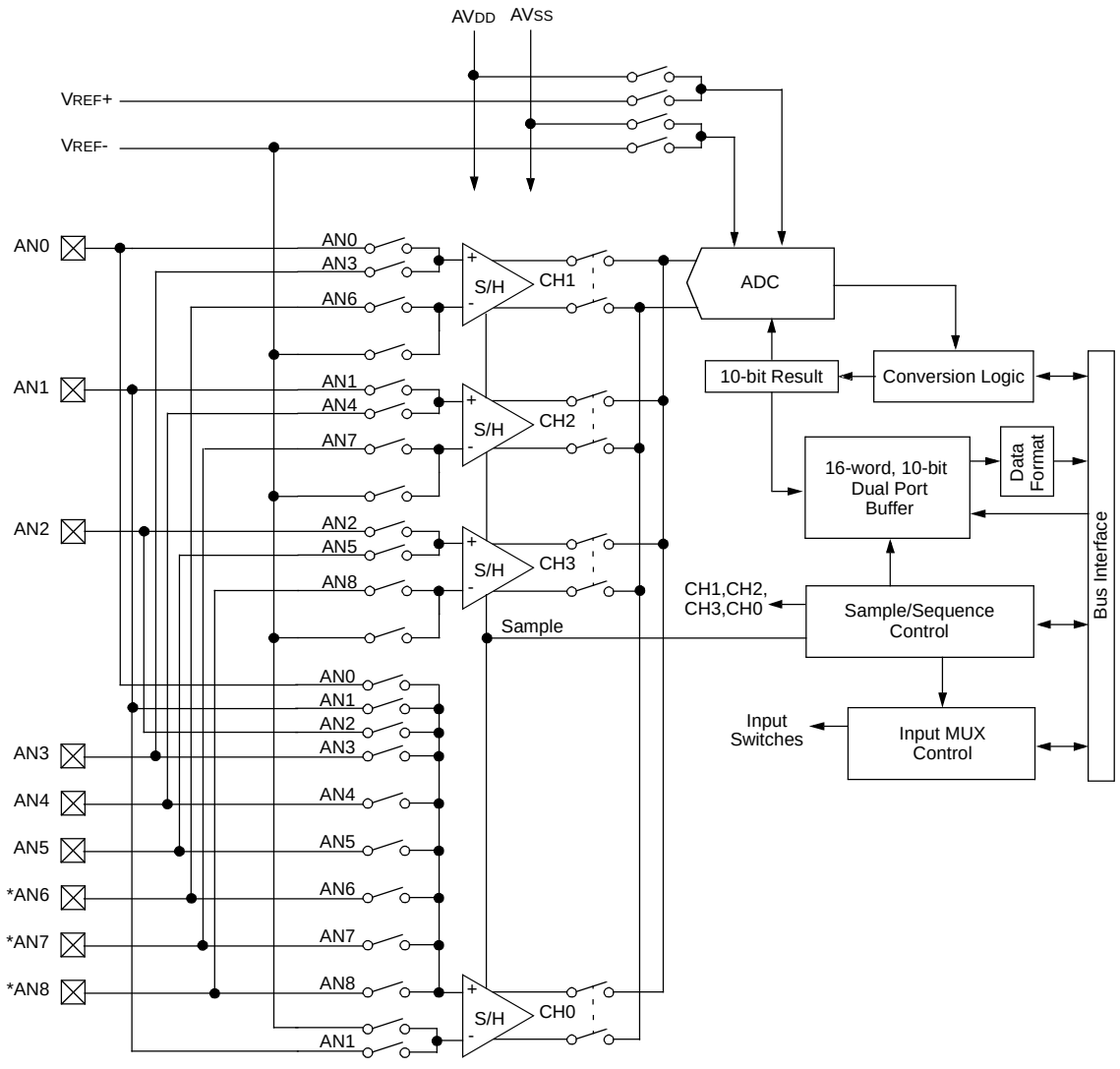
7.10.1. Introducción

Algunos modelos de la familia dsPIC disponen de un módulo conversor analógico digital de 10 bits de alta velocidad (hasta 1MSps[10]) o de 12 bits de velocidad inferior (depende del modelo). Algunas de las características:

- Hasta 16 canales de entrada analógica.
- Entradas de tensión de referencia.

[10]Megamuestras por segundo.

- Hasta cuatro amplificadores *sample/hold*. Lo que permite el muestreo simultaneo de hasta cuatro entradas analógicas.
- Modo de muestreo de canales automático.
- Buffer de 16 posiciones de los valores digitales obtenidos.
- Cuatro opciones de alineación.
- Operación durante modos de bajo consumo *Sleep* e *Idle*.



Según el modelo se pueden tener hasta 16 pines de entradas analógicas **AN0**..**AN15**. Además, se pueden poner referencias externas para el voltaje inferior y superior a través de dos pines que pueden estar compartidos con otros pines de entrada analógica o no.

Las entradas analógicas se conectan a través de multiplexores analógicos a hasta cuatro amplificadores S/H (CH0-CH3). Se pueden utilizar uno, dos, o cuatro amplificadores S/H para la adquisición de datos de entrada.

Los multiplexores de entrada pueden conmutarse entre dos conjuntos de entradas analógicas durante las conversiones.

Son posibles las conversiones diferenciales o simples en todos los canales de entrada.

El conversor A/D de 10/12 bits está conectado a un buffer de 16 posiciones. Cada resultado se alinea a uno de los cuatro formatos de salida posibles.

7.10.2. Registros

El conversor A/D dispone de seis registros de control:

- **ADCON1**: Registro de control 1.
- **ADCON2**: Registro de control 2.
- **ADCON3**: Registro de control 3.
- **ADCHS**: Registro de selección del canal de entrada del conversor.
- **ADPCFG**: Registro de configuración de puerto analógico/digital.
- **ADCSSL**: Registro de selección de entradas a muestrear.
- Además dispone de hasta 16 registros de captura **ADCBUF0**, **ADCBUF1** ...,**ADCBUFE**, **ADCBUFF**.

A continuación examinamos las funciones de cada registro:

ADCON1

ADCON1: A/D Control Register 1

Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
ADON	—	ADSIDL	—	—	—	FORM<1:0>	
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0 HC, HS	R/C-0 HC, HS
SSRC<2:0>			—	—	ASAM	SAMP	DONE
bit 7							bit 0

bit 15 **ADON:** A/D Operating Mode bit
1 = A/D converter module is operating
0 = A/D converter is off

bit 14 **Unimplemented:** Read as '0'

bit 13 **ADSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12-10 **Unimplemented:** Read as '0'

bit 9-8 **FORM<1:0>:** Data Output Format bits
11 = Signed fractional (DOUT = sddd dddd dddd 0000)
10 = Fractional (DOUT = dddd dddd dddd 0000)
01 = Signed integer (DOUT = ssss sddd dddd dddd)
00 = Integer (DOUT = 0000 dddd dddd dddd)

bit 7-5 **SSRC<2:0>:** Conversion Trigger Source Select bits
111 = Internal counter ends sampling and starts conversion (auto convert)
110 = Reserved
101 = Reserved
100 = Reserved
011 = Motor Control PWM interval ends sampling and starts conversion
010 = General purpose Timer3 compare ends sampling and starts conversion
001 = Active transition on INT0 pin ends sampling and starts conversion
000 = Clearing SAMP bit ends sampling and starts conversion

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **ASAM:** A/D Sample Auto-Start bit
1 = Sampling begins immediately after last conversion completes. SAMP bit is auto set.
0 = Sampling begins when SAMP bit set

bit 1 **SAMP:** A/D Sample Enable bit
1 = At least one A/D sample/hold amplifier is sampling
0 = A/D sample/hold amplifiers are holding
When ASAM = 0, writing '1' to this bit will start sampling.
When SSRC = 000, writing '0' to this bit will end sampling and start conversion.

bit 0 **DONE:** A/D Conversion Status bit
1 = A/D conversion is done
0 = A/D conversion is not done
Clearing this bit will not effect any operation in progress.
Cleared by software or start of a new conversion.

ADON Control de operación.

0: Convertidor A/D apagado.

1: Convertidor A/D funcionando.

ADSIDL Parar en modo *Idle*.

0: Continua funcionando en modo *Idle*.

1: Deja de funcionar en modo *Idle*.

FORM[1:0] Formato de los datos de salida.

00: = Entero sin signo (DOUT = 0000 00dd dddd dddd).

01: = Entero con signo (DOUT = ssss sssd dddd dddd).

10: = Fraccionario 1.15 (DOUT = dddd dddd dd00 0000).

11: = Fraccionario 1.15 con signo (DOUT = sddd dddd dd00 0000).

SSRC[2:0] Selecciona la fuente de disparo de la conversión.

111: Un contador interno finaliza el muestreo e inicializa la conversión (automático).

110,101,100: Reservado.

011: Sincronizado con señal PWM (finaliza el muestreo e inicializa la conversión).

010: Sincronizado con Timer3 (función de comparación).

001: Sincronizado con transición en el pin INT0.

000: Borrar el bit SAMP finaliza el muestreo e inicializa la conversión.

SIMSAM Bit de selección de muestreo simultaneo (solo aplicable cuando CHPS = 01 o 1x).

1: Muestreo simultaneo de los canales CH0, CH1, CH2, CH3 (cuando CHPS = 1x) o muestreo simultaneo en los canales CH0 y CH1 (cuando CHPS = 01).

0: Muestro de múltiples canales en secuencia individual.

ASAM Bit de auto-comienzo de muestreo del convertidor A/D.

1: Muestreo empieza inmediatamente después que la última conversión halla finalizado. Bit SAMP se pone a uno automáticamente.

0: Muestreo empieza cuando el bit SAMP se pone a uno.

SAMP Habilita el muestreo del convertidor.

1: Al menos un amplificador sample/hold está muestreando.

0: Todos los amplificadores sample and hold están ocupados manteniendo un dato. Cuando el bit ASAM es cero, escribiendo '1' en este bit empieza el muestreo. Cuando los bit SSRC = 000 finaliza el muestreo y empieza la conversión.

DONE Bit de estado de la conversión.

1: Conversión A/D está hecha.

0: Conversión A/D no está hecha. Se borra por software o al empezar una nueva conversión. Si se borra este bit no afecta a las operaciones que estén en progreso.

ADCON2

ADCON2: A/D Control Register 2

Upper Byte:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0
VCFG<2:0>			—	—	CSCNA	—	—
bit 15							bit 8

Lower Byte:							
R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	SMPI<3:0>				BUFM	ALTS
bit 7							bit 0

bit 15-13 **VCFG<2:0>:** Voltage Reference Configuration bits

	A/D VREFH	**A/D VREFL**
000	AVDD	AVSS
001	External VREF+ pin	AVSS
010	AVDD	External VREF- pin
011	External VREF+ pin	External VREF- pin
1xx	AVDD	AVSS

bit 12 **Reserved:** User should write '0' to this location

bit 11 **Unimplemented:** Read as '0'

bit 10 **CSCNA:** Scan Input Selections for CH0+ S/H Input for MUX A Input Multiplexer Setting bit
1 = Scan inputs
0 = Do not scan inputs

bit 9-8 **Unimplemented:** Read as '0'

bit 7 **BUFS:** Buffer Fill Status bit
Only valid when BUFM = 1 (ADRES split into 2 x 8-word buffers)
1 = A/D is currently filling buffer 0x8-0xF, user should access data in 0x0-0x7
0 = A/D is currently filling buffer 0x0-0x7, user should access data in 0x8-0xF

bit 6 **Unimplemented:** Read as '0'

bit 5-2 **SMPI<3:0>:** Sample/Convert Sequences Per Interrupt Selection bits
1111 = Interrupts at the completion of conversion for each 16th sample/convert sequence
1110 = Interrupts at the completion of conversion for each 15th sample/convert sequence
.....
0001 = Interrupts at the completion of conversion for each 2nd sample/convert sequence
0000 = Interrupts at the completion of conversion for each sample/convert sequence

bit 1 **BUFM:** Buffer Mode Select bit
1 = Buffer configured as two 8-word buffers ADCBUF(15...8), ADCBUF(7...0)
0 = Buffer configured as one 16-word buffer ADCBUF(15...0)

bit 0 **ALTS:** Alternate Input Sample Mode Select bit
1 = Uses MUX A input multiplexer settings for first sample, then alternate between MUX B and MUX A input multiplexer settings for all subsequent samples
0 = Always use MUX A input multiplexer settings

VCFG[2:0] Bits de configuración de la tensión de referencia.

	A/D VREFH	**A/D VREFL**
000	AVDD	AVSS
001	External VREF+ pin	AVSS
010	AVDD	External VREF- pin
011	External VREF+ pin	External VREF- pin
1XX	AVDD	AVSS

CSCNA Escanea la selección de entradas para el canal CH0+ S/H a través de MUX A.

1: Escanea las entradas.

0: No escanea las entradas.

CHPS[1:0]Selección de los canales empleados.

1x: Convierte los canales CH0, CH1, CH2 y CH3.

01: Convierte los canales CH0 y CH1.

00: Convierte el canal CH0.

Cuando el bit ADCON1.SIMSAM = 0 los canales son muestreados secuencialmente. Cuando el bit ADCON1.SIMSAM = 1 los canales son muestreados según CHPS[1:0].

BUFS Bit de estado del búfer. Solo válido cuando BUFM = 1 (ADRES dividida en 2 x 8 búferes).

1: Posiciones 0x8-0xF del buffer no disponibles, el usuario debe acceder a los datos de las posiciones 0x0-0x7.

0: Posiciones 0x0-0x7 del buffer no disponibles, el usuario debe acceder a los datos de las posiciones 0x8-0xF.

SMPI[3:0] Muestreos para generar una interrupción.

1111: Se genera una interrupción cada 16ª muestreo/conversión.

1110: Se genera una interrupción cada 15º muestreo/conversión.

1101: Se genera una interrupción cada 15º muestreo/conversión.

- ...

0010: Se genera una interrupción cada 3º muestreo/conversión.

0001: Se genera una interrupción cada 2º muestreo/conversión.

0000: Se genera una interrupción cada vez que se termina una secuencia muestreo/conversión.

BUFM Modo de funcionamiento del búfer.

1: Búfer configurado como dos búferes de 8 palabras cada una, ADCBUF(15..8) y ADCBUF(7...0).

0: Búfer configurado como un único búfer de 16 palabras ADCBUF(15..0).

ALTS Modo alternado. Selección de multiplexor.

1: Usa la configuración de entrada del multiplexor MUX A para el primer muestreo, para luego alternar entre los multiplexores MUX A y MUX B.

0: Utiliza siempre el multiplexor MUX A.

ADCON3

ADCON3: A/D Control Register 3

Upper Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SAMC<4:0>				
bit 15							bit 8

Lower Byte:							
R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	—	ADCS<5:0>					
bit 7							bit 0

bit 15-13 **Unimplemented:** Read as '0'

bit 12-8 **SAMC<4:0>:** Auto Sample Time bits
11111 = 31 TAD
.....
00001 = 1 TAD
00000 = 0 TAD

bit 7 **ADRC:** A/D Conversion Clock Source bit
1 = A/D internal RC clock
0 = Clock derived from system clock

bit 6 **Unimplemented:** Read as '0'

bit 5-0 **ADCS<5:0>:** A/D Conversion Clock Select bits
111111 = TCY/2 • (ADCS<5:0> + 1) = 32 • TCY
.....
000001 = TCY/2 • (ADCS<5:0> + 1) = TCY
000000 = TCY/2 • (ADCS<5:0> + 1) = TCY/2

SAMC[4:0] Indica el tiempo de muestreo automático.

11111: $31T_{AD}$

...

00001: $1T_{AD}$

00000: $0T_{AD}$ (sólo si se permite realizar conversiones secuencial usando más de un amplificador S/H).

ADRC: Fuente del reloj del conversor A/D.

1: Reloj RC interno para el convertidor A/D.

0: Reloj obtenido del reloj del sistema.

ADCS[5:0] Bits que realiza la selección del reloj de conversión.

111111: $T_{CY}/2$ * (ADCS[5:0] + 1) = $32T_{CY}$.

...

000001: $T_{CY}/2$ * (ADCS[5:0] + 1) = T_{CY}.

000000: $T_{CY}/2$ * (ADCS[5:0] + 1) = $T_{CY}/2$.

ADCHS

Este registro se encarga de seleccionar los pines de entrada que van a ser conectados a los amplificadores S/H.

ADCHS: A/D Input Select Register

Upper Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CH0NB	CH0SB<3:0>			
bit 15							bit 8

Lower Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CH0NA	CH0SA<3:0>			
bit 7							bit 0

bit 15-13 **Unimplemented:** Read as '0'

bit 12 **CH0NB:** Channel 0 Negative Input Select for MUX B Multiplexer Setting bit
Same definition as bit <4> (see **Note**).

bit 11-8 **CH0SB<3:0>:** Channel 0 Positive Input Select for MUX B Multiplexer Setting bit
Same definition as bits <3:0> (see **Note**).

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **CH0NA:** Channel 0 Negative Input Select for MUX A Multiplexer Setting bit
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VREF-

bit 3-0 **CH0SA<3:0>:** Channel 0 Positive Input Select for MUX A Multiplexer Setting bit
1111 = Channel 0 positive input is AN15
1110 = Channel 0 positive input is AN14
1101 = Channel 0 positive input is AN13
.....
0001 = Channel 0 positive input is AN1
0000 = Channel 0 positive input is AN0

Note: The analog input multiplexer supports two input setting configurations, denoted MUX A and MUX B. ADCHS<15:8> determines the settings for MUX B, and ADCHS<7:0> determines the settings for MUX A. Both sets of control bits function identically.

CH123NB[1:0]: Selecciona la entrada negativa de los canales 1,2,3 para la configuración del MUX B.

11: El CH1 negativo es la entrada AN9.
El CH2 negativo es la entrada AN10.
El CH3 negativo es la entrada AN11.

10: El CH1 negativo es la entrada AN6.
El CH2 negativo es la entrada AN7.
El CH3 negativo es la entrada AN8

0x: Las entradas negativas de CH1, CH2 y CH3 son la entrada V_{REF-}.

CH123SB Selecciona la entrada positiva de los canales 1,2,3 para la configuración del MUX B.

1: El CH1 positivo es la entrada AN3.
El CH2 positivo es la entrada AN4.
El CH3 positivo es la entrada AN5.

0: El CH1 positivo es la entrada AN0.
El CH2 positivo es la entrada AN1.
El CH3 positivo es la entrada AN2.

CH0NB Selecciona la entrada negativa del canal 0 para la configuración del MUX B

1: El CH0 negativo es la entrada AN1.

0: El CH0 negativo es la entrada V_{REF-}.

CH0SB[3:0] Selecciona la entrada negativa del canal 0 para la configuración del MUX B.

1111: El CH0 positivo es la entrada AN15.

1110: El CH0 positivo es la entrada AN14.

1101: El CH0 positivo es la entrada AN13.

....:

0001: El CH0 positivo es la entrada AN1.

0000: El CH0 positivo es la entrada AN0.

CH123NA[1:0] Selecciona la entrada negativa de los canales 1,2,3 para la configuración del multiplexor MUX A.

11: El CH1 negativo es la entrada AN9.
El CH2 negativo es la entrada AN10.
El CH3 negativo es la entrada AN11.

10: El CH1 negativo es la entrada AN6.
El CH2 negativo es la entrada AN7.
El CH3 negativo es la entrada AN8

0x: Las entradas negativas de CH1, CH2 y CH3 son la entrada V_{REF-}.

CH123SA Selecciona la entrada positiva de los canales 1,2,3 para la configuración del multiplexor MUX A.

1: El CH1 positivo es la entrada AN3.
El CH2 positivo es la entrada AN4.
El CH3 positivo es la entrada AN5.

0: El CH1 positivo es la entrada AN0.
El CH2 positivo es la entrada AN1.
El CH3 positivo es la entrada AN2.

CH0NA Selecciona la entrada negativa del canal 0 para la configuración del MUX A.

1: El CH0 negativo es la entrada AN1.

0: El CH0 negativo es la entrada V_{REF-}.

CH0SA[3:0] Selecciona la entrada negativa del canal 0 para la configuración del MUXA.

1111: El CH0 positivo es la entrada AN15.

1110: El CH0 positivo es la entrada AN14.

1101: El CH0 positivo es la entrada AN13.

....:

0001: El CH0 positivo es la entrada AN1.

0000: El CH0 positivo es la entrada AN0.

ADPCFG

Es el encargado de configurar las patillas de entrada como entradas analógicas o como entradas digitales.

ADPCFG: A/D Port Configuration Register

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 15-0 **PCFG<15:0>:** Analog Input Pin Configuration Control bits
1 = Analog input pin in Digital mode, port read input enabled, A/D input multiplexer input connected to AVSS
0 = Analog input pin in Analog mode, port read input disabled, A/D samples pin voltage

ADCSSL

Este registro selecciona que entradas serán muestreadas.

ADCSSL: A/D Input Scan Select Register

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0
bit 7							bit 0

bit 15-0 **CSSL<15:0>:** A/D Input Pin Scan Selection bits
1 = Select ANx for input scan
0 = Skip ANx for input scan

ADCBUFx. Búfer de resultados

El módulo contiene una RAM de 16 palabras de doble puerto, llamada ADCBUF, para almacenar las conversiones. Las posiciones del mismos se denominan **ADCBUF0**, **ADCBUF1**, **ADCBUF2**, ... **ADCBUFE**, **ADCBUFF**.

7.10.3. Funcionamiento del conversor A/D

La conversión analógica/digital tiene dos fases:

Muestreo. La fase de muestreo captura a través de los amplificadores *sample/hold* (S/H) el valor de tensión analógica de uno de los pines de entrada.

Según el modelo de dsPIC se pueden tener uno o varios de estos S/H que se denominarán CH0-CH3. El conexionado de estos con los pines de entrada se hace a través de multiplexores analógicos controlados por el registro **ADCHS**.

El tiempo de muestreo es el tiempo que el S/H está conectado a la entrada analógica. El muestreo puede comenzar/finalizar manualmente activando el bit **ADCON1.SAMP** o realizarse automáticamente según la configuración.

Conversión. La fase de conversión obtiene el valor digital del valor analógico muestreado previamente.

El conversor A/D requiere un tiempo T_{AD} para convertir cada bit del resultado. Se necesitan $12T_{AD}$ para una la conversión completa (si la precisión es de 12 bits).

El resultado se almacena en uno de 16 registros `ADCBUFx` y automáticamente el S/H puede ser reconectado al pin de entrada y se puede generar una interrupción.

La suma del tiempo de muestreo y el tiempo de conversión del ADC proporciona el tiempo total de tiempo de conversión.

Hay un tiempo mínimo de muestreo para asegurar que el S/H dé la precisión deseada para la conversión A/D .

Un proceso de muestreo/conversión que utilice varios canales S/H las entradas pueden ser muestreadas simultáneamente, esto está controlado por el bit **ADCON1.SIMSAM**. El muestreo simultáneo asegura que las entradas analógicas se producen en el mismo instante de tiempo para todos los canales.

Con el muestreo secuencial se toma una instantánea de cada entrada analógica justo antes de la conversión, y la toma de muestras de múltiples entradas no se correlacionan.

El inicio del muestro puede ser controlado por software mediante el bit **ADCON1.SAMP**, o controlado automáticamente por el hardware controlado por el bit **ADCON1.ASAM**. La fuente de disparo es seleccionada por los bits de control **ADCON1.SSRC2**`..0`.

El número de secuencias muestreo/conversión entre interrupciones puede variar entre 1 y 16. El número total de resultados de conversión entre las interrupciones es el producto de los canales por muestra y el valor de **ADCON2.SMPI3**`..0`. El número total de conversiones entre las interrupciones no deben exceder el tamaño del buffer.

7.10.4. Pasos a seguir

Configuración del módulo A/D

- Los pines que van a ser entradas analógicos hay que configurarlos como entradas (registros `TRISx`).

- Además se debe configurar el funcionamiento analógico de las mismas mediante los bits **ADPCFG**.PCFGx que valdrán 0 si son analógicos.
- Se seleccionan las referencia del voltaje con **ADCON2**.**VCFG2**..0
- Se seleccionan los canales de entrada mediante el registro **ADCHS**.

  El S/H denominado CH0 dispone de dos configuraciones para funcionar:

  La configuración denominada MUX A se controla con los campos **ADCHS**.**CHOSA3**..0 y **ADCHS**.**CHONA**.

  La configuración denominada MUX B se controla con los campos **ADCHS**.**CHOSB3**..0 y **ADCHS**.**CHONB**.

  Cuando el campo **ADCON2**.**ALTS**=1 el módulo A/D alterna entre las entradas del MUX A en un muestreo y las de MUX B en el siguiente.

  En los canales S/H 1,2 y 3 se puede elegir entre dos grupos de 3 entradas. El registro **ADCHS** selecciona las fuentes de las entradas positivas y negativas de estos canales.
- Se determina qué canales se van a muestrear **ADCSSL**.
- Se selecciona la secuencia muestro/conversión adecuada **ADCON1**.
- Se selecciona la forma de presentar los resultados en el buffer mediante **ADCON1**.**FORM1**..0.
- Se selecciona la frecuencia del reloj mediante **ADCON3**.
- Selecciona el disparo del convertidor con **ADCON1**.**SSRC2**..0.
- Se habilita el módulo **ADCON1**.ADON=1.
- Se configura la interrupción de conversor (si es necesario). Para ello borramos el bit **ADIF**. Seleccionamos la prioridad de la interrupción y activamos la interrupción **ADIE**=1.

Inicio del muestreo

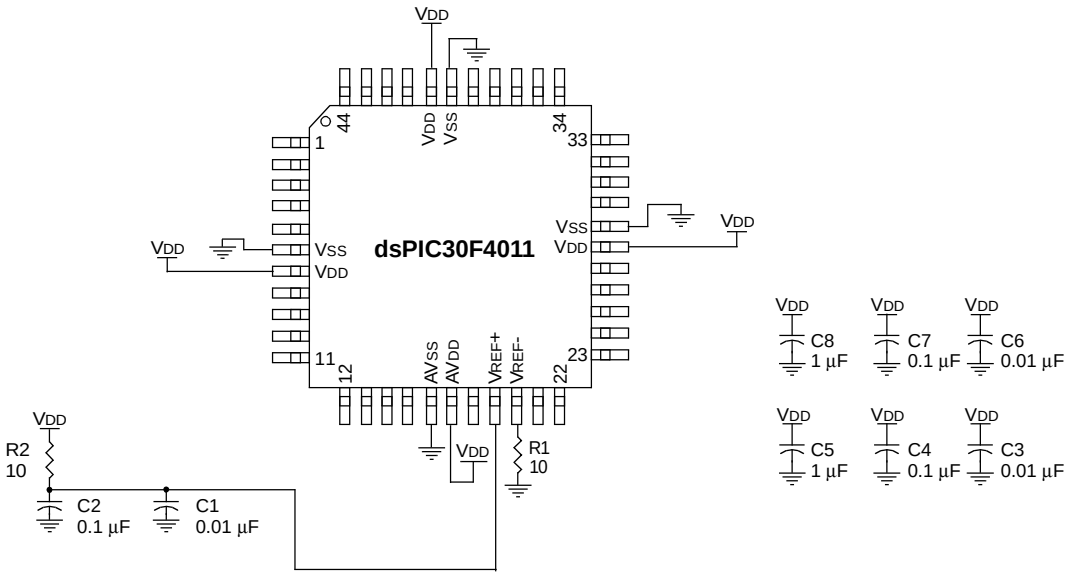
En modo manual se inicia con el bit **ADCON1**.**SAMP**. En modo automático **ADCON1**.**ASAM**=1 el muestreo se inicia cuando termina la conversión anterior.

Se espera el tiempo necesario de adquisición de la muestra. La conversión se inicia cuando termina el muestreo. Con los bits **ADCON1**.**SSRC2**..0 se selecciona la fuente de disparo del conversor.

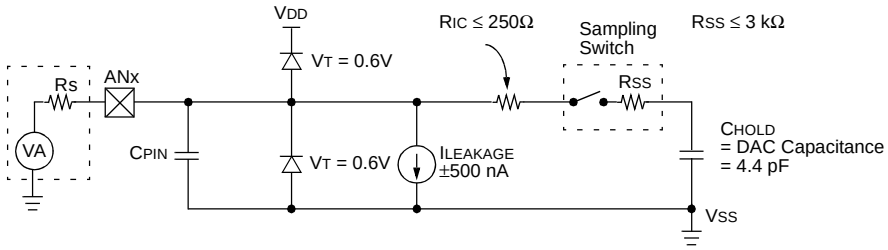
Se espera a que se complete la conversión **ADCON1**.**DONE**=1. Se lee el resultado del buffer y borra el bit **ADIF** si es necesario.

7.10.5. Consideraciones eléctricas

El dsPIC debe disponer de los desacoplos de alimentación indicados:

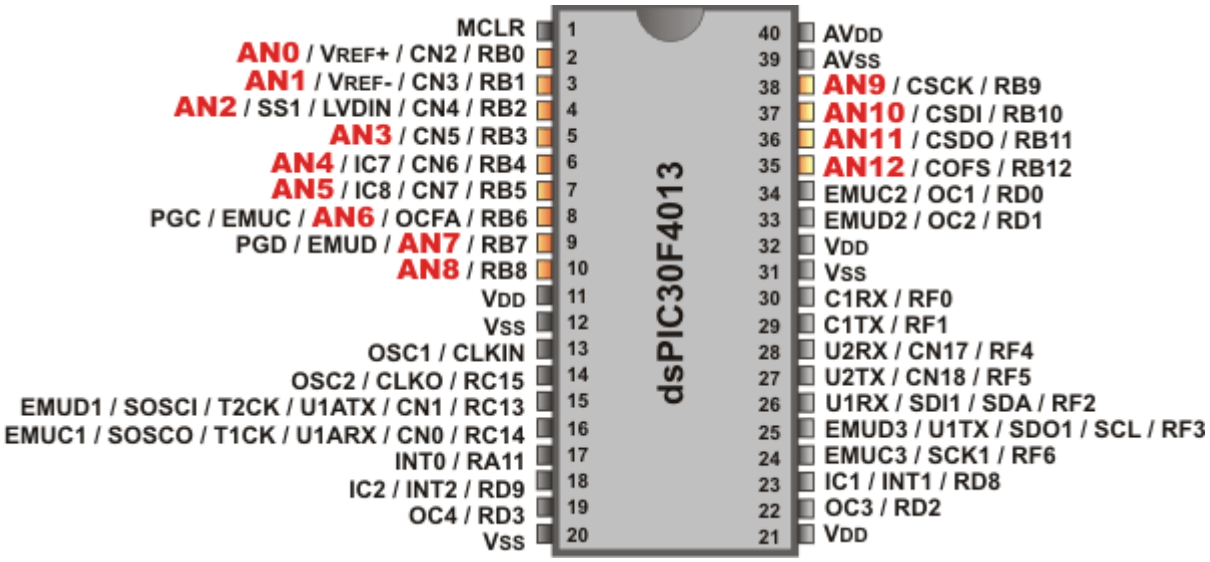


El modelo que representa el convertidor se puede ver a continuación:



Legend:	CPIN	= Input Capacitance
	VT	= Threshold Voltage
	I leakage	= Leakage Current at the pin due to various junctions
	RIC	= Interconnect Resistance
	RSS	= Sampling Switch Resistance
	CHOLD	= Sample/Hold Capacitance (from DAC)

Esquema de los pines del ADC del dsPIC30F4013



7.10.6. Resumen de registros

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
ADCBUF0	0280	—	—	—	—	—	—	ADC Data Buffer 0										0000 00uu uuuu uuuu
ADCBUF1	0282	—	—	—	—	—	—	ADC Data Buffer 1										0000 00uu uuuu uuuu
ADCBUF2	0284	—	—	—	—	—	—	ADC Data Buffer 2										0000 00uu uuuu uuuu
ADCBUF3	0286	—	—	—	—	—	—	ADC Data Buffer 3										0000 00uu uuuu uuuu
ADCBUF4	0288	—	—	—	—	—	—	ADC Data Buffer 4										0000 00uu uuuu uuuu
ADCBUF5	028A	—	—	—	—	—	—	ADC Data Buffer 5										0000 00uu uuuu uuuu
ADCBUF6	028C	—	—	—	—	—	—	ADC Data Buffer 6										0000 00uu uuuu uuuu
ADCBUF7	028E	—	—	—	—	—	—	ADC Data Buffer 7										0000 00uu uuuu uuuu
ADCBUF8	0290	—	—	—	—	—	—	ADC Data Buffer 8										0000 00uu uuuu uuuu
ADCBUF9	0292	—	—	—	—	—	—	ADC Data Buffer 9										0000 00uu uuuu uuuu
ADCBUFA	0294	—	—	—	—	—	—	ADC Data Buffer 10										0000 00uu uuuu uuuu
ADCBUFB	0296	—	—	—	—	—	—	ADC Data Buffer 11										0000 00uu uuuu uuuu
ADCBUFC	0298	—	—	—	—	—	—	ADC Data Buffer 12										0000 00uu uuuu uuuu
ADCBUFD	029A	—	—	—	—	—	—	ADC Data Buffer 13										0000 00uu uuuu uuuu
ADCBUFE	029C	—	—	—	—	—	—	ADC Data Buffer 14										0000 00uu uuuu uuuu
ADCBUFF	029E	—	—	—	—	—	—	ADC Data Buffer 15										0000 00uu uuuu uuuu
ADCON1	02A0	ADON	—	ADSIDL	—	—	—	FORM<1:0>		SSRC<2:0>			—	SIMSAM	ASAM	SAMP	DONE	0000 0000 0000 0000
ADCON2	02A2	VCFG<2:0>			—	—	CSCNA	CHPS<1:0>		BUFS	—	SMPI<3:0>				BUFM	ALTS	0000 0000 0000 0000
ADCON3	02A4	—	—	—	SAMC<4:0>					ADRC	—	ADCS<5:0>						0000 0000 0000 0000
ADCHS	02A6	CH123NB<1:0>		CH123SB	CH0NB	CH0SB<3:0>				CH123NA<1:0>		CH123SA	CH0NA	CH0SA<3:0>				0000 0000 0000 0000
ADPCFG	02A8	—	—	—	—	—	—	—	PCFG8*	PCFG7*	PCFG6*	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000
ADCSSL	02AA	—	—	—	—	—	—	—	CSSL8*	CSSL7*	CSSL6*	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0	0000 0000 0000 0000

Legend: u = uninitialized bit
* Not available on dsPIC30F4012.
Note: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

7.11. Módulo SPI

7.11.1. Introducción

La familia de procesadores digitales de señal dsPIC pueden implementar uno o varios módulos SPI (*Serial Peripheral Interface*). Se enumerarán como SPI1, etc. Nos centraremos en el SPI1.

Registros

Cada módulo SPI consiste en un registro de desplazamiento de 16 bits (`SPI1SR`), empleado para introducir y sacar los datos serie, un búfer (**`SPI1BUF`**), un registro de control (**`SPI1CON`**) que configura el módulo y un registro de estado (**`SPI1STAT`**).

Patillas

El interfaz consiste en 4 pines: **`SDI1`** (serial data input), **`SDO1`** (serial data output), **`SCK1`** (reloj) y **`/SS1`** (selección de esclavo activo en bajo).

Funcionamiento

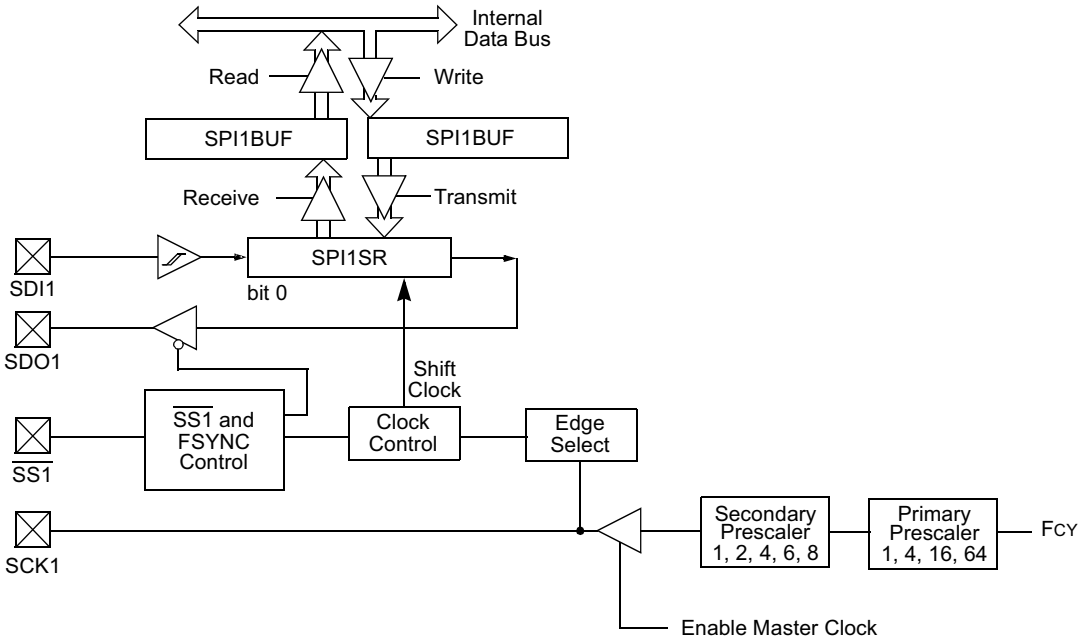
En modo maestro **`SCK1`** es una señal de reloj saliente. En modo esclavo es una señal de entrada.

Series de 8 o 16 pulsos de reloj desplazan los bits desde el registro de desplazamiento hasta el pin **`SDO1`** y los toma del pin **`SDI1`**.

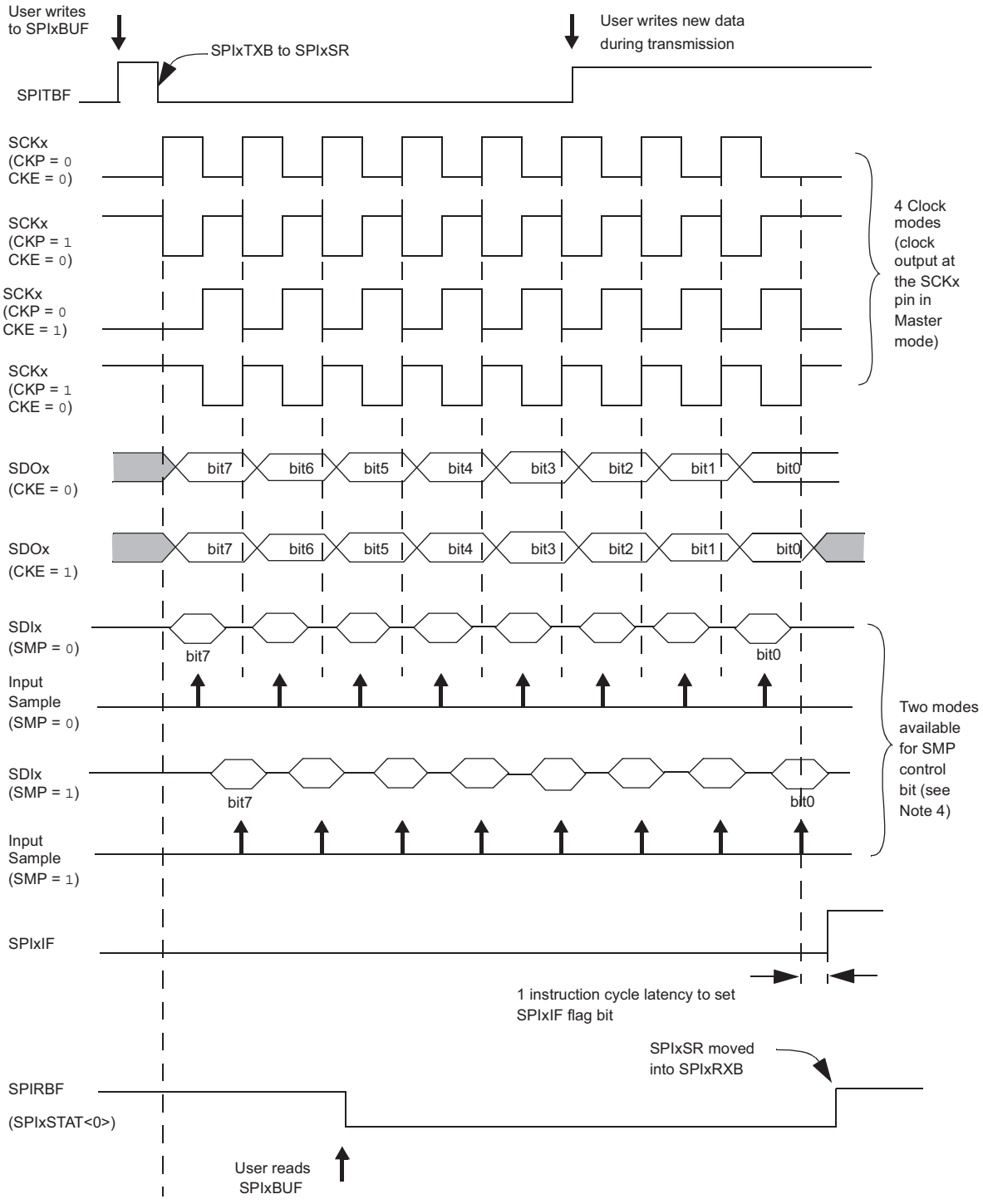
Cuando se completa una transferencia se notifica mediante el bit **`SPI1IF`**, o **`SPI2IF`** y si las interrupciones correspondientes están habilitadas (**`SPI1IE`**, **`SPI2IE`**) se salta a una rutina de interrupciones.

La operación de recepción tiene un búfer doble. Si el búfer está lleno se notifica (**`SPI1CON`**.`SPIROV`). Necesitaremos borrar ese bit para continuar recibiendo datos.

La transmisión también dispone de un doble búfer. Se copian los datos de **SPI1BUF** a SPI1SR una vez que se ha completado la transferencia anterior.



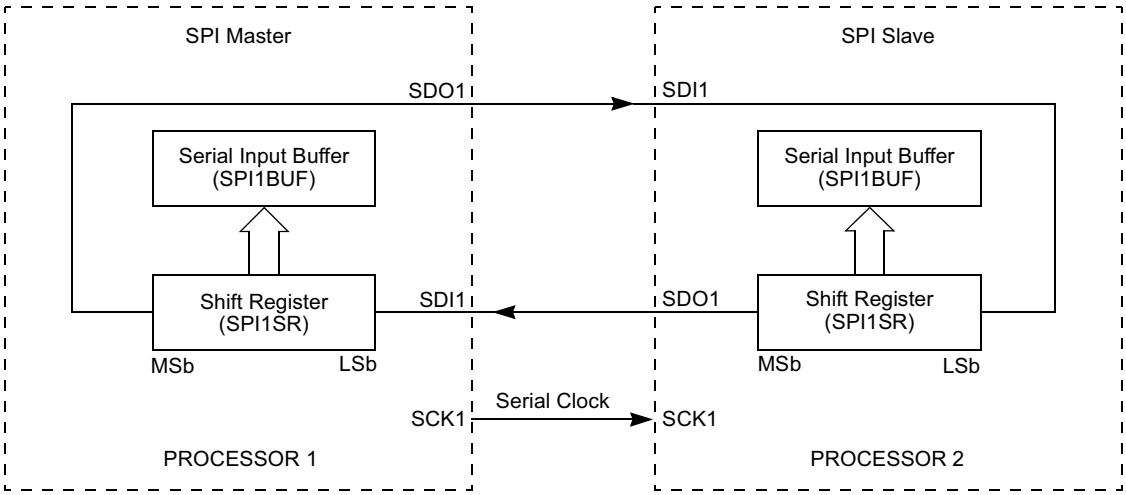
En modo maestro el reloj se genera subdividiendo la frecuencia del reloj del sistema. Los datos se transmiten tan pronto se escribe en **SPI1BUF**. La interrupción se genera en el punto medio de la transferencia del último bit.



Note 1: Four SPI Clock modes shown to demonstrate CKP (SPIxCON<6>) and CKE (SPIxCON<8>) bit functionality only. Only one of the four modes can be chosen for operation.
2: SDI and input sample shown for two different values of the SMP (SPIxCON<9>) bit, for demonstration purposes only. Only one of the two configurations of the SMP bit can be chosen during operation.
3: If there are no pending transmissions, SPIxTXB is transferred to SPIxSR as soon as the user writes to SPIxBUF.
4: Operation for 8-bit mode shown. The 16-bit mode is similar.

En modo esclavo los datos se transmiten/reciben en cuanto aparece un pulso en la señal de reloj. Si el control de la patilla **/SS1** está activado, solo se activará la transmisión/recepción cuando haya un valor lógico cero en esa patilla. El reloj proporcionado al módulo

SPI es $F_{OSC}/4$. Se dispone de dos preescalas de esta señal: la primaria (**SPI1CON**.PPRE) y la secundaria (**SPI1CON**.SPRE). El bit **SPI1CON**.CKE determina si la transferencia ocurre cuando hay una transición del estado activo del reloj al estado ocioso o si es al revés. El bit **SPI1CON**.CKP selecciona el estado ocioso de la señal de reloj.

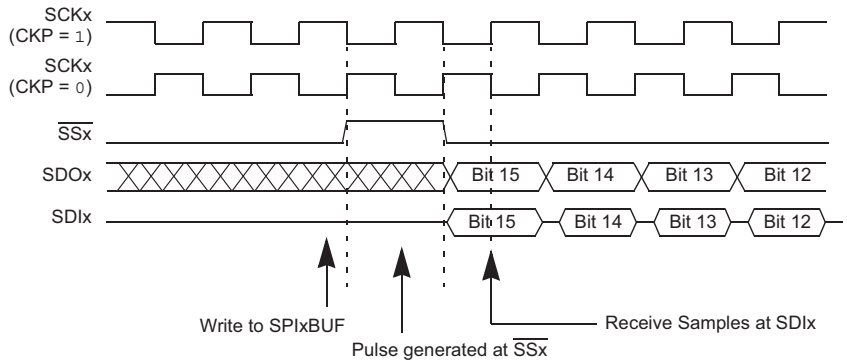


El bit de control **SPI1CON**.MODE16 permite al módulo la transferencia de datos de 16 bits en vez de datos de 8 bits. Se debe desactivar el módulo antes de cambiar ese bit.

El bit de control **SPI1CON**.DISSDO permite desactivar la patilla **SDO1**. Esto es útil en configuraciones de solo entrada para aprovechar el pin como de propósito general.

Soporte de SPI enmarcado.

El módulo SPI soporta un protocolo SPI enmarcado (*framed*) básico tanto en modo maestro como en modo esclavo. El bit **SPI1CON**.FRMEN habilita este modo haciendo que el pin **/SS1** realice un pulso de sincronización de trama (FSYNC). EL bit **SPI1CON**.SPIFSD determinará si el pin **/SS1** actuará como entrada o salida. El pulso de enmarcado es un pulso activo en alta durante un único ciclo de reloj de la señal **SCK1**.



7.11.2. Registros

SPIxCON: SPI*x* Control Register

Upper Byte:							
U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSEN	CKP	MSTEN	SPRE<2:0>			PPRE<1:0>	
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14 **FRMEN:** Framed SPI Support bit
1 = Framed SPI support enabled
0 = Framed SPI support disabled

bit 13 **SPIFSD:** Frame Sync Pulse Direction Control on $\overline{\text{SS}}$x pin bit
1 = Frame sync pulse input (slave)
0 = Frame sync pulse output (master)

bit 12 **Unimplemented:** Read as ‘0’

bit 11 **DISSDO:** Disable SDOx pin bit
1 = SDOx pin is not used by module. Pin is controlled by associated port register.
0 = SDOx pin is controlled by the module

bit 10 **MODE16:** Word/Byte Communication Select bit
1 = Communication is word-wide (16 bits)
0 = Communication is byte-wide (8 bits)

bit 9 **SMP:** SPI Data Input Sample Phase bit
Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
Slave mode:
SMP must be cleared when SPI is used in Slave mode.

bit 8 **CKE:** SPI Clock Edge Select bit
1 = Serial output data changes on transition from active clock state to Idle clock state (see bit 6)
0 = Serial output data changes on transition from Idle clock state to active clock state (see bit 6)

Note: The CKE bit is not used in the Framed SPI modes. The user should program this bit to ‘0’ for the Framed SPI modes (FRMEN = 1).

bit 7 **SSEN:** Slave Select Enable (Slave mode) bit
1 = $\overline{\text{SS}}$ pin used for Slave mode
0 = $\overline{\text{SS}}$ pin not used by module. Pin controlled by port function.

bit 6 **CKP:** Clock Polarity Select bit
1 = Idle state for clock is a high level; active state is a low level
0 = Idle state for clock is a low level; active state is a high level

bit 5 **MSTEN:** Master Mode Enable bit
1 = Master mode
0 = Slave mode

SPIxCON: SPIx Control Register (Continued)

bit 4-2 **SPRE<2:0>:** Secondary Prescale (Master Mode) bits
(Supported settings: 1:1, 2:1 through 8:1, all inclusive)
111 = Secondary prescale 1:1
110 = Secondary prescale 2:1
...
000 = Secondary prescale 8:1

bit 1-0 **PPRE<1:0>:** Primary Prescale (Master Mode) bits
11 = Primary prescale 1:1
10 = Primary prescale 4:1
01 = Primary prescale 16:1
00 = Primary prescale 64:1

SPIxSTAT: SPI Status and Control Register

Upper Byte:

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
SPIEN	—	SPISIDL	—	—	—	—	—
bit 15							bit 8

Lower Byte:

U-0	R/W-0 HS	U-0	U-0	U-0	U-0	R-0	R-0
—	SPIROV	—	—	—	—	SPITBF	SPIRBF
bit 7							bit 0

bit 15 **SPIEN:** SPI Enable bit
1 = Enables module and configures SCKx, SDOx, SDIx and SSx as serial port pins
0 = Disables module

bit 14 **Unimplemented:** Read as '0'

bit 13 **SPISIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12-7 **Unimplemented:** Read as '0'

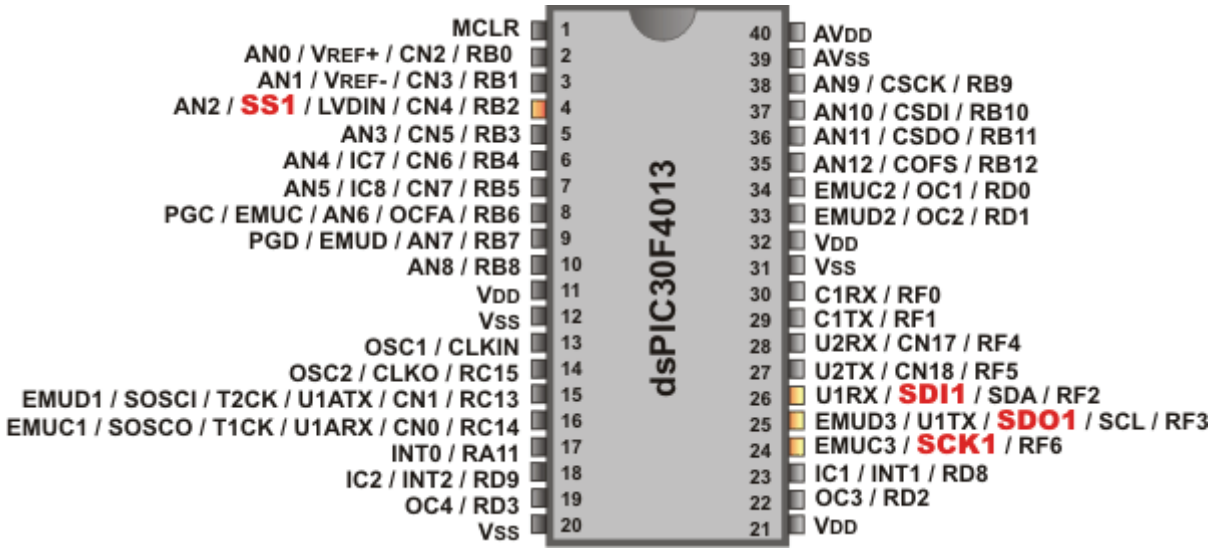
bit 6 **SPIROV:** Receive Overflow Flag bit
1 = A new byte/word is completely received and discarded. The user software has not read the previous data in the SPIxBUF register.
0 = No overflow has occurred

bit 5-2 **Unimplemented:** Read as '0'

bit 1 **SPITBF:** SPI Transmit Buffer Full Status bit
1 = Transmit not yet started, SPIxTXB is full
0 = Transmit started, SPIxTXB is empty
Automatically set in hardware when CPU writes SPIxBUF location, loading SPIxTXB.
Automatically cleared in hardware when SPIx module transfers data from SPIxTXB to SPIxSR.

bit 0 **SPIRBF:** SPI Receive Buffer Full Status bit
1 = Receive complete, SPIxRXB is full
0 = Receive is not complete, SPIxRXB is empty
Automatically set in hardware when SPIx transfers data from SPIxSR to SPIxRXB.
Automatically cleared in hardware when core reads SPIxBUF location, reading SPIxRXB.

Esquema de las líneas del módulo SPI del dsPIC30F4013



Resumen de registros

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
SPI1STAT	0220	SPIEN	—	SPISIDL	—	—	—	—	—	—	SPIROV	—	—	—	—	SPITBF	SPIRBF	0000 0000 0000 0000
SPI1CON	0222	—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	0000 0000 0000 0000
SPI1BUF	0224	Transmit and Receive Buffer																0000 0000 0000 0000

Legend: u = uninitialized bit

Note: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

7.12. Módulo UART

7.12.1. Introducción

Según el modelo de dsPIC nos podemos encontrar con uno o más de estos módulos que permiten una comunicación serie asíncrona bidireccional y simultánea (para interfaces del tipo RS-232 o RS-485).

Características

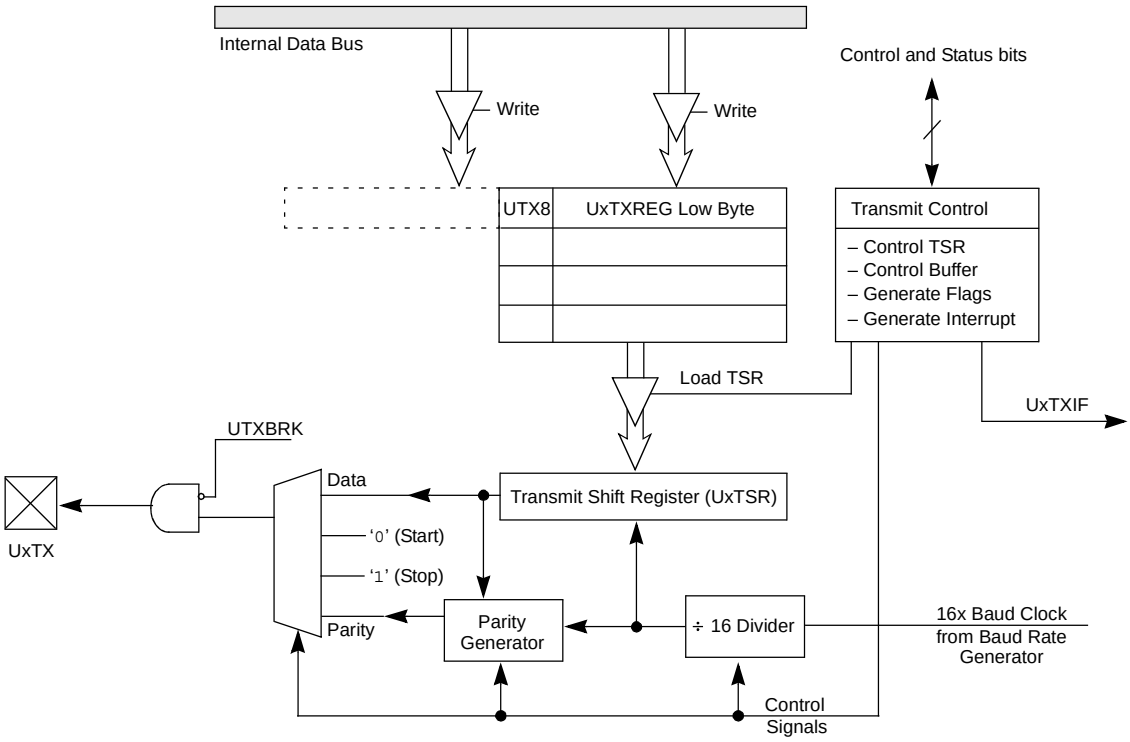
Las características de estos módulos son:

- Comunicación con 8 o 9 bits de tipo *full-duplex*.
- En modo 8 bits tiene la opción de usar paridad par o impar o no usar paridad.
- Permite seleccionar entre uno o dos *bits* de *stop*.
- Incluye un generador de baudios (BRG) con una preescala de 16 bits.
- Tiene un búfer de transmisión de 4 niveles y un búfer de recepción de 4 niveles.
- Detecta errores de paridad, de marcaje de datos y de sobreescritura en los búferes.
- Permite interrupciones al detectar una dirección (noveno bit a uno).

- Dispone de interrupciones separadas para la transmisión y la recepción.
- Dispone de un modo *loopback* para diagnóstico.
- Permite escoger los pines TX/RX entre dos opciones.

Módulo de transmisión

El módulo de transmisión se representa en la figura siguiente:



1. Primero hay que configurar el módulo mediante los registros **U1MODE**[11] y **U1STA**. El pin **U1TX** deberá ser configurado como salida y el pin **U1RX** como entrada.

   Si el campo **U1MODE**.ALTIO=1 se emplearían los pines **U1ATX** y **U1ARX**.

2. También hay que definir la velocidad de transmisión mediante el registro **U1BRG**.

   Para la generación del ritmo de transmisión se pone **U1BRG**= $\frac{F_{CY}}{Baudios*16} - 1$, donde F_{CY} es la frecuencia del ciclo de instrucción.

   Debemos recordar que el error máximo de velocidades entre el emisor y el receptor no debe superar el 4 %.

3. Hay que activar la UART:

Listado 7.7:

```
U1MODE.UARTEN=1     ; Habilita la UART
U1STA.UTXEN = 1     ; Habilita la transmision
```

[11] OJO. El nombre de los campos de los registros en el compilador de C de Hitech difieren de los aquí expresados. Ver apéndice "Definiciones para dsPICC ..."

La desactivación de la UART vaciaría los búferes asociados. Además colocaría en su valor por defecto todos los bits de notificación asociados a la UART. Además abortaría cualquier transmisión o recepción en curso. El resto de la configuración no se vería afectada.

4. Se selecciona la longitud de los datos y el uso o no de paridad mediante **U1MODE**.PDSEL y el número de bits de parada con **U1MODE**.STSEL. Por defecto está en 8 bits, sin paridad y con un bit de parada.

5. Si se necesitan las interrupciones hay que configurarlas.

6. Los datos a transmitir se escriben en el registro **U1TXREG**. Si el dato es de 8 bits se transmitirán 8 bits y si el dato es de 16 bits se transmitirán los 9 bits de menor peso. Existe un búfer FIFO de 4 niveles.

7. En cuanto el registro U1TSR se encuentre vacío se copiará el primer dato del búfer FIFO hacia él y comenzará la transmisión al ritmo indicado por el generador de baudios y su divisor de frecuencia. Automáticamente se insertará la paridad si así está configurado y siempre que no se quieran transmitir 9 bits.

8. Una vez finalizada la transmisión se procederá de nuevo con otro dato del búfer (el campo **U1STA**.UTXBF indicará si el búfer está lleno). En caso de encontrarse éste vacío, se activará el bit U1TIF para indicar que el búfer está vacío.

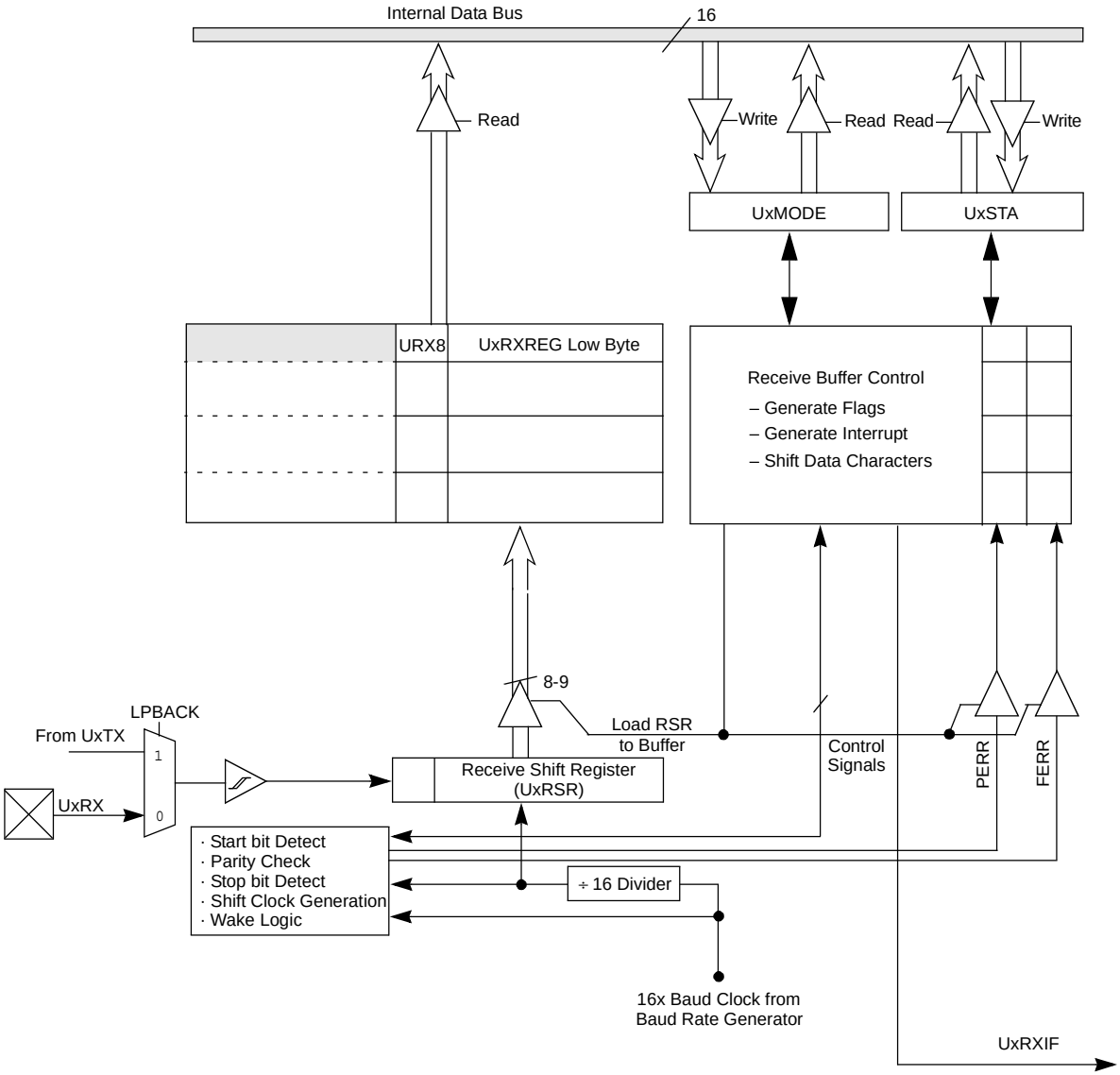
   Si **U1STA**.UTXISEL=0 se generará una interrupción cuando se transmita un dato del búfer a U1TSR. El búfer contendrá siempre como mínimo un dato.

   Si vale 1, se generará la interrupción al transferir un dato y que el búfer esté vacío.

9. Un caso especial de transmisión es el carácter Break empleado en algunos protocolos de comunicaciones basados en UART (por ejemplo el protocolo LIN). Consiste en poner un cero lógico durante al menos 13 tiempos de bit (dato no valido). Esta condición se consigue haciendo **U1STA**.UTXBRK= 1. Inmediatamente después de la generación de esta condición el flag citado se pone a cero. Esta condición no genera interrupciones.

Módulo de recepción

El módulo de recepción se representa en la figura siguiente:



La secuencia a llevar a cabo para recibir datos es la siguiente:

1. Se configura la UART.
2. Se habilita la UART.
3. Se generará una interrupción de recepción cuando se reciban uno o más datos según lo especificado en el campo **U1STA**.URXISEL
4. Comprobaremos el campo **U1STA**.OERR para verificar que no ha ocurrido una sobre-escritura de ningún dato del búfer.
5. Se lee del búfer de recepción (**U1RXREG**) el dato. Esto actualiza los campos PERR y FERR con respecto al nuevo dato del búfer disponible.
6. El campo **U1STA**.URXDA indicará si hay un dato disponible en el búfer.
7. La generación de interrupciones dependerá del campo **U1STA**.URXISEL:

00,01 Se generará una interrupción cada vez que se transfiera un dato al búfer.

10 Se generará una interrupción siempre que al transferir un dato al búfer este contenga 3 datos.

11 Se generará una interrupción cuando al transferir un dato al búfer, este contenga 4 datos.

8. La comprobación de errores es importante al recibir los datos:

OERR Se activa cuando el búfer está lleno y se trata de transferir un dato al mismo. Para continuar con el funcionamiento hay que ponerlo a cero.

FERR Error de marcaje. Se genera cuando se recibe un '1' en vez del bit de parada ('0').

PERR Error de paridad. La paridad del dato recibido no es correcta.

9. Mientras que la UART está ocupada recibiendo un dato, se pone el campo `U1STA.RIDLE=0`.

10. Cuando se recibe una condición de Break lo detectaremos comprobando que el dato recibido tiene todos sus bits a cero y hay un error de marcaje.

7.12.2. Registros

Los registros empleados son los siguientes:

UxMODE: UARTx Mode Register

Upper Byte:							
R/W-0	U-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0
UARTEN	—	USIDL	—	reserved	ALTIO	reserved	reserved
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
WAKE	LPBACK	ABAUD	—	—	PDSEL<1:0>		STSEL
bit 7							bit 0

bit 15 **UARTEN:** UART Enable bit
1 = UART is enabled. UART pins are controlled by UART as defined by UEN<1:0> and UTXEN control bits.
0 = UART is disabled. UART pins are controlled by corresponding PORT, LAT, and TRIS bits.

bit 14 **Unimplemented:** Read as '0'

bit 13 **USIDL:** Stop in Idle Mode bit
1 = Discontinue operation when device enters Idle mode
0 = Continue operation in Idle mode

bit 12 **Unimplemented:** Read as '0'

bit 11 **Reserved:** Write '0' to this location

bit 10 **ALTIO:** UART Alternate I/O Selection bit
1 = UART communicates using UxATX and UxARX I/O pins
0 = UART communicates using UxTX and UxRX I/O pins

Note: The alternate UART I/O pins are not available on all devices. See device data sheet for details.

bit 9-8 **Reserved:** Write '0' to these locations

bit 7 **WAKE:** Enable Wake-up on Start bit Detect During Sleep Mode bit
1 = Wake-up enabled
0 = Wake-up disabled

bit 6 **LPBACK:** UART Loopback Mode Select bit
1 = Enable Loopback mode
0 = Loopback mode is disabled

bit 5 **ABAUD:** Auto Baud Enable bit
1 = Input to Capture module from UxRX pin
0 = Input to Capture module from ICx pin

bit 4-3 **Unimplemented:** Read as '0'

bit 2-1 **PDSEL<1:0>:** Parity and Data Selection bits
11 = 9-bit data, no parity
10 = 8-bit data, odd parity
01 = 8-bit data, even parity
00 = 8-bit data, no parity

bit 0 **STSEL:** Stop Selection bit
1 = 2 Stop bits
0 = 1 Stop bit

UxSTA: UARTx Status and Control Register

Upper Byte:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-1
UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	URXDA
bit 7							bit 0

bit 15 **UTXISEL:** Transmission Interrupt Mode Selection bit
1 = Interrupt when a character is transferred to the Transmit Shift register and as result, the transmit buffer becomes empty
0 = Interrupt when a character is transferred to the Transmit Shift register (this implies that there is at least one character open in the transmit buffer)

bit 14-12 **Unimplemented:** Read as '0'

bit 11 **UTXBRK:** Transmit Break bit
1 = UxTX pin is driven low, regardless of transmitter state
0 = UxTX pin operates normally

bit 10 **UTXEN:** Transmit Enable bit
1 = UART transmitter enabled, UxTX pin controlled by UART (if UARTEN = 1)
0 = UART transmitter disabled, any pending transmission is aborted and buffer is reset. UxTX pin controlled by PORT.

bit 9 **UTXBF:** Transmit Buffer Full Status bit (Read Only)
1 = Transmit buffer is full
0 = Transmit buffer is not full, at least one more data word can be written

bit 8 **TRMT:** Transmit Shift Register is Empty bit (Read Only)
1 = Transmit shift register is empty and transmit buffer is empty (the last transmission has completed)
0 = Transmit shift register is not empty, a transmission is in progress or queued in the transmit buffer

bit 7-6 **URXISEL<1:0>:** Receive Interrupt Mode Selection bit
11 = Interrupt flag bit is set when Receive Buffer is full (i.e., has 4 data characters)
10 = Interrupt flag bit is set when Receive Buffer is 3/4 full (i.e., has 3 data characters)
0x = Interrupt flag bit is set when a character is received

bit 5 **ADDEN:** Address Character Detect (bit 8 of received data = 1)
1 = Address Detect mode enabled. If 9-bit mode is not selected, this control bit has no effect.
0 = Address Detect mode disabled

bit 4 **RIDLE:** Receiver Idle bit (Read Only)
1 = Receiver is Idle
0 = Data is being received

bit 3 **PERR:** Parity Error Status bit (Read Only)
1 = Parity error has been detected for the current character
0 = Parity error has not been detected

bit 2 **FERR:** Framing Error Status bit (Read Only)
1 = Framing Error has been detected for the current character
0 = Framing Error has not been detected

UxSTA: UARTx Status and Control Register (Continued)

bit 1 **OERR:** Receive Buffer Overrun Error Status bit (Read/Clear Only)
1 = Receive buffer has overflowed
0 = Receive buffer has not overflowed

bit 0 **URXDA:** Receive Buffer Data Available bit (Read Only)
1 = Receive buffer has data, at least one more character can be read
0 = Receive buffer is empty

UxBRG: UARTx Baud Rate Register

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRG<15:8>							
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRG<7:0>							
bit 7							bit 0

bit 15-0 **BRG<15:0>:** Baud Rate Divisor bits

UxTXREG: UARTx Transmit Register (Write Only)

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	W-x
—	—	—	—	—	—	—	UTX8
bit 15							bit 8

Lower Byte:							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
UTX<7:0>							
bit 7							bit 0

bit 15-9 **Unimplemented:** Read as '0'

bit 8 **UTX8:** Data bit 8 of the Character to be Transmitted (in 9-bit mode)

bit 7-0 **UTX<7:0>:** Data bits 7-0 of the Character to be Transmitted

UxRXREG: UARTx Receive Register

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	R-0
—	—	—	—	—	—	—	URX8
bit 15							bit 8

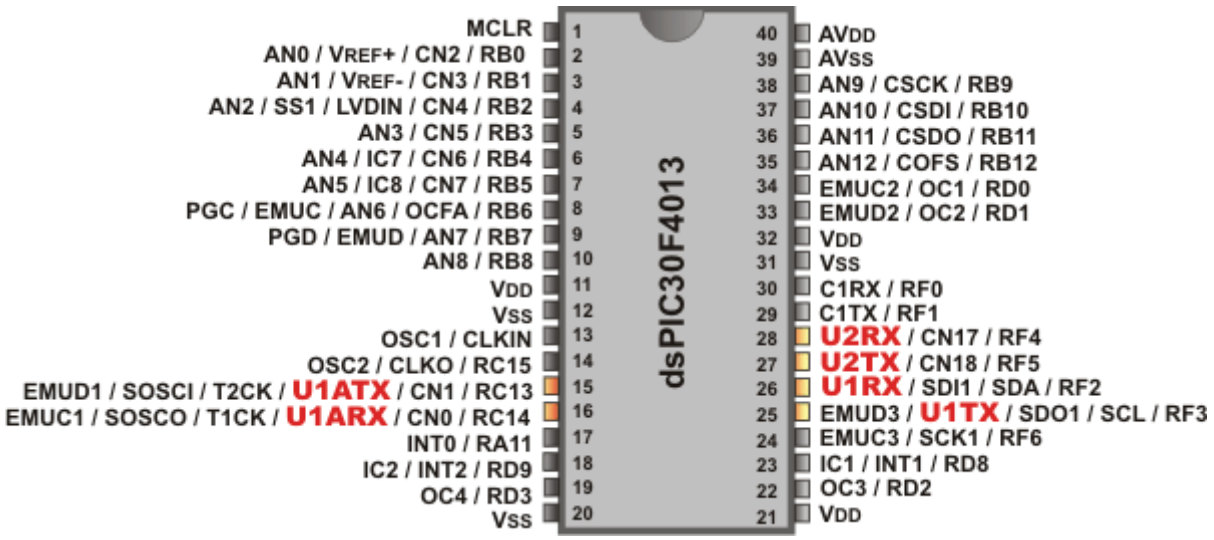
Lower Byte:							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
URX<7:0>							
bit 7							bit 0

bit 15-9 **Unimplemented:** Read as '0'

bit 8 **URX8:** Data bit 8 of the Received Character (in 9-bit mode)

bit 7-0 **URX<7:0>:** Data bits 7-0 of the Received Character

Esquema de los pines de la UART del dsPIC30F4013



7.12.3. Resumen de registros

Vemos el resumen de registros:

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
U1MODE	020C	UARTEN	—	USIDL	—	—	ALTIO	—	—	WAKE	LPBACK	ABAUD	—	—	PDSEL1	PDSEL0	STSEL	0000 0000 0000 0000
U1STA	020E	UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0000 0001 0001 0000
U1TXREG	0210	—	—	—	—	—	—	—	UTX8	Transmit Register								0000 000u uuuu uuuu
U1RXREG	0212	—	—	—	—	—	—	—	URX8	Receive Register								0000 0000 0000 0000
U1BRG	0214	Baud Rate Generator Prescaler																0000 0000 0000 0000

Legend: u = uninitialized bit

Note: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

7.13. Integración del sistema

7.13.1. Sistema de osciladores

Proporciona la señal de reloj a la CPU y a todos los dispositivos periféricos integrados.

El sistema oscilador dispone de las siguientes funcionalidades:

- Varias opciones de oscilador internas y externas.
- Un circuito PLL para elevar las frecuencias de funcionamiento interno.
- Un mecanismo para conmutar entre las fuentes de reloj.
- Un postscaler programable para el ahorro energético.
- Un sistema de monitorización de la fuente primaria de reloj segura (FSCM).
- Un registro de control de opciones del reloj (`OSCCON`).
- Bits de configuración no volátiles para la selección del oscilador principal.

Modos de operación

El sistema de osciladores permitirá configurar su funcionamiento empleando cualquiera de los sistemas siguientes:

Oscilador primario. Permite tres modos de funcionamiento:

- XTL: Con cristal de cuarzo o resonador cerámico para un rango de frecuencias comprendido entre 200KHz y 4MHz.
- XT: Cristal o resonador entre 4MHz y 10MHz.
- HS: Para un cristal de cuarzo entre 10MHz y 25MHz.

Todos los osciladores primarios utilizan los pines **`OSC1`** y **`OSC2`**.

Oscilador secundario. El oscilador secundario LP (*Low Power*) está diseñado para consumir poca potencia y necesita un cristal o resonador cerámico de 32KHz, siendo **`SOSC1`** y **`SOSC2`** las patillas que se utilizan.

Este oscilador también puede controlar el temporizador Timer1 para implementar un reloj (RTC).

Osciladores internos. Existen dos osciladores internos.

- El FRC *(Fast RC)* trabaja a 7.37 MHz. Está diseñado para trabajar a frecuencias altas sin necesidad de conectar un cristal externo.
- El segundo oscilador interno LPRC *(Low power RC)* está conectado al perro guardián y trabaja a 512 kHz. Hace de fuente de reloj para el temporizador PWRT, perro guardián, y los circuitos de monitorización del reloj. También es utilizado como fuente de reloj en aplicaciones que no requieren una elevada frecuencia pero tienen un consumo de potencia crítico.

La frecuencia de oscilación depende de la temperatura y del voltaje al que trabaje el dispositivo.

Oscilador externo. El oscilador externo para una red RC (EXTRC) trabaja a frecuencias que llegan a los 4 MHz. Utiliza una resistencia y un condensador externos conectados al pin **OSC1**, el cual también puede conectarse a una señal de reloj externa (modo EC).

TABLE 20-1: OSCILLATOR OPERATING MODES

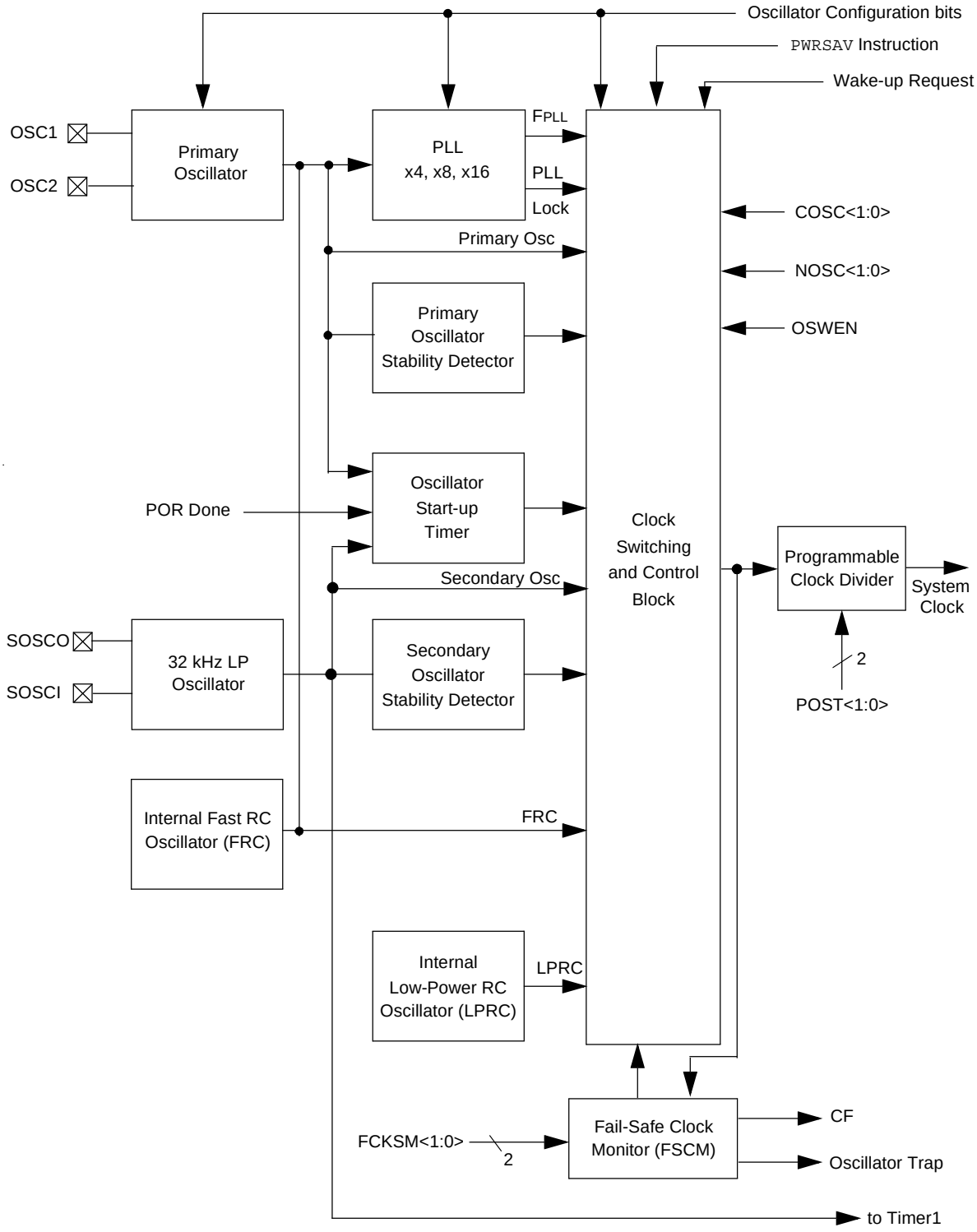
Oscillator Mode	Description
XTL	200 kHz-4 MHz crystal on OSC1:OSC2
XT	4 MHz-10 MHz crystal on OSC1:OSC2
XT w/PLL 4x	4 MHz-10 MHz crystal on OSC1:OSC2, 4x PLL enabled
XT w/PLL 8x	4 MHz-10 MHz crystal on OSC1:OSC2, 8x PLL enabled
XT w/PLL 16x	4 MHz-10 MHz crystal on OSC1:OSC2, 16x PLL enabled(1)
LP	32 kHz crystal on SOSCO:SOSCI(2)
HS	10 MHz-25 MHz crystal
HS/2 w/PLL 4x	10 MHz-25 MHz crystal, divide by 2, 4x PLL enabled
HS/2 w/PLL 8x	10 MHz-25 MHz crystal, divide by 2, 8x PLL enabled
HS/2 w/PLL 16x	10 MHz-25 MHz crystal, divide by 2, 16x PLL enabled
HS/3 w/PLL 4x	10 MHz-25 MHz crystal, divide by 3, 4x PLL enabled
HS/3 w/PLL 8x	10 MHz-25 MHz crystal, divide by 3, 8x PLL enabled
HS/3 w/PLL 16x	10 MHz-25 MHz crystal, divide by 3, 16x PLL enabled
EC	External clock input (0-40 MHz)
ECIO	External clock input (0-40 MHz), OSC2 pin is I/O
EC w/PLL 4x	External clock input (0-40 MHz), OSC2 pin is I/O, 4x PLL enabled(1)
EC w/PLL 8x	External clock input (0-40 MHz), OSC2 pin is I/O, 8x PLL enabled(1)
EC w/PLL 16x	External clock input (0-40 MHz), OSC2 pin is I/O, 16x PLL enabled(1)
ERC	External RC oscillator, OSC2 pin is FOSC/4 output(3)
ERCIO	External RC oscillator, OSC2 pin is I/O(3)
FRC	7.37 MHz internal RC oscillator
FRC w/PLL 4x	7.37 MHz Internal RC oscillator, 4x PLL enabled
FRC w/PLL 8x	7.37 MHz Internal RC oscillator, 8x PLL enabled
FRC w/PLL 16x	7.37 MHz Internal RC oscillator, 16x PLL enabled
LPRC	512 kHz internal RC oscillator

Note 1: dsPIC30F maximum operating frequency of 120 MHz must be met.

2: LP oscillator can be conveniently shared as system clock, as well as real-time clock for Timer1.

3: Requires external R and C. Frequency operation up to 4 MHz.

Diagrama de bloques del sistema de osciladores



Configuraciones del oscilador

Configuración inicial. Cuando el dsPIC se inicia desde un POR o un BOR, la selección se basa en los bits de configuración FOSC.

Temporizador de arranque del oscilador (OST). A continuación este sistema cuenta 1024 ciclos para asegurarse el correcto arranque del oscilador seleccionado. Se aplica en los modos LP, XT, XTL y HS para el oscilador primario.

Control del oscilador LP. La habilitación del oscilador LP se lleva a cabo mediante `OSCCON.NOSC2..0` y `OSCCON.LPOSCEN`. Este oscilador funcionará incluso en modo *sleep*. Está pensado para poner un cristal de 32768 Hz y llevar un reloj en tiempo real (RTC).

Lazo de enganche de fase (PLL). Permite multiplicar la frecuencia generada por el oscilador primario por 4, 8 ó 16. Se puede comprobar si el PLL está enganchado en el bit `OSCCON.LOCK`

Oscilador Fast RC. Se trata de un circuito oscilador autónomo basado en una red resistencia-condensador rápido. Oscila a 7.37 MHz ±2 %. Se selecciona mediante el registro `OSCCON` y mediante los bits de configuración FOSC. Dispone de un campo: `OSCTUN.TUN3..0` que permite sintonizar (calibrar) el FRC permitiendo variaciones de entre +10.5 % y -12 % en pasos de 1.5 %

TUN<3:0> Bits	FRC Frequency
0111	+10.5%
0110	+9.0%
0101	+7.5%
0100	+6.0%
0011	+4.5%
0010	+3.0%
0001	+1.5%
0000	Center Frequency (oscillator is running at calibrated frequency)
1111	-1.5%
1110	-3.0%
1101	-4.5%
1100	-6.0%
1011	-7.5%
1010	-9.0%
1001	-10.5%
1000	-12.0%

Oscilador RC de bajo consumo (LPRC). Este oscilador alimenta al perro guardián y oscila a 512 kHz. También es señal de reloj para el sistema de temporización de arranque (PWRTE) y el monitor de fallo del reloj.

Monitor de fallo del oscilador primario (FSCM). Permite monitorizar el funcionamiento correcto del oscilador primario. Si este falló entonces puede entrar a generar la señal de reloj del sistema un oscilador interno secundario. La política de conmutación de reloj se elige con el registro `OSCCON`.

Registros

REGISTER 20-1: OSCCON: OSCILLATOR CONTROL REGISTER

U-0	R-y	R-y	R-y	U-0	R/W-y	R/W-y	R/W-y
—	COSC<2:0>			—	NOSC<2:0>		
bit 15							bit 8

R/W-0	R/W-0	R-0	U-0	R/W-0	U-0	R/W-0	R/W-0
POST<1:0>		LOCK	—	CF	—	LPOSCEN	OSWEN
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **Unimplemented:** Read as '0'

bit 14-12 **COSC<2:0>:** Current Oscillator Group Selection bits (Read-Only)
111 = PLL Oscillator; PLL source selected by FPR<4:0> bits
011 = External Oscillator; OSC1/OSC2 pins; External Oscillator configuration selected by FPR<4:0> bits
010 = LPRC internal low-power RC
001 = FRC internal fast RC
000 = LP crystal oscillator; SOSCI/SOSCO pins
Set to FOS<2:0> values on POR or BOR
Loaded with NOSC<2:0> at the completion of a successful clock switch
Set to FRC value when FSCM detects a failure and switches clock to FRC

bit 11 **Unimplemented:** Read as '0'

bit 10-8 **NOSC<2:0>:** New Oscillator Group Selection bits
111 = PLL Oscillator; PLL source selected by FPR<4:0> bits
011 = External Oscillator; OSC1/OSC2 pins; External Oscillator configuration selected by FPR<4:0> bits
010 = LPRC internal low-power RC
001 = FRC internal fast RC
000 = LP crystal oscillator; SOSCI/SOSCO pins
Set to FOS<2:0> values on POR or BOR

bit 7-6 **POST<1:0>:** Oscillator Postscaler Selection bits
11 = Oscillator postscaler divides clock by 64
10 = Oscillator postscaler divides clock by 16
01 = Oscillator postscaler divides clock by 4
00 = Oscillator postscaler does not alter clock

bit 5 **LOCK:** PLL Lock Status bit (Read-Only)
1 = Indicates that PLL is in lock
0 = Indicates that PLL is out of lock (or disabled)
Reset on POR or BOR
Reset when a valid clock switching sequence is initiated
Set when PLL lock is achieved after a PLL start
Reset when lock is lost
Read zero when PLL is not selected as a system clock

REGISTER 20-1: OSCCON: OSCILLATOR CONTROL REGISTER (CONTINUED)

bit 4 **Unimplemented:** Read as '0'

bit 3 **CF:** Clock Fail Detect bit (Read/Clearable by application)
1 = FSCM has detected clock failure
0 = FSCM has NOT detected clock failure
Reset on POR or BOR
Reset when a valid clock switching sequence is initiated
Set when clock fail detected

bit 2 **Unimplemented:** Read as '0'

bit 1 **LPOSCEN:** 32 KHz Secondary (LP) Oscillator Enable bit
1 = Secondary oscillator is enabled
0 = Secondary oscillator is disabled
Reset on POR or BOR

bit 0 **OSWEN:** Oscillator Switch Enable bit
1 = Request oscillator switch to selection specified by NOSC<2:0> bits
0 = Oscillator switch is complete
Reset on POR or BOR
Reset after a successful clock switch
Reset after a redundant clock switch
Reset after FSCM switches the oscillator to (Group 1) FRC

REGISTER 20-2: OSCTUN: FRC OSCILLATOR TUNING REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	TUN<3:0>			
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-4 **Unimplemented:** Read as '0'

bit 3-0 **TUN<3:0>:** Lower two bits of TUN field. The four-bit field specified by TUN<3:0> specifies the user tuning capability for the internal fast RC oscillator (nominal 7.37 MHz).
0111 = Maximum Frequency
0110 =
0101 =
0100 =
0011 =
0010 =
0001 =
0000 = Center Frequency, Oscillator is running at calibrated frequency
1111 =
1110 =
1101 =
1100 =
1011 =
1010 =
1001 =
1000 = Minimum Frequency

7.13.2. Palabras de configuración

Existen cuatro palabras de configuración que permiten, en tiempo de programación del dispositivo, modificar el comportamiento del mismo. Estas palabras no se podrán modificar en tiempo de ejecución.

Las palabras son:

FOSC Palabra de configuración del oscilador.

REGISTER 20-3: FOSC: OSCILLATOR CONFIGURATION REGISTER

U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23							bit 16

R/P	R/P	U	U	U	R/P	R/P	R/P
FCKSM<1:0>		—	—	—	FOS<2:0>		
bit 15							bit 8

U	U	U	R/P	R/P	R/P	R/P	R/P
—	—	—	FPR<4:0>				
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 23-16 **Unimplemented:** Read as '0'

bit 15-14 **FCKSM<1:0>:** Clock Switching and Monitor Selection Configuration bits
1x = Clock switching is disabled, Fail-Safe Clock Monitor is disabled
01 = Clock switching is enabled, Fail-Safe Clock Monitor is disabled
00 = Clock switching is enabled, Fail-Safe Clock Monitor is enabled

bit 13-11 **Unimplemented:** Read as '0'

bit 10-8 **FOS<2:0>:** Oscillator Group Selection on POR bits
111 = PLL Oscillator; PLL source selected by FPR<4:0> bits. See Table 20-2.
011 = EXT: External Oscillator; OSC1/OSC2 pins; External Oscillator configuration selected by FPR<4:0> bits
010 = LPRC: Internal Low-Power RC
001 = FRC: Internal Fast RC
000 = LPOSC: Low-Power Crystal Oscillator; SOSCI/SOSCO pins

bit 7-4 **Unimplemented:** Read as '0'

bit 3-0 **FPR<4:0>:** Oscillator Selection within Primary Group bits. See Table 20-2.

La palabra de configuración define el comportamiento de algunas características del circuito.

En lenguaje C de Hi-Tech se emplea:

Listado 7.8:

```
__CONFIG(FOSC,XTPLL16); // Primero el registro de configuracion
                        // Luego la opcion
```

Ver más adelante las distintas opciones de cada palabra de configuración

FWDT Palabra de configuración del perro guardián.

Register 24-1: FWDT: Watchdog Timer Configuration Register

Upper Byte:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23							bit 16

Middle Byte:							
R/P	U	U	U	U	U	U	U
FWDTEN	—	—	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U	U	R/P	R/P	R/P	R/P	R/P	R/P
		FWPSA<1:0>		FWPSB<3:0>			
bit 7							bit 0

bit 23-16 **Unimplemented:** Read as '0'

bit 15 **FWDTEN:** Watchdog Enable Configuration bit
1 = Watchdog Enabled (LPRC oscillator cannot be disabled. Clearing the SWDTEN bit in the RCON register. Will have no effect.)
0 = Watchdog Disabled (LPRC oscillator can be disabled by clearing the SWDTEN bit in the RCON register.)

bit 14-6 **Unimplemented:** Read as '0'

bit 5-4: **FWPSA<1:0>:** Prescale Value Selection for Watchdog Timer Prescaler A bits
11 = 1:512
10 = 1:64
01 = 1:8
00 = 1:1

bit 3-0 **FWPSB<3:0>:** Prescale Value Selection for Watchdog Timer Prescaler B bits
1111 = 1:16
1110 = 1:15
•
•
•
0001 = 1:2
0000 = 1:1

Permite habilitar el perro guardián y seleccionar la preescala del mismo.

FBORPOR Palabra de configuración de los sistemas de reset por voltaje insuficiente (BOR) y de reset de arranque (POR) y otros.

Register 24-2: FBORPOR: BOR and POR Configuration Register

Upper Byte:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23							bit 16

Middle Byte:							
R/P	U	U	U	U	R/P	R/P	R/P
MCLREN	—	—	—	—	PWMPIN	HPOL	LPOL
bit 15							bit 8

Lower Byte:							
R/P	U	R/P	R/P	U	U	R/P	R/P
BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
bit 7							bit 0

bit 23-16 **Unimplemented:** Read as '0'

bit 15 **MCLREN:** $\overline{\text{MCLR}}$ Pin Function Enable bit
1 = Pin function is $\overline{\text{MCLR}}$ (default case)
0 = Pin is disabled

bit 14-11 **Unimplemented:** Read as '0'

bit 10 **PWMPIN:** Motor Control PWM Module Pin Mode bit
1 = PWM module pins controlled by PORT register at device Reset (tri-stated)
0 = PWM module pins controlled by PWM module at device Reset (configured as output pins)

bit 9 **HPOL:** Motor Control PWM Module High Side Polarity bit
1 = PWM module high-side output pins have active-high output polarity
0 = PWM module high-side output pins have active-low output polarity

bit 8 **LPOL:** Motor Control PWM Module Low Side Polarity bit
1 = PWM module low-side output pins have active-high output polarity
0 = PWM module low-side output pins have active-low output polarity

bit 7 **BOREN:** PBOR Enable bit
1 = PBOR Enabled
0 = PBOR Disabled

bit 6 **Unimplemented:** Read as '0'

bit 5-4 **BORV<1:0>:** Brown-out Voltage Select bits
11 = 2.0V
10 = 2.7V
01 = 4.2V
00 = 4.5V

bit 3-2 **Unimplemented:** Read as '0'

bit 1-0 **FPWRT<1:0>:** Power-on Reset Timer Value Selection bits
11 = PWRT = 64 ms
10 = PWRT = 16 ms
01 = PWRT = 4 ms
00 = Power-up timer disabled

Note: PWMPIN, HPOL, and LPOL Configuration bits are only available on devices that feature a Motor Control PWM module.

FGS Palabra de configuración del segmento general de código: protección.

Register 24-3: FGS: General Code Segment Configuration Register

Upper Byte:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23							bit 16

Middle Byte:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Byte:							
U	U	U	U	U	U	P	P
—	—	—	—	—	—	GCP	GWRP
bit 7							bit 0

bit 23-2 **Unimplemented:** Read as '0'

bit 1 **GCP:** General Code Segment Code-Protect bit
1 = User program memory is not code-protected
0 = User program memory is code-protected

bit 0 **GWRP:** General Code Segment Write-Protect bit
1 = User program memory is not write-protected
0 = User program memory is write-protected

Note: The BCP and GWRP Configuration bits can only be programmed to a '0'.

dsPIC30F. Lista de opciones.

Description	Config Register	Symbols
Primary oscillator types	FOSC	ECPLL16, ECPLL8, ECPLL4, ECIO, EC, ERC, ERCIO, XTPLL16, XTPLL8, XTPLL4, XT, HS, XTL
Oscillator select	FOSC	POSC, LP, FRC, LPRC
Oscillator system clock switch	FOSC	CLKSWDIS, CLKSWEN, FSCMDIS, FCSMEN
Watchdog timer enable	FWDT	WDTEN, WDTDIS
Watchdog timer pre-scale select	FWDT	WDTPSA512, WDTPSA64, WDTPSA8, WDTPSA1, WDTPSB1-WDTPSB16
Powerup timer enable	FBORPOR	PWRT64, PWRT16, PWRT4, PWRTDIS
Brown-out reset enable	FBORPOR	BOREN, BORDIS
Brown-out reset voltage	FBORPOR	BORV20, BORV27, BORV42, BORV45
MCLR pin function	FBORPOR	MCLREN, MCLRDIS
Motor control PWM	FBORPOR	PWMBIN, HPOL, LPOL [1]
Code protection	FGS	GCPU, GCPP, GWRU, GWRP

dsPIC33F y PIC24H. Lista de opciones.

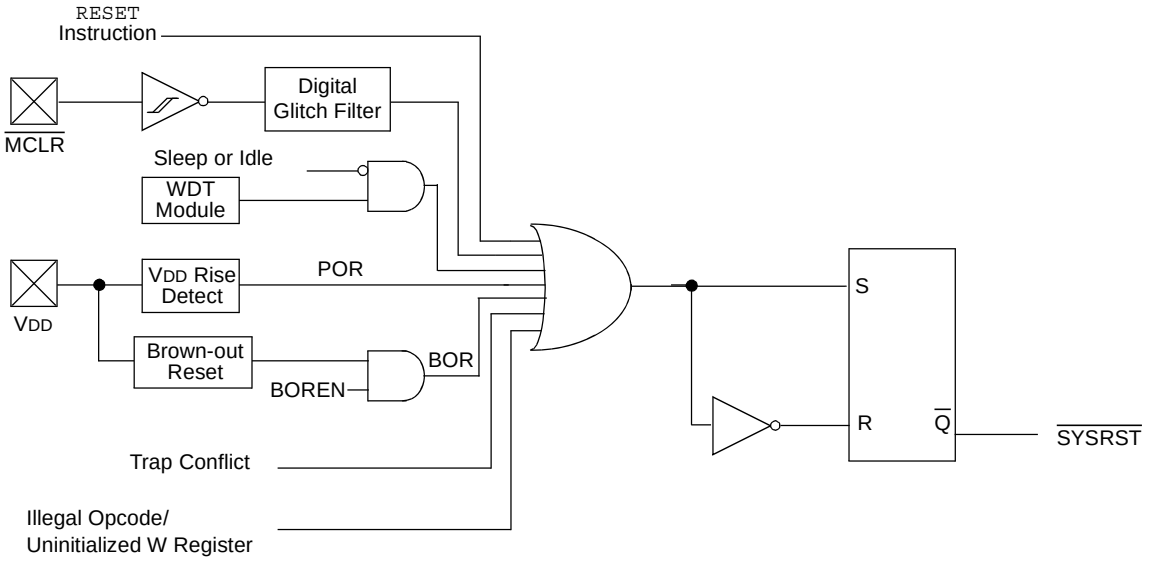
Description	Config Register	Symbols
Code protection	FGS	GCPU, GCPP, GWRU, GWRP
Oscillator two-speed startup	FOSCSEL	IESOEN, IESODIS
Temperature protection	FOSCSEL	TEMPDIS, TEMPEN
Initial oscillator source selection	FOSCSEL	FRCPS, LPRC, LP, OSCPLL, OSC, FRCPLL, FRC
Oscillator clock switching modes	FOSC	FCKSMDIS, CLKSWEN, FCKSMEN
OSC2 pin function	FOSC	OSC2OUT, OSC2DIO
Primary oscillator modes	FOSC	POSCDIS, POSCHS, POSCXT, POSCEC
Watchdog timer enable	FWDT	WDTEN, WDTDIS, WINDIS, WINEN
Watchdog timer prescaler	FWDT	WDTPRE128, WDTPRE32
Watchdog timer postscaler	FWDT	WDTPS32768, WDTPS16384, WDTPS8192, WDTPS4096, WDTPS2048, WDTPS1024, WDTPS512, WDTPS256, WDTPS128, WDTPS64, WDTPS32, WDTPS16, WDTPS8, WDTPS4, WDTPS2, WDTPS1
Motor Control[2]	FPOR	PWMPORT, PWMPWM, PWMHPAH, PWMHPAL, PWMLPAH, PWMLPAL
Power-on Reset Timer	FPOR	PWRT128, PWRT64, PWRT32, PWRT16, PWRT8, PWRT4, PWRT2, PWRTDIS

7.13.3. Sistema de reset

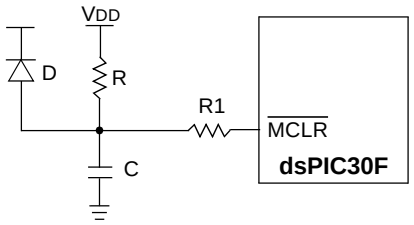
Se dispone en los dsPIC de distintas fuentes de reset, que son:

- Reset de arranque (POR:*Power-on reset*).
- Activación de **/MCLR** durante el modo de funcionamiento normal.
- Activación de **/MCLR** durante el modo de funcionamiento de bajo consumo (*sleep*).
- Reset generado por el perro guardián (WDT) durante el modo de funcionamiento normal.
- Reset del *Brown-out Reset (BOR)* programable, cuando baja la tensión de alimentación.

- Instrucción `reset`.
- Reset causado por la excepción TRAPR.
- Reset causado por la excepción de código de operación ilegal u otros.



Circuito de reset externo . Se recomiendas un circuito de reset externo como el de la figura si tenemos un arranque lento de la alimentación (cuando la fuente tiene poca capacidad y mucha carga conectada).



Note 1: External Power-on Reset circuit is required only if the VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R should be suitably chosen so as to make sure that the voltage drop across R does not violate the device's electrical specification.

3: R1 should be suitably chosen so as to limit any current flowing into $\overline{\text{MCLR}}$ from external capacitor C, in the event of $\overline{\text{MCLR}}$ pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

Brown-out reset (BOR). Los bits de configuración permiten elegir el umbral para disparar el reset cuando la tensión de alimentación cae por debajo de ese umbral.

- 2.6V - 2.71V
- 4.1V - 4.4V
- 4.58V - 4.73V

Registro RCON. Es el registro que contiene la causa del reset por el que ha arrancado el sistema.

Register 8-1: RCON: Reset Control Register

Upper Byte:							
R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
TRAPR	IOPUWR	BGST	LVDEN	LVDL<3:0>			
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR
bit 7							bit 0

bit 15 **TRAPR:** Trap Reset Flag bit
1 = A Trap Conflict Reset has occurred
0 = A Trap Conflict Reset has not occurred

bit 14 **IOPUWR:** Illegal Opcode or Uninitialized W Access Reset Flag bit
1 = An illegal opcode detection, an illegal Address mode, or uninitialized W register used as an address pointer caused a Reset
0 = An illegal opcode or uninitialized W Reset has not occurred

bit 13 **BGST:** Bandgap Stable bit
1 = The bandgap has stabilized
0 = Bandgap is not stable and LVD interrupts should be disabled

bit 12 **LVDEN:** Low Voltage Detect Power Enable bit
1 = Enables LVD, powers up LVD circuit
0 = Disables LVD, powers down LVD circuit

bit 11-8 **LVDL<3:0>:** Low Voltage Detection Limit bits
Refer to **Section 9. "Low Voltage Detect (LVD)"** for further details.

bit 7 **EXTR:** External Reset ($\overline{\text{MCLR}}$) Pin bit
1 = A Master Clear (pin) Reset has occurred
0 = A Master Clear (pin) Reset has not occurred

bit 6 **SWR:** Software RESET (Instruction) Flag bit
1 = A RESET instruction has been executed
0 = A RESET instruction has not been executed

bit 5 **SWDTEN:** Software Enable/Disable of WDT bit
1 = WDT is turned on
0 = WDT is turned off

Note: If FWDTEN fuse bit is '1' (unprogrammed), the WDT is ALWAYS ENABLED, regardless of the SWDTEN bit setting.

bit 4 **WDTO:** Watchdog Timer Time-out Flag bit
1 = WDT Time-out has occurred
0 = WDT Time-out has not occurred

bit 3 **SLEEP:** Wake From Sleep Flag bit
1 = Device has been in Sleep mode
0 = Device has not been in Sleep mode

bit 2 **IDLE:** Wake-up From Idle Flag bit
1 = Device was in Idle mode
0 = Device was not in Idle mode

Register 8-1: RCON: Reset Control Register (Continued)

bit 1 **BOR:** Brown-out Reset Flag bit
1 = A Brown-out Reset has occurred. Note that BOR is also set after Power-on Reset.
0 = A Brown-out Reset has not occurred

bit 0 **POR:** Power-on Reset Flag bit
1 = A Power-up Reset has occurred
0 = A Power-up Reset has not occurred

> **Note:** All of the Reset status bits may be set or cleared in software. Setting one of these bits in software does not cause a device Reset.

En la siguiente tabla se resume el estado de algunos bits del registro **RCON** tras un reset dependiendo de la causa de este.

Condition	Program Counter	TRAPR	IOPUWR	EXTR	SWR	WDTO	IDLE	SLEEP	POR	BOR
Power-on Reset	0x000000	0	0	0	0	0	0	0	1	1
Brown-out Reset	0x000000	0	0	0	0	0	0	0	0	1
$\overline{\text{MCLR}}$ Reset during normal operation	0x000000	0	0	1	0	0	0	0	0	0
Software Reset during normal operation	0x000000	0	0	0	1	0	0	0	0	0
$\overline{\text{MCLR}}$ Reset during Sleep	0x000000	0	0	1	0	0	0	1	0	0
$\overline{\text{MCLR}}$ Reset during Idle	0x000000	0	0	1	0	0	1	0	0	0
WDT Time-out Reset	0x000000	0	0	0	0	1	0	0	0	0
WDT Wake-up	PC + 2	0	0	0	0	1	0	1	0	0
Interrupt Wake-up from Sleep	PC + 2(1)	0	0	0	0	0	0	1	0	0
Clock Failure Trap	0x000004	0	0	0	0	0	0	0	0	0
Trap Reset	0x000000	1	0	0	0	0	0	0	0	0
Illegal Operation Trap	0x000000	0	1	0	0	0	0	0	0	0

Note 1: When the wake-up is due to an enabled interrupt, the PC is loaded with the corresponding interrupt vector.

Resumen de registros

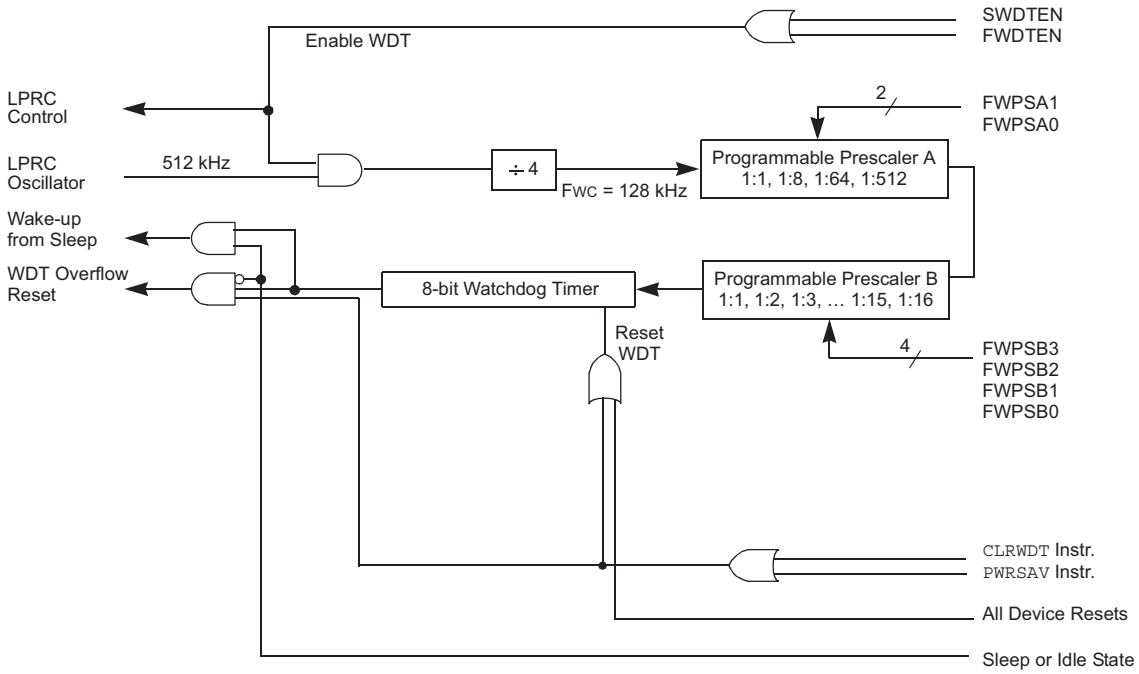
SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
RCON	0740	TRAPR	IOPUWR	BGST	—					EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR	Depends on type of Reset.
OSCCON	0742	TUN3	TUN2	COSC<1:0>		TUN1	TUN0	NOSC<1:0>		POST<1:0>		LOCK	—	CF	—	LPOSCEN	OSWEN	Depends on Configuration bits.

Note: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

File Name	Addr.	Bits 23-16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FOSC	F80000	—	FCKSM<1:0>		—	—	—	—	FOS<1:0>		—	—	—	—	FPR<3:0>			
FWDT	F80002	—	FWDTEN	—	—	—	—	—	—	—	—	—	FWPSA<1:0>		FWPSB<3:0>			
FBORPOR	F80004	—	MCLREN	—	—	—	—	PWMPIN	HPOL	LPOL	BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
FGS	F8000A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	GCP	GWRP

Note: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

7.13.4. Perro guardián



Si el campo FWDTEN está activado entonces el perro guardián estará operativo y no se podrá desactivar en tiempo de ejecución.

Si no está activo entonces a través del campo **RCON.SWDTEN** se puede activar o desactivar el perro guardián en tiempo de ejecución. La preescala utilizada será la que indiquen los bits de configuración contando que cada ciclo son 2 ms. Luego en total el periodo de desbordamiento será de:

$$T_{WDT} = 2ms * PreescalaA * PreescalaB \tag{7.2}$$

Para refrescar el perro guardián se podrá hacer empleando la instrucción:

Listado 7.9:

```
asm("clrwdt");
```

o las instrucciones de paso a modo de bajo consumo (cuando cumpla el periodo del WDT se despertarán)

Listado 7.10:

```
#define IDLE  #1
#define SLEEP #0

// Modo ocioso
asm("pwrsav␣IDLE");
// Modo dormido
asm("pwrsav␣SLEEP");
```

Capítulo 8

Juego de Instrucciones del dsPIC3xF

8.1. Instrucciones de copia de datos

Assembly	Syntax	Description	Words	Cycles
EXCH	Wns,Wnd	Swap Wns and Wnd	1	1
MOV	f {,WREG}	Move f to destination	1	1
MOV	WREG,f	Move WREG to f	1	1
MOV	f,Wnd	Move f to Wnd	1	1
MOV	Wns,f	Move Wns to f	1	1
MOV.b	#lit8,Wnd	Move 8-bit literal to Wnd	1	1
MOV	#lit16,Wnd	Move 16-bit literal to Wnd	1	1
MOV	[Ws+Slit10],Wnd	Move [Ws + signed 10-bit offset] to Wnd	1	1
MOV	Wns,[Wd+Slit10]	Move Wns to [Wd + signed 10-bit offset]	1	1
MOV	Ws,Wd	Move Ws to Wd	1	1
MOV.D	Ws,Wnd	Move double Ws to Wnd:Wnd + 1	1	2
MOV.D	Wns,Wd	Move double Wns:Wns + 1 to Wd	1	2
SWAP	Wn	Wn = byte or nibble swap Wn	1	1
TBLRDH	Ws,Wd	Read high program word to Wd	1	2
TBLRDL	Ws,Wd	Read low program word to Wd	1	2
TBLWTH	Ws,Wd	Write Ws to high program word	1	2
TBLWTL	Ws,Wd	Write Ws to low program word	1	2

Note: When the optional {,WREG} operand is specified, the destination of the instruction is WREG. When {,WREG} is not specified, the destination of the instruction is the file register f.

8.2. Instrucciones matemáticas

Assembly	Syntax	Description	Words	Cycles
ADD	f {,WREG}	Destination = f + WREG	1	1
ADD	#lit10,Wn	Wn = lit10 + Wn	1	1
ADD	Wb,#lit5,Wd	Wd = Wb + lit5	1	1
ADD	Wb,Ws,Wd	Wd = Wb + Ws	1	1
ADDC	f {,WREG}	Destination = f + WREG + (C)	1	1
ADDC	#lit10,Wn	Wn = lit10 + Wn + (C)	1	1
ADDC	Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	1	1
ADDC	Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1
DAW.B	Wn	Wn = decimal adjust Wn	1	1
DEC	f {,WREG}	Destination = f – 1	1	1
DEC	Ws,Wd	Wd = Ws – 1	1	1
DEC2	f {,WREG}	Destination = f – 2	1	1
DEC2	Ws,Wd	Wd = Ws – 2	1	1
DIV.S	Wm,Wn	Signed 16/16-bit integer divide*	1	18
DIV.SD	Wm,Wn	Signed 32/16-bit integer divide*	1	18
DIV.U	Wm,Wn	Unsigned 16/16-bit integer divide*	1	18
DIV.UD	Wm,Wn	Unsigned 32/16-bit integer divide*	1	18
DIVF	Wm,Wn	Signed 16/16-bit fractional divide*	1	18
INC	f {,WREG}	Destination = f + 1	1	1
INC	Ws,Wd	Wd = Ws + 1	1	1
INC2	f {,WREG}	Destination = f + 2	1	1
INC2	Ws,Wd	Wd = Ws + 2	1	1
MUL	f	W3:W2 = f * WREG	1	1
MUL.SS	Wb,Ws,Wnd	{Wnd+1,Wnd} = sign(Wb) * sign(Ws)	1	1
MUL.SU	Wb,#lit5,Wnd	{Wnd+1,Wnd} = sign(Wb) * unsign(lit5)	1	1
MUL.SU	Wb,Ws,Wnd	{Wnd+1,Wnd} = sign(Wb) * unsign(Ws)	1	1
MUL.US	Wb,Ws,Wnd	{Wnd+1,Wnd} = unsign(Wb) * sign(Ws)	1	1
MUL.UU	Wb,#lit5,Wnd	{Wnd+1,Wnd} = unsign(Wb) * unsign(lit5)	1	1
MUL.UU	Wb,Ws,Wnd	{Wnd+1,Wnd} = unsign(Wb) * unsign(Ws)	1	1
SE	Ws,Wnd	Wnd = sign-extended Ws	1	1
SUB	f {,WREG}	Destination = f – WREG	1	1
SUB	#lit10, Wn	Wn = Wn – lit10	1	1
SUB	Wb,#lit5,Wd	Wd = Wb – lit5	1	1
SUB	Wb,Ws,Wd	Wd = Wb – Ws	1	1
SUBB	f {,WREG}	Destination = f – WREG – (C)	1	1
SUBB	#lit10, Wn	Wn = Wn – lit10 – (C)	1	1
SUBB	Wb,#lit5,Wd	Wd = Wb – lit5 – ($\overline{C}$)	1	1
SUBB	Wb,Ws,Wd	Wd = Wb – Ws – ($\overline{C}$)	1	1
SUBBR	f {,WREG}	Destination = WREG – f – ($\overline{C}$)	1	1
SUBBR	Wb,#lit5,Wd	Wd = lit5 – Wb – ($\overline{C}$)	1	1
SUBBR	Wb,Ws,Wd	Wd = Ws – Wb – ($\overline{C}$)	1	1
SUBR	f {,WREG}	Destination = WREG – f	1	1
SUBR	Wb,#lit5,Wd	Wd = lit5 – Wb	1	1
SUBR	Wb,Ws,Wd	Wd = Ws – Wb	1	1
ZE	Ws,Wnd	Wnd = zero-extended Ws	1	1

* Divide instructions are interruptible on a cycle-by-cycle basis. Also, divide instructions must be accompanied by a REPEAT instruction, which adds 1 extra cycle.

8.3. Instrucciones lógicas

Assembly	Syntax	Description	Words	Cycles
AND	f {,WREG}	Destination = f .AND. WREG	1	1
AND	#lit10,Wn	Wn = lit10 .AND. Wn	1	1
AND	Wb,#lit5,Wd	Wd = Wb .AND. lit5	1	1
AND	Wb,Ws,Wd	Wd = Wb .AND. Ws	1	1
CLR	f	f = 0x0000	1	1
CLR	WREG	WREG = 0x0000	1	1
CLR	Wd	Wd = 0x0000	1	1
COM	f {,WREG}	Destination = $\overline{f}$	1	1
COM	Ws,Wd	Wd = $\overline{Ws}$	1	1
IOR	f {,WREG}	Destination = f .IOR. WREG	1	1
IOR	#lit10,Wn	Wn = lit10 .IOR. Wn	1	1
IOR	Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	1
IOR	Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	1
NEG	f {,WREG}	Destination = $\overline{f}$ + 1	1	1
NEG	Ws,Wd	Wd = $\overline{Ws}$ + 1	1	1
SETM	f	f = 0xFFFF	1	1
SETM	WREG	WREG = 0xFFFF	1	1
SETM	Wd	Wd = 0xFFFF	1	1
XOR	f {,WREG}	Destination = f .XOR. WREG	1	1
XOR	#lit10,Wn	Wn = lit10 .XOR. Wn	1	1
XOR	Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1
XOR	Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1

Note: When the optional {,WREG} operand is specified, the destination of the instruction is WREG. When {,WREG} is not specified, the destination of the instruction is the file register f.

8.4. Instrucciones de rotación y desplazamiento

Assembly	Syntax	Description	Words	Cycles
ASR	f {,WREG}	Destination = arithmetic right shift f	1	1
ASR	Ws,Wd	Wd = arithmetic right shift Ws	1	1
ASR	Wb,#lit4,Wnd	Wnd = arithmetic right shift Wb by lit4	1	1
ASR	Wb,Wns,Wnd	Wnd = arithmetic right shift Wb by Wns	1	1
LSR	f {,WREG}	Destination = logical right shift f	1	1
LSR	Ws,Wd	Wd = logical right shift Ws	1	1
LSR	Wb,#lit4,Wnd	Wnd = logical right shift Wb by lit4	1	1
LSR	Wb,Wns,Wnd	Wnd = logical right shift Wb by Wns	1	1
RLC	f {,WREG}	Destination = rotate left through Carry f	1	1
RLC	Ws,Wd	Wd = rotate left through Carry Ws	1	1
RLNC	f {,WREG}	Destination = rotate left (no Carry) f	1	1
RLNC	Ws,Wd	Wd = rotate left (no Carry) Ws	1	1
RRC	f {,WREG}	Destination = rotate right through Carry f	1	1
RRC	Ws,Wd	Wd = rotate right through Carry Ws	1	1
RRNC	f {,WREG}	Destination = rotate right (no Carry) f	1	1
RRNC	Ws,Wd	Wd = rotate right (no Carry) Ws	1	1
SL	f {,WREG}	Destination = left shift f	1	1
SL	Ws,Wd	Wd = left shift Ws	1	1
SL	Wb,#lit4,Wnd	Wnd = left shift Wb by lit4	1	1
SL	Wb,Wns,Wnd	Wnd = left shift Wb by Wns	1	1

Note: When the optional {,WREG} operand is specified, the destination of the instruction is WREG. When {,WREG} is not specified, the destination of the instruction is the file register f.

8.5. Instrucciones de bit

Assembly	Syntax	Description	Words	Cycles
BCLR	f,#bit4	Bit clear f	1	1
BCLR	Ws,#bit4	Bit clear Ws	1	1
BSET	f,#bit4	Bit set f	1	1
BSET	Ws,#bit4	Bit set Ws	1	1
BSW.C	Ws,Wb	Write C bit to Ws<Wb>	1	1
BSW.Z	Ws,Wb	Write $\overline{SZ}$ bit to Ws<Wb>	1	1
BTG	f,#bit4	Bit toggle f	1	1
BTG	Ws,#bit4	Bit toggle Ws	1	1
BTST	f,#bit4	Bit test f	1	1
BTST.C	Ws,#bit4	Bit test Ws to C	1	1
BTST.Z	Ws,#bit4	Bit test Ws to SZ	1	1
BTST.C	Ws,Wb	Bit test Ws<Wb> to C	1	1
BTST.Z	Ws,Wb	Bit test Ws<Wb> to SZ	1	1
BTSTS	f,#bit4	Bit test f then set f	1	1
BTSTS.C	Ws,#bit4	Bit test Ws to C then set Ws	1	1
BTSTS.Z	Ws,#bit4	Bit test Ws to SZ then set Ws	1	1
FBCL	Ws,Wnd	Find bit change from left (MSb) side	1	1
FF1L	Ws,Wnd	Find first one from left (MSb) side	1	1
FF1R	Ws,Wnd	Find first one from right (LSb) side	1	1

Note: Bit positions are specified by bit4 (0:15) for word operations.

8.6. Instrucciones de comparación y salto

Assembly	Syntax	Description	Words	Cycles
BTSC	f,#bit4	Bit test f, skip if clear	1	1 (2 or 3)
BTSC	Ws,#bit4	Bit test Ws, skip if clear	1	1 (2 or 3)
BTSS	f,#bit4	Bit test f, skip if set	1	1 (2 or 3)
BTSS	Ws,#bit4	Bit test Ws, skip if set	1	1 (2 or 3)
CP	f	Compare (f – WREG)	1	1
CP	Wb,#lit5	Compare (Wb – lit5)	1	1
CP	Wb,Ws	Compare (Wb – Ws)	1	1
CP0	f	Compare (f – 0x0000)	1	1
CP0	Ws	Compare (Ws – 0x0000)	1	1
CPB	f	Compare with Borrow (f – WREG – $\overline{C}$)	1	1
CPB	Wb,#lit5	Compare with Borrow (Wb – lit5 – $\overline{C}$)	1	1
CPB	Wb,Ws	Compare with Borrow (Wb – Ws – $\overline{C}$)	1	1
CPSEQ	Wb,Wn	Compare Wb with Wn, Skip if Equal (Wb = Wn)	1	1 (2 or 3)
CPSGT	Wb,Wn	Signed Compare Wb with Wn, Skip if Greater Than (Wb > Wn)	1	1 (2 or 3)
CPSLT	Wb,Wn	Signed Compare Wb with Wn, Skip if Less Than (Wb < Wn)	1	1 (2 or 3)
CPSNE	Wb,Wn	Signed Compare Wb with Wn, Skip if Not Equal (Wb ≠ Wn)	1	1 (2 or 3)

Note 1: Bit positions are specified by bit4 (0:15) for word operations.

2: Conditional skip instructions execute in 1 cycle if the skip is not taken, 2 cycles if the skip is taken over a one-word instruction and 3 cycles if the skip is taken over a two-word instruction.

8.7. Instrucciones de flujo de programa

Assembly	Syntax	Description	Words	Cycles
BRA	Expr	Branch unconditionally	1	2
BRA	Wn	Computed branch	1	2
BRA	C,Expr	Branch if Carry (no Borrow)	1	1 (2)
BRA	GE,Expr	Branch if greater than or equal	1	1 (2)
BRA	GEU,Expr	Branch if unsigned greater than or equal	1	1 (2)
BRA	GT,Expr	Branch if greater than	1	1 (2)
BRA	GTU,Expr	Branch if unsigned greater than	1	1 (2)
BRA	LE,Expr	Branch if less than or equal	1	1 (2)
BRA	LEU,Expr	Branch if unsigned less than or equal	1	1 (2)
BRA	LT,Expr	Branch if less than	1	1 (2)
BRA	LTU,Expr	Branch if unsigned less than	1	1 (2)
BRA	N,Expr	Branch if Negative	1	1 (2)
BRA	NC,Expr	Branch if not Carry (Borrow)	1	1 (2)
BRA	NN,Expr	Branch if not Negative	1	1 (2)
BRA	NOV,Expr	Branch if not Overflow	1	1 (2)
BRA	NZ,Expr	Branch if not Zero	1	1 (2)
BRA	OA,Expr	Branch if Accumulator A Overflow	1	1 (2)
BRA	OB,Expr	Branch if Accumulator B Overflow	1	1 (2)
BRA	OV,Expr	Branch if Overflow	1	1 (2)
BRA	SA,Expr	Branch if Accumulator A Saturate	1	1 (2)
BRA	SB,Expr	Branch if Accumulator B Saturate	1	1 (2)
BRA	Z,Expr	Branch if Zero	1	1 (2)
CALL	Expr	Call subroutine	2	2
CALL	Wn	Call indirect subroutine	1	2
DO	#lit14,Expr	Do code through PC + Expr, (lit14 + 1) times	2	2
DO	Wn,Expr	Do code through PC + Expr, (Wn + 1) times	2	2
GOTO	Expr	Go to address	2	2
GOTO	Wn	Go to address indirectly	1	2
RCALL	Expr	Relative call	1	2
RCALL	Wn	Computed call	1	2
REPEAT	#lit14	Repeat next instruction (lit14 + 1) times	1	1
REPEAT	Wn	Repeat next instruction (Wn + 1) times	1	1
RETFIE		Return from interrupt enable	1	3 (2)
RETLW	#lit10,Wn	Return with lit10 in Wn	1	3 (2)
RETURN		Return from subroutine	1	3 (2)

Note 1: Conditional branch instructions execute in 1 cycle if the branch is not taken, or 2 cycles if the branch is taken.

2: RETURN normally executes in 3 cycles. However, it executes in 2 cycles if an interrupt is pending.

8.8. Instrucciones de acceso a pilas

Assembly	Syntax	Description	Words	Cycles
LNK	#lit14	Link Frame Pointer	1	1
POP	f	Pop TOS to f	1	1
POP	Wd	Pop TOS to Wd	1	1
POP.D	Wnd	Double pop from TOS to Wnd:Wnd + 1	1	2
POP.S		Pop shadow registers	1	1
PUSH	f	Push f to TOS	1	1
PUSH	Ws	Push Ws to TOS	1	1
PUSH.D	Wns	Push double Wns:Wns + 1 to TOS	1	2
PUSH.S		Push shadow registers	1	1
ULNK		Unlink Frame Pointer	1	1

8.9. Instrucciones de control

Assembly	Syntax	Description	Words	Cycles
CLRWDT		Clear Watchdog Timer	1	1
DISI	#lit14	Disable interrupts for (lit14 + 1) instruction cycles	1	1
NOP		No operation	1	1
NOPR		No operation	1	1
PWRSAV	#lit1	Enter Power-Saving mode lit1	1	1
RESET		Software device Reset	1	1

8.10. Instrucciones del motor DSP

Assembly	Syntax	Description	Words	Cycles
ADD	Acc	Add accumulators	1	1
ADD	Ws,#Slit4,Acc	16-bit signed add to Acc	1	1
CLR	Acc,Wx,Wxd,Wy,Wyd,AWB	Clear Acc	1	1
ED	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean distance (no accumulate)	1	1
EDAC	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean distance	1	1
LAC	Ws,#Slit4,Acc	Load Acc	1	1
MAC	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd,AWB	Multiply and accumulate	1	1
MAC	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square and accumulate	1	1
MOVSAC	Acc,Wx,Wxd,Wy,Wyd,AWB	Move Wx to Wxd and Wy to Wyd	1	1
MPY	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	Multiply Wn by Wm to Acc	1	1
MPY	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square to Acc	1	1
MPY.N	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wn by Wm) to Acc	1	1
MSC	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd,AWB	Multiply and subtract from Acc	1	1
NEG	Acc	Negate Acc	1	1
SAC	Acc,#Slit4,Wd	Store Acc	1	1
SAC.R	Acc,#Slit4,Wd	Store rounded Acc	1	1
SFTAC	Acc,#Slit6	Arithmetic shift Acc by Slit6	1	1
SFTAC	Acc,Wn	Arithmetic shift Acc by (Wn)	1	1
SUB	Acc	Subtract accumulators	1	1

Capítulo 9

Familias de DSPs

En este tema trataremos de analizar cuales son las principales familias de procesadores de señal digitales. La intención es tener una imagen global de los fabricantes y arquitecturas disponibles en el mercado a fecha de hoy (Abril 2009). Se puede encontrar una guía actualizada en: `http://www.bdti.com/pocket/pocket.htm`

En las páginas siguientes puedes encontrar dos tablas: una de los Chips de DSPs actuales con sus características más notables y otra con los núcleos (*cores*) de algunos de ellos de los que se puede obtener una licencia para incluirlos en diseños a medida (*full custom*).

Updated April 2009

CHIPS								
Vendor	Family	Floating, Fixed, or Both	Data Width	Core Clock Speed	**BDTImark 2000**	Total On-Chip Memory,	Unit Price	Notes
Analog Devices	ADSP-218x	Fixed point	16 bits	80 MHz	**240**	8 K–104 K	$6–27	Many family members w/ assorted peripherals
	ADSP-219x	Fixed point	16 bits	160 MHz	**410**	8 K–64 K	$12–30	Enhanced version of the ADSP-218x
	ADSP-2126x (SHARC)	Floating point	32/40 bits	200 MHz	**1090**	512 K–768 K	$6–20	Features SIMD, strong multiprocessor support
	ADSP-213xx (SHARC)	Floating point	32/40 bits	400 MHz	**2050**	384 K–1024 K	$8–40	SHARC with a lengthened pipeline for higher clock speeds
	ADSP-BF5xx (Blackfin)	Fixed point	16 bits	750 MHz	4190	52 K–308 K	$5–32	Dual-MAC DSP with variable speed and voltage
	ADSP-TS20x (TigerSHARC)	Both	8/16/32/40 bits	600 MHz	6400	512 K–3 M	$133–207	4-way VLIW with SIMD capabilities; uses eDRAM
Freescale	DSP563xx	Fixed point	24 bits	275 MHz	**820**	24 K–648 K	n/a	Many audio-oriented parts; binary-compatible with '560xx
	DSP56F8xx (56800)	Fixed point	16 bits	80 MHz	**110**	28 K–152 K	n/a	Contains many microcontroller-like features
	DSP5685x/ 56F8xxx (56800E)	Fixed point	16 bits	120 MHz	*340*	20 K–612 K	n/a	Enhanced version of the '568xx
	MSC71xx (SC1400)	Fixed point	16 bits	300 MHz	*3370*	408 K–472 K	$32–38	Based on SC1400 licensable core
	MSC81xx (SC140)	Fixed point	16 bits	500 MHz	5610	514 K–1440 K	$55–122	Based on SC1400-compatible core; most chips use 4 cores
	MSC81xx (SC3400)	Fixed point	16 bits	1 GHz	11900	11.2 M	$123–156	Based on SC3400 core; quad-core chip
Microchip	dsPIC3xF	Fixed point	16 bits	40 MHz	*130*	13 K–287 K	$2–7	Hybrid microcontroller/DSP
NXP	TriMedia 3270 core	Both	8/16/32 bits	350 MHz	n/a	16 M	n/a	VLIW media processor with SIMD capabilities
picoChip	PC102	Fixed Point	16 bits	160 MHz	n/a	1 M	$99	Massively parallel chip with 344 processors
Renesas	SH77xx	Both	16/32 bits	400 MHz	**1250**	48 K–304 K	$18–32	Superscalar microprocessor with 3D geometry instructions
Sandbridge	SB3500	Fixed point	16 bits	500 MHz	n/a	928 K	n/a	Three 4-way multithreaded with 16-way SIMD unit DSP's and one ARM9E
Texas Instruments	TMS320C28x/ F28x	Fixed point	32 bits	150 MHz	n/a	40 K–326 K	$3–16	Hybrid microcontroller/DSP; assembly-compatible w/ 'C24x
	TMS320F283xx	Floating point	32 bits	300 MHz	n/a	182 K–582 K	$12–15	Adds floating-point unit to 'C28x
	TMS320C54x	Fixed point	16 bits	160 MHz	500	20 K–272 K	$3–24	Many specialized instructions
	TMS320C55x	Fixed point	16 bits	300 MHz	**1460**	80 K–376 K	$4–15	Dual-issue, dual-MAC DSP; assembly-compatible w/ 'C54x
	TMS320C64x/ DM64x	Fixed point	8/16 bits	1 GHz	**9130**	128 K–1032 K	$9–182	Adds quad-MAC capabilities and specialized operations to 'C62x
	TMS320DM6446	Fixed point	8/16 bits	600 MHz	**6590**	240 K	$32	ARM9, C64x+ and video accelerator (BDTImark2000 for C64x+ only)
	'C64x+	Fixed point	8/16 bits	1.2 GHz	**13170**	128 K–60 M	$10–191	Adds 8-MAC capabilities and specialized operations to 'C64x
	TMS320C67x	Floating point	32 bits	300 MHz	**1500**	72 K–264 K	$12–28	Floating-point version of 'C62x
	TMS320C67x+	Floating point	32 bits	300 MHz	n/a	480 K–672 K	$6–25	Adds registers and audio-oriented instructions to the 'C67x
	OMAP35x	Fixed point	8/16/32 bits	600 MHz	*4540*	320 K	$20–37	Metrics for ARM Cortex-A8 core only (optional 'C64x+ DSP available)

Tilera	TILE64	Fixed point	8/16 bits	866 MHz	n/a	5120 K	$896	64 core chip, 3-way VLIW with SIMD capabilities
VeriSilicon	VSI40x (ZSP400)	Fixed point	16/32 bits	200 MHz	**940**	96 K–252 K	n/a	Based on ZSP400 licensable core (see below)

CORES								
Licensor	Family	Floating, Fixed, or Both	Data Width	Core Clock Speed [1,8]	BDTImark2000™ BDTIsimMark2000™ [2]	Total Core Memory Space, Bytes	Die area [8]	Notes
ARC	ARC 600/ ARC XY	Fixed point	16/32 bits [9]	180 MHz	n/a	4 G	0.77 mm²	Customizable core with optional DSP features
	ARC 700/ ARC XY	Fixed point	16/32 bits [9]	265 MHz	n/a	4 G	1.1 mm²	Longer pipeline enables higher clock rate
	AV 401V	Fixed-point	16/32 bits	n/a	n/a	4 G	n/a	Licensable video subsystem based on ARC 700 plus accelerators
ARM	ARM7	Fixed point	32 bits	145 MHz	*160*	4 G	0.28 mm²	Widely licensed 32-bit microprocessor core
	ARM9	Fixed point	32 bits	255 MHz	*320*	4 G	n/a	Adds separate bus for data access, deeper pipeline to ARM7
	ARM9E	Fixed point	16/32 bits	265 MHz	*550*	4 G	1.7 mm²	ARM9 enhanced with single-cycle MAC unit
	ARM1136	Fixed point	16/32 bits	330 MHz	*1160*	4 G	2.3 mm²	Adds SIMD, load/store unit, branch prediction, deeper pipeline
	ARM1176	Fixed-point	16/32 bits	335 MHz	*1200*	4 G	2.5 mm²	Very similar to ARM1136
	Cortex-A8	Fixed-point	8/16/32 bits	n/a	*7.6 per MHz*	4 G	n/a	Dual-issue superscalar architecture with NEON DSP extensions
	Cortex-R4	Fixed-point	16/32 bits	n/a	*3.8 per MHz*	4 G	n/a	Dual-issue superscalar architecture, software compatible with ARM9E
CEVA	CEVA-TeakLite	Fixed point	16 bits	170 MHz	n/a	256 K	0.4 mm²	Single-MAC, single-issue DSP core
	CEVA-TeakLite II	Fixed point	16 bits	200 MHz	n/a	4 M	0.5 mm²	Faster version of CEVA-TeakLite
	CEVA-TeakLite III	Fixed point	16/32 bits	550 MHz	n/a	4 G	n/a	Dual-MAC DSP core; backward compatible with TeakLite II
	CEVA-Teak	Fixed point	16 bits	150 MHz	n/a	8 M	0.9 mm²	Dual-MAC DSP core
	CEVA-X1620	Fixed point	8/16 bits	330 MHz	*2660*	4 G	2.6 mm²	8-way VLIW, dual-MAC DSP core
	CEVA-X1641	Fixed point	8/16 bits	600 MHz	n/a	4 G	n/a	8-way VLIW, quad-MAC DSP core supporting SIMD operations
MIPS	MIPS32 24KE (with DSP ASE)	Fixed point	16/32 bits	335 MHz	*1000*	4 G	2.0 mm²	MIPS core with SIMD DSP extensions
NXP	CoolFlux DSP	Fixed point	24 bits	175 MHz	n/a	640 K	0.34 mm²	Dual-MAC core targets low-power audio applications
Tensilica	Diamond 545CK	Fixed point	18 bits [9]	230 MHz	*3820*	4 G	3.7 mm²	VLIW-based customizable core; with optional DSP features
	ZSP*neo*	Fixed point	16/32 bits	165 MHz	n/a	256 K	0.45 mm²	Single-MAC, scalar variant of the ZSP400
	ZSP200	Fixed point	16/32 bits	165 MHz	n/a	256 K	0.7 mm²	Single-MAC, 2-way superscalar variant of the ZSP400
	ZSP400	Fixed point	16/32 bits	165 MHz	**780**	256 K	1.3 mm²	Dual-MAC, 4-way superscalar DSP core
	ZSP410	Fixed point	16/32 bits	185 MHz	**870**	4 G	1.4 mm²	Enhanced ZSP400 with instruction cache
	ZSP500	Fixed point	16/32 bits	205 MHz	*1620*	64 M	2.2 mm²	Second-generation ZSP; dual-MAC, 4-way superscalar
	ZSP540	Fixed point	16/32 bits	200 MHz	n/a	64 M	2.7 mm²	Quad-MAC, 4-way variant of the ZSP500

VeriSilicon	ZSP600	Fixed point	16/32 bits	175 MHz	n/a	64 M	3.1 mm²	Quad-MAC, 6-way variant of the ZSP500

FPGAs				
Vendor	Family	Speed Grade	Unit Price	Notes
Altera	Stratix II EP2S15F672C5	Slow speed grade	$38	FPGA with hardwired DSP features, such as multipliers
Xilinx	Virtex-4 SX25-10FF668C	Slow speed grade	$89	FPGA with hardwired DSP features, such as multipliers
	Virtex-4 XC4VFX140-11FF1760C4006	Medium speed grade	$1280	FPGA with hardwired DSP features, such as multipliers

Apéndice A

Definiciones para PICC del PIC16F88

Listado A.1: Codigo/pic16f87.h

```
#ifndef _HTC_H_
#warning Header file pic16f87.h included directly. Use #include <htc.h>.
#endif

/* header file for the MICROCHIP PIC microcontrollers
       PIC16F87
       PIC16F88
*/

#ifndef __PIC16F87_H
#define __PIC16F87_H

// Special function register definitions

volatile        unsigned char     TMR0            @ 0x01;
volatile        unsigned char     PCL             @ 0x02;
volatile        unsigned char     STATUS          @ 0x03;
                unsigned char     FSR             @ 0x04;
volatile        unsigned char     PORTA           @ 0x05;
volatile        unsigned char     PORTB           @ 0x06;
volatile        unsigned char     PCLATH          @ 0x0A;
volatile        unsigned char     INTCON          @ 0x0B;
volatile        unsigned char     PIR1            @ 0x0C;
volatile        unsigned char     PIR2            @ 0x0D;
volatile        unsigned char     TMR1L           @ 0x0E;
volatile        unsigned char     TMR1H           @ 0x0F;
volatile        unsigned char     T1CON           @ 0x10;
volatile        unsigned char     TMR2            @ 0x11;
volatile        unsigned char     T2CON           @ 0x12;
volatile        unsigned char     SSPBUF          @ 0x13;
volatile        unsigned char     SSPCON          @ 0x14;
volatile        unsigned char     CCPR1L          @ 0x15;
volatile        unsigned char     CCPR1H          @ 0x16;
volatile        unsigned char     CCP1CON         @ 0x17;
volatile        unsigned char     RCSTA           @ 0x18;
volatile        unsigned char     TXREG           @ 0x19;
volatile        unsigned char     RCREG           @ 0x1A;
#if defined(_16F88)
volatile        unsigned char     ADRESH          @ 0x1E;
volatile        unsigned char     ADCON0          @ 0x1F;
#endif
                unsigned char     OPTION          @ 0x81;
volatile        unsigned char     TRISA           @ 0x85;
volatile        unsigned char     TRISB           @ 0x86;
                unsigned char     PIE1            @ 0x8C;
                unsigned char     PIE2            @ 0x8D;
volatile        unsigned char     PCON            @ 0x8E;
volatile        unsigned char     OSCCON          @ 0x8F;
volatile        unsigned char     OSCTUNE         @ 0x90;
volatile        unsigned char     PR2          @ 0x92;
volatile        unsigned char     SSPADD          @ 0x93;
```

```
53  volatile        unsigned char     SSPSTAT          @ 0x94;
    volatile        unsigned char     TXSTA            @ 0x98;
                    unsigned char     SPBRG            @ 0x99;
    #if defined(_16F88)
                    unsigned char     ANSEL            @ 0x9B;
58  #endif
    volatile        unsigned char     CMCON            @ 0x9C;
    volatile        unsigned char     CVRCON           @ 0x9D;
    #if defined(_16F88)
    volatile        unsigned char     ADRESL           @ 0x9E;
63  volatile        unsigned char     ADCON1           @ 0x9F;
    #endif
    volatile        unsigned char     WDTCON           @ 0x105;
    volatile        unsigned char     EEDATA           @ 0x10C;
    // Alternate definition
68  volatile        unsigned char     EEDAT            @ 0x10C;
    volatile        unsigned char     EEADR            @ 0x10D;
    // Alternate definition
    volatile        unsigned char     EEADRL           @ 0x10D;
    volatile        unsigned char     EEDATH           @ 0x10E;
73  volatile        unsigned char     EEADRH           @ 0x10F;
    volatile        unsigned char     EECON1           @ 0x18C;
    volatile        unsigned char     EECON2           @ 0x18D;


78  /* Definitions for STATUS register */
    volatile        bit       CARRY   @ ((unsigned)&STATUS*8)+0;
    volatile        bit       DC              @ ((unsigned)&STATUS*8)+1;
    volatile        bit       ZERO    @ ((unsigned)&STATUS*8)+2;
    volatile        bit       PD              @ ((unsigned)&STATUS*8)+3;
83  volatile        bit       TO              @ ((unsigned)&STATUS*8)+4;
    volatile        bit       RP0             @ ((unsigned)&STATUS*8)+5;
    volatile        bit       RP1             @ ((unsigned)&STATUS*8)+6;
    volatile        bit       IRP             @ ((unsigned)&STATUS*8)+7;

88  /* Definitions for PORTA register */
    volatile        bit       RA0             @ ((unsigned)&PORTA*8)+0;
    volatile        bit       RA1             @ ((unsigned)&PORTA*8)+1;
    volatile        bit       RA2             @ ((unsigned)&PORTA*8)+2;
    volatile        bit       RA3             @ ((unsigned)&PORTA*8)+3;
93  volatile        bit       RA4             @ ((unsigned)&PORTA*8)+4;
    volatile        bit       RA5             @ ((unsigned)&PORTA*8)+5;
    volatile        bit       RA6             @ ((unsigned)&PORTA*8)+6;
    volatile        bit       RA7             @ ((unsigned)&PORTA*8)+7;

98  /* Definitions for PORTB register */
    volatile        bit       RB0             @ ((unsigned)&PORTB*8)+0;
    volatile        bit       RB1             @ ((unsigned)&PORTB*8)+1;
    volatile        bit       RB2             @ ((unsigned)&PORTB*8)+2;
    volatile        bit       RB3             @ ((unsigned)&PORTB*8)+3;
103 volatile        bit       RB4             @ ((unsigned)&PORTB*8)+4;
    volatile        bit       RB5             @ ((unsigned)&PORTB*8)+5;
    volatile        bit       RB6             @ ((unsigned)&PORTB*8)+6;
    volatile        bit       RB7             @ ((unsigned)&PORTB*8)+7;

108 /* Definitions for INTCON register */
    volatile        bit       RBIF            @ ((unsigned)&INTCON*8)+0;
    volatile        bit       INT0IF          @ ((unsigned)&INTCON*8)+1;
    volatile        bit       TMR0IF          @ ((unsigned)&INTCON*8)+2;
    volatile        bit       RBIE            @ ((unsigned)&INTCON*8)+3;
113                 bit       INT0IE          @ ((unsigned)&INTCON*8)+4;
    volatile        bit       TMR0IE          @ ((unsigned)&INTCON*8)+5;
    volatile        bit       PEIE            @ ((unsigned)&INTCON*8)+6;
    volatile            bit   GIE                     @ ((unsigned)&INTCON*8)+7;

118 /* Definitions for PIR1 register */
    volatile        bit       TMR1IF          @ ((unsigned)&PIR1*8)+0;
    volatile        bit       TMR2IF          @ ((unsigned)&PIR1*8)+1;
    volatile        bit       CCP1IF          @ ((unsigned)&PIR1*8)+2;
    volatile        bit       SSPIF           @ ((unsigned)&PIR1*8)+3;
123 volatile        bit       TXIF            @ ((unsigned)&PIR1*8)+4;
    volatile        bit       RCIF            @ ((unsigned)&PIR1*8)+5;
    volatile        bit       ADIF            @ ((unsigned)&PIR1*8)+6;

    /* Definitions for PIR2 register */
128 volatile        bit       EEIF            @ ((unsigned)&PIR2*8)+4;
    volatile        bit       CMIF            @ ((unsigned)&PIR2*8)+6;
    volatile        bit       OSFIF           @ ((unsigned)&PIR2*8)+7;
```

```
    /* Definitions for T1CON register */
133 volatile        bit         T1R1ON          @ ((unsigned)&T1CON*8)+0;
    volatile        bit         T1R1CS          @ ((unsigned)&T1CON*8)+1;
    volatile        bit         T1SYNC          @ ((unsigned)&T1CON*8)+2;
    volatile        bit         T1OSCEN         @ ((unsigned)&T1CON*8)+3;
    volatile        bit         T1CKPS0         @ ((unsigned)&T1CON*8)+4;
138 volatile        bit         T1CKPS1         @ ((unsigned)&T1CON*8)+5;
                    bit         T1RUN           @ ((unsigned)&T1CON*8)+6;

    /* Definitions for T2CON register */
    volatile        bit         T2CKPS0         @ ((unsigned)&T2CON*8)+0;
143 volatile        bit         T2CKPS1         @ ((unsigned)&T2CON*8)+1;
    volatile        bit         TMR2ON          @ ((unsigned)&T2CON*8)+2;
    volatile        bit         TOUTPS0         @ ((unsigned)&T2CON*8)+3;
    volatile        bit         TOUTPS1         @ ((unsigned)&T2CON*8)+4;
    volatile        bit         TOUTPS2         @ ((unsigned)&T2CON*8)+5;
148 volatile        bit         TOUTPS3         @ ((unsigned)&T2CON*8)+6;

    /* Definitions for SSPCON register */
    volatile        bit         SSPM0           @ ((unsigned)&SSPCON*8)+0;
    volatile        bit         SSPM1           @ ((unsigned)&SSPCON*8)+1;
153 volatile        bit         SSPM2           @ ((unsigned)&SSPCON*8)+2;
    volatile        bit         SSPM3           @ ((unsigned)&SSPCON*8)+3;
    volatile        bit         CKP             @ ((unsigned)&SSPCON*8)+4;
    volatile        bit         SSPEN           @ ((unsigned)&SSPCON*8)+5;
    volatile        bit         SSPOV           @ ((unsigned)&SSPCON*8)+6;
158 volatile        bit         WCOL            @ ((unsigned)&SSPCON*8)+7;

    /* Definitions for CCP1CON register */
                    bit         CCP1M0          @ ((unsigned)&CCP1CON*8)+0;
                    bit         CCP1M1          @ ((unsigned)&CCP1CON*8)+1;
163                 bit         CCP1M2          @ ((unsigned)&CCP1CON*8)+2;
                    bit         CCP1M3          @ ((unsigned)&CCP1CON*8)+3;
    volatile        bit         CCP1Y           @ ((unsigned)&CCP1CON*8)+4;
    volatile        bit         CCP1X           @ ((unsigned)&CCP1CON*8)+5;

168 /* Definitions for RCSTA register */
    volatile        bit         RX9D            @ ((unsigned)&RCSTA*8)+0;
    volatile        bit         OERR            @ ((unsigned)&RCSTA*8)+1;
    volatile        bit         FERR            @ ((unsigned)&RCSTA*8)+2;
                    bit         ADDEN           @ ((unsigned)&RCSTA*8)+3;
173                 bit         CREN            @ ((unsigned)&RCSTA*8)+4;
                    bit         SREN            @ ((unsigned)&RCSTA*8)+5;
                    bit         RX9             @ ((unsigned)&RCSTA*8)+6;
                    bit         SPEN            @ ((unsigned)&RCSTA*8)+7;

178 #if defined(_16F88)
    /* Definitions for ADCON0 register */
    volatile        bit         ADON            @ ((unsigned)&ADCON0*8)+0;
    volatile        bit         GODONE          @ ((unsigned)&ADCON0*8)+2;
    volatile        bit         CHS0            @ ((unsigned)&ADCON0*8)+3;
183 volatile        bit         CHS1            @ ((unsigned)&ADCON0*8)+4;
    volatile        bit         CHS2            @ ((unsigned)&ADCON0*8)+5;
    volatile        bit         ADCS0           @ ((unsigned)&ADCON0*8)+6;
    volatile        bit         ADCS1           @ ((unsigned)&ADCON0*8)+7;
    /* Alternate definitions for ADCON0 register */
188 volatile        bit         ADGO            @ ((unsigned)&ADCON0*8)+2;
    #endif

    /* Definitions for OPTION register */
                    bit         PS0             @ ((unsigned)&OPTION*8)+0;
193                 bit         PS1             @ ((unsigned)&OPTION*8)+1;
                    bit         PS2             @ ((unsigned)&OPTION*8)+2;
                    bit         PSA             @ ((unsigned)&OPTION*8)+3;
                    bit         T0SE            @ ((unsigned)&OPTION*8)+4;
                    bit         T0CS            @ ((unsigned)&OPTION*8)+5;
198                 bit         INTEDG          @ ((unsigned)&OPTION*8)+6;
                    bit         RBPU            @ ((unsigned)&OPTION*8)+7;

    /* Definitions for TRISA register */
    volatile        bit         TRISA0          @ ((unsigned)&TRISA*8)+0;
203 volatile        bit         TRISA1          @ ((unsigned)&TRISA*8)+1;
    volatile        bit         TRISA2          @ ((unsigned)&TRISA*8)+2;
    volatile        bit         TRISA3          @ ((unsigned)&TRISA*8)+3;
    volatile        bit         TRISA4          @ ((unsigned)&TRISA*8)+4;
    volatile        bit         TRISA5          @ ((unsigned)&TRISA*8)+5;
208 volatile        bit         TRISA6          @ ((unsigned)&TRISA*8)+6;
```

```
    volatile        bit     TRISA7          @ ((unsigned)&TRISA*8)+7;

    /* Definitions for TRISB register */
    volatile        bit     TRISB0          @ ((unsigned)&TRISB*8)+0;
213 volatile        bit     TRISB1          @ ((unsigned)&TRISB*8)+1;
    volatile        bit     TRISB2          @ ((unsigned)&TRISB*8)+2;
    volatile        bit     TRISB3          @ ((unsigned)&TRISB*8)+3;
    volatile        bit     TRISB4          @ ((unsigned)&TRISB*8)+4;
    volatile        bit     TRISB5          @ ((unsigned)&TRISB*8)+5;
218 volatile        bit     TRISB6          @ ((unsigned)&TRISB*8)+6;
    volatile        bit     TRISB7          @ ((unsigned)&TRISB*8)+7;

    /* Definitions for PIE1 register */
                    bit     TMR1IE          @ ((unsigned)&PIE1*8)+0;
223                 bit     TMR2IE          @ ((unsigned)&PIE1*8)+1;
                    bit     CCP1IE          @ ((unsigned)&PIE1*8)+2;
                    bit     SSPIE           @ ((unsigned)&PIE1*8)+3;
                    bit     TXIE            @ ((unsigned)&PIE1*8)+4;
                    bit     RCIE            @ ((unsigned)&PIE1*8)+5;
228                 bit     ADIE            @ ((unsigned)&PIE1*8)+6;

    /* Definitions for PIE2 register */
                    bit     EEIE            @ ((unsigned)&PIE2*8)+4;
                    bit     CMIE            @ ((unsigned)&PIE2*8)+6;
233                 bit     OSFIE           @ ((unsigned)&PIE2*8)+7;

    /* Definitions for PCON register */
    volatile        bit     BOR             @ ((unsigned)&PCON*8)+0;
    volatile        bit     POR             @ ((unsigned)&PCON*8)+1;
238
    /* Definitions for OSCCON register */
                    bit     SCS0            @ ((unsigned)&OSCCON*8)+0;
                    bit     SCS1            @ ((unsigned)&OSCCON*8)+1;
    volatile        bit     IOFS            @ ((unsigned)&OSCCON*8)+2;
243 volatile        bit     OSTS            @ ((unsigned)&OSCCON*8)+3;
    volatile        bit     IRCF0           @ ((unsigned)&OSCCON*8)+4;
    volatile        bit     IRCF1           @ ((unsigned)&OSCCON*8)+5;
    volatile        bit     IRCF2           @ ((unsigned)&OSCCON*8)+6;

248 /* Definitions for OSCTUNE register */
                    bit     TUN0            @ ((unsigned)&OSCTUNE*8)+0;
                    bit     TUN1            @ ((unsigned)&OSCTUNE*8)+1;
                    bit     TUN2            @ ((unsigned)&OSCTUNE*8)+2;
                    bit     TUN3            @ ((unsigned)&OSCTUNE*8)+3;
253                 bit     TUN4            @ ((unsigned)&OSCTUNE*8)+4;
                    bit     TUN5            @ ((unsigned)&OSCTUNE*8)+5;

    /* Definitions for SSPSTAT register */
    volatile        bit     BF              @ ((unsigned)&SSPSTAT*8)+0;
258 volatile        bit     UA              @ ((unsigned)&SSPSTAT*8)+1;
    volatile        bit     RW              @ ((unsigned)&SSPSTAT*8)+2;
    volatile        bit     START           @ ((unsigned)&SSPSTAT*8)+3;
    volatile        bit     STOP            @ ((unsigned)&SSPSTAT*8)+4;
    volatile        bit     DA              @ ((unsigned)&SSPSTAT*8)+5;
263                 bit     CKE             @ ((unsigned)&SSPSTAT*8)+6;
                    bit     SMP             @ ((unsigned)&SSPSTAT*8)+7;

    /* Definitions for TXSTA register */
    volatile        bit     TX9D            @ ((unsigned)&TXSTA*8)+0;
268 volatile        bit     TRMT            @ ((unsigned)&TXSTA*8)+1;
                    bit     BRGH            @ ((unsigned)&TXSTA*8)+2;
                    bit     SYNC            @ ((unsigned)&TXSTA*8)+4;
                    bit     TXEN            @ ((unsigned)&TXSTA*8)+5;
                    bit     TX9             @ ((unsigned)&TXSTA*8)+6;
273                 bit     CSRC            @ ((unsigned)&TXSTA*8)+7;

    #if defined(_16F88)
    /* Definitions for ANSEL register */
                    bit     ANS0            @ ((unsigned)&ANSEL*8)+0;
278                 bit     ANS1            @ ((unsigned)&ANSEL*8)+1;
                    bit     ANS2            @ ((unsigned)&ANSEL*8)+2;
                    bit     ANS3            @ ((unsigned)&ANSEL*8)+3;
                    bit     ANS4            @ ((unsigned)&ANSEL*8)+4;
                    bit     ANS5            @ ((unsigned)&ANSEL*8)+5;
283                 bit     ANS6            @ ((unsigned)&ANSEL*8)+6;
    #endif

    /* Definitions for CMCON register */
```

```
                bit     CM0             @ ((unsigned)&CMCON*8)+0;
                bit     CM1             @ ((unsigned)&CMCON*8)+1;
                bit     CM2             @ ((unsigned)&CMCON*8)+2;
                bit     CIS             @ ((unsigned)&CMCON*8)+3;
                bit     C1INV           @ ((unsigned)&CMCON*8)+4;
                bit     C2INV           @ ((unsigned)&CMCON*8)+5;
volatile        bit     C1OUT           @ ((unsigned)&CMCON*8)+6;
volatile        bit     C2OUT           @ ((unsigned)&CMCON*8)+7;

/* Definitions for CVRCON register */
                bit     CVR0            @ ((unsigned)&CVRCON*8)+0;
                bit     CVR1            @ ((unsigned)&CVRCON*8)+1;
                bit     CVR2            @ ((unsigned)&CVRCON*8)+2;
                bit     CVR3            @ ((unsigned)&CVRCON*8)+3;
                bit     CVRR            @ ((unsigned)&CVRCON*8)+5;
                bit     CVROE           @ ((unsigned)&CVRCON*8)+6;
                bit     CVREN           @ ((unsigned)&CVRCON*8)+7;

#if defined(_16F88)
/* Definitions for ADCON1 register */
                bit     VCFG0           @ ((unsigned)&ADCON1*8)+4;
                bit     VCFG1           @ ((unsigned)&ADCON1*8)+5;
                bit     ADCS2           @ ((unsigned)&ADCON1*8)+6;
                bit     ADFM            @ ((unsigned)&ADCON1*8)+7;
#endif

/* Definitions for WDTCON register */
                bit     SWDTEN          @ ((unsigned)&WDTCON*8)+0;
                bit     WDTPS0          @ ((unsigned)&WDTCON*8)+1;
                bit     WDTPS1          @ ((unsigned)&WDTCON*8)+2;
                bit     WDTPS2          @ ((unsigned)&WDTCON*8)+3;
                bit     WDTPS3          @ ((unsigned)&WDTCON*8)+4;

/* Definitions for EECON1 register */
volatile        bit     RD              @ ((unsigned)&EECON1*8)+0;
volatile        bit     WR              @ ((unsigned)&EECON1*8)+1;
volatile        bit     WREN            @ ((unsigned)&EECON1*8)+2;
volatile        bit     WRERR           @ ((unsigned)&EECON1*8)+3;
volatile        bit     FREE            @ ((unsigned)&EECON1*8)+4;
volatile        bit     EEPGD           @ ((unsigned)&EECON1*8)+7;

// Configuration Mask Definitions
#define CONFIG_ADDR     0x2007
// Protection of program code
#define PROTECT         0x1FFF
#define UNPROTECT       0x3FFF
// CCP1 Pin selection
#define CCPRB0          0x3FFF
#define CCPRB3          0x2FFF
// In-Circuit Debugger Mode
#define DEBUGEN         0x37FF
#define DEBUGDIS        0x3FFF
// Flash Program Memory Write Enable
#define UNPROTECT       0x3FFF
#define WP0             0x3DFF
#define WP1             0x3BFF
#define WPA             0x39FF
// Data EE Memory Code Protection
#define UNPROTECT       0x3FFF
#define CPD             0x3EFF
// Low Voltage Programming Enable
#define LVPEN           0x3FFF
#define LVPDIS          0x3F7F
// Brown out detection enable
#define BOREN           0x3FFF
#define BORDIS          0x3FBF
// Master clear reset pin function
#define MCLREN          0x3FFF
#define MCLRDIS         0x3FDF
// Power up timer enable
#define PWRTDIS         0x3FFF
#define PWRTEN          0x3FF7
// Watchdog timer enable
#define WDTEN           0x3FFF
#define WDTDIS          0x3FFB
// Oscillator configurations
#define RCCLK           0x3FFF
```

```
    #define RCIO            0x3FFE
    #define INTCLK          0x3FFD
    #define INTIO           0x3FFC
368 #define EC              0x3FEF
    #define HS              0x3FEE
    #define XT              0x3FED
    #define LP              0x3FEC

373 #define CONFIG_ADDR2    0x2008
    // Fail Clock Monitor Enable
    #define FCMEN           0x3FFF
    #define FCMDIS          0x3FFE
    // Internal External Switch Over
378 #define IESOEN          0x3FFF
    #define IESODIS         0x3FFD


383 #endif
```

Apéndice B

Juego de Instrucciones del dsPIC3xF

B.1. Instrucciones de copia de datos

Assembly	Syntax	Description	Words	Cycles
EXCH	Wns,Wnd	Swap Wns and Wnd	1	1
MOV	f {,WREG}	Move f to destination	1	1
MOV	WREG,f	Move WREG to f	1	1
MOV	f,Wnd	Move f to Wnd	1	1
MOV	Wns,f	Move Wns to f	1	1
MOV.b	#lit8,Wnd	Move 8-bit literal to Wnd	1	1
MOV	#lit16,Wnd	Move 16-bit literal to Wnd	1	1
MOV	[Ws+Slit10],Wnd	Move [Ws + signed 10-bit offset] to Wnd	1	1
MOV	Wns,[Wd+Slit10]	Move Wns to [Wd + signed 10-bit offset]	1	1
MOV	Ws,Wd	Move Ws to Wd	1	1
MOV.D	Ws,Wnd	Move double Ws to Wnd:Wnd + 1	1	2
MOV.D	Wns,Wd	Move double Wns:Wns + 1 to Wd	1	2
SWAP	Wn	Wn = byte or nibble swap Wn	1	1
TBLRDH	Ws,Wd	Read high program word to Wd	1	2
TBLRDL	Ws,Wd	Read low program word to Wd	1	2
TBLWTH	Ws,Wd	Write Ws to high program word	1	2
TBLWTL	Ws,Wd	Write Ws to low program word	1	2

Note: When the optional {,WREG} operand is specified, the destination of the instruction is WREG. When {,WREG} is not specified, the destination of the instruction is the file register f.

B.2. Instrucciones matemáticas

Assembly	Syntax	Description	Words	Cycles
ADD	f {,WREG}	Destination = f + WREG	1	1
ADD	#lit10,Wn	Wn = lit10 + Wn	1	1
ADD	Wb,#lit5,Wd	Wd = Wb + lit5	1	1
ADD	Wb,Ws,Wd	Wd = Wb + Ws	1	1
ADDC	f {,WREG}	Destination = f + WREG + (C)	1	1
ADDC	#lit10,Wn	Wn = lit10 + Wn + (C)	1	1
ADDC	Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	1	1
ADDC	Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1
DAW.B	Wn	Wn = decimal adjust Wn	1	1
DEC	f {,WREG}	Destination = f – 1	1	1
DEC	Ws,Wd	Wd = Ws – 1	1	1
DEC2	f {,WREG}	Destination = f – 2	1	1
DEC2	Ws,Wd	Wd = Ws – 2	1	1
DIV.S	Wm,Wn	Signed 16/16-bit integer divide*	1	18
DIV.SD	Wm,Wn	Signed 32/16-bit integer divide*	1	18
DIV.U	Wm,Wn	Unsigned 16/16-bit integer divide*	1	18
DIV.UD	Wm,Wn	Unsigned 32/16-bit integer divide*	1	18
DIVF	Wm,Wn	Signed 16/16-bit fractional divide*	1	18
INC	f {,WREG}	Destination = f + 1	1	1
INC	Ws,Wd	Wd = Ws + 1	1	1
INC2	f {,WREG}	Destination = f + 2	1	1
INC2	Ws,Wd	Wd = Ws + 2	1	1
MUL	f	W3:W2 = f * WREG	1	1
MUL.SS	Wb,Ws,Wnd	{Wnd+1,Wnd} = sign(Wb) * sign(Ws)	1	1
MUL.SU	Wb,#lit5,Wnd	{Wnd+1,Wnd} = sign(Wb) * unsign(lit5)	1	1
MUL.SU	Wb,Ws,Wnd	{Wnd+1,Wnd} = sign(Wb) * unsign(Ws)	1	1
MUL.US	Wb,Ws,Wnd	{Wnd+1,Wnd} = unsign(Wb) * sign(Ws)	1	1
MUL.UU	Wb,#lit5,Wnd	{Wnd+1,Wnd} = unsign(Wb) * unsign(lit5)	1	1
MUL.UU	Wb,Ws,Wnd	{Wnd+1,Wnd} = unsign(Wb) * unsign(Ws)	1	1
SE	Ws,Wnd	Wnd = sign-extended Ws	1	1
SUB	f {,WREG}	Destination = f – WREG	1	1
SUB	#lit10, Wn	Wn = Wn – lit10	1	1
SUB	Wb,#lit5,Wd	Wd = Wb – lit5	1	1
SUB	Wb,Ws,Wd	Wd = Wb – Ws	1	1
SUBB	f {,WREG}	Destination = f – WREG – (C)	1	1
SUBB	#lit10, Wn	Wn = Wn – lit10 – (C)	1	1
SUBB	Wb,#lit5,Wd	Wd = Wb – lit5 – ($\overline{C}$)	1	1
SUBB	Wb,Ws,Wd	Wd = Wb – Ws – ($\overline{C}$)	1	1
SUBBR	f {,WREG}	Destination = WREG – f – ($\overline{C}$)	1	1
SUBBR	Wb,#lit5,Wd	Wd = lit5 – Wb – ($\overline{C}$)	1	1
SUBBR	Wb,Ws,Wd	Wd = Ws – Wb – ($\overline{C}$)	1	1
SUBR	f {,WREG}	Destination = WREG – f	1	1
SUBR	Wb,#lit5,Wd	Wd = lit5 – Wb	1	1
SUBR	Wb,Ws,Wd	Wd = Ws – Wb	1	1
ZE	Ws,Wnd	Wnd = zero-extended Ws	1	1

* Divide instructions are interruptible on a cycle-by-cycle basis. Also, divide instructions must be accompanied by a REPEAT instruction, which adds 1 extra cycle.

B.3. Instrucciones lógicas

Assembly	Syntax	Description	Words	Cycles
AND	f {,WREG}	Destination = f .AND. WREG	1	1
AND	#lit10,Wn	Wn = lit10 .AND. Wn	1	1
AND	Wb,#lit5,Wd	Wd = Wb .AND. lit5	1	1
AND	Wb,Ws,Wd	Wd = Wb .AND. Ws	1	1
CLR	f	f = 0x0000	1	1
CLR	WREG	WREG = 0x0000	1	1
CLR	Wd	Wd = 0x0000	1	1
COM	f {,WREG}	Destination = $\overline{f}$	1	1
COM	Ws,Wd	Wd = $\overline{Ws}$	1	1
IOR	f {,WREG}	Destination = f .IOR. WREG	1	1
IOR	#lit10,Wn	Wn = lit10 .IOR. Wn	1	1
IOR	Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	1
IOR	Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	1
NEG	f {,WREG}	Destination = $\overline{f}$ + 1	1	1
NEG	Ws,Wd	Wd = $\overline{Ws}$ + 1	1	1
SETM	f	f = 0xFFFF	1	1
SETM	WREG	WREG = 0xFFFF	1	1
SETM	Wd	Wd = 0xFFFF	1	1
XOR	f {,WREG}	Destination = f .XOR. WREG	1	1
XOR	#lit10,Wn	Wn = lit10 .XOR. Wn	1	1
XOR	Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1
XOR	Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1

Note: When the optional {,WREG} operand is specified, the destination of the instruction is WREG. When {,WREG} is not specified, the destination of the instruction is the file register f.

B.4. Instrucciones de rotación y desplazamiento

Assembly	Syntax	Description	Words	Cycles
ASR	f {,WREG}	Destination = arithmetic right shift f	1	1
ASR	Ws,Wd	Wd = arithmetic right shift Ws	1	1
ASR	Wb,#lit4,Wnd	Wnd = arithmetic right shift Wb by lit4	1	1
ASR	Wb,Wns,Wnd	Wnd = arithmetic right shift Wb by Wns	1	1
LSR	f {,WREG}	Destination = logical right shift f	1	1
LSR	Ws,Wd	Wd = logical right shift Ws	1	1
LSR	Wb,#lit4,Wnd	Wnd = logical right shift Wb by lit4	1	1
LSR	Wb,Wns,Wnd	Wnd = logical right shift Wb by Wns	1	1
RLC	f {,WREG}	Destination = rotate left through Carry f	1	1
RLC	Ws,Wd	Wd = rotate left through Carry Ws	1	1
RLNC	f {,WREG}	Destination = rotate left (no Carry) f	1	1
RLNC	Ws,Wd	Wd = rotate left (no Carry) Ws	1	1
RRC	f {,WREG}	Destination = rotate right through Carry f	1	1
RRC	Ws,Wd	Wd = rotate right through Carry Ws	1	1
RRNC	f {,WREG}	Destination = rotate right (no Carry) f	1	1
RRNC	Ws,Wd	Wd = rotate right (no Carry) Ws	1	1
SL	f {,WREG}	Destination = left shift f	1	1
SL	Ws,Wd	Wd = left shift Ws	1	1
SL	Wb,#lit4,Wnd	Wnd = left shift Wb by lit4	1	1
SL	Wb,Wns,Wnd	Wnd = left shift Wb by Wns	1	1

Note: When the optional {,WREG} operand is specified, the destination of the instruction is WREG. When {,WREG} is not specified, the destination of the instruction is the file register f.

B.5. Instrucciones de bit

Assembly	Syntax	Description	Words	Cycles
BCLR	f,#bit4	Bit clear f	1	1
BCLR	Ws,#bit4	Bit clear Ws	1	1
BSET	f,#bit4	Bit set f	1	1
BSET	Ws,#bit4	Bit set Ws	1	1
BSW.C	Ws,Wb	Write C bit to Ws<Wb>	1	1
BSW.Z	Ws,Wb	Write $\overline{SZ}$ bit to Ws<Wb>	1	1
BTG	f,#bit4	Bit toggle f	1	1
BTG	Ws,#bit4	Bit toggle Ws	1	1
BTST	f,#bit4	Bit test f	1	1
BTST.C	Ws,#bit4	Bit test Ws to C	1	1
BTST.Z	Ws,#bit4	Bit test Ws to SZ	1	1
BTST.C	Ws,Wb	Bit test Ws<Wb> to C	1	1
BTST.Z	Ws,Wb	Bit test Ws<Wb> to SZ	1	1
BTSTS	f,#bit4	Bit test f then set f	1	1
BTSTS.C	Ws,#bit4	Bit test Ws to C then set Ws	1	1
BTSTS.Z	Ws,#bit4	Bit test Ws to SZ then set Ws	1	1
FBCL	Ws,Wnd	Find bit change from left (MSb) side	1	1
FF1L	Ws,Wnd	Find first one from left (MSb) side	1	1
FF1R	Ws,Wnd	Find first one from right (LSb) side	1	1

Note: Bit positions are specified by bit4 (0:15) for word operations.

B.6. Instrucciones de comparación y salto

Assembly	Syntax	Description	Words	Cycles
BTSC	f,#bit4	Bit test f, skip if clear	1	1 (2 or 3)
BTSC	Ws,#bit4	Bit test Ws, skip if clear	1	1 (2 or 3)
BTSS	f,#bit4	Bit test f, skip if set	1	1 (2 or 3)
BTSS	Ws,#bit4	Bit test Ws, skip if set	1	1 (2 or 3)
CP	f	Compare (f – WREG)	1	1
CP	Wb,#lit5	Compare (Wb – lit5)	1	1
CP	Wb,Ws	Compare (Wb – Ws)	1	1
CP0	f	Compare (f – 0x0000)	1	1
CP0	Ws	Compare (Ws – 0x0000)	1	1
CPB	f	Compare with Borrow (f – WREG – $\overline{C}$)	1	1
CPB	Wb,#lit5	Compare with Borrow (Wb – lit5 – $\overline{C}$)	1	1
CPB	Wb,Ws	Compare with Borrow (Wb – Ws – $\overline{C}$)	1	1
CPSEQ	Wb,Wn	Compare Wb with Wn, Skip if Equal (Wb = Wn)	1	1 (2 or 3)
CPSGT	Wb,Wn	Signed Compare Wb with Wn, Skip if Greater Than (Wb > Wn)	1	1 (2 or 3)
CPSLT	Wb,Wn	Signed Compare Wb with Wn, Skip if Less Than (Wb < Wn)	1	1 (2 or 3)
CPSNE	Wb,Wn	Signed Compare Wb with Wn, Skip if Not Equal (Wb ≠ Wn)	1	1 (2 or 3)

Note 1: Bit positions are specified by bit4 (0:15) for word operations.

2: Conditional skip instructions execute in 1 cycle if the skip is not taken, 2 cycles if the skip is taken over a one-word instruction and 3 cycles if the skip is taken over a two-word instruction.

B.7. Instrucciones de flujo de programa

Assembly	Syntax	Description	Words	Cycles
BRA	Expr	Branch unconditionally	1	2
BRA	Wn	Computed branch	1	2
BRA	C,Expr	Branch if Carry (no Borrow)	1	1 (2)
BRA	GE,Expr	Branch if greater than or equal	1	1 (2)
BRA	GEU,Expr	Branch if unsigned greater than or equal	1	1 (2)
BRA	GT,Expr	Branch if greater than	1	1 (2)
BRA	GTU,Expr	Branch if unsigned greater than	1	1 (2)
BRA	LE,Expr	Branch if less than or equal	1	1 (2)
BRA	LEU,Expr	Branch if unsigned less than or equal	1	1 (2)
BRA	LT,Expr	Branch if less than	1	1 (2)
BRA	LTU,Expr	Branch if unsigned less than	1	1 (2)
BRA	N,Expr	Branch if Negative	1	1 (2)
BRA	NC,Expr	Branch if not Carry (Borrow)	1	1 (2)
BRA	NN,Expr	Branch if not Negative	1	1 (2)
BRA	NOV,Expr	Branch if not Overflow	1	1 (2)
BRA	NZ,Expr	Branch if not Zero	1	1 (2)
BRA	OA,Expr	Branch if Accumulator A Overflow	1	1 (2)
BRA	OB,Expr	Branch if Accumulator B Overflow	1	1 (2)
BRA	OV,Expr	Branch if Overflow	1	1 (2)
BRA	SA,Expr	Branch if Accumulator A Saturate	1	1 (2)
BRA	SB,Expr	Branch if Accumulator B Saturate	1	1 (2)
BRA	Z,Expr	Branch if Zero	1	1 (2)
CALL	Expr	Call subroutine	2	2
CALL	Wn	Call indirect subroutine	1	2
DO	#lit14,Expr	Do code through PC + Expr, (lit14 + 1) times	2	2
DO	Wn,Expr	Do code through PC + Expr, (Wn + 1) times	2	2
GOTO	Expr	Go to address	2	2
GOTO	Wn	Go to address indirectly	1	2
RCALL	Expr	Relative call	1	2
RCALL	Wn	Computed call	1	2
REPEAT	#lit14	Repeat next instruction (lit14 + 1) times	1	1
REPEAT	Wn	Repeat next instruction (Wn + 1) times	1	1
RETFIE		Return from interrupt enable	1	3 (2)
RETLW	#lit10,Wn	Return with lit10 in Wn	1	3 (2)
RETURN		Return from subroutine	1	3 (2)

Note 1: Conditional branch instructions execute in 1 cycle if the branch is not taken, or 2 cycles if the branch is taken.

2: RETURN normally executes in 3 cycles. However, it executes in 2 cycles if an interrupt is pending.

B.8. Instrucciones de acceso a pilas

Assembly	Syntax	Description	Words	Cycles
LNK	#lit14	Link Frame Pointer	1	1
POP	f	Pop TOS to f	1	1
POP	Wd	Pop TOS to Wd	1	1
POP.D	Wnd	Double pop from TOS to Wnd:Wnd + 1	1	2
POP.S		Pop shadow registers	1	1
PUSH	f	Push f to TOS	1	1
PUSH	Ws	Push Ws to TOS	1	1
PUSH.D	Wns	Push double Wns:Wns + 1 to TOS	1	2
PUSH.S		Push shadow registers	1	1
ULNK		Unlink Frame Pointer	1	1

B.9. Instrucciones de control

Assembly	Syntax	Description	Words	Cycles
CLRWDT		Clear Watchdog Timer	1	1
DISI	#lit14	Disable interrupts for (lit14 + 1) instruction cycles	1	1
NOP		No operation	1	1
NOPR		No operation	1	1
PWRSAV	#lit1	Enter Power-Saving mode lit1	1	1
RESET		Software device Reset	1	1

B.10. Instrucciones del motor DSP

Assembly	Syntax	Description	Words	Cycles
ADD	Acc	Add accumulators	1	1
ADD	Ws,#Slit4,Acc	16-bit signed add to Acc	1	1
CLR	Acc,Wx,Wxd,Wy,Wyd,AWB	Clear Acc	1	1
ED	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean distance (no accumulate)	1	1
EDAC	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean distance	1	1
LAC	Ws,#Slit4,Acc	Load Acc	1	1
MAC	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd,AWB	Multiply and accumulate	1	1
MAC	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square and accumulate	1	1
MOVSAC	Acc,Wx,Wxd,Wy,Wyd,AWB	Move Wx to Wxd and Wy to Wyd	1	1
MPY	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	Multiply Wn by Wm to Acc	1	1
MPY	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square to Acc	1	1
MPY.N	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wn by Wm) to Acc	1	1
MSC	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd,AWB	Multiply and subtract from Acc	1	1
NEG	Acc	Negate Acc	1	1
SAC	Acc,#Slit4,Wd	Store Acc	1	1
SAC.R	Acc,#Slit4,Wd	Store rounded Acc	1	1
SFTAC	Acc,#Slit6	Arithmetic shift Acc by Slit6	1	1
SFTAC	Acc,Wn	Arithmetic shift Acc by (Wn)	1	1
SUB	Acc	Subtract accumulators	1	1

Apéndice C

Definiciones para dsPICC del dsPIC30F4011

Listado C.1: Codigo/dspic30f4011.h

```
/*
 * Header file for the dspic30f4011 microcontroller.
 */

#ifndef __DSPIC30F4011_H
#define __DSPIC30F4011_H

//----------------------------------------------------------------------------
// Special Function Registers
//----------------------------------------------------------------------------

static volatile unsigned int  WREG0             @ 0x0;
static volatile unsigned int  WREG1             @ 0x2;
static volatile unsigned int  WREG2             @ 0x4;
static volatile unsigned int  WREG3             @ 0x6;
static volatile unsigned int  WREG4             @ 0x8;
static volatile unsigned int  WREG5             @ 0xA;
static volatile unsigned int  WREG6             @ 0xC;
static volatile unsigned int  WREG7             @ 0xE;
static volatile unsigned int  WREG8             @ 0x10;
static volatile unsigned int  WREG9             @ 0x12;
static volatile unsigned int  WREG10            @ 0x14;
static volatile unsigned int  WREG11            @ 0x16;
static volatile unsigned int  WREG12            @ 0x18;
static volatile unsigned int  WREG13            @ 0x1A;
static volatile unsigned int  WREG14            @ 0x1C;
static volatile unsigned int  WREG15            @ 0x1E;
static volatile unsigned int  SPLIM             @ 0x20;
static volatile unsigned int  ACCAL             @ 0x22;
static volatile unsigned int  ACCAH             @ 0x24;
static volatile unsigned int  ACCAU             @ 0x26;
static volatile bit           ACCAU0            @ ((unsigned)&ACCAU*8)+0;
static volatile bit           ACCAU1            @ ((unsigned)&ACCAU*8)+1;
static volatile bit           ACCAU2            @ ((unsigned)&ACCAU*8)+2;
static volatile bit           ACCAU3            @ ((unsigned)&ACCAU*8)+3;
static volatile bit           ACCAU4            @ ((unsigned)&ACCAU*8)+4;
static volatile bit           ACCAU5            @ ((unsigned)&ACCAU*8)+5;
static volatile bit           ACCAU6            @ ((unsigned)&ACCAU*8)+6;
static volatile bit           ACCAU7            @ ((unsigned)&ACCAU*8)+7;
static volatile bit           AA_SIGNEXT0       @ ((unsigned)&ACCAU*8)+8;
static volatile bit           AA_SIGNEXT1       @ ((unsigned)&ACCAU*8)+9;
static volatile bit           AA_SIGNEXT2       @ ((unsigned)&ACCAU*8)+10;
static volatile bit           AA_SIGNEXT3       @ ((unsigned)&ACCAU*8)+11;
static volatile bit           AA_SIGNEXT4       @ ((unsigned)&ACCAU*8)+12;
static volatile bit           AA_SIGNEXT5       @ ((unsigned)&ACCAU*8)+13;
static volatile bit           AA_SIGNEXT6       @ ((unsigned)&ACCAU*8)+14;
static volatile bit           AA_SIGNEXT7       @ ((unsigned)&ACCAU*8)+15;
/* Microchip compatible bit field */
static volatile struct {
       volatile unsigned      SIGNEXT           : 8;
       volatile unsigned      ACCAU             : 8;
} ACCAUbits @ 0x26;
```

```
    static volatile unsigned int   ACCBL                @ 0x28;
    static volatile unsigned int   ACCBH                @ 0x2A;
    static volatile unsigned int   ACCBU                @ 0x2C;
57  static volatile bit            ACCBU0               @ ((unsigned)&ACCBU*8)+0;
    static volatile bit            ACCBU1               @ ((unsigned)&ACCBU*8)+1;
    static volatile bit            ACCBU2               @ ((unsigned)&ACCBU*8)+2;
    static volatile bit            ACCBU3               @ ((unsigned)&ACCBU*8)+3;
    static volatile bit            ACCBU4               @ ((unsigned)&ACCBU*8)+4;
62  static volatile bit            ACCBU5               @ ((unsigned)&ACCBU*8)+5;
    static volatile bit            ACCBU6               @ ((unsigned)&ACCBU*8)+6;
    static volatile bit            ACCBU7               @ ((unsigned)&ACCBU*8)+7;
    static volatile bit            AB_SIGNEXT0          @ ((unsigned)&ACCBU*8)+8;
    static volatile bit            AB_SIGNEXT1          @ ((unsigned)&ACCBU*8)+9;
67  static volatile bit            AB_SIGNEXT2          @ ((unsigned)&ACCBU*8)+10;
    static volatile bit            AB_SIGNEXT3          @ ((unsigned)&ACCBU*8)+11;
    static volatile bit            AB_SIGNEXT4          @ ((unsigned)&ACCBU*8)+12;
    static volatile bit            AB_SIGNEXT5          @ ((unsigned)&ACCBU*8)+13;
    static volatile bit            AB_SIGNEXT6          @ ((unsigned)&ACCBU*8)+14;
72  static volatile bit            AB_SIGNEXT7          @ ((unsigned)&ACCBU*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
           volatile unsigned       SIGNEXT              : 8;
           volatile unsigned       ACCBU                : 8;
77  } ACCBUbits @ 0x2C;

    static volatile unsigned int   PCL                  @ 0x2E;
    static volatile unsigned int   PCH                  @ 0x30;
    static volatile unsigned int   TBLPAG               @ 0x32;
82  static volatile unsigned int   PSVPAG               @ 0x34;
    static volatile unsigned int   RCOUNT               @ 0x36;
    static volatile unsigned int   DCOUNT               @ 0x38;
    static volatile unsigned int   DOSTARTL             @ 0x3A;
    static volatile unsigned int   DOSTARTH             @ 0x3C;
87  static volatile unsigned int   DOENDL               @ 0x3E;
    static volatile unsigned int   DOENDH               @ 0x40;
    static volatile unsigned int   SR                   @ 0x42;
    static volatile bit            C                    @ ((unsigned)&SR*8)+0;
    static volatile bit            Z                    @ ((unsigned)&SR*8)+1;
92  static volatile bit            OV                   @ ((unsigned)&SR*8)+2;
    static volatile bit            N                    @ ((unsigned)&SR*8)+3;
    static volatile bit            RA                   @ ((unsigned)&SR*8)+4;
    static volatile bit            IPL0                 @ ((unsigned)&SR*8)+5;
    static volatile bit            IPL1                 @ ((unsigned)&SR*8)+6;
97  static volatile bit            IPL2                 @ ((unsigned)&SR*8)+7;
    static volatile bit            DC                   @ ((unsigned)&SR*8)+8;
    static volatile bit            DA                   @ ((unsigned)&SR*8)+9;
    static volatile bit            SAB                  @ ((unsigned)&SR*8)+10;
    static volatile bit            OAB                  @ ((unsigned)&SR*8)+11;
102 static volatile bit            SB                   @ ((unsigned)&SR*8)+12;
    static volatile bit            SA                   @ ((unsigned)&SR*8)+13;
    static volatile bit            OB                   @ ((unsigned)&SR*8)+14;
    static volatile bit            OA                   @ ((unsigned)&SR*8)+15;
    /* Microchip compatible bit field */
107 static volatile struct {
           volatile unsigned       OA                   : 1;
           volatile unsigned       OB                   : 1;
           volatile unsigned       SA                   : 1;
           volatile unsigned       SB                   : 1;
112        volatile unsigned       OAB                  : 1;
           volatile unsigned       SAB                  : 1;
           volatile unsigned       DA                   : 1;
           volatile unsigned       DC                   : 1;
           volatile unsigned       IPL                  : 3;
117        volatile unsigned       RA                   : 1;
           volatile unsigned       N                    : 1;
           volatile unsigned       OV                   : 1;
           volatile unsigned       Z                    : 1;
           volatile unsigned       C                    : 1;
122 } SRbits @ 0x42;

    static volatile unsigned int   CORCON               @ 0x44;
    static volatile bit            IF                   @ ((unsigned)&CORCON*8)+0;
    static volatile bit            RND                  @ ((unsigned)&CORCON*8)+1;
127 static volatile bit            PSV                  @ ((unsigned)&CORCON*8)+2;
    static volatile bit            IPL3                 @ ((unsigned)&CORCON*8)+3;
    static volatile bit            ACCSAT               @ ((unsigned)&CORCON*8)+4;
    static volatile bit            SATDW                @ ((unsigned)&CORCON*8)+5;
```

```
    static volatile bit           SATB              @ ((unsigned)&CORCON*8)+6;
132 static volatile bit           SATA              @ ((unsigned)&CORCON*8)+7;
    static volatile bit           DL0               @ ((unsigned)&CORCON*8)+8;
    static volatile bit           DL1               @ ((unsigned)&CORCON*8)+9;
    static volatile bit           DL2               @ ((unsigned)&CORCON*8)+10;
    static volatile bit           EDT               @ ((unsigned)&CORCON*8)+11;
137 static volatile bit           US                @ ((unsigned)&CORCON*8)+12;
    /* Microchip compatible bit field */
    static volatile struct {
                     unsigned                       : 1;
                     unsigned                       : 1;
142                  unsigned                       : 1;
            volatile unsigned     US                : 1;
            volatile unsigned     EDT               : 1;
            volatile unsigned     DL                : 3;
            volatile unsigned     SATA              : 1;
147         volatile unsigned     SATB              : 1;
            volatile unsigned     SATDW             : 1;
            volatile unsigned     ACCSAT            : 1;
            volatile unsigned     IPL3              : 1;
            volatile unsigned     PSV               : 1;
152         volatile unsigned     RND               : 1;
            volatile unsigned     IF                : 1;
    } CORCONbits @ 0x44;

    static volatile unsigned int  MODCON            @ 0x46;
157 static volatile bit           XWM0              @ ((unsigned)&MODCON*8)+0;
    static volatile bit           XWM1              @ ((unsigned)&MODCON*8)+1;
    static volatile bit           XWM2              @ ((unsigned)&MODCON*8)+2;
    static volatile bit           XWM3              @ ((unsigned)&MODCON*8)+3;
    static volatile bit           YWM0              @ ((unsigned)&MODCON*8)+4;
162 static volatile bit           YWM1              @ ((unsigned)&MODCON*8)+5;
    static volatile bit           YWM2              @ ((unsigned)&MODCON*8)+6;
    static volatile bit           YWM3              @ ((unsigned)&MODCON*8)+7;
    static volatile bit           BWM0              @ ((unsigned)&MODCON*8)+8;
    static volatile bit           BWM1              @ ((unsigned)&MODCON*8)+9;
167 static volatile bit           BWM2              @ ((unsigned)&MODCON*8)+10;
    static volatile bit           BWM3              @ ((unsigned)&MODCON*8)+11;
    static volatile bit           YMODEN            @ ((unsigned)&MODCON*8)+14;
    static volatile bit           XMODEN            @ ((unsigned)&MODCON*8)+15;
    /* Microchip compatible bit field */
172 static volatile struct {
            volatile unsigned     XMODEN            : 1;
            volatile unsigned     YMODEN            : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
177         volatile unsigned     BWM               : 4;
            volatile unsigned     YWM               : 4;
            volatile unsigned     XWM               : 4;
    } MODCONbits @ 0x46;

182 static volatile unsigned int  XMODSRT           @ 0x48;
    static volatile unsigned int  XMODEND           @ 0x4A;
    static volatile unsigned int  YMODSRT           @ 0x4C;
    static volatile unsigned int  YMODEND           @ 0x4E;
    static volatile unsigned int  XBREV             @ 0x50;
187 static volatile bit           XB0               @ ((unsigned)&XBREV*8)+0;
    static volatile bit           XB1               @ ((unsigned)&XBREV*8)+1;
    static volatile bit           XB2               @ ((unsigned)&XBREV*8)+2;
    static volatile bit           XB3               @ ((unsigned)&XBREV*8)+3;
    static volatile bit           XB4               @ ((unsigned)&XBREV*8)+4;
192 static volatile bit           XB5               @ ((unsigned)&XBREV*8)+5;
    static volatile bit           XB6               @ ((unsigned)&XBREV*8)+6;
    static volatile bit           XB7               @ ((unsigned)&XBREV*8)+7;
    static volatile bit           XB8               @ ((unsigned)&XBREV*8)+8;
    static volatile bit           XB9               @ ((unsigned)&XBREV*8)+9;
197 static volatile bit           XB10              @ ((unsigned)&XBREV*8)+10;
    static volatile bit           XB11              @ ((unsigned)&XBREV*8)+11;
    static volatile bit           XB12              @ ((unsigned)&XBREV*8)+12;
    static volatile bit           XB13              @ ((unsigned)&XBREV*8)+13;
    static volatile bit           XB14              @ ((unsigned)&XBREV*8)+14;
202 static volatile bit           BREN              @ ((unsigned)&XBREV*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned     BREN              : 1;
            volatile unsigned     XB                : 15;
207 } XBREVbits @ 0x50;
```

```
    static volatile unsigned int   DISICNT             @ 0x52;
    static volatile unsigned int   INTCON1             @ 0x80;
    static volatile bit            OSCFAIL             @ ((unsigned)&INTCON1*8)+1;
212 static volatile bit            STKERR              @ ((unsigned)&INTCON1*8)+2;
    static volatile bit            ADDRERR             @ ((unsigned)&INTCON1*8)+3;
    static volatile bit            MATHERR             @ ((unsigned)&INTCON1*8)+4;
    static volatile bit            COVTE               @ ((unsigned)&INTCON1*8)+8;
    static volatile bit            OVBTE               @ ((unsigned)&INTCON1*8)+9;
217 static volatile bit            OVATE               @ ((unsigned)&INTCON1*8)+10;
    static volatile bit            NSTDIS              @ ((unsigned)&INTCON1*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
           volatile unsigned       NSTDIS              : 1;
222                unsigned                           : 1;
                   unsigned                           : 1;
                   unsigned                           : 1;
                   unsigned                           : 1;
           volatile unsigned       OVATE               : 1;
227        volatile unsigned       OVBTE               : 1;
           volatile unsigned       COVTE               : 1;
                   unsigned                           : 1;
                   unsigned                           : 1;
                   unsigned                           : 1;
232        volatile unsigned       MATHERR             : 1;
           volatile unsigned       ADDRERR             : 1;
           volatile unsigned       STKERR              : 1;
           volatile unsigned       OSCFAIL             : 1;
                   unsigned                           : 1;
237 } INTCON1bits @ 0x80;

    static volatile unsigned int   INTCON2             @ 0x82;
    static volatile bit            INT0EP              @ ((unsigned)&INTCON2*8)+0;
    static volatile bit            INT1EP              @ ((unsigned)&INTCON2*8)+1;
242 static volatile bit            INT2EP              @ ((unsigned)&INTCON2*8)+2;
    static volatile bit            DISI                @ ((unsigned)&INTCON2*8)+14;
    static volatile bit            ALTIVT              @ ((unsigned)&INTCON2*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
247        volatile unsigned       ALTIVT              : 1;
           volatile unsigned       DISI                : 1;
                   unsigned                           : 1;
                   unsigned                           : 1;
                   unsigned                           : 1;
252                unsigned                           : 1;
                   unsigned                           : 1;
                   unsigned                           : 1;
                   unsigned                           : 1;
                   unsigned                           : 1;
257                unsigned                           : 1;
                   unsigned                           : 1;
                   unsigned                           : 1;
           volatile unsigned       INT2EP              : 1;
           volatile unsigned       INT1EP              : 1;
262        volatile unsigned       INT0EP              : 1;
    } INTCON2bits @ 0x82;

    static volatile unsigned int   IFS0                @ 0x84;
    static volatile bit            INT0IF              @ ((unsigned)&IFS0*8)+0;
267 static volatile bit            IC1IF               @ ((unsigned)&IFS0*8)+1;
    static volatile bit            OC1IF               @ ((unsigned)&IFS0*8)+2;
    static volatile bit            T1IF                @ ((unsigned)&IFS0*8)+3;
    static volatile bit            IC2IF               @ ((unsigned)&IFS0*8)+4;
    static volatile bit            OC2IF               @ ((unsigned)&IFS0*8)+5;
272 static volatile bit            T2IF                @ ((unsigned)&IFS0*8)+6;
    static volatile bit            T3IF                @ ((unsigned)&IFS0*8)+7;
    static volatile bit            SPI1IF              @ ((unsigned)&IFS0*8)+8;
    static volatile bit            U1RXIF              @ ((unsigned)&IFS0*8)+9;
    static volatile bit            U1TXIF              @ ((unsigned)&IFS0*8)+10;
277 static volatile bit            ADIF                @ ((unsigned)&IFS0*8)+11;
    static volatile bit            NVMIF               @ ((unsigned)&IFS0*8)+12;
    static volatile bit            I2CIF               @ ((unsigned)&IFS0*8)+13;
    static volatile bit            BCLIF               @ ((unsigned)&IFS0*8)+14;
    static volatile bit            CNIF                @ ((unsigned)&IFS0*8)+15;
282 /* Microchip compatible bit field */
    static volatile struct {
           volatile unsigned       CNIF                : 1;
           volatile unsigned       BCLIF               : 1;
           volatile unsigned       I2CIF               : 1;
```

```
287         volatile unsigned       NVMIF               : 1;
            volatile unsigned       ADIF                : 1;
            volatile unsigned       U1TXIF              : 1;
            volatile unsigned       U1RXIF              : 1;
            volatile unsigned       SPI1IF              : 1;
292         volatile unsigned       T3IF                : 1;
            volatile unsigned       T2IF                : 1;
            volatile unsigned       OC2IF               : 1;
            volatile unsigned       IC2IF               : 1;
            volatile unsigned       T1IF                : 1;
297         volatile unsigned       OC1IF               : 1;
            volatile unsigned       IC1IF               : 1;
            volatile unsigned       INT0IF              : 1;
    } IFS0bits @ 0x84;

302 static volatile unsigned int  IFS1                @ 0x86;
    static volatile bit             INT1IF              @ ((unsigned)&IFS1*8)+0;
    static volatile bit             IC7IF               @ ((unsigned)&IFS1*8)+1;
    static volatile bit             IC8IF               @ ((unsigned)&IFS1*8)+2;
    static volatile bit             OC3IF               @ ((unsigned)&IFS1*8)+3;
307 static volatile bit             OC4IF               @ ((unsigned)&IFS1*8)+4;
    static volatile bit             T4IF                @ ((unsigned)&IFS1*8)+5;
    static volatile bit             T5IF                @ ((unsigned)&IFS1*8)+6;
    static volatile bit             INT2IF              @ ((unsigned)&IFS1*8)+7;
    static volatile bit             U2RXIF              @ ((unsigned)&IFS1*8)+8;
312 static volatile bit             U2TXIF              @ ((unsigned)&IFS1*8)+9;
    static volatile bit             C1IF                @ ((unsigned)&IFS1*8)+11;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                            : 1;
317                 unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
            volatile unsigned       C1IF                : 1;
                    unsigned                            : 1;
322         volatile unsigned       U2TXIF              : 1;
            volatile unsigned       U2RXIF              : 1;
            volatile unsigned       INT2IF              : 1;
            volatile unsigned       T5IF                : 1;
            volatile unsigned       T4IF                : 1;
327         volatile unsigned       OC4IF               : 1;
            volatile unsigned       OC3IF               : 1;
            volatile unsigned       IC8IF               : 1;
            volatile unsigned       IC7IF               : 1;
            volatile unsigned       INT1IF              : 1;
332 } IFS1bits @ 0x86;

    static volatile unsigned int  IFS2                @ 0x88;
    static volatile bit             PWMIF               @ ((unsigned)&IFS2*8)+7;
    static volatile bit             QEIIF               @ ((unsigned)&IFS2*8)+8;
337 static volatile bit             FLTAIF              @ ((unsigned)&IFS2*8)+11;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                            : 1;
                    unsigned                            : 1;
342                 unsigned                            : 1;
                    unsigned                            : 1;
            volatile unsigned       FLTAIF              : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
347         volatile unsigned       QEIIF               : 1;
            volatile unsigned       PWMIF               : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
352                 unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
    } IFS2bits @ 0x88;
357
    static volatile unsigned int  IEC0                @ 0x8C;
    static volatile bit             INT0IE              @ ((unsigned)&IEC0*8)+0;
    static volatile bit             IC1IE               @ ((unsigned)&IEC0*8)+1;
    static volatile bit             OC1IE               @ ((unsigned)&IEC0*8)+2;
362 static volatile bit             T1IE                @ ((unsigned)&IEC0*8)+3;
    static volatile bit             IC2IE               @ ((unsigned)&IEC0*8)+4;
    static volatile bit             OC2IE               @ ((unsigned)&IEC0*8)+5;
```

```
    static volatile bit             T2IE                @ ((unsigned)&IEC0*8)+6;
    static volatile bit             T3IE                @ ((unsigned)&IEC0*8)+7;
367 static volatile bit             SPI1IE              @ ((unsigned)&IEC0*8)+8;
    static volatile bit             U1RXIE              @ ((unsigned)&IEC0*8)+9;
    static volatile bit             U1TXIE              @ ((unsigned)&IEC0*8)+10;
    static volatile bit             ADIE                @ ((unsigned)&IEC0*8)+11;
    static volatile bit             NVMIE               @ ((unsigned)&IEC0*8)+12;
372 static volatile bit             I2CIE               @ ((unsigned)&IEC0*8)+13;
    static volatile bit             BCLIE               @ ((unsigned)&IEC0*8)+14;
    static volatile bit             CNIE                @ ((unsigned)&IEC0*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
377         volatile unsigned       CNIE                : 1;
            volatile unsigned       BCLIE               : 1;
            volatile unsigned       I2CIE               : 1;
            volatile unsigned       NVMIE               : 1;
            volatile unsigned       ADIE                : 1;
382         volatile unsigned       U1TXIE              : 1;
            volatile unsigned       U1RXIE              : 1;
            volatile unsigned       SPI1IE              : 1;
            volatile unsigned       T3IE                : 1;
            volatile unsigned       T2IE                : 1;
387         volatile unsigned       OC2IE               : 1;
            volatile unsigned       IC2IE               : 1;
            volatile unsigned       T1IE                : 1;
            volatile unsigned       OC1IE               : 1;
            volatile unsigned       IC1IE               : 1;
392         volatile unsigned       INT0IE              : 1;
    } IEC0bits @ 0x8C;

    static volatile unsigned int    IEC1                @ 0x8E;
    static volatile bit             INT1IE              @ ((unsigned)&IEC1*8)+0;
397 static volatile bit             IC7IE               @ ((unsigned)&IEC1*8)+1;
    static volatile bit             IC8IE               @ ((unsigned)&IEC1*8)+2;
    static volatile bit             OC3IE               @ ((unsigned)&IEC1*8)+3;
    static volatile bit             OC4IE               @ ((unsigned)&IEC1*8)+4;
    static volatile bit             T4IE                @ ((unsigned)&IEC1*8)+5;
402 static volatile bit             T5IE                @ ((unsigned)&IEC1*8)+6;
    static volatile bit             INT2IE              @ ((unsigned)&IEC1*8)+7;
    static volatile bit             U2RXIE              @ ((unsigned)&IEC1*8)+8;
    static volatile bit             U2TXIE              @ ((unsigned)&IEC1*8)+9;
    static volatile bit             C1IE                @ ((unsigned)&IEC1*8)+11;
407 /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
412                 unsigned                            : 1;
            volatile unsigned       C1IE                : 1;
                    unsigned                            : 1;
            volatile unsigned       U2TXIE              : 1;
            volatile unsigned       U2RXIE              : 1;
417         volatile unsigned       INT2IE              : 1;
            volatile unsigned       T5IE                : 1;
            volatile unsigned       T4IE                : 1;
            volatile unsigned       OC4IE               : 1;
            volatile unsigned       OC3IE               : 1;
422         volatile unsigned       IC8IE               : 1;
            volatile unsigned       IC7IE               : 1;
            volatile unsigned       INT1IE              : 1;
    } IEC1bits @ 0x8E;

427 static volatile unsigned int    IEC2                @ 0x90;
    static volatile bit             PWMIE               @ ((unsigned)&IEC2*8)+7;
    static volatile bit             QEIIE               @ ((unsigned)&IEC2*8)+8;
    static volatile bit             FLTAIE              @ ((unsigned)&IEC2*8)+11;
    /* Microchip compatible bit field */
432 static volatile struct {
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
437         volatile unsigned       FLTAIE              : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
            volatile unsigned       QEIIE               : 1;
            volatile unsigned       PWMIE               : 1;
442                 unsigned                            : 1;
```

```
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
447             unsigned                        : 1;
                unsigned                        : 1;
} IEC2bits @ 0x90;

static volatile unsigned int   IPC0             @ 0x94;
452 static volatile bit        INT0IP0          @ ((unsigned)&IPC0*8)+0;
static volatile bit            INT0IP1          @ ((unsigned)&IPC0*8)+1;
static volatile bit            INT0IP2          @ ((unsigned)&IPC0*8)+2;
static volatile bit            IC1IP0           @ ((unsigned)&IPC0*8)+4;
static volatile bit            IC1IP1           @ ((unsigned)&IPC0*8)+5;
457 static volatile bit        IC1IP2           @ ((unsigned)&IPC0*8)+6;
static volatile bit            OC1IP0           @ ((unsigned)&IPC0*8)+8;
static volatile bit            OC1IP1           @ ((unsigned)&IPC0*8)+9;
static volatile bit            OC1IP2           @ ((unsigned)&IPC0*8)+10;
static volatile bit            T1IP0            @ ((unsigned)&IPC0*8)+12;
462 static volatile bit        T1IP1            @ ((unsigned)&IPC0*8)+13;
static volatile bit            T1IP2            @ ((unsigned)&IPC0*8)+14;
/* Microchip compatible bit field */
static volatile struct {
                unsigned                        : 1;
467         volatile unsigned      T1IP         : 3;
                unsigned                        : 1;
            volatile unsigned      OC1IP        : 3;
                unsigned                        : 1;
            volatile unsigned      IC1IP        : 3;
472             unsigned                        : 1;
            volatile unsigned      INT0IP       : 3;
} IPC0bits @ 0x94;

static volatile unsigned int   IPC1             @ 0x96;
477 static volatile bit         IC2IP0          @ ((unsigned)&IPC1*8)+0;
static volatile bit            IC2IP1           @ ((unsigned)&IPC1*8)+1;
static volatile bit            IC2IP2           @ ((unsigned)&IPC1*8)+2;
static volatile bit            OC2IP0           @ ((unsigned)&IPC1*8)+4;
static volatile bit            OC2IP1           @ ((unsigned)&IPC1*8)+5;
482 static volatile bit         OC2IP2           @ ((unsigned)&IPC1*8)+6;
static volatile bit            T2IP0            @ ((unsigned)&IPC1*8)+8;
static volatile bit            T2IP1            @ ((unsigned)&IPC1*8)+9;
static volatile bit            T2IP2            @ ((unsigned)&IPC1*8)+10;
static volatile bit            T3IP0            @ ((unsigned)&IPC1*8)+12;
487 static volatile bit         T3IP1            @ ((unsigned)&IPC1*8)+13;
static volatile bit            T3IP2            @ ((unsigned)&IPC1*8)+14;
/* Microchip compatible bit field */
static volatile struct {
                unsigned                        : 1;
492         volatile unsigned      T3IP         : 3;
                unsigned                        : 1;
            volatile unsigned      T2IP         : 3;
                unsigned                        : 1;
            volatile unsigned      OC2IP        : 3;
497             unsigned                        : 1;
            volatile unsigned      IC2IP        : 3;
} IPC1bits @ 0x96;

static volatile unsigned int   IPC2             @ 0x98;
502 static volatile bit         SPI1IP0         @ ((unsigned)&IPC2*8)+0;
static volatile bit            SPI1IP1          @ ((unsigned)&IPC2*8)+1;
static volatile bit            SPI1IP2          @ ((unsigned)&IPC2*8)+2;
static volatile bit            U1RXIP0          @ ((unsigned)&IPC2*8)+4;
static volatile bit            U1RXIP1          @ ((unsigned)&IPC2*8)+5;
507 static volatile bit         U1RXIP2          @ ((unsigned)&IPC2*8)+6;
static volatile bit            U1TXIP0          @ ((unsigned)&IPC2*8)+8;
static volatile bit            U1TXIP1          @ ((unsigned)&IPC2*8)+9;
static volatile bit            U1TXIP2          @ ((unsigned)&IPC2*8)+10;
static volatile bit            ADIP0            @ ((unsigned)&IPC2*8)+12;
512 static volatile bit         ADIP1            @ ((unsigned)&IPC2*8)+13;
static volatile bit            ADIP2            @ ((unsigned)&IPC2*8)+14;
/* Microchip compatible bit field */
static volatile struct {
                unsigned                        : 1;
517         volatile unsigned      ADIP         : 3;
                unsigned                        : 1;
            volatile unsigned      U1TXIP       : 3;
                unsigned                        : 1;
```

```
            volatile unsigned       U1RXIP              : 3;
522                 unsigned                            : 1;
            volatile unsigned       SPI1IP              : 3;
    } IPC2bits @ 0x98;

    static volatile unsigned int   IPC3                 @ 0x9A;
527 static volatile bit             NVMIP0              @ ((unsigned)&IPC3*8)+0;
    static volatile bit             NVMIP1              @ ((unsigned)&IPC3*8)+1;
    static volatile bit             NVMIP2              @ ((unsigned)&IPC3*8)+2;
    static volatile bit             I2CIP0              @ ((unsigned)&IPC3*8)+4;
    static volatile bit             I2CIP1              @ ((unsigned)&IPC3*8)+5;
532 static volatile bit             I2CIP2              @ ((unsigned)&IPC3*8)+6;
    static volatile bit             BCLIP0              @ ((unsigned)&IPC3*8)+8;
    static volatile bit             BCLIP1              @ ((unsigned)&IPC3*8)+9;
    static volatile bit             BCLIP2              @ ((unsigned)&IPC3*8)+10;
    static volatile bit             CNIP0               @ ((unsigned)&IPC3*8)+12;
537 static volatile bit             CNIP1               @ ((unsigned)&IPC3*8)+13;
    static volatile bit             CNIP2               @ ((unsigned)&IPC3*8)+14;
    /* Microchip compatible bit field */
    static volatile struct {
                        unsigned                        : 1;
542         volatile unsigned       CNIP                : 3;
                        unsigned                        : 1;
            volatile unsigned       BCLIP               : 3;
                        unsigned                        : 1;
            volatile unsigned       I2CIP               : 3;
547                 unsigned                            : 1;
            volatile unsigned       NVMIP               : 3;
    } IPC3bits @ 0x9A;

    static volatile unsigned int   IPC4                 @ 0x9C;
552 static volatile bit             INT1IP0             @ ((unsigned)&IPC4*8)+0;
    static volatile bit             INT1IP1             @ ((unsigned)&IPC4*8)+1;
    static volatile bit             INT1IP2             @ ((unsigned)&IPC4*8)+2;
    static volatile bit             IC7IP0              @ ((unsigned)&IPC4*8)+4;
    static volatile bit             IC7IP1              @ ((unsigned)&IPC4*8)+5;
557 static volatile bit             IC7IP2              @ ((unsigned)&IPC4*8)+6;
    static volatile bit             IC8IP0              @ ((unsigned)&IPC4*8)+8;
    static volatile bit             IC8IP1              @ ((unsigned)&IPC4*8)+9;
    static volatile bit             IC8IP2              @ ((unsigned)&IPC4*8)+10;
    static volatile bit             OC3IP0              @ ((unsigned)&IPC4*8)+12;
562 static volatile bit             OC3IP1              @ ((unsigned)&IPC4*8)+13;
    static volatile bit             OC3IP2              @ ((unsigned)&IPC4*8)+14;
    /* Microchip compatible bit field */
    static volatile struct {
                        unsigned                        : 1;
567         volatile unsigned       OC3IP               : 3;
                        unsigned                        : 1;
            volatile unsigned       IC8IP               : 3;
                        unsigned                        : 1;
            volatile unsigned       IC7IP               : 3;
572                 unsigned                            : 1;
            volatile unsigned       INT1IP              : 3;
    } IPC4bits @ 0x9C;

    static volatile unsigned int   IPC5                 @ 0x9E;
577 static volatile bit             OC4IP0              @ ((unsigned)&IPC5*8)+0;
    static volatile bit             OC4IP1              @ ((unsigned)&IPC5*8)+1;
    static volatile bit             OC4IP2              @ ((unsigned)&IPC5*8)+2;
    static volatile bit             T4IP0               @ ((unsigned)&IPC5*8)+4;
    static volatile bit             T4IP1               @ ((unsigned)&IPC5*8)+5;
582 static volatile bit             T4IP2               @ ((unsigned)&IPC5*8)+6;
    static volatile bit             T5IP0               @ ((unsigned)&IPC5*8)+8;
    static volatile bit             T5IP1               @ ((unsigned)&IPC5*8)+9;
    static volatile bit             T5IP2               @ ((unsigned)&IPC5*8)+10;
    static volatile bit             INT2IP0             @ ((unsigned)&IPC5*8)+12;
587 static volatile bit             INT2IP1             @ ((unsigned)&IPC5*8)+13;
    static volatile bit             INT2IP2             @ ((unsigned)&IPC5*8)+14;
    /* Microchip compatible bit field */
    static volatile struct {
                        unsigned                        : 1;
592         volatile unsigned       INT2IP              : 3;
                        unsigned                        : 1;
            volatile unsigned       T5IP                : 3;
                        unsigned                        : 1;
            volatile unsigned       T4IP                : 3;
597                 unsigned                            : 1;
            volatile unsigned       OC4IP               : 3;
```

```
} IPC5bits @ 0x9E;

static volatile unsigned int  IPC6              @ 0xA0;
static volatile bit           U2RXIP0           @ ((unsigned)&IPC6*8)+0;
static volatile bit           U2RXIP1           @ ((unsigned)&IPC6*8)+1;
static volatile bit           U2RXIP2           @ ((unsigned)&IPC6*8)+2;
static volatile bit           U2TXIP0           @ ((unsigned)&IPC6*8)+4;
static volatile bit           U2TXIP1           @ ((unsigned)&IPC6*8)+5;
static volatile bit           U2TXIP2           @ ((unsigned)&IPC6*8)+6;
static volatile bit           C1IP0             @ ((unsigned)&IPC6*8)+12;
static volatile bit           C1IP1             @ ((unsigned)&IPC6*8)+13;
static volatile bit           C1IP2             @ ((unsigned)&IPC6*8)+14;
/* Microchip compatible bit field */
static volatile struct {
                unsigned                        : 1;
        volatile unsigned       C1IP            : 3;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
        volatile unsigned       U2TXIP          : 3;
                unsigned                        : 1;
        volatile unsigned       U2RXIP          : 3;
} IPC6bits @ 0xA0;

static volatile unsigned int  IPC9              @ 0xA6;
static volatile bit           PWMIP0            @ ((unsigned)&IPC9*8)+12;
static volatile bit           PWMIP1            @ ((unsigned)&IPC9*8)+13;
static volatile bit           PWMIP2            @ ((unsigned)&IPC9*8)+14;
/* Microchip compatible bit field */
static volatile struct {
                unsigned                        : 1;
        volatile unsigned       PWMIP           : 3;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
} IPC9bits @ 0xA6;

static volatile unsigned int  IPC10             @ 0xA8;
static volatile bit           QEIIP0            @ ((unsigned)&IPC10*8)+0;
static volatile bit           QEIIP1            @ ((unsigned)&IPC10*8)+1;
static volatile bit           QEIIP2            @ ((unsigned)&IPC10*8)+2;
static volatile bit           FLTAIP0           @ ((unsigned)&IPC10*8)+12;
static volatile bit           FLTAIP1           @ ((unsigned)&IPC10*8)+13;
static volatile bit           FLTAIP2           @ ((unsigned)&IPC10*8)+14;
/* Microchip compatible bit field */
static volatile struct {
                unsigned                        : 1;
        volatile unsigned       FLTAIP          : 3;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
        volatile unsigned       QEIIP           : 3;
} IPC10bits @ 0xA8;

static volatile unsigned int  INTREG            @ 0xB0;
static volatile bit           VECNUM0           @ ((unsigned)&INTREG*8)+0;
static volatile bit           VECNUM1           @ ((unsigned)&INTREG*8)+1;
static volatile bit           VECNUM2           @ ((unsigned)&INTREG*8)+2;
static volatile bit           VECNUM3           @ ((unsigned)&INTREG*8)+3;
static volatile bit           VECNUM4           @ ((unsigned)&INTREG*8)+4;
static volatile bit           VECNUM5           @ ((unsigned)&INTREG*8)+5;
```

```
677 static volatile bit           ILR0              @ ((unsigned)&INTREG*8)+8;
    static volatile bit           ILR1              @ ((unsigned)&INTREG*8)+9;
    static volatile bit           ILR2              @ ((unsigned)&INTREG*8)+10;
    static volatile bit           ILR3              @ ((unsigned)&INTREG*8)+11;
    static volatile bit           TMODE             @ ((unsigned)&INTREG*8)+14;
682 static volatile bit           IRQTOCPU          @ ((unsigned)&INTREG*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
           volatile unsigned      IRQtoCPU          : 1;
           volatile unsigned      TMODE             : 1;
687                 unsigned                        : 1;
                    unsigned                        : 1;
           volatile unsigned      ILR               : 4;
                    unsigned                        : 1;
                    unsigned                        : 1;
692        volatile unsigned      VECNUM            : 6;
    } INTREGbits @ 0xB0;

    static volatile unsigned int  CNEN1             @ 0xC0;
    static volatile bit           CN0IE             @ ((unsigned)&CNEN1*8)+0;
697 static volatile bit           CN1IE             @ ((unsigned)&CNEN1*8)+1;
    static volatile bit           CN2IE             @ ((unsigned)&CNEN1*8)+2;
    static volatile bit           CN3IE             @ ((unsigned)&CNEN1*8)+3;
    static volatile bit           CN4IE             @ ((unsigned)&CNEN1*8)+4;
    static volatile bit           CN5IE             @ ((unsigned)&CNEN1*8)+5;
702 static volatile bit           CN6IE             @ ((unsigned)&CNEN1*8)+6;
    static volatile bit           CN7IE             @ ((unsigned)&CNEN1*8)+7;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                        : 1;
707                 unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
712                 unsigned                        : 1;
                    unsigned                        : 1;
           volatile unsigned      CN7IE             : 1;
           volatile unsigned      CN6IE             : 1;
           volatile unsigned      CN5IE             : 1;
717        volatile unsigned      CN4IE             : 1;
           volatile unsigned      CN3IE             : 1;
           volatile unsigned      CN2IE             : 1;
           volatile unsigned      CN1IE             : 1;
           volatile unsigned      CN0IE             : 1;
722 } CNEN1bits @ 0xC0;

    static volatile unsigned int  CNEN2             @ 0xC2;
    static volatile bit           CN17IE            @ ((unsigned)&CNEN2*8)+1;
    static volatile bit           CN18IE            @ ((unsigned)&CNEN2*8)+2;
727 /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
732                 unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
737                 unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
742        volatile unsigned      CN18IE            : 1;
           volatile unsigned      CN17IE            : 1;
                    unsigned                        : 1;
    } CNEN2bits @ 0xC2;

747 static volatile unsigned int  CNPU1             @ 0xC4;
    static volatile bit           CN0PUE            @ ((unsigned)&CNPU1*8)+0;
    static volatile bit           CN1PUE            @ ((unsigned)&CNPU1*8)+1;
    static volatile bit           CN2PUE            @ ((unsigned)&CNPU1*8)+2;
    static volatile bit           CN3PUE            @ ((unsigned)&CNPU1*8)+3;
752 static volatile bit           CN4PUE            @ ((unsigned)&CNPU1*8)+4;
    static volatile bit           CN5PUE            @ ((unsigned)&CNPU1*8)+5;
    static volatile bit           CN6PUE            @ ((unsigned)&CNPU1*8)+6;
```

```
    static volatile bit           CN7PUE          @ ((unsigned)&CNPU1*8)+7;
    /* Microchip compatible bit field */
757 static volatile struct {
                    unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
762                 unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
            volatile unsigned       CN7PUE        : 1;
767         volatile unsigned       CN6PUE        : 1;
            volatile unsigned       CN5PUE        : 1;
            volatile unsigned       CN4PUE        : 1;
            volatile unsigned       CN3PUE        : 1;
            volatile unsigned       CN2PUE        : 1;
772         volatile unsigned       CN1PUE        : 1;
            volatile unsigned       CN0PUE        : 1;
    } CNPU1bits @ 0xC4;

    static volatile unsigned int  CNPU2           @ 0xC6;
777 static volatile bit            CN17PUE         @ ((unsigned)&CNPU2*8)+1;
    static volatile bit           CN18PUE         @ ((unsigned)&CNPU2*8)+2;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                      : 1;
782                 unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
787                 unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
792                 unsigned                      : 1;
                    unsigned                      : 1;
            volatile unsigned       CN18PUE       : 1;
            volatile unsigned       CN17PUE       : 1;
                    unsigned                      : 1;
797 } CNPU2bits @ 0xC6;

    static volatile unsigned int  TMR1            @ 0x100;
    static volatile unsigned int  PR1             @ 0x102;
    static volatile unsigned int  T1CON           @ 0x104;
802 static volatile bit            T1CS            @ ((unsigned)&T1CON*8)+1;
    static volatile bit           T1SYNC          @ ((unsigned)&T1CON*8)+2;
    static volatile bit           T1CKPS0         @ ((unsigned)&T1CON*8)+4;
    static volatile bit           T1CKPS1         @ ((unsigned)&T1CON*8)+5;
    static volatile bit           T1GATE          @ ((unsigned)&T1CON*8)+6;
807 static volatile bit           T1SIDL          @ ((unsigned)&T1CON*8)+13;
    static volatile bit           T1ON            @ ((unsigned)&T1CON*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned       TON           : 1;
812                 unsigned                      : 1;
            volatile unsigned       TSIDL         : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
817                 unsigned                      : 1;
                    unsigned                      : 1;
                    unsigned                      : 1;
            volatile unsigned       TGATE         : 1;
            volatile unsigned       TCKPS         : 2;
822                 unsigned                      : 1;
            volatile unsigned       TSYNC         : 1;
            volatile unsigned       TCS           : 1;
                    unsigned                      : 1;
    } T1CONbits @ 0x104;
827
    static volatile unsigned int  TMR2            @ 0x106;
    static volatile unsigned int  TMR3HLD         @ 0x108;
    static volatile unsigned int  TMR3            @ 0x10A;
    static volatile unsigned int  PR2             @ 0x10C;
832 static volatile unsigned int  PR3             @ 0x10E;
```

```
    static volatile unsigned int  T2CON             @ 0x110;
    static volatile bit           T2CS              @ ((unsigned)&T2CON*8)+1;
    static volatile bit           T2T32             @ ((unsigned)&T2CON*8)+3;
    static volatile bit           T2CKPS0           @ ((unsigned)&T2CON*8)+4;
837 static volatile bit           T2CKPS1           @ ((unsigned)&T2CON*8)+5;
    static volatile bit           T2GATE            @ ((unsigned)&T2CON*8)+6;
    static volatile bit           T2SIDL            @ ((unsigned)&T2CON*8)+13;
    static volatile bit           T2ON              @ ((unsigned)&T2CON*8)+15;
    /* Microchip compatible bit field */
842 static volatile struct {
            volatile unsigned     TON               : 1;
                     unsigned                       : 1;
            volatile unsigned     TSIDL             : 1;
                     unsigned                       : 1;
847                  unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
852         volatile unsigned     TGATE             : 1;
            volatile unsigned     TCKPS             : 2;
            volatile unsigned     T32               : 1;
                     unsigned                       : 1;
            volatile unsigned     TCS               : 1;
857                  unsigned                       : 1;
    } T2CONbits @ 0x110;

    static volatile unsigned int  T3CON             @ 0x112;
    static volatile bit           T3CS              @ ((unsigned)&T3CON*8)+1;
862 static volatile bit           T3CKPS0           @ ((unsigned)&T3CON*8)+4;
    static volatile bit           T3CKPS1           @ ((unsigned)&T3CON*8)+5;
    static volatile bit           T3GATE            @ ((unsigned)&T3CON*8)+6;
    static volatile bit           T3SIDL            @ ((unsigned)&T3CON*8)+13;
    static volatile bit           T3ON              @ ((unsigned)&T3CON*8)+15;
867 /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned     TON               : 1;
                     unsigned                       : 1;
            volatile unsigned     TSIDL             : 1;
872                  unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
877                  unsigned                       : 1;
            volatile unsigned     TGATE             : 1;
            volatile unsigned     TCKPS             : 2;
                     unsigned                       : 1;
                     unsigned                       : 1;
882         volatile unsigned     TCS               : 1;
                     unsigned                       : 1;
    } T3CONbits @ 0x112;

    static volatile unsigned int  TMR4              @ 0x114;
887 static volatile unsigned int  TMR5HLD          @ 0x116;
    static volatile unsigned int  TMR5              @ 0x118;
    static volatile unsigned int  PR4               @ 0x11A;
    static volatile unsigned int  PR5               @ 0x11C;
    static volatile unsigned int  T4CON             @ 0x11E;
892 static volatile bit           T4CS              @ ((unsigned)&T4CON*8)+1;
    static volatile bit           T4T45             @ ((unsigned)&T4CON*8)+3;
    static volatile bit           T4CKPS0           @ ((unsigned)&T4CON*8)+4;
    static volatile bit           T4CKPS1           @ ((unsigned)&T4CON*8)+5;
    static volatile bit           T4GATE            @ ((unsigned)&T4CON*8)+6;
897 static volatile bit           T4SIDL            @ ((unsigned)&T4CON*8)+13;
    static volatile bit           T4ON              @ ((unsigned)&T4CON*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned     TON               : 1;
902                  unsigned                       : 1;
            volatile unsigned     TSIDL             : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
907                  unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
            volatile unsigned     TGATE             : 1;
```

```
            volatile unsigned       TCKPS               : 2;
912         volatile unsigned       T45                 : 1;
                     unsigned                           : 1;
            volatile unsigned       TCS                 : 1;
                     unsigned                           : 1;
    } T4CONbits @ 0x11E;
917
    static volatile unsigned int   T5CON               @ 0x120;
    static volatile bit            T5CS                @ ((unsigned)&T5CON*8)+1;
    static volatile bit            T5CKPS0             @ ((unsigned)&T5CON*8)+4;
    static volatile bit            T5CKPS1             @ ((unsigned)&T5CON*8)+5;
922 static volatile bit            T5GATE              @ ((unsigned)&T5CON*8)+6;
    static volatile bit            T5SIDL              @ ((unsigned)&T5CON*8)+13;
    static volatile bit            T5ON                @ ((unsigned)&T5CON*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
927         volatile unsigned       TON                 : 1;
                     unsigned                           : 1;
            volatile unsigned       TSIDL               : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
932                  unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
            volatile unsigned       TGATE               : 1;
937         volatile unsigned       TCKPS               : 2;
                     unsigned                           : 1;
                     unsigned                           : 1;
            volatile unsigned       TCS                 : 1;
                     unsigned                           : 1;
942 } T5CONbits @ 0x120;

    static volatile unsigned int   QEICON              @ 0x122;
    static volatile bit            UPDN_CNT            @ ((unsigned)&QEICON*8)+0;
    static volatile bit            TQCS                @ ((unsigned)&QEICON*8)+1;
947 static volatile bit            POSRES              @ ((unsigned)&QEICON*8)+2;
    static volatile bit            TQCKPS0             @ ((unsigned)&QEICON*8)+3;
    static volatile bit            TQCKPS1             @ ((unsigned)&QEICON*8)+4;
    static volatile bit            TQGATE              @ ((unsigned)&QEICON*8)+5;
    static volatile bit            PCDOUT              @ ((unsigned)&QEICON*8)+6;
952 static volatile bit            SWPAB               @ ((unsigned)&QEICON*8)+7;
    static volatile bit            QEIM0               @ ((unsigned)&QEICON*8)+8;
    static volatile bit            QEIM1               @ ((unsigned)&QEICON*8)+9;
    static volatile bit            QEIM2               @ ((unsigned)&QEICON*8)+10;
    static volatile bit            UPDN                @ ((unsigned)&QEICON*8)+11;
957 static volatile bit            INDX                @ ((unsigned)&QEICON*8)+12;
    static volatile bit            QEISIDL             @ ((unsigned)&QEICON*8)+13;
    static volatile bit            CNTERR              @ ((unsigned)&QEICON*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
962         volatile unsigned       CNTERR              : 1;
            volatile unsigned                           : 1;
            volatile unsigned       QEISIDL             : 1;
            volatile unsigned       INDX                : 1;
            volatile unsigned       UPDN                : 1;
967         volatile unsigned       QEIM2               : 1;
            volatile unsigned       QEIM1               : 1;
            volatile unsigned       QEIM0               : 1;
            volatile unsigned       SWPAB               : 1;
            volatile unsigned       PCDOUT              : 1;
972         volatile unsigned       TQGATE              : 1;
            volatile unsigned       TQCKPS1             : 1;
            volatile unsigned       TQCKPS0             : 1;
            volatile unsigned       POSRES              : 1;
            volatile unsigned       TQCS                : 1;
977         volatile unsigned       UPDN_CNT            : 1;
    } QEICONbits @ 0x122;

    static volatile unsigned int   DFLTCON             @ 0x124;
    static volatile bit            QECK0               @ ((unsigned)&DFLTCON*8)+4;
982 static volatile bit            QECK1               @ ((unsigned)&DFLTCON*8)+5;
    static volatile bit            QECK2               @ ((unsigned)&DFLTCON*8)+6;
    static volatile bit            QEOUT               @ ((unsigned)&DFLTCON*8)+7;
    static volatile bit            CEID                @ ((unsigned)&DFLTCON*8)+8;
    static volatile bit            IMV0                @ ((unsigned)&DFLTCON*8)+9;
987 static volatile bit            IMV1                @ ((unsigned)&DFLTCON*8)+10;
    /* Microchip compatible bit field */
```

```
    static volatile struct {
                    unsigned                        : 1;
                    unsigned                        : 1;
992                 unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       IMV1            : 1;
            volatile unsigned       IMV0            : 1;
997         volatile unsigned       CEID            : 1;
            volatile unsigned       QEOUT           : 1;
            volatile unsigned       QECK2           : 1;
            volatile unsigned       QECK1           : 1;
            volatile unsigned       QECK0           : 1;
1002                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
    } DFLTCONbits @ 0x124;
1007
    static volatile unsigned int  POSCNT            @ 0x126;
    static volatile unsigned int  MAXCNT            @ 0x128;
    static volatile unsigned int  IC1BUF            @ 0x140;
    static volatile unsigned int  IC1CON            @ 0x142;
1012 static volatile bit           IC1_M0            @ ((unsigned)&IC1CON*8)+0;
    static volatile bit           IC1_M1            @ ((unsigned)&IC1CON*8)+1;
    static volatile bit           IC1_M2            @ ((unsigned)&IC1CON*8)+2;
    static volatile bit           IC1_BNE           @ ((unsigned)&IC1CON*8)+3;
    static volatile bit           IC1_OV            @ ((unsigned)&IC1CON*8)+4;
1017 static volatile bit           IC1_IO            @ ((unsigned)&IC1CON*8)+5;
    static volatile bit           IC1_I1            @ ((unsigned)&IC1CON*8)+6;
    static volatile bit           IC1_TMR           @ ((unsigned)&IC1CON*8)+7;
    static volatile bit           IC1_SIDL          @ ((unsigned)&IC1CON*8)+13;
    /* Microchip compatible bit field */
1022 static volatile struct {
                    unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       ICSIDL          : 1;
                    unsigned                        : 1;
1027                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       ICTMR           : 1;
1032        volatile unsigned       ICI             : 2;
            volatile unsigned       ICOV            : 1;
            volatile unsigned       ICBNE           : 1;
            volatile unsigned       ICM             : 3;
    } IC1CONbits @ 0x142;
1037
    static volatile unsigned int  IC2BUF            @ 0x144;
    static volatile unsigned int  IC2CON            @ 0x146;
    static volatile bit           IC2_M0            @ ((unsigned)&IC2CON*8)+0;
    static volatile bit           IC2_M1            @ ((unsigned)&IC2CON*8)+1;
1042 static volatile bit           IC2_M2            @ ((unsigned)&IC2CON*8)+2;
    static volatile bit           IC2_BNE           @ ((unsigned)&IC2CON*8)+3;
    static volatile bit           IC2_OV            @ ((unsigned)&IC2CON*8)+4;
    static volatile bit           IC2_IO            @ ((unsigned)&IC2CON*8)+5;
    static volatile bit           IC2_I1            @ ((unsigned)&IC2CON*8)+6;
1047 static volatile bit           IC2_TMR           @ ((unsigned)&IC2CON*8)+7;
    static volatile bit           IC2_SIDL          @ ((unsigned)&IC2CON*8)+13;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                        : 1;
1052                unsigned                        : 1;
            volatile unsigned       ICSIDL          : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
1057                unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       ICTMR           : 1;
            volatile unsigned       ICI             : 2;
            volatile unsigned       ICOV            : 1;
1062        volatile unsigned       ICBNE           : 1;
            volatile unsigned       ICM             : 3;
    } IC2CONbits @ 0x146;

    static volatile unsigned int  IC7BUF            @ 0x158;
```

```
1067  static volatile unsigned int  IC7CON              @ 0x15A;
      static volatile bit           IC7_M0              @ ((unsigned)&IC7CON*8)+0;
      static volatile bit           IC7_M1              @ ((unsigned)&IC7CON*8)+1;
      static volatile bit           IC7_M2              @ ((unsigned)&IC7CON*8)+2;
      static volatile bit           IC7_BNE             @ ((unsigned)&IC7CON*8)+3;
1072  static volatile bit           IC7_OV              @ ((unsigned)&IC7CON*8)+4;
      static volatile bit           IC7_IO              @ ((unsigned)&IC7CON*8)+5;
      static volatile bit           IC7_I1              @ ((unsigned)&IC7CON*8)+6;
      static volatile bit           IC7_TMR             @ ((unsigned)&IC7CON*8)+7;
      static volatile bit           IC7_SIDL            @ ((unsigned)&IC7CON*8)+13;
1077  /* Microchip compatible bit field */
      static volatile struct {
                      unsigned                          : 1;
                      unsigned                          : 1;
              volatile unsigned     ICSIDL              : 1;
1082                  unsigned                          : 1;
                      unsigned                          : 1;
                      unsigned                          : 1;
                      unsigned                          : 1;
                      unsigned                          : 1;
1087          volatile unsigned     ICTMR               : 1;
              volatile unsigned     ICI                 : 2;
              volatile unsigned     ICOV                : 1;
              volatile unsigned     ICBNE               : 1;
              volatile unsigned     ICM                 : 3;
1092  } IC7CONbits @ 0x15A;

      static volatile unsigned int  IC8BUF              @ 0x15C;
      static volatile unsigned int  IC8CON              @ 0x15E;
      static volatile bit           IC8_M0              @ ((unsigned)&IC8CON*8)+0;
1097  static volatile bit           IC8_M1              @ ((unsigned)&IC8CON*8)+1;
      static volatile bit           IC8_M2              @ ((unsigned)&IC8CON*8)+2;
      static volatile bit           IC8_BNE             @ ((unsigned)&IC8CON*8)+3;
      static volatile bit           IC8_OV              @ ((unsigned)&IC8CON*8)+4;
      static volatile bit           IC8_IO              @ ((unsigned)&IC8CON*8)+5;
1102  static volatile bit           IC8_I1              @ ((unsigned)&IC8CON*8)+6;
      static volatile bit           IC8_TMR             @ ((unsigned)&IC8CON*8)+7;
      static volatile bit           IC8_SIDL            @ ((unsigned)&IC8CON*8)+13;
      /* Microchip compatible bit field */
      static volatile struct {
1107                  unsigned                          : 1;
                      unsigned                          : 1;
              volatile unsigned     ICSIDL              : 1;
                      unsigned                          : 1;
                      unsigned                          : 1;
1112                  unsigned                          : 1;
                      unsigned                          : 1;
                      unsigned                          : 1;
              volatile unsigned     ICTMR               : 1;
              volatile unsigned     ICI                 : 2;
1117          volatile unsigned     ICOV                : 1;
              volatile unsigned     ICBNE               : 1;
              volatile unsigned     ICM                 : 3;
      } IC8CONbits @ 0x15E;

1122  static volatile unsigned int  OC1RS               @ 0x180;
      static volatile unsigned int  OC1R                @ 0x182;
      static volatile unsigned int  OC1CON              @ 0x184;
      static volatile bit           OC1_M0              @ ((unsigned)&OC1CON*8)+0;
      static volatile bit           OC1_M1              @ ((unsigned)&OC1CON*8)+1;
1127  static volatile bit           OC1_M2              @ ((unsigned)&OC1CON*8)+2;
      static volatile bit           OC1_TSEL            @ ((unsigned)&OC1CON*8)+3;
      static volatile bit           OC1_FLT             @ ((unsigned)&OC1CON*8)+4;
      static volatile bit           OC1_SIDL            @ ((unsigned)&OC1CON*8)+13;
      /* Microchip compatible bit field */
1132  static volatile struct {
                      unsigned                          : 1;
                      unsigned                          : 1;
              volatile unsigned     OCSIDL              : 1;
                      unsigned                          : 1;
1137                  unsigned                          : 1;
                      unsigned                          : 1;
                      unsigned                          : 1;
                      unsigned                          : 1;
                      unsigned                          : 1;
1142                  unsigned                          : 1;
                      unsigned                          : 1;
              volatile unsigned     OCFLT               : 1;
```

```
            volatile unsigned       OCTSEL              : 1;
            volatile unsigned       OCM                 : 3;
} OC1CONbits @ 0x184;

    static volatile unsigned int   OC2RS                @ 0x186;
    static volatile unsigned int   OC2R                 @ 0x188;
    static volatile unsigned int   OC2CON               @ 0x18A;
    static volatile bit            OC2_M0               @ ((unsigned)&OC2CON*8)+0;
    static volatile bit            OC2_M1               @ ((unsigned)&OC2CON*8)+1;
    static volatile bit            OC2_M2               @ ((unsigned)&OC2CON*8)+2;
    static volatile bit            OC2_TSEL             @ ((unsigned)&OC2CON*8)+3;
    static volatile bit            OC2_FLT              @ ((unsigned)&OC2CON*8)+4;
    static volatile bit            OC2_SIDL             @ ((unsigned)&OC2CON*8)+13;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                            : 1;
                    unsigned                            : 1;
            volatile unsigned       OCSIDL              : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
            volatile unsigned       OCFLT               : 1;
            volatile unsigned       OCTSEL              : 1;
            volatile unsigned       OCM                 : 3;
    } OC2CONbits @ 0x18A;

    static volatile unsigned int   OC3RS                @ 0x18C;
    static volatile unsigned int   OC3R                 @ 0x18E;
    static volatile unsigned int   OC3CON               @ 0x190;
    static volatile bit            OC3_M0               @ ((unsigned)&OC3CON*8)+0;
    static volatile bit            OC3_M1               @ ((unsigned)&OC3CON*8)+1;
    static volatile bit            OC3_M2               @ ((unsigned)&OC3CON*8)+2;
    static volatile bit            OC3_TSEL             @ ((unsigned)&OC3CON*8)+3;
    static volatile bit            OC3_FLT              @ ((unsigned)&OC3CON*8)+4;
    static volatile bit            OC3_SIDL             @ ((unsigned)&OC3CON*8)+13;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                            : 1;
                    unsigned                            : 1;
            volatile unsigned       OCSIDL              : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
            volatile unsigned       OCFLT               : 1;
            volatile unsigned       OCTSEL              : 1;
            volatile unsigned       OCM                 : 3;
    } OC3CONbits @ 0x190;

    static volatile unsigned int   OC4RS                @ 0x192;
    static volatile unsigned int   OC4R                 @ 0x194;
    static volatile unsigned int   OC4CON               @ 0x196;
    static volatile bit            OC4_M0               @ ((unsigned)&OC4CON*8)+0;
    static volatile bit            OC4_M1               @ ((unsigned)&OC4CON*8)+1;
    static volatile bit            OC4_M2               @ ((unsigned)&OC4CON*8)+2;
    static volatile bit            OC4_TSEL             @ ((unsigned)&OC4CON*8)+3;
    static volatile bit            OC4_FLT              @ ((unsigned)&OC4CON*8)+4;
    static volatile bit            OC4_SIDL             @ ((unsigned)&OC4CON*8)+13;
/* Microchip compatible bit field */
    static volatile struct {
                    unsigned                            : 1;
                    unsigned                            : 1;
            volatile unsigned       OCSIDL              : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
```

```
                unsigned                        : 1;
                unsigned                        : 1;
        volatile unsigned       OCFLT           : 1;
        volatile unsigned       OCTSEL          : 1;
1227    volatile unsigned       OCM             : 3;
} OC4CONbits @ 0x196;

static volatile unsigned int   PTCON            @ 0x1C0;
static volatile bit            PTMOD0           @ ((unsigned)&PTCON*8)+0;
1232 static volatile bit            PTMOD1           @ ((unsigned)&PTCON*8)+1;
static volatile bit            PTCKPS0          @ ((unsigned)&PTCON*8)+2;
static volatile bit            PTCKPS1          @ ((unsigned)&PTCON*8)+3;
static volatile bit            PTOPS0           @ ((unsigned)&PTCON*8)+4;
static volatile bit            PTOPS1           @ ((unsigned)&PTCON*8)+5;
1237 static volatile bit            PTOPS2           @ ((unsigned)&PTCON*8)+6;
static volatile bit            PTOPS3           @ ((unsigned)&PTCON*8)+7;
static volatile bit            PTSIDL           @ ((unsigned)&PTCON*8)+13;
static volatile bit            PTEN             @ ((unsigned)&PTCON*8)+15;
/* Microchip compatible bit field */
1242 static volatile struct {
        volatile unsigned       PTEN            : 1;
                unsigned                        : 1;
        volatile unsigned       PTSIDL          : 1;
                unsigned                        : 1;
1247            unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
                unsigned                        : 1;
        volatile unsigned       PTOPS           : 4;
1252    volatile unsigned       PTCKPS          : 2;
        volatile unsigned       PTMOD           : 2;
} PTCONbits @ 0x1C0;

static volatile unsigned int   PTMR             @ 0x1C2;
1257 static volatile bit            PTMR0            @ ((unsigned)&PTMR*8)+0;
static volatile bit            PTMR1            @ ((unsigned)&PTMR*8)+1;
static volatile bit            PTMR2            @ ((unsigned)&PTMR*8)+2;
static volatile bit            PTMR3            @ ((unsigned)&PTMR*8)+3;
static volatile bit            PTMR4            @ ((unsigned)&PTMR*8)+4;
1262 static volatile bit            PTMR5            @ ((unsigned)&PTMR*8)+5;
static volatile bit            PTMR6            @ ((unsigned)&PTMR*8)+6;
static volatile bit            PTMR7            @ ((unsigned)&PTMR*8)+7;
static volatile bit            PTMR8            @ ((unsigned)&PTMR*8)+8;
static volatile bit            PTMR9            @ ((unsigned)&PTMR*8)+9;
1267 static volatile bit            PTMR10           @ ((unsigned)&PTMR*8)+10;
static volatile bit            PTMR11           @ ((unsigned)&PTMR*8)+11;
static volatile bit            PTMR12           @ ((unsigned)&PTMR*8)+12;
static volatile bit            PTMR13           @ ((unsigned)&PTMR*8)+13;
static volatile bit            PTMR14           @ ((unsigned)&PTMR*8)+14;
1272 static volatile bit            PTDIR            @ ((unsigned)&PTMR*8)+15;
/* Microchip compatible bit field */
static volatile struct {
        volatile unsigned       PTDIR           : 1;
        volatile unsigned       PTMR            : 15;
1277 } PTMRbits @ 0x1C2;

static volatile unsigned int   PTPER            @ 0x1C4;
static volatile unsigned int   SEVTCMP          @ 0x1C6;
static volatile bit            SEVTCMP0         @ ((unsigned)&SEVTCMP*8)+0;
1282 static volatile bit            SEVTCMP1         @ ((unsigned)&SEVTCMP*8)+1;
static volatile bit            SEVTCMP2         @ ((unsigned)&SEVTCMP*8)+2;
static volatile bit            SEVTCMP3         @ ((unsigned)&SEVTCMP*8)+3;
static volatile bit            SEVTCMP4         @ ((unsigned)&SEVTCMP*8)+4;
static volatile bit            SEVTCMP5         @ ((unsigned)&SEVTCMP*8)+5;
1287 static volatile bit            SEVTCMP6         @ ((unsigned)&SEVTCMP*8)+6;
static volatile bit            SEVTCMP7         @ ((unsigned)&SEVTCMP*8)+7;
static volatile bit            SEVTCMP8         @ ((unsigned)&SEVTCMP*8)+8;
static volatile bit            SEVTCMP9         @ ((unsigned)&SEVTCMP*8)+9;
static volatile bit            SEVTCMP10        @ ((unsigned)&SEVTCMP*8)+10;
1292 static volatile bit            SEVTCMP11        @ ((unsigned)&SEVTCMP*8)+11;
static volatile bit            SEVTCMP12        @ ((unsigned)&SEVTCMP*8)+12;
static volatile bit            SEVTCMP13        @ ((unsigned)&SEVTCMP*8)+13;
static volatile bit            SEVTCMP14        @ ((unsigned)&SEVTCMP*8)+14;
static volatile bit            SEVTDIR          @ ((unsigned)&SEVTCMP*8)+15;
1297 /* Microchip compatible bit field */
static volatile struct {
        volatile unsigned       SEVTDIR         : 1;
        volatile unsigned       SEVTCMP         : 15;
```

```
    } SEVTCMPbits @ 0x1C6;
1302
    static volatile unsigned int  PWMCON1             @ 0x1C8;
    static volatile bit           PWMEN1L             @ ((unsigned)&PWMCON1*8)+0;
    static volatile bit           PWMEN2L             @ ((unsigned)&PWMCON1*8)+1;
    static volatile bit           PWMEN3L             @ ((unsigned)&PWMCON1*8)+2;
1307 static volatile bit           PWMEN4L             @ ((unsigned)&PWMCON1*8)+3;
    static volatile bit           PWMEN1H             @ ((unsigned)&PWMCON1*8)+4;
    static volatile bit           PWMEN2H             @ ((unsigned)&PWMCON1*8)+5;
    static volatile bit           PWMEN3H             @ ((unsigned)&PWMCON1*8)+6;
    static volatile bit           PWMEN4H             @ ((unsigned)&PWMCON1*8)+7;
1312 static volatile bit           PWMTMOD1            @ ((unsigned)&PWMCON1*8)+8;
    static volatile bit           PWMTMOD2            @ ((unsigned)&PWMCON1*8)+9;
    static volatile bit           PWMTMOD3            @ ((unsigned)&PWMCON1*8)+10;
    static volatile bit           PWMTMOD4            @ ((unsigned)&PWMCON1*8)+11;
    /* Microchip compatible bit field */
1317 static volatile struct {
                  unsigned                          : 1;
                  unsigned                          : 1;
                  unsigned                          : 1;
                  unsigned                          : 1;
1322        volatile unsigned       PTMOD4          : 1;
            volatile unsigned       PTMOD3          : 1;
            volatile unsigned       PTMOD2          : 1;
            volatile unsigned       PTMOD1          : 1;
            volatile unsigned       PEN4H           : 1;
1327        volatile unsigned       PEN3H           : 1;
            volatile unsigned       PEN2H           : 1;
            volatile unsigned       PEN1H           : 1;
            volatile unsigned       PEN4L           : 1;
            volatile unsigned       PEN3L           : 1;
1332        volatile unsigned       PEN2L           : 1;
            volatile unsigned       PEN1L           : 1;
    } PWMCON1bits @ 0x1C8;

    static volatile unsigned int  PWMCON2             @ 0x1CA;
1337 static volatile bit           UDIS                @ ((unsigned)&PWMCON2*8)+0;
    static volatile bit           OSYNC               @ ((unsigned)&PWMCON2*8)+1;
    static volatile bit           SEVOPS0             @ ((unsigned)&PWMCON2*8)+8;
    static volatile bit           SEVOPS1             @ ((unsigned)&PWMCON2*8)+9;
    static volatile bit           SEVOPS2             @ ((unsigned)&PWMCON2*8)+10;
1342 static volatile bit           SEVOPS3             @ ((unsigned)&PWMCON2*8)+11;
    /* Microchip compatible bit field */
    static volatile struct {
                  unsigned                          : 1;
                  unsigned                          : 1;
1347              unsigned                          : 1;
                  unsigned                          : 1;
            volatile unsigned       SEVOPS          : 4;
                  unsigned                          : 1;
                  unsigned                          : 1;
1352              unsigned                          : 1;
                  unsigned                          : 1;
                  unsigned                          : 1;
                  unsigned                          : 1;
            volatile unsigned       OSYNC           : 1;
1357        volatile unsigned       UDIS            : 1;
    } PWMCON2bits @ 0x1CA;

    static volatile unsigned int  DTCON1              @ 0x1CC;
    static volatile bit           DTAVAL0             @ ((unsigned)&DTCON1*8)+0;
1362 static volatile bit           DTAVAL1             @ ((unsigned)&DTCON1*8)+1;
    static volatile bit           DTAVAL2             @ ((unsigned)&DTCON1*8)+2;
    static volatile bit           DTAVAL3             @ ((unsigned)&DTCON1*8)+3;
    static volatile bit           DTAVAL4             @ ((unsigned)&DTCON1*8)+4;
    static volatile bit           DTAVAL5             @ ((unsigned)&DTCON1*8)+5;
1367 static volatile bit           DTAPS0              @ ((unsigned)&DTCON1*8)+6;
    static volatile bit           DTAPS1              @ ((unsigned)&DTCON1*8)+7;
    static volatile bit           DTBVAL0             @ ((unsigned)&DTCON1*8)+8;
    static volatile bit           DTBVAL1             @ ((unsigned)&DTCON1*8)+9;
    static volatile bit           DTBVAL2             @ ((unsigned)&DTCON1*8)+10;
1372 static volatile bit           DTBVAL3             @ ((unsigned)&DTCON1*8)+11;
    static volatile bit           DTBVAL4             @ ((unsigned)&DTCON1*8)+12;
    static volatile bit           DTBVAL5             @ ((unsigned)&DTCON1*8)+13;
    static volatile bit           DTBPS0              @ ((unsigned)&DTCON1*8)+14;
    static volatile bit           DTBPS1              @ ((unsigned)&DTCON1*8)+15;
1377 /* Microchip compatible bit field */
    static volatile struct {
```

```
            volatile unsigned       DTBPS               : 2;
            volatile unsigned       DTBVAL              : 6;
            volatile unsigned       DTAPS               : 2;
1382        volatile unsigned       DTAVAL              : 6;
    } DTCON1bits @ 0x1CC;

    static volatile unsigned int   FLTACON             @ 0x1D0;
    static volatile bit            FAEN1               @ ((unsigned)&FLTACON*8)+0;
1387 static volatile bit            FAEN2               @ ((unsigned)&FLTACON*8)+1;
    static volatile bit            FAEN3               @ ((unsigned)&FLTACON*8)+2;
    static volatile bit            FAEN4               @ ((unsigned)&FLTACON*8)+3;
    static volatile bit            FLTAM               @ ((unsigned)&FLTACON*8)+7;
    static volatile bit            FAOV1L              @ ((unsigned)&FLTACON*8)+8;
1392 static volatile bit            FAOV1H              @ ((unsigned)&FLTACON*8)+9;
    static volatile bit            FAOV2L              @ ((unsigned)&FLTACON*8)+10;
    static volatile bit            FAOV2H              @ ((unsigned)&FLTACON*8)+11;
    static volatile bit            FAOV3L              @ ((unsigned)&FLTACON*8)+12;
    static volatile bit            FAOV3H              @ ((unsigned)&FLTACON*8)+13;
1397 static volatile bit            FAOV4L              @ ((unsigned)&FLTACON*8)+14;
    static volatile bit            FAOV4H              @ ((unsigned)&FLTACON*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned       FAOV4H              : 1;
1402        volatile unsigned       FAOV4L              : 1;
            volatile unsigned       FAOV3H              : 1;
            volatile unsigned       FAOV3L              : 1;
            volatile unsigned       FAOV2H              : 1;
            volatile unsigned       FAOV2L              : 1;
1407        volatile unsigned       FAOV1H              : 1;
            volatile unsigned       FAOV1L              : 1;
            volatile unsigned       FLTAM               : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
1412                 unsigned                           : 1;
            volatile unsigned       FAEN4               : 1;
            volatile unsigned       FAEN3               : 1;
            volatile unsigned       FAEN2               : 1;
            volatile unsigned       FAEN1               : 1;
1417 } FLTACONbits @ 0x1D0;

    static volatile unsigned int   OVDCON              @ 0x1D4;
    static volatile bit            POUT1L              @ ((unsigned)&OVDCON*8)+0;
    static volatile bit            POUT1H              @ ((unsigned)&OVDCON*8)+1;
1422 static volatile bit            POUT2L              @ ((unsigned)&OVDCON*8)+2;
    static volatile bit            POUT2H              @ ((unsigned)&OVDCON*8)+3;
    static volatile bit            POUT3L              @ ((unsigned)&OVDCON*8)+4;
    static volatile bit            POUT3H              @ ((unsigned)&OVDCON*8)+5;
    static volatile bit            POUT4L              @ ((unsigned)&OVDCON*8)+6;
1427 static volatile bit            POUT4H              @ ((unsigned)&OVDCON*8)+7;
    static volatile bit            POVD1L              @ ((unsigned)&OVDCON*8)+8;
    static volatile bit            POVD1H              @ ((unsigned)&OVDCON*8)+9;
    static volatile bit            POVD2L              @ ((unsigned)&OVDCON*8)+10;
    static volatile bit            POVD2H              @ ((unsigned)&OVDCON*8)+11;
1432 static volatile bit            POVD3L              @ ((unsigned)&OVDCON*8)+12;
    static volatile bit            POVD3H              @ ((unsigned)&OVDCON*8)+13;
    static volatile bit            POVD4L              @ ((unsigned)&OVDCON*8)+14;
    static volatile bit            POVD4H              @ ((unsigned)&OVDCON*8)+15;
    /* Microchip compatible bit field */
1437 static volatile struct {
            volatile unsigned       POVD4H              : 1;
            volatile unsigned       POVD4L              : 1;
            volatile unsigned       POVD3H              : 1;
            volatile unsigned       POVD3L              : 1;
1442        volatile unsigned       POVD2H              : 1;
            volatile unsigned       POVD2L              : 1;
            volatile unsigned       POVD1H              : 1;
            volatile unsigned       POVD1L              : 1;
            volatile unsigned       POUT4H              : 1;
1447        volatile unsigned       POUT4L              : 1;
            volatile unsigned       POUT3H              : 1;
            volatile unsigned       POUT3L              : 1;
            volatile unsigned       POUT2H              : 1;
            volatile unsigned       POUT2L              : 1;
1452        volatile unsigned       POUT1H              : 1;
            volatile unsigned       POUT1L              : 1;
    } OVDCONbits @ 0x1D4;

    static volatile unsigned int   PDC1                @ 0x1D6;
```

```
1457  static volatile unsigned int   PDC2                @ 0x1D8;
      static volatile unsigned int   PDC3                @ 0x1DA;
      static volatile unsigned int   I2CRCV              @ 0x200;
      static volatile unsigned int   I2CTRN              @ 0x202;
      static volatile unsigned int   I2CBRG              @ 0x204;
1462  static volatile unsigned int   I2CCON              @ 0x206;
      static volatile bit            I2C_SEN             @ ((unsigned)&I2CCON*8)+0;
      static volatile bit            I2C_RSEN            @ ((unsigned)&I2CCON*8)+1;
      static volatile bit            I2C_PEN             @ ((unsigned)&I2CCON*8)+2;
      static volatile bit            I2C_RCEN            @ ((unsigned)&I2CCON*8)+3;
1467  static volatile bit            I2C_ACKEN           @ ((unsigned)&I2CCON*8)+4;
      static volatile bit            I2C_ACKDT           @ ((unsigned)&I2CCON*8)+5;
      static volatile bit            I2C_STREN           @ ((unsigned)&I2CCON*8)+6;
      static volatile bit            I2C_GCEN            @ ((unsigned)&I2CCON*8)+7;
      static volatile bit            I2C_SMEN            @ ((unsigned)&I2CCON*8)+8;
1472  static volatile bit            I2C_DISSLW          @ ((unsigned)&I2CCON*8)+9;
      static volatile bit            I2C_A10M            @ ((unsigned)&I2CCON*8)+10;
      static volatile bit            I2C_IPMIEN          @ ((unsigned)&I2CCON*8)+11;
      static volatile bit            I2C_SCLREL          @ ((unsigned)&I2CCON*8)+12;
      static volatile bit            I2C_SIDL            @ ((unsigned)&I2CCON*8)+13;
1477  static volatile bit            I2C_EN              @ ((unsigned)&I2CCON*8)+15;
      /* Microchip compatible bit field */
      static volatile struct {
              volatile unsigned      I2CEN               : 1;
                       unsigned                          : 1;
1482          volatile unsigned      I2CSIDL             : 1;
              volatile unsigned      SCLREL              : 1;
              volatile unsigned      IPMIEN              : 1;
              volatile unsigned      A10M                : 1;
              volatile unsigned      DISSLW              : 1;
1487          volatile unsigned      SMEN                : 1;
              volatile unsigned      GCEN                : 1;
              volatile unsigned      STREN               : 1;
              volatile unsigned      ACKDT               : 1;
              volatile unsigned      ACKEN               : 1;
1492          volatile unsigned      RCEN                : 1;
              volatile unsigned      PEN                 : 1;
              volatile unsigned      RSEN                : 1;
              volatile unsigned      SEN                 : 1;
      } I2CCONbits @ 0x206;
1497
      static volatile unsigned int   I2CSTAT             @ 0x208;
      static volatile bit            I2C_TBF             @ ((unsigned)&I2CSTAT*8)+0;
      static volatile bit            I2C_RBF             @ ((unsigned)&I2CSTAT*8)+1;
      static volatile bit            I2C_R_NW            @ ((unsigned)&I2CSTAT*8)+2;
1502  static volatile bit            I2C_S               @ ((unsigned)&I2CSTAT*8)+3;
      static volatile bit            I2C_P               @ ((unsigned)&I2CSTAT*8)+4;
      static volatile bit            I2C_D_NA            @ ((unsigned)&I2CSTAT*8)+5;
      static volatile bit            I2C_OV              @ ((unsigned)&I2CSTAT*8)+6;
      static volatile bit            I2C_IWCOL           @ ((unsigned)&I2CSTAT*8)+7;
1507  static volatile bit            I2C_ADD10           @ ((unsigned)&I2CSTAT*8)+8;
      static volatile bit            I2C_GCSTAT          @ ((unsigned)&I2CSTAT*8)+9;
      static volatile bit            I2C_BCL             @ ((unsigned)&I2CSTAT*8)+10;
      static volatile bit            I2C_TRSTAT          @ ((unsigned)&I2CSTAT*8)+14;
      static volatile bit            I2C_ACKSTAT         @ ((unsigned)&I2CSTAT*8)+15;
1512  /* Microchip compatible bit field */
      static volatile struct {
              volatile unsigned      ACKSTAT             : 1;
              volatile unsigned      TRSTAT              : 1;
                       unsigned                          : 1;
1517                   unsigned                          : 1;
                       unsigned                          : 1;
              volatile unsigned      BCL                 : 1;
              volatile unsigned      GCSTAT              : 1;
              volatile unsigned      ADD10               : 1;
1522          volatile unsigned      IWCOL               : 1;
              volatile unsigned      I2COV               : 1;
              volatile unsigned      D_nA                : 1;
              volatile unsigned      P                   : 1;
              volatile unsigned      S                   : 1;
1527          volatile unsigned      R_nW                : 1;
              volatile unsigned      RBF                 : 1;
              volatile unsigned      TBF                 : 1;
      } I2CSTATbits @ 0x208;

1532  static volatile unsigned int   I2CADD              @ 0x20A;
      static volatile unsigned int   U1MODE              @ 0x20C;
      static volatile bit            U1_STSEL            @ ((unsigned)&U1MODE*8)+0;
```

```
    static volatile bit             U1_PDSEL0           @ ((unsigned)&U1MODE*8)+1;
    static volatile bit             U1_PDSEL1           @ ((unsigned)&U1MODE*8)+2;
1537 static volatile bit             U1_ABAUD            @ ((unsigned)&U1MODE*8)+5;
    static volatile bit             U1_LPBACK           @ ((unsigned)&U1MODE*8)+6;
    static volatile bit             U1_WAKE             @ ((unsigned)&U1MODE*8)+7;
    static volatile bit             U1_ALTIO            @ ((unsigned)&U1MODE*8)+10;
    static volatile bit             U1_USIDL            @ ((unsigned)&U1MODE*8)+13;
1542 static volatile bit             U1_UARTEN           @ ((unsigned)&U1MODE*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned       UARTEN              : 1;
                     unsigned                           : 1;
1547        volatile unsigned       USIDL               : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
            volatile unsigned       ALTIO               : 1;
                     unsigned                           : 1;
1552                 unsigned                           : 1;
            volatile unsigned       WAKE                : 1;
            volatile unsigned       LPBACK              : 1;
            volatile unsigned       ABAUD               : 1;
                     unsigned                           : 1;
1557                 unsigned                           : 1;
            volatile unsigned       PDSEL               : 2;
            volatile unsigned       STSEL               : 1;
    } U1MODEbits @ 0x20C;

1562 static volatile unsigned int  U1STA               @ 0x20E;
    static volatile bit             U1_URXDA            @ ((unsigned)&U1STA*8)+0;
    static volatile bit             U1_OERR             @ ((unsigned)&U1STA*8)+1;
    static volatile bit             U1_FERR             @ ((unsigned)&U1STA*8)+2;
    static volatile bit             U1_PERR             @ ((unsigned)&U1STA*8)+3;
1567 static volatile bit             U1_RIDLE            @ ((unsigned)&U1STA*8)+4;
    static volatile bit             U1_ADDEN            @ ((unsigned)&U1STA*8)+5;
    static volatile bit             U1_URXISEL0         @ ((unsigned)&U1STA*8)+6;
    static volatile bit             U1_URXISEL1         @ ((unsigned)&U1STA*8)+7;
    static volatile bit             U1_TRMT             @ ((unsigned)&U1STA*8)+8;
1572 static volatile bit             U1_UTXBF            @ ((unsigned)&U1STA*8)+9;
    static volatile bit             U1_UTXEN            @ ((unsigned)&U1STA*8)+10;
    static volatile bit             U1_UTXBRK           @ ((unsigned)&U1STA*8)+11;
    static volatile bit             U1_UTXISEL          @ ((unsigned)&U1STA*8)+15;
    /* Microchip compatible bit field */
1577 static volatile struct {
            volatile unsigned       UTXISEL             : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
1582        volatile unsigned       UTXBRK              : 1;
            volatile unsigned       UTXEN               : 1;
            volatile unsigned       UTXBF               : 1;
            volatile unsigned       TRMT                : 1;
            volatile unsigned       URXISEL             : 2;
1587        volatile unsigned       ADDEN               : 1;
            volatile unsigned       RIDLE               : 1;
            volatile unsigned       PERR                : 1;
            volatile unsigned       FERR                : 1;
            volatile unsigned       OERR                : 1;
1592        volatile unsigned       URXDA               : 1;
    } U1STAbits @ 0x20E;

    static volatile unsigned int  U1TXREG             @ 0x210;
    static volatile unsigned int  U1RXREG             @ 0x212;
1597 static volatile unsigned int  U1BRG               @ 0x214;
    static volatile unsigned int  U2MODE              @ 0x216;
    static volatile bit             U2_STSEL            @ ((unsigned)&U2MODE*8)+0;
    static volatile bit             U2_PDSEL0           @ ((unsigned)&U2MODE*8)+1;
    static volatile bit             U2_PDSEL1           @ ((unsigned)&U2MODE*8)+2;
1602 static volatile bit             U2_ABAUD            @ ((unsigned)&U2MODE*8)+5;
    static volatile bit             U2_LPBACK           @ ((unsigned)&U2MODE*8)+6;
    static volatile bit             U2_WAKE             @ ((unsigned)&U2MODE*8)+7;
    static volatile bit             U2_USIDL            @ ((unsigned)&U2MODE*8)+13;
    static volatile bit             U2_UARTEN           @ ((unsigned)&U2MODE*8)+15;
1607 /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned       UARTEN              : 1;
                     unsigned                           : 1;
            volatile unsigned       USIDL               : 1;
1612                 unsigned                           : 1;
```

```
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
1617        volatile unsigned       WAKE            : 1;
            volatile unsigned       LPBACK          : 1;
            volatile unsigned       ABAUD           : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
1622        volatile unsigned       PDSEL           : 2;
            volatile unsigned       STSEL           : 1;
    } U2MODEbits @ 0x216;

    static volatile unsigned int  U2STA             @ 0x218;
1627 static volatile bit           U2_URXDA          @ ((unsigned)&U2STA*8)+0;
    static volatile bit           U2_OERR           @ ((unsigned)&U2STA*8)+1;
    static volatile bit           U2_FERR           @ ((unsigned)&U2STA*8)+2;
    static volatile bit           U2_PERR           @ ((unsigned)&U2STA*8)+3;
    static volatile bit           U2_RIDLE          @ ((unsigned)&U2STA*8)+4;
1632 static volatile bit           U2_ADDEN          @ ((unsigned)&U2STA*8)+5;
    static volatile bit           U2_URXISEL0       @ ((unsigned)&U2STA*8)+6;
    static volatile bit           U2_URXISEL1       @ ((unsigned)&U2STA*8)+7;
    static volatile bit           U2_TRMT           @ ((unsigned)&U2STA*8)+8;
    static volatile bit           U2_UTXBF          @ ((unsigned)&U2STA*8)+9;
1637 static volatile bit           U2_UTXEN          @ ((unsigned)&U2STA*8)+10;
    static volatile bit           U2_UTXBRK         @ ((unsigned)&U2STA*8)+11;
    static volatile bit           U2_UTXISEL        @ ((unsigned)&U2STA*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
1642        volatile unsigned       UTXISEL         : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       UTXBRK          : 1;
1647        volatile unsigned       UTXEN           : 1;
            volatile unsigned       UTXBF           : 1;
            volatile unsigned       TRMT            : 1;
            volatile unsigned       URXISEL         : 2;
            volatile unsigned       ADDEN           : 1;
1652        volatile unsigned       RIDLE           : 1;
            volatile unsigned       PERR            : 1;
            volatile unsigned       FERR            : 1;
            volatile unsigned       OERR            : 1;
            volatile unsigned       URXDA           : 1;
1657 } U2STAbits @ 0x218;

    static volatile unsigned int  U2TXREG           @ 0x21A;
    static volatile unsigned int  U2RXREG           @ 0x21C;
    static volatile unsigned int  U2BRG             @ 0x21E;
1662 static volatile unsigned int  SPI1STAT          @ 0x220;
    static volatile bit           SPI1_RBF          @ ((unsigned)&SPI1STAT*8)+0;
    static volatile bit           SPI1_TBF          @ ((unsigned)&SPI1STAT*8)+1;
    static volatile bit           SPI1_ROV          @ ((unsigned)&SPI1STAT*8)+6;
    static volatile bit           SPI1_SIDL         @ ((unsigned)&SPI1STAT*8)+13;
1667 static volatile bit           SPI1_EN           @ ((unsigned)&SPI1STAT*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned       SPIEN           : 1;
                    unsigned                        : 1;
1672        volatile unsigned       SPISIDL         : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
1677                unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       SPIROV          : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
1682                unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       SPITBF          : 1;
            volatile unsigned       SPIRBF          : 1;
    } SPI1STATbits @ 0x220;
1687
    static volatile unsigned int  SPI1CON           @ 0x222;
    static volatile bit           SPI1_PPRE0        @ ((unsigned)&SPI1CON*8)+0;
    static volatile bit           SPI1_PPRE1        @ ((unsigned)&SPI1CON*8)+1;
```

```
    static volatile bit           SPI1_SPRE0        @ ((unsigned)&SPI1CON*8)+2;
1692 static volatile bit           SPI1_SPRE1        @ ((unsigned)&SPI1CON*8)+3;
    static volatile bit           SPI1_SPRE2        @ ((unsigned)&SPI1CON*8)+4;
    static volatile bit           SPI1_MSTEN        @ ((unsigned)&SPI1CON*8)+5;
    static volatile bit           SPI1_CKP          @ ((unsigned)&SPI1CON*8)+6;
    static volatile bit           SPI1_SSEN         @ ((unsigned)&SPI1CON*8)+7;
1697 static volatile bit           SPI1_CKE          @ ((unsigned)&SPI1CON*8)+8;
    static volatile bit           SPI1_SMP          @ ((unsigned)&SPI1CON*8)+9;
    static volatile bit           SPI1_MODE16       @ ((unsigned)&SPI1CON*8)+10;
    static volatile bit           SPI1_DISSDO       @ ((unsigned)&SPI1CON*8)+11;
    static volatile bit           SPI1_FSD          @ ((unsigned)&SPI1CON*8)+13;
1702 static volatile bit           SPI1_FRMEN        @ ((unsigned)&SPI1CON*8)+14;
    /* Microchip compatible bit field */
    static volatile struct {
                     unsigned                       : 1;
            volatile unsigned      FRMEN            : 1;
1707        volatile unsigned      SPIFSD           : 1;
                     unsigned                       : 1;
            volatile unsigned      DISSDO           : 1;
            volatile unsigned      MODE16           : 1;
            volatile unsigned      SMP              : 1;
1712        volatile unsigned      CKE              : 1;
            volatile unsigned      SSEN             : 1;
            volatile unsigned      CKP              : 1;
            volatile unsigned      MSTEN            : 1;
            volatile unsigned      SPRE2            : 1;
1717        volatile unsigned      SPRE1            : 1;
            volatile unsigned      SPRE0            : 1;
            volatile unsigned      PPRE1            : 1;
            volatile unsigned      PPRE0            : 1;
    } SPI1CONbits @ 0x222;
1722
    static volatile unsigned int  SPI1BUF           @ 0x224;
    static volatile unsigned int  ADCBUF0           @ 0x280;
    static volatile unsigned int  ADCBUF1           @ 0x282;
    static volatile unsigned int  ADCBUF2           @ 0x284;
1727 static volatile unsigned int  ADCBUF3           @ 0x286;
    static volatile unsigned int  ADCBUF4           @ 0x288;
    static volatile unsigned int  ADCBUF5           @ 0x28A;
    static volatile unsigned int  ADCBUF6           @ 0x28C;
    static volatile unsigned int  ADCBUF7           @ 0x28E;
1732 static volatile unsigned int  ADCBUF8           @ 0x290;
    static volatile unsigned int  ADCBUF9           @ 0x292;
    static volatile unsigned int  ADCBUFA           @ 0x294;
    static volatile unsigned int  ADCBUFB           @ 0x296;
    static volatile unsigned int  ADCBUFC           @ 0x298;
1737 static volatile unsigned int  ADCBUFD           @ 0x29A;
    static volatile unsigned int  ADCBUFE           @ 0x29C;
    static volatile unsigned int  ADCBUFF           @ 0x29E;
    static volatile unsigned int  ADCON1            @ 0x2A0;
    static volatile bit           DONE              @ ((unsigned)&ADCON1*8)+0;
1742 static volatile bit           SAMP              @ ((unsigned)&ADCON1*8)+1;
    static volatile bit           ASAM              @ ((unsigned)&ADCON1*8)+2;
    static volatile bit           SIMSAM            @ ((unsigned)&ADCON1*8)+3;
    static volatile bit           SSRC0             @ ((unsigned)&ADCON1*8)+5;
    static volatile bit           SSRC1             @ ((unsigned)&ADCON1*8)+6;
1747 static volatile bit           SSRC2             @ ((unsigned)&ADCON1*8)+7;
    static volatile bit           FORM0             @ ((unsigned)&ADCON1*8)+8;
    static volatile bit           FORM1             @ ((unsigned)&ADCON1*8)+9;
    static volatile bit           ADSIDL            @ ((unsigned)&ADCON1*8)+13;
    static volatile bit           ADON              @ ((unsigned)&ADCON1*8)+15;
1752 /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned      ADON             : 1;
                     unsigned                       : 1;
            volatile unsigned      ADSIDL           : 1;
1757                 unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
            volatile unsigned      FORM             : 2;
            volatile unsigned      SSRC             : 3;
1762                 unsigned                       : 1;
            volatile unsigned      SIMSAM           : 1;
            volatile unsigned      ASAM             : 1;
            volatile unsigned      SAMP             : 1;
            volatile unsigned      DONE             : 1;
1767 } ADCON1bits @ 0x2A0;
```

```
    static volatile unsigned int   ADCON2              @ 0x2A2;
    static volatile bit            ALTS                @ ((unsigned)&ADCON2*8)+0;
    static volatile bit            BUFM                @ ((unsigned)&ADCON2*8)+1;
1772 static volatile bit            SMPI0               @ ((unsigned)&ADCON2*8)+2;
    static volatile bit            SMPI1               @ ((unsigned)&ADCON2*8)+3;
    static volatile bit            SMPI2               @ ((unsigned)&ADCON2*8)+4;
    static volatile bit            SMPI3               @ ((unsigned)&ADCON2*8)+5;
    static volatile bit            BUFS                @ ((unsigned)&ADCON2*8)+7;
1777 static volatile bit            CHPS0               @ ((unsigned)&ADCON2*8)+8;
    static volatile bit            CHPS1               @ ((unsigned)&ADCON2*8)+9;
    static volatile bit            CSCNA               @ ((unsigned)&ADCON2*8)+10;
    static volatile bit            OFFCAL              @ ((unsigned)&ADCON2*8)+12;
    static volatile bit            VCFG0               @ ((unsigned)&ADCON2*8)+13;
1782 static volatile bit            VCFG1               @ ((unsigned)&ADCON2*8)+14;
    static volatile bit            VCFG2               @ ((unsigned)&ADCON2*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
           volatile unsigned       VCFG                : 3;
1787       volatile unsigned       OFFCAL              : 1;
                    unsigned                           : 1;
           volatile unsigned       CSCNA               : 1;
           volatile unsigned       CHPS                : 2;
           volatile unsigned       BUFS                : 1;
1792                unsigned                           : 1;
           volatile unsigned       SMPI                : 4;
           volatile unsigned       BUFM                : 1;
           volatile unsigned       ALTS                : 1;
    } ADCON2bits @ 0x2A2;
1797
    static volatile unsigned int   ADCON3              @ 0x2A4;
    static volatile bit            ADCS0               @ ((unsigned)&ADCON3*8)+0;
    static volatile bit            ADCS1               @ ((unsigned)&ADCON3*8)+1;
    static volatile bit            ADCS2               @ ((unsigned)&ADCON3*8)+2;
1802 static volatile bit            ADCS3               @ ((unsigned)&ADCON3*8)+3;
    static volatile bit            ADCS4               @ ((unsigned)&ADCON3*8)+4;
    static volatile bit            ADCS5               @ ((unsigned)&ADCON3*8)+5;
    static volatile bit            ADRC                @ ((unsigned)&ADCON3*8)+7;
    static volatile bit            SAMC0               @ ((unsigned)&ADCON3*8)+8;
1807 static volatile bit            SAMC1               @ ((unsigned)&ADCON3*8)+9;
    static volatile bit            SAMC2               @ ((unsigned)&ADCON3*8)+10;
    static volatile bit            SAMC3               @ ((unsigned)&ADCON3*8)+11;
    static volatile bit            SAMC4               @ ((unsigned)&ADCON3*8)+12;
    /* Microchip compatible bit field */
1812 static volatile struct {
                    unsigned                           : 1;
                    unsigned                           : 1;
                    unsigned                           : 1;
           volatile unsigned       SAMC                : 5;
1817       volatile unsigned       ADRC                : 1;
                    unsigned                           : 1;
           volatile unsigned       ADCS                : 6;
    } ADCON3bits @ 0x2A4;

1822 static volatile unsigned int   ADCHS              @ 0x2A6;
    static volatile bit            CHOSA0              @ ((unsigned)&ADCHS*8)+0;
    static volatile bit            CHOSA1              @ ((unsigned)&ADCHS*8)+1;
    static volatile bit            CHOSA2              @ ((unsigned)&ADCHS*8)+2;
    static volatile bit            CHOSA3              @ ((unsigned)&ADCHS*8)+3;
1827 static volatile bit            CHONA               @ ((unsigned)&ADCHS*8)+4;
    static volatile bit            CHXSA               @ ((unsigned)&ADCHS*8)+5;
    static volatile bit            CHXNA0              @ ((unsigned)&ADCHS*8)+6;
    static volatile bit            CHXNA1              @ ((unsigned)&ADCHS*8)+7;
    static volatile bit            CHOSB0              @ ((unsigned)&ADCHS*8)+8;
1832 static volatile bit            CHOSB1              @ ((unsigned)&ADCHS*8)+9;
    static volatile bit            CHOSB2              @ ((unsigned)&ADCHS*8)+10;
    static volatile bit            CHOSB3              @ ((unsigned)&ADCHS*8)+11;
    static volatile bit            CHONB               @ ((unsigned)&ADCHS*8)+12;
    static volatile bit            CHXSB               @ ((unsigned)&ADCHS*8)+13;
1837 static volatile bit            CHXNB0              @ ((unsigned)&ADCHS*8)+14;
    static volatile bit            CHXNB1              @ ((unsigned)&ADCHS*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
           volatile unsigned       CHXNB               : 2;
1842       volatile unsigned       CHXSB               : 1;
           volatile unsigned       CHONB               : 1;
           volatile unsigned       CHOSB               : 4;
           volatile unsigned       CHXNA               : 2;
           volatile unsigned       CHXSA               : 1;
```

```
1847          volatile unsigned       CH0NA               : 1;
              volatile unsigned       CH0SA               : 4;
      } ADCHSbits @ 0x2A6;

      static volatile unsigned int   ADPCFG              @ 0x2A8;
1852  static volatile unsigned int   ADCSSL              @ 0x2AA;
      static volatile unsigned int   TRISB               @ 0x2C6;
      static volatile bit            TRISB0              @ ((unsigned)&TRISB*8)+0;
      static volatile bit            TRISB1              @ ((unsigned)&TRISB*8)+1;
      static volatile bit            TRISB2              @ ((unsigned)&TRISB*8)+2;
1857  static volatile bit            TRISB3              @ ((unsigned)&TRISB*8)+3;
      static volatile bit            TRISB4              @ ((unsigned)&TRISB*8)+4;
      static volatile bit            TRISB5              @ ((unsigned)&TRISB*8)+5;
      static volatile bit            TRISB6              @ ((unsigned)&TRISB*8)+6;
      static volatile bit            TRISB7              @ ((unsigned)&TRISB*8)+7;
1862  static volatile bit            TRISB8              @ ((unsigned)&TRISB*8)+8;
      /* Microchip compatible bit field */
      static volatile struct {
                      unsigned                           : 1;
                      unsigned                           : 1;
1867                  unsigned                           : 1;
                      unsigned                           : 1;
                      unsigned                           : 1;
                      unsigned                           : 1;
                      unsigned                           : 1;
1872          volatile unsigned       TRISB8              : 1;
              volatile unsigned       TRISB7              : 1;
              volatile unsigned       TRISB6              : 1;
              volatile unsigned       TRISB5              : 1;
              volatile unsigned       TRISB4              : 1;
1877          volatile unsigned       TRISB3              : 1;
              volatile unsigned       TRISB2              : 1;
              volatile unsigned       TRISB1              : 1;
              volatile unsigned       TRISB0              : 1;
      } TRISBbits @ 0x2C6;
1882
      static volatile unsigned int   PORTB               @ 0x2C8;
      static volatile bit            RB0                 @ ((unsigned)&PORTB*8)+0;
      static volatile bit            RB1                 @ ((unsigned)&PORTB*8)+1;
      static volatile bit            RB2                 @ ((unsigned)&PORTB*8)+2;
1887  static volatile bit            RB3                 @ ((unsigned)&PORTB*8)+3;
      static volatile bit            RB4                 @ ((unsigned)&PORTB*8)+4;
      static volatile bit            RB5                 @ ((unsigned)&PORTB*8)+5;
      static volatile bit            RB6                 @ ((unsigned)&PORTB*8)+6;
      static volatile bit            RB7                 @ ((unsigned)&PORTB*8)+7;
1892  static volatile bit            RB8                 @ ((unsigned)&PORTB*8)+8;
      /* Microchip compatible bit field */
      static volatile struct {
                      unsigned                           : 1;
                      unsigned                           : 1;
1897                  unsigned                           : 1;
                      unsigned                           : 1;
                      unsigned                           : 1;
                      unsigned                           : 1;
                      unsigned                           : 1;
1902          volatile unsigned       RB8                 : 1;
              volatile unsigned       RB7                 : 1;
              volatile unsigned       RB6                 : 1;
              volatile unsigned       RB5                 : 1;
              volatile unsigned       RB4                 : 1;
1907          volatile unsigned       RB3                 : 1;
              volatile unsigned       RB2                 : 1;
              volatile unsigned       RB1                 : 1;
              volatile unsigned       RB0                 : 1;
      } PORTBbits @ 0x2C8;
1912
      static volatile unsigned int   LATB                @ 0x2CA;
      static volatile bit            LATB0               @ ((unsigned)&LATB*8)+0;
      static volatile bit            LATB1               @ ((unsigned)&LATB*8)+1;
      static volatile bit            LATB2               @ ((unsigned)&LATB*8)+2;
1917  static volatile bit            LATB3               @ ((unsigned)&LATB*8)+3;
      static volatile bit            LATB4               @ ((unsigned)&LATB*8)+4;
      static volatile bit            LATB5               @ ((unsigned)&LATB*8)+5;
      static volatile bit            LATB6               @ ((unsigned)&LATB*8)+6;
      static volatile bit            LATB7               @ ((unsigned)&LATB*8)+7;
1922  static volatile bit            LATB8               @ ((unsigned)&LATB*8)+8;
      /* Microchip compatible bit field */
      static volatile struct {
```

```
                    unsigned                        : 1;
                    unsigned                        : 1;
1927                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
1932        volatile unsigned       LATB8           : 1;
            volatile unsigned       LATB7           : 1;
            volatile unsigned       LATB6           : 1;
            volatile unsigned       LATB5           : 1;
            volatile unsigned       LATB4           : 1;
1937        volatile unsigned       LATB3           : 1;
            volatile unsigned       LATB2           : 1;
            volatile unsigned       LATB1           : 1;
            volatile unsigned       LATB0           : 1;
    } LATBbits @ 0x2CA;
1942
    static volatile unsigned int  TRISC             @ 0x2CC;
    static volatile bit           TRISC13           @ ((unsigned)&TRISC*8)+13;
    static volatile bit           TRISC14           @ ((unsigned)&TRISC*8)+14;
    static volatile bit           TRISC15           @ ((unsigned)&TRISC*8)+15;
1947 /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned       TRISC15         : 1;
            volatile unsigned       TRISC14         : 1;
            volatile unsigned       TRISC13         : 1;
1952                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
1957                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
1962                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
    } TRISCbits @ 0x2CC;

1967 static volatile unsigned int  PORTC             @ 0x2CE;
    static volatile bit           RC13              @ ((unsigned)&PORTC*8)+13;
    static volatile bit           RC14              @ ((unsigned)&PORTC*8)+14;
    static volatile bit           RC15              @ ((unsigned)&PORTC*8)+15;
    /* Microchip compatible bit field */
1972 static volatile struct {
            volatile unsigned       RC15            : 1;
            volatile unsigned       RC14            : 1;
            volatile unsigned       RC13            : 1;
                    unsigned                        : 1;
1977                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
1982                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
1987                unsigned                        : 1;
                    unsigned                        : 1;
    } PORTCbits @ 0x2CE;

    static volatile unsigned int  LATC              @ 0x2D0;
1992 static volatile bit           LATC13            @ ((unsigned)&LATC*8)+13;
    static volatile bit           LATC14            @ ((unsigned)&LATC*8)+14;
    static volatile bit           LATC15            @ ((unsigned)&LATC*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
1997        volatile unsigned       LATC15          : 1;
            volatile unsigned       LATC14          : 1;
            volatile unsigned       LATC13          : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2002                unsigned                        : 1;
```

```
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2007                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2012                unsigned                        : 1;
    } LATCbits @ 0x2D0;

    static volatile unsigned int  TRISD             @ 0x2D2;
    static volatile bit           TRISD0            @ ((unsigned)&TRISD*8)+0;
2017 static volatile bit           TRISD1            @ ((unsigned)&TRISD*8)+1;
    static volatile bit           TRISD2            @ ((unsigned)&TRISD*8)+2;
    static volatile bit           TRISD3            @ ((unsigned)&TRISD*8)+3;
    /* Microchip compatible bit field */
    static volatile struct {
2022                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2027                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2032                unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       TRISD3          : 1;
            volatile unsigned       TRISD2          : 1;
            volatile unsigned       TRISD1          : 1;
2037        volatile unsigned       TRISD0          : 1;
    } TRISDbits @ 0x2D2;

    static volatile unsigned int  PORTD             @ 0x2D4;
    static volatile bit           RD0               @ ((unsigned)&PORTD*8)+0;
2042 static volatile bit           RD1               @ ((unsigned)&PORTD*8)+1;
    static volatile bit           RD2               @ ((unsigned)&PORTD*8)+2;
    static volatile bit           RD3               @ ((unsigned)&PORTD*8)+3;
    /* Microchip compatible bit field */
    static volatile struct {
2047                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2052                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2057                unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       RD3             : 1;
            volatile unsigned       RD2             : 1;
            volatile unsigned       RD1             : 1;
2062        volatile unsigned       RD0             : 1;
    } PORTDbits @ 0x2D4;

    static volatile unsigned int  LATD              @ 0x2D6;
    static volatile bit           LATD0             @ ((unsigned)&LATD*8)+0;
2067 static volatile bit           LATD1             @ ((unsigned)&LATD*8)+1;
    static volatile bit           LATD2             @ ((unsigned)&LATD*8)+2;
    static volatile bit           LATD3             @ ((unsigned)&LATD*8)+3;
    /* Microchip compatible bit field */
    static volatile struct {
2072                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2077                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
```

```
                    unsigned                        : 1;
2082                unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       LATD3           : 1;
            volatile unsigned       LATD2           : 1;
            volatile unsigned       LATD1           : 1;
2087        volatile unsigned       LATD0           : 1;
    } LATDbits @ 0x2D6;

    static volatile unsigned int  TRISE             @ 0x2D8;
    static volatile bit           TRISE0            @ ((unsigned)&TRISE*8)+0;
2092 static volatile bit           TRISE1            @ ((unsigned)&TRISE*8)+1;
    static volatile bit           TRISE2            @ ((unsigned)&TRISE*8)+2;
    static volatile bit           TRISE3            @ ((unsigned)&TRISE*8)+3;
    static volatile bit           TRISE4            @ ((unsigned)&TRISE*8)+4;
    static volatile bit           TRISE5            @ ((unsigned)&TRISE*8)+5;
2097 static volatile bit           TRISE8            @ ((unsigned)&TRISE*8)+8;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                        : 1;
                    unsigned                        : 1;
2102                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2107        volatile unsigned       TRISE8          : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       TRISE5          : 1;
            volatile unsigned       TRISE4          : 1;
2112        volatile unsigned       TRISE3          : 1;
            volatile unsigned       TRISE2          : 1;
            volatile unsigned       TRISE1          : 1;
            volatile unsigned       TRISE0          : 1;
    } TRISEbits @ 0x2D8;
2117
    static volatile unsigned int  PORTE             @ 0x2DA;
    static volatile bit           RE0               @ ((unsigned)&PORTE*8)+0;
    static volatile bit           RE1               @ ((unsigned)&PORTE*8)+1;
    static volatile bit           RE2               @ ((unsigned)&PORTE*8)+2;
2122 static volatile bit           RE3               @ ((unsigned)&PORTE*8)+3;
    static volatile bit           RE4               @ ((unsigned)&PORTE*8)+4;
    static volatile bit           RE5               @ ((unsigned)&PORTE*8)+5;
    static volatile bit           RE8               @ ((unsigned)&PORTE*8)+8;
    /* Microchip compatible bit field */
2127 static volatile struct {
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2132                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned       RE8             : 1;
                    unsigned                        : 1;
2137                unsigned                        : 1;
            volatile unsigned       RE5             : 1;
            volatile unsigned       RE4             : 1;
            volatile unsigned       RE3             : 1;
            volatile unsigned       RE2             : 1;
2142        volatile unsigned       RE1             : 1;
            volatile unsigned       RE0             : 1;
    } PORTEbits @ 0x2DA;

    static volatile unsigned int  LATE              @ 0x2DC;
2147 static volatile bit           LATE0             @ ((unsigned)&LATE*8)+0;
    static volatile bit           LATE1             @ ((unsigned)&LATE*8)+1;
    static volatile bit           LATE2             @ ((unsigned)&LATE*8)+2;
    static volatile bit           LATE3             @ ((unsigned)&LATE*8)+3;
    static volatile bit           LATE4             @ ((unsigned)&LATE*8)+4;
2152 static volatile bit           LATE5             @ ((unsigned)&LATE*8)+5;
    static volatile bit           LATE8             @ ((unsigned)&LATE*8)+8;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                        : 1;
2157                unsigned                        : 1;
                    unsigned                        : 1;
```

```
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2162                unsigned                        : 1;
          volatile  unsigned        LATE8           : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
          volatile  unsigned        LATE5           : 1;
2167      volatile  unsigned        LATE4           : 1;
          volatile  unsigned        LATE3           : 1;
          volatile  unsigned        LATE2           : 1;
          volatile  unsigned        LATE1           : 1;
          volatile  unsigned        LATE0           : 1;
2172 } LATEbits @ 0x2DC;

     static volatile unsigned int   TRISF           @ 0x2DE;
     static volatile bit            TRISF0          @ ((unsigned)&TRISF*8)+0;
     static volatile bit            TRISF1          @ ((unsigned)&TRISF*8)+1;
2177 static volatile bit            TRISF2          @ ((unsigned)&TRISF*8)+2;
     static volatile bit            TRISF3          @ ((unsigned)&TRISF*8)+3;
     static volatile bit            TRISF4          @ ((unsigned)&TRISF*8)+4;
     static volatile bit            TRISF5          @ ((unsigned)&TRISF*8)+5;
     static volatile bit            TRISF6          @ ((unsigned)&TRISF*8)+6;
2182 /* Microchip compatible bit field */
     static volatile struct {
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2187                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2192                unsigned                        : 1;
          volatile  unsigned        TRISF6          : 1;
          volatile  unsigned        TRISF5          : 1;
          volatile  unsigned        TRISF4          : 1;
          volatile  unsigned        TRISF3          : 1;
2197      volatile  unsigned        TRISF2          : 1;
          volatile  unsigned        TRISF1          : 1;
          volatile  unsigned        TRISF0          : 1;
     } TRISFbits @ 0x2DE;

2202 static volatile unsigned int   PORTF           @ 0x2E0;
     static volatile bit            RF0             @ ((unsigned)&PORTF*8)+0;
     static volatile bit            RF1             @ ((unsigned)&PORTF*8)+1;
     static volatile bit            RF2             @ ((unsigned)&PORTF*8)+2;
     static volatile bit            RF3             @ ((unsigned)&PORTF*8)+3;
2207 static volatile bit            RF4             @ ((unsigned)&PORTF*8)+4;
     static volatile bit            RF5             @ ((unsigned)&PORTF*8)+5;
     static volatile bit            RF6             @ ((unsigned)&PORTF*8)+6;
     /* Microchip compatible bit field */
     static volatile struct {
2212                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
2217                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
          volatile  unsigned        RF6             : 1;
2222      volatile  unsigned        RF5             : 1;
          volatile  unsigned        RF4             : 1;
          volatile  unsigned        RF3             : 1;
          volatile  unsigned        RF2             : 1;
          volatile  unsigned        RF1             : 1;
2227      volatile  unsigned        RF0             : 1;
     } PORTFbits @ 0x2E0;

     static volatile unsigned int   LATF            @ 0x2E2;
     static volatile bit            LATF0           @ ((unsigned)&LATF*8)+0;
2232 static volatile bit            LATF1           @ ((unsigned)&LATF*8)+1;
     static volatile bit            LATF2           @ ((unsigned)&LATF*8)+2;
     static volatile bit            LATF3           @ ((unsigned)&LATF*8)+3;
     static volatile bit            LATF4           @ ((unsigned)&LATF*8)+4;
     static volatile bit            LATF5           @ ((unsigned)&LATF*8)+5;
```

```
2237 static volatile bit           LATF6              @ ((unsigned)&LATF*8)+6;
     /* Microchip compatible bit field */
     static volatile struct {
                     unsigned                        : 1;
                     unsigned                        : 1;
2242                 unsigned                        : 1;
                     unsigned                        : 1;
                     unsigned                        : 1;
                     unsigned                        : 1;
                     unsigned                        : 1;
2247                 unsigned                        : 1;
                     unsigned                        : 1;
             volatile unsigned       LATF6           : 1;
             volatile unsigned       LATF5           : 1;
             volatile unsigned       LATF4           : 1;
2252         volatile unsigned       LATF3           : 1;
             volatile unsigned       LATF2           : 1;
             volatile unsigned       LATF1           : 1;
             volatile unsigned       LATF0           : 1;
     } LATFbits @ 0x2E2;
2257
     static volatile unsigned int  C1RXF0SID         @ 0x300;
     static volatile bit           C1RXF0_EXIDE      @ ((unsigned)&C1RXF0SID*8)+0;
     static volatile bit           C1RXF0_SID0       @ ((unsigned)&C1RXF0SID*8)+2;
     static volatile bit           C1RXF0_SID1       @ ((unsigned)&C1RXF0SID*8)+3;
2262 static volatile bit           C1RXF0_SID2       @ ((unsigned)&C1RXF0SID*8)+4;
     static volatile bit           C1RXF0_SID3       @ ((unsigned)&C1RXF0SID*8)+5;
     static volatile bit           C1RXF0_SID4       @ ((unsigned)&C1RXF0SID*8)+6;
     static volatile bit           C1RXF0_SID5       @ ((unsigned)&C1RXF0SID*8)+7;
     static volatile bit           C1RXF0_SID6       @ ((unsigned)&C1RXF0SID*8)+8;
2267 static volatile bit           C1RXF0_SID7       @ ((unsigned)&C1RXF0SID*8)+9;
     static volatile bit           C1RXF0_SID8       @ ((unsigned)&C1RXF0SID*8)+10;
     static volatile bit           C1RXF0_SID9       @ ((unsigned)&C1RXF0SID*8)+11;
     static volatile bit           C1RXF0_SID10      @ ((unsigned)&C1RXF0SID*8)+12;
     /* Microchip compatible bit field */
2272 static volatile struct {
                     unsigned                        : 1;
                     unsigned                        : 1;
                     unsigned                        : 1;
             volatile unsigned       SID10           : 1;
2277         volatile unsigned       SID9            : 1;
             volatile unsigned       SID8            : 1;
             volatile unsigned       SID7            : 1;
             volatile unsigned       SID6            : 1;
             volatile unsigned       SID5            : 1;
2282         volatile unsigned       SID4            : 1;
             volatile unsigned       SID3            : 1;
             volatile unsigned       SID2            : 1;
             volatile unsigned       SID1            : 1;
             volatile unsigned       SID0            : 1;
2287                 unsigned                        : 1;
             volatile unsigned       EXIDE           : 1;
     } C1RXF0SIDbits @ 0x300;

     static volatile unsigned int  C1RXF0EIDH        @ 0x302;
2292 static volatile unsigned int  C1RXF0EIDL        @ 0x304;
     static volatile unsigned int  C1RXF1SID         @ 0x308;
     static volatile bit           C1RXF1_EXIDE      @ ((unsigned)&C1RXF1SID*8)+0;
     static volatile bit           C1RXF1_SID0       @ ((unsigned)&C1RXF1SID*8)+2;
     static volatile bit           C1RXF1_SID1       @ ((unsigned)&C1RXF1SID*8)+3;
2297 static volatile bit           C1RXF1_SID2       @ ((unsigned)&C1RXF1SID*8)+4;
     static volatile bit           C1RXF1_SID3       @ ((unsigned)&C1RXF1SID*8)+5;
     static volatile bit           C1RXF1_SID4       @ ((unsigned)&C1RXF1SID*8)+6;
     static volatile bit           C1RXF1_SID5       @ ((unsigned)&C1RXF1SID*8)+7;
     static volatile bit           C1RXF1_SID6       @ ((unsigned)&C1RXF1SID*8)+8;
2302 static volatile bit           C1RXF1_SID7       @ ((unsigned)&C1RXF1SID*8)+9;
     static volatile bit           C1RXF1_SID8       @ ((unsigned)&C1RXF1SID*8)+10;
     static volatile bit           C1RXF1_SID9       @ ((unsigned)&C1RXF1SID*8)+11;
     static volatile bit           C1RXF1_SID10      @ ((unsigned)&C1RXF1SID*8)+12;
     /* Microchip compatible bit field */
2307 static volatile struct {
                     unsigned                        : 1;
                     unsigned                        : 1;
                     unsigned                        : 1;
             volatile unsigned       SID10           : 1;
2312         volatile unsigned       SID9            : 1;
             volatile unsigned       SID8            : 1;
             volatile unsigned       SID7            : 1;
```

```
            volatile unsigned       SID6                : 1;
            volatile unsigned       SID5                : 1;
2317        volatile unsigned       SID4                : 1;
            volatile unsigned       SID3                : 1;
            volatile unsigned       SID2                : 1;
            volatile unsigned       SID1                : 1;
            volatile unsigned       SID0                : 1;
2322                 unsigned                           : 1;
            volatile unsigned       EXIDE               : 1;
    } C1RXF1SIDbits @ 0x308;

    static volatile unsigned int   C1RXF1EIDH          @ 0x30A;
2327 static volatile unsigned int   C1RXF1EIDL          @ 0x30C;
    static volatile unsigned int   C1RXF2SID           @ 0x310;
    static volatile bit            C1RXF2_EXIDE        @ ((unsigned)&C1RXF2SID*8)+0;
    static volatile bit            C1RXF2_SID0         @ ((unsigned)&C1RXF2SID*8)+2;
    static volatile bit            C1RXF2_SID1         @ ((unsigned)&C1RXF2SID*8)+3;
2332 static volatile bit            C1RXF2_SID2         @ ((unsigned)&C1RXF2SID*8)+4;
    static volatile bit            C1RXF2_SID3         @ ((unsigned)&C1RXF2SID*8)+5;
    static volatile bit            C1RXF2_SID4         @ ((unsigned)&C1RXF2SID*8)+6;
    static volatile bit            C1RXF2_SID5         @ ((unsigned)&C1RXF2SID*8)+7;
    static volatile bit            C1RXF2_SID6         @ ((unsigned)&C1RXF2SID*8)+8;
2337 static volatile bit            C1RXF2_SID7         @ ((unsigned)&C1RXF2SID*8)+9;
    static volatile bit            C1RXF2_SID8         @ ((unsigned)&C1RXF2SID*8)+10;
    static volatile bit            C1RXF2_SID9         @ ((unsigned)&C1RXF2SID*8)+11;
    static volatile bit            C1RXF2_SID10        @ ((unsigned)&C1RXF2SID*8)+12;
    /* Microchip compatible bit field */
2342 static volatile struct {
                     unsigned                          : 1;
                     unsigned                          : 1;
                     unsigned                          : 1;
            volatile unsigned       SID10               : 1;
2347        volatile unsigned       SID9                : 1;
            volatile unsigned       SID8                : 1;
            volatile unsigned       SID7                : 1;
            volatile unsigned       SID6                : 1;
            volatile unsigned       SID5                : 1;
2352        volatile unsigned       SID4                : 1;
            volatile unsigned       SID3                : 1;
            volatile unsigned       SID2                : 1;
            volatile unsigned       SID1                : 1;
            volatile unsigned       SID0                : 1;
2357                 unsigned                           : 1;
            volatile unsigned       EXIDE               : 1;
    } C1RXF2SIDbits @ 0x310;

    static volatile unsigned int   C1RXF2EIDH          @ 0x312;
2362 static volatile unsigned int   C1RXF2EIDL          @ 0x314;
    static volatile unsigned int   C1RXF3SID           @ 0x318;
    static volatile bit            C1RXF3_EXIDE        @ ((unsigned)&C1RXF3SID*8)+0;
    static volatile bit            C1RXF3_SID0         @ ((unsigned)&C1RXF3SID*8)+2;
    static volatile bit            C1RXF3_SID1         @ ((unsigned)&C1RXF3SID*8)+3;
2367 static volatile bit            C1RXF3_SID2         @ ((unsigned)&C1RXF3SID*8)+4;
    static volatile bit            C1RXF3_SID3         @ ((unsigned)&C1RXF3SID*8)+5;
    static volatile bit            C1RXF3_SID4         @ ((unsigned)&C1RXF3SID*8)+6;
    static volatile bit            C1RXF3_SID5         @ ((unsigned)&C1RXF3SID*8)+7;
    static volatile bit            C1RXF3_SID6         @ ((unsigned)&C1RXF3SID*8)+8;
2372 static volatile bit            C1RXF3_SID7         @ ((unsigned)&C1RXF3SID*8)+9;
    static volatile bit            C1RXF3_SID8         @ ((unsigned)&C1RXF3SID*8)+10;
    static volatile bit            C1RXF3_SID9         @ ((unsigned)&C1RXF3SID*8)+11;
    static volatile bit            C1RXF3_SID10        @ ((unsigned)&C1RXF3SID*8)+12;
    /* Microchip compatible bit field */
2377 static volatile struct {
                     unsigned                          : 1;
                     unsigned                          : 1;
                     unsigned                          : 1;
            volatile unsigned       SID10               : 1;
2382        volatile unsigned       SID9                : 1;
            volatile unsigned       SID8                : 1;
            volatile unsigned       SID7                : 1;
            volatile unsigned       SID6                : 1;
            volatile unsigned       SID5                : 1;
2387        volatile unsigned       SID4                : 1;
            volatile unsigned       SID3                : 1;
            volatile unsigned       SID2                : 1;
            volatile unsigned       SID1                : 1;
            volatile unsigned       SID0                : 1;
2392                 unsigned                           : 1;
```

```
        volatile unsigned       EXIDE               : 1;
    } C1RXF3SIDbits @ 0x318;

    static volatile unsigned int  C1RXF3EIDH          @ 0x31A;
    static volatile unsigned int  C1RXF3EIDL          @ 0x31C;
    static volatile unsigned int  C1RXF4SID           @ 0x320;
    static volatile bit           C1RXF4_EXIDE        @ ((unsigned)&C1RXF4SID*8)+0;
    static volatile bit           C1RXF4_SID0         @ ((unsigned)&C1RXF4SID*8)+2;
    static volatile bit           C1RXF4_SID1         @ ((unsigned)&C1RXF4SID*8)+3;
    static volatile bit           C1RXF4_SID2         @ ((unsigned)&C1RXF4SID*8)+4;
    static volatile bit           C1RXF4_SID3         @ ((unsigned)&C1RXF4SID*8)+5;
    static volatile bit           C1RXF4_SID4         @ ((unsigned)&C1RXF4SID*8)+6;
    static volatile bit           C1RXF4_SID5         @ ((unsigned)&C1RXF4SID*8)+7;
    static volatile bit           C1RXF4_SID6         @ ((unsigned)&C1RXF4SID*8)+8;
    static volatile bit           C1RXF4_SID7         @ ((unsigned)&C1RXF4SID*8)+9;
    static volatile bit           C1RXF4_SID8         @ ((unsigned)&C1RXF4SID*8)+10;
    static volatile bit           C1RXF4_SID9         @ ((unsigned)&C1RXF4SID*8)+11;
    static volatile bit           C1RXF4_SID10        @ ((unsigned)&C1RXF4SID*8)+12;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
            volatile unsigned       SID10             : 1;
            volatile unsigned       SID9              : 1;
            volatile unsigned       SID8              : 1;
            volatile unsigned       SID7              : 1;
            volatile unsigned       SID6              : 1;
            volatile unsigned       SID5              : 1;
            volatile unsigned       SID4              : 1;
            volatile unsigned       SID3              : 1;
            volatile unsigned       SID2              : 1;
            volatile unsigned       SID1              : 1;
            volatile unsigned       SID0              : 1;
                    unsigned                          : 1;
            volatile unsigned       EXIDE             : 1;
    } C1RXF4SIDbits @ 0x320;

    static volatile unsigned int  C1RXF4EIDH          @ 0x322;
    static volatile unsigned int  C1RXF4EIDL          @ 0x324;
    static volatile unsigned int  C1RXF5SID           @ 0x328;
    static volatile bit           C1RXF5_EXIDE        @ ((unsigned)&C1RXF5SID*8)+0;
    static volatile bit           C1RXF5_SID0         @ ((unsigned)&C1RXF5SID*8)+2;
    static volatile bit           C1RXF5_SID1         @ ((unsigned)&C1RXF5SID*8)+3;
    static volatile bit           C1RXF5_SID2         @ ((unsigned)&C1RXF5SID*8)+4;
    static volatile bit           C1RXF5_SID3         @ ((unsigned)&C1RXF5SID*8)+5;
    static volatile bit           C1RXF5_SID4         @ ((unsigned)&C1RXF5SID*8)+6;
    static volatile bit           C1RXF5_SID5         @ ((unsigned)&C1RXF5SID*8)+7;
    static volatile bit           C1RXF5_SID6         @ ((unsigned)&C1RXF5SID*8)+8;
    static volatile bit           C1RXF5_SID7         @ ((unsigned)&C1RXF5SID*8)+9;
    static volatile bit           C1RXF5_SID8         @ ((unsigned)&C1RXF5SID*8)+10;
    static volatile bit           C1RXF5_SID9         @ ((unsigned)&C1RXF5SID*8)+11;
    static volatile bit           C1RXF5_SID10        @ ((unsigned)&C1RXF5SID*8)+12;
    /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
            volatile unsigned       SID10             : 1;
            volatile unsigned       SID9              : 1;
            volatile unsigned       SID8              : 1;
            volatile unsigned       SID7              : 1;
            volatile unsigned       SID6              : 1;
            volatile unsigned       SID5              : 1;
            volatile unsigned       SID4              : 1;
            volatile unsigned       SID3              : 1;
            volatile unsigned       SID2              : 1;
            volatile unsigned       SID1              : 1;
            volatile unsigned       SID0              : 1;
                    unsigned                          : 1;
            volatile unsigned       EXIDE             : 1;
    } C1RXF5SIDbits @ 0x328;

    static volatile unsigned int  C1RXF5EIDH          @ 0x32A;
    static volatile unsigned int  C1RXF5EIDL          @ 0x32C;
    static volatile unsigned int  C1RXM0SID           @ 0x330;
    static volatile bit           C1RXM0_MIDE         @ ((unsigned)&C1RXM0SID*8)+0;
    static volatile bit           C1RXM0_SID0         @ ((unsigned)&C1RXM0SID*8)+2;
```

```
    static volatile bit           C1RXM0_SID1       @ ((unsigned)&C1RXM0SID*8)+3;
2472 static volatile bit           C1RXM0_SID2       @ ((unsigned)&C1RXM0SID*8)+4;
    static volatile bit           C1RXM0_SID3       @ ((unsigned)&C1RXM0SID*8)+5;
    static volatile bit           C1RXM0_SID4       @ ((unsigned)&C1RXM0SID*8)+6;
    static volatile bit           C1RXM0_SID5       @ ((unsigned)&C1RXM0SID*8)+7;
    static volatile bit           C1RXM0_SID6       @ ((unsigned)&C1RXM0SID*8)+8;
2477 static volatile bit           C1RXM0_SID7       @ ((unsigned)&C1RXM0SID*8)+9;
    static volatile bit           C1RXM0_SID8       @ ((unsigned)&C1RXM0SID*8)+10;
    static volatile bit           C1RXM0_SID9       @ ((unsigned)&C1RXM0SID*8)+11;
    static volatile bit           C1RXM0_SID10      @ ((unsigned)&C1RXM0SID*8)+12;
    /* Microchip compatible bit field */
2482 static volatile struct {
                  unsigned                        : 1;
                  unsigned                        : 1;
                  unsigned                        : 1;
           volatile unsigned      SID10            : 1;
2487       volatile unsigned      SID9             : 1;
           volatile unsigned      SID8             : 1;
           volatile unsigned      SID7             : 1;
           volatile unsigned      SID6             : 1;
           volatile unsigned      SID5             : 1;
2492       volatile unsigned      SID4             : 1;
           volatile unsigned      SID3             : 1;
           volatile unsigned      SID2             : 1;
           volatile unsigned      SID1             : 1;
           volatile unsigned      SID0             : 1;
2497              unsigned                        : 1;
           volatile unsigned      MIDE             : 1;
    } C1RXM0SIDbits @ 0x330;

    static volatile unsigned int  C1RXM0EIDH        @ 0x332;
2502 static volatile unsigned int  C1RXM0EIDL        @ 0x334;
    static volatile unsigned int  C1RXM1SID         @ 0x338;
    static volatile bit           C1RXM1_MIDE       @ ((unsigned)&C1RXM1SID*8)+0;
    static volatile bit           C1RXM1_SID0       @ ((unsigned)&C1RXM1SID*8)+2;
    static volatile bit           C1RXM1_SID1       @ ((unsigned)&C1RXM1SID*8)+3;
2507 static volatile bit           C1RXM1_SID2       @ ((unsigned)&C1RXM1SID*8)+4;
    static volatile bit           C1RXM1_SID3       @ ((unsigned)&C1RXM1SID*8)+5;
    static volatile bit           C1RXM1_SID4       @ ((unsigned)&C1RXM1SID*8)+6;
    static volatile bit           C1RXM1_SID5       @ ((unsigned)&C1RXM1SID*8)+7;
    static volatile bit           C1RXM1_SID6       @ ((unsigned)&C1RXM1SID*8)+8;
2512 static volatile bit           C1RXM1_SID7       @ ((unsigned)&C1RXM1SID*8)+9;
    static volatile bit           C1RXM1_SID8       @ ((unsigned)&C1RXM1SID*8)+10;
    static volatile bit           C1RXM1_SID9       @ ((unsigned)&C1RXM1SID*8)+11;
    static volatile bit           C1RXM1_SID10      @ ((unsigned)&C1RXM1SID*8)+12;
    /* Microchip compatible bit field */
2517 static volatile struct {
                  unsigned                        : 1;
                  unsigned                        : 1;
                  unsigned                        : 1;
           volatile unsigned      SID10            : 1;
2522       volatile unsigned      SID9             : 1;
           volatile unsigned      SID8             : 1;
           volatile unsigned      SID7             : 1;
           volatile unsigned      SID6             : 1;
           volatile unsigned      SID5             : 1;
2527       volatile unsigned      SID4             : 1;
           volatile unsigned      SID3             : 1;
           volatile unsigned      SID2             : 1;
           volatile unsigned      SID1             : 1;
           volatile unsigned      SID0             : 1;
2532              unsigned                        : 1;
           volatile unsigned      MIDE             : 1;
    } C1RXM1SIDbits @ 0x338;

    static volatile unsigned int  C1RXM1EIDH        @ 0x33A;
2537 static volatile unsigned int  C1RXM1EIDL        @ 0x33C;
    static volatile unsigned int  C1TX2SID          @ 0x340;
    static volatile bit           C1TX2_TXIDE       @ ((unsigned)&C1TX2SID*8)+0;
    static volatile bit           C1TX2_SRR         @ ((unsigned)&C1TX2SID*8)+1;
    static volatile bit           C1TX2_SID0        @ ((unsigned)&C1TX2SID*8)+2;
2542 static volatile bit           C1TX2_SID1        @ ((unsigned)&C1TX2SID*8)+3;
    static volatile bit           C1TX2_SID2        @ ((unsigned)&C1TX2SID*8)+4;
    static volatile bit           C1TX2_SID3        @ ((unsigned)&C1TX2SID*8)+5;
    static volatile bit           C1TX2_SID4        @ ((unsigned)&C1TX2SID*8)+6;
    static volatile bit           C1TX2_SID5        @ ((unsigned)&C1TX2SID*8)+7;
2547 static volatile bit           C1TX2_SID6        @ ((unsigned)&C1TX2SID*8)+11;
    static volatile bit           C1TX2_SID7        @ ((unsigned)&C1TX2SID*8)+12;
```

```
    static volatile bit           C1TX2_SID8          @ ((unsigned)&C1TX2SID*8)+13;
    static volatile bit           C1TX2_SID9          @ ((unsigned)&C1TX2SID*8)+14;
    static volatile bit           C1TX2_SID10         @ ((unsigned)&C1TX2SID*8)+15;
/* Microchip compatible bit field */
    static volatile struct {
           volatile unsigned      SID10               : 1;
           volatile unsigned      SID9                : 1;
           volatile unsigned      SID8                : 1;
           volatile unsigned      SID7                : 1;
           volatile unsigned      SID6                : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
           volatile unsigned      SID5                : 1;
           volatile unsigned      SID4                : 1;
           volatile unsigned      SID3                : 1;
           volatile unsigned      SID2                : 1;
           volatile unsigned      SID1                : 1;
           volatile unsigned      SID0                : 1;
           volatile unsigned      SRR                 : 1;
           volatile unsigned      TXIDE               : 1;
    } C1TX2SIDbits @ 0x340;

    static volatile unsigned int  C1TX2EID            @ 0x342;
    static volatile unsigned int  C1TX2DLC            @ 0x344;
    static volatile bit           C1TX2_DLC0          @ ((unsigned)&C1TX2DLC*8)+3;
    static volatile bit           C1TX2_DLC1          @ ((unsigned)&C1TX2DLC*8)+4;
    static volatile bit           C1TX2_DLC2          @ ((unsigned)&C1TX2DLC*8)+5;
    static volatile bit           C1TX2_DLC3          @ ((unsigned)&C1TX2DLC*8)+6;
    static volatile bit           C1TX2_TXRB0         @ ((unsigned)&C1TX2DLC*8)+7;
    static volatile bit           C1TX2_TXRB1         @ ((unsigned)&C1TX2DLC*8)+8;
    static volatile bit           C1TX2_TXRTR         @ ((unsigned)&C1TX2DLC*8)+9;
    static volatile bit           C1TX2_EID0          @ ((unsigned)&C1TX2DLC*8)+10;
    static volatile bit           C1TX2_EID1          @ ((unsigned)&C1TX2DLC*8)+11;
    static volatile bit           C1TX2_EID2          @ ((unsigned)&C1TX2DLC*8)+12;
    static volatile bit           C1TX2_EID3          @ ((unsigned)&C1TX2DLC*8)+13;
    static volatile bit           C1TX2_EID4          @ ((unsigned)&C1TX2DLC*8)+14;
    static volatile bit           C1TX2_EID5          @ ((unsigned)&C1TX2DLC*8)+15;
/* Microchip compatible bit field */
    static volatile struct {
           volatile unsigned      EID5                : 1;
           volatile unsigned      EID4                : 1;
           volatile unsigned      EID3                : 1;
           volatile unsigned      EID2                : 1;
           volatile unsigned      EID1                : 1;
           volatile unsigned      EID0                : 1;
           volatile unsigned      TXRTR               : 1;
           volatile unsigned      TXRB1               : 1;
           volatile unsigned      TXRB0               : 1;
           volatile unsigned      DLC                 : 4;
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
} C1TX2DLCbits @ 0x344;

    static volatile unsigned int  C1TX2B1             @ 0x346;
    static volatile unsigned int  C1TX2B2             @ 0x348;
    static volatile unsigned int  C1TX2B3             @ 0x34A;
    static volatile unsigned int  C1TX2B4             @ 0x34C;
    static volatile unsigned int  C1TX2CON            @ 0x34E;
    static volatile bit           C1TX2_TXPRI0        @ ((unsigned)&C1TX2CON*8)+0;
    static volatile bit           C1TX2_TXPRI1        @ ((unsigned)&C1TX2CON*8)+1;
    static volatile bit           C1TX2_TXREQ         @ ((unsigned)&C1TX2CON*8)+3;
    static volatile bit           C1TX2_TXERR         @ ((unsigned)&C1TX2CON*8)+4;
    static volatile bit           C1TX2_TXLARB        @ ((unsigned)&C1TX2CON*8)+5;
    static volatile bit           C1TX2_TXABT         @ ((unsigned)&C1TX2CON*8)+6;
/* Microchip compatible bit field */
    static volatile struct {
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
                    unsigned                          : 1;
           volatile unsigned      TXABT               : 1;
```

```
2627            volatile unsigned       TXLARB              : 1;
                volatile unsigned       TXERR               : 1;
                volatile unsigned       TXREQ               : 1;
                         unsigned                           : 1;
                volatile unsigned       TXPRI               : 2;
2632    } C1TX2CONbits @ 0x34E;

        static volatile unsigned int    C1TX1SID            @ 0x350;
        static volatile bit             C1TX1_TXIDE         @ ((unsigned)&C1TX1SID*8)+0;
        static volatile bit             C1TX1_SRR           @ ((unsigned)&C1TX1SID*8)+1;
2637    static volatile bit             C1TX1_SID0          @ ((unsigned)&C1TX1SID*8)+2;
        static volatile bit             C1TX1_SID1          @ ((unsigned)&C1TX1SID*8)+3;
        static volatile bit             C1TX1_SID2          @ ((unsigned)&C1TX1SID*8)+4;
        static volatile bit             C1TX1_SID3          @ ((unsigned)&C1TX1SID*8)+5;
        static volatile bit             C1TX1_SID4          @ ((unsigned)&C1TX1SID*8)+6;
2642    static volatile bit             C1TX1_SID5          @ ((unsigned)&C1TX1SID*8)+7;
        static volatile bit             C1TX1_SID6          @ ((unsigned)&C1TX1SID*8)+11;
        static volatile bit             C1TX1_SID7          @ ((unsigned)&C1TX1SID*8)+12;
        static volatile bit             C1TX1_SID8          @ ((unsigned)&C1TX1SID*8)+13;
        static volatile bit             C1TX1_SID9          @ ((unsigned)&C1TX1SID*8)+14;
2647    static volatile bit             C1TX1_SID10         @ ((unsigned)&C1TX1SID*8)+15;
        /* Microchip compatible bit field */
        static volatile struct {
                volatile unsigned       SID10               : 1;
                volatile unsigned       SID9                : 1;
2652            volatile unsigned       SID8                : 1;
                volatile unsigned       SID7                : 1;
                volatile unsigned       SID6                : 1;
                         unsigned                           : 1;
                         unsigned                           : 1;
2657                     unsigned                           : 1;
                volatile unsigned       SID5                : 1;
                volatile unsigned       SID4                : 1;
                volatile unsigned       SID3                : 1;
                volatile unsigned       SID2                : 1;
2662            volatile unsigned       SID1                : 1;
                volatile unsigned       SID0                : 1;
                volatile unsigned       SRR                 : 1;
                volatile unsigned       TXIDE               : 1;
        } C1TX1SIDbits @ 0x350;
2667
        static volatile unsigned int    C1TX1EID            @ 0x352;
        static volatile unsigned int    C1TX1DLC            @ 0x354;
        static volatile bit             C1TX1_DLC0          @ ((unsigned)&C1TX1DLC*8)+3;
        static volatile bit             C1TX1_DLC1          @ ((unsigned)&C1TX1DLC*8)+4;
2672    static volatile bit             C1TX1_DLC2          @ ((unsigned)&C1TX1DLC*8)+5;
        static volatile bit             C1TX1_DLC3          @ ((unsigned)&C1TX1DLC*8)+6;
        static volatile bit             C1TX1_TXRB0         @ ((unsigned)&C1TX1DLC*8)+7;
        static volatile bit             C1TX1_TXRB1         @ ((unsigned)&C1TX1DLC*8)+8;
        static volatile bit             C1TX1_TXRTR         @ ((unsigned)&C1TX1DLC*8)+9;
2677    static volatile bit             C1TX1_EID0          @ ((unsigned)&C1TX1DLC*8)+10;
        static volatile bit             C1TX1_EID1          @ ((unsigned)&C1TX1DLC*8)+11;
        static volatile bit             C1TX1_EID2          @ ((unsigned)&C1TX1DLC*8)+12;
        static volatile bit             C1TX1_EID3          @ ((unsigned)&C1TX1DLC*8)+13;
        static volatile bit             C1TX1_EID4          @ ((unsigned)&C1TX1DLC*8)+14;
2682    static volatile bit             C1TX1_EID5          @ ((unsigned)&C1TX1DLC*8)+15;
        /* Microchip compatible bit field */
        static volatile struct {
                volatile unsigned       EID5                : 1;
                volatile unsigned       EID4                : 1;
2687            volatile unsigned       EID3                : 1;
                volatile unsigned       EID2                : 1;
                volatile unsigned       EID1                : 1;
                volatile unsigned       EID0                : 1;
                volatile unsigned       TXRTR               : 1;
2692            volatile unsigned       TXRB1               : 1;
                volatile unsigned       TXRB0               : 1;
                volatile unsigned       DLC                 : 4;
                         unsigned                           : 1;
                         unsigned                           : 1;
2697                     unsigned                           : 1;
        } C1TX1DLCbits @ 0x354;

        static volatile unsigned int    C1TX1B1             @ 0x356;
        static volatile unsigned int    C1TX1B2             @ 0x358;
2702    static volatile unsigned int    C1TX1B3             @ 0x35A;
        static volatile unsigned int    C1TX1B4             @ 0x35C;
        static volatile unsigned int    C1TX1CON            @ 0x35E;
```

```
    static volatile bit             C1TX1_TXPRI0        @ ((unsigned)&C1TX1CON*8)+0;
    static volatile bit             C1TX1_TXPRI1        @ ((unsigned)&C1TX1CON*8)+1;
2707 static volatile bit            C1TX1_TXREQ         @ ((unsigned)&C1TX1CON*8)+3;
    static volatile bit             C1TX1_TXERR         @ ((unsigned)&C1TX1CON*8)+4;
    static volatile bit             C1TX1_TXLARB        @ ((unsigned)&C1TX1CON*8)+5;
    static volatile bit             C1TX1_TXABT         @ ((unsigned)&C1TX1CON*8)+6;
    /* Microchip compatible bit field */
2712 static volatile struct {
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
2717                unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
                    unsigned                            : 1;
2722        volatile unsigned       TXABT               : 1;
            volatile unsigned       TXLARB              : 1;
            volatile unsigned       TXERR               : 1;
            volatile unsigned       TXREQ               : 1;
                    unsigned                            : 1;
2727        volatile unsigned       TXPRI               : 2;
    } C1TX1CONbits @ 0x35E;

    static volatile unsigned int    C1TX0SID            @ 0x360;
    static volatile bit             C1TX0_TXIDE         @ ((unsigned)&C1TX0SID*8)+0;
2732 static volatile bit             C1TX0_SRR           @ ((unsigned)&C1TX0SID*8)+1;
    static volatile bit             C1TX0_SID0          @ ((unsigned)&C1TX0SID*8)+2;
    static volatile bit             C1TX0_SID1          @ ((unsigned)&C1TX0SID*8)+3;
    static volatile bit             C1TX0_SID2          @ ((unsigned)&C1TX0SID*8)+4;
    static volatile bit             C1TX0_SID3          @ ((unsigned)&C1TX0SID*8)+5;
2737 static volatile bit             C1TX0_SID4          @ ((unsigned)&C1TX0SID*8)+6;
    static volatile bit             C1TX0_SID5          @ ((unsigned)&C1TX0SID*8)+7;
    static volatile bit             C1TX0_SID6          @ ((unsigned)&C1TX0SID*8)+11;
    static volatile bit             C1TX0_SID7          @ ((unsigned)&C1TX0SID*8)+12;
    static volatile bit             C1TX0_SID8          @ ((unsigned)&C1TX0SID*8)+13;
2742 static volatile bit             C1TX0_SID9          @ ((unsigned)&C1TX0SID*8)+14;
    static volatile bit             C1TX0_SID10         @ ((unsigned)&C1TX0SID*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned       SID10               : 1;
2747        volatile unsigned       SID9                : 1;
            volatile unsigned       SID8                : 1;
            volatile unsigned       SID7                : 1;
            volatile unsigned       SID6                : 1;
                    unsigned                            : 1;
2752                unsigned                            : 1;
                    unsigned                            : 1;
            volatile unsigned       SID5                : 1;
            volatile unsigned       SID4                : 1;
            volatile unsigned       SID3                : 1;
2757        volatile unsigned       SID2                : 1;
            volatile unsigned       SID1                : 1;
            volatile unsigned       SID0                : 1;
            volatile unsigned       SRR                 : 1;
            volatile unsigned       TXIDE               : 1;
2762 } C1TX0SIDbits @ 0x360;

    static volatile unsigned int    C1TX0EID            @ 0x362;
    static volatile unsigned int    C1TX0DLC            @ 0x364;
    static volatile bit             C1TX0_DLC0          @ ((unsigned)&C1TX0DLC*8)+3;
2767 static volatile bit             C1TX0_DLC1          @ ((unsigned)&C1TX0DLC*8)+4;
    static volatile bit             C1TX0_DLC2          @ ((unsigned)&C1TX0DLC*8)+5;
    static volatile bit             C1TX0_DLC3          @ ((unsigned)&C1TX0DLC*8)+6;
    static volatile bit             C1TX0_TXRB0         @ ((unsigned)&C1TX0DLC*8)+7;
    static volatile bit             C1TX0_TXRB1         @ ((unsigned)&C1TX0DLC*8)+8;
2772 static volatile bit             C1TX0_TXRTR         @ ((unsigned)&C1TX0DLC*8)+9;
    static volatile bit             C1TX0_EID0          @ ((unsigned)&C1TX0DLC*8)+10;
    static volatile bit             C1TX0_EID1          @ ((unsigned)&C1TX0DLC*8)+11;
    static volatile bit             C1TX0_EID2          @ ((unsigned)&C1TX0DLC*8)+12;
    static volatile bit             C1TX0_EID3          @ ((unsigned)&C1TX0DLC*8)+13;
2777 static volatile bit             C1TX0_EID4          @ ((unsigned)&C1TX0DLC*8)+14;
    static volatile bit             C1TX0_EID5          @ ((unsigned)&C1TX0DLC*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned       EID5                : 1;
2782        volatile unsigned       EID4                : 1;
```

```
            volatile unsigned       EID3                : 1;
            volatile unsigned       EID2                : 1;
            volatile unsigned       EID1                : 1;
            volatile unsigned       EID0                : 1;
2787        volatile unsigned       TXRTR               : 1;
            volatile unsigned       TXRB1               : 1;
            volatile unsigned       TXRB0               : 1;
            volatile unsigned       DLC                 : 4;
                     unsigned                           : 1;
2792                 unsigned                           : 1;
                     unsigned                           : 1;
    } C1TX0DLCbits @ 0x364;

    static volatile unsigned int   C1TX0B1             @ 0x366;
2797 static volatile unsigned int   C1TX0B2             @ 0x368;
    static volatile unsigned int   C1TX0B3             @ 0x36A;
    static volatile unsigned int   C1TX0B4             @ 0x36C;
    static volatile unsigned int   C1TX0CON            @ 0x36E;
    static volatile bit            C1TX0_TXPRI0        @ ((unsigned)&C1TX0CON*8)+0;
2802 static volatile bit            C1TX0_TXPRI1        @ ((unsigned)&C1TX0CON*8)+1;
    static volatile bit            C1TX0_TXREQ         @ ((unsigned)&C1TX0CON*8)+3;
    static volatile bit            C1TX0_TXERR         @ ((unsigned)&C1TX0CON*8)+4;
    static volatile bit            C1TX0_TXLARB        @ ((unsigned)&C1TX0CON*8)+5;
    static volatile bit            C1TX0_TXABT         @ ((unsigned)&C1TX0CON*8)+6;
2807 /* Microchip compatible bit field */
    static volatile struct {
                     unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
2812                 unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
2817                 unsigned                           : 1;
            volatile unsigned       TXABT               : 1;
            volatile unsigned       TXLARB              : 1;
            volatile unsigned       TXERR               : 1;
            volatile unsigned       TXREQ               : 1;
2822                 unsigned                           : 1;
            volatile unsigned       TXPRI               : 2;
    } C1TX0CONbits @ 0x36E;

    static volatile unsigned int   C1RX1SID            @ 0x370;
2827 static volatile bit            C1RX1_RXIDE         @ ((unsigned)&C1RX1SID*8)+0;
    static volatile bit            C1RX1_SRR           @ ((unsigned)&C1RX1SID*8)+1;
    static volatile bit            C1RX1_SID0          @ ((unsigned)&C1RX1SID*8)+2;
    static volatile bit            C1RX1_SID1          @ ((unsigned)&C1RX1SID*8)+3;
    static volatile bit            C1RX1_SID2          @ ((unsigned)&C1RX1SID*8)+4;
2832 static volatile bit            C1RX1_SID3          @ ((unsigned)&C1RX1SID*8)+5;
    static volatile bit            C1RX1_SID4          @ ((unsigned)&C1RX1SID*8)+6;
    static volatile bit            C1RX1_SID5          @ ((unsigned)&C1RX1SID*8)+7;
    static volatile bit            C1RX1_SID6          @ ((unsigned)&C1RX1SID*8)+8;
    static volatile bit            C1RX1_SID7          @ ((unsigned)&C1RX1SID*8)+9;
2837 static volatile bit            C1RX1_SID8          @ ((unsigned)&C1RX1SID*8)+10;
    static volatile bit            C1RX1_SID9          @ ((unsigned)&C1RX1SID*8)+11;
    static volatile bit            C1RX1_SID10         @ ((unsigned)&C1RX1SID*8)+12;
    /* Microchip compatible bit field */
    static volatile struct {
2842                 unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
            volatile unsigned       SID10               : 1;
            volatile unsigned       SID9                : 1;
2847        volatile unsigned       SID8                : 1;
            volatile unsigned       SID7                : 1;
            volatile unsigned       SID6                : 1;
            volatile unsigned       SID5                : 1;
            volatile unsigned       SID4                : 1;
2852        volatile unsigned       SID3                : 1;
            volatile unsigned       SID2                : 1;
            volatile unsigned       SID1                : 1;
            volatile unsigned       SID0                : 1;
            volatile unsigned       SRR                 : 1;
2857        volatile unsigned       RXIDE               : 1;
    } C1RX1SIDbits @ 0x370;

    static volatile unsigned int   C1RX1EID            @ 0x372;
```

```
    static volatile unsigned int  C1RX1DLC          @ 0x374;
2862 static volatile bit           C1RX1_DLC0        @ ((unsigned)&C1RX1DLC*8)+0;
    static volatile bit           C1RX1_DLC1        @ ((unsigned)&C1RX1DLC*8)+1;
    static volatile bit           C1RX1_DLC2        @ ((unsigned)&C1RX1DLC*8)+2;
    static volatile bit           C1RX1_DLC3        @ ((unsigned)&C1RX1DLC*8)+3;
    static volatile bit           C1RX1_RXRB0       @ ((unsigned)&C1RX1DLC*8)+4;
2867 static volatile bit           C1RX1_RXRB1       @ ((unsigned)&C1RX1DLC*8)+8;
    static volatile bit           C1RX1_RXRTR       @ ((unsigned)&C1RX1DLC*8)+9;
    static volatile bit           C1RX1_EID0        @ ((unsigned)&C1RX1DLC*8)+10;
    static volatile bit           C1RX1_EID1        @ ((unsigned)&C1RX1DLC*8)+11;
    static volatile bit           C1RX1_EID2        @ ((unsigned)&C1RX1DLC*8)+12;
2872 static volatile bit           C1RX1_EID3        @ ((unsigned)&C1RX1DLC*8)+13;
    static volatile bit           C1RX1_EID4        @ ((unsigned)&C1RX1DLC*8)+14;
    static volatile bit           C1RX1_EID5        @ ((unsigned)&C1RX1DLC*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
2877        volatile unsigned     EID5              : 1;
            volatile unsigned     EID4              : 1;
            volatile unsigned     EID3              : 1;
            volatile unsigned     EID2              : 1;
            volatile unsigned     EID1              : 1;
2882        volatile unsigned     EID0              : 1;
            volatile unsigned     RXRTR             : 1;
            volatile unsigned     RXRB1             : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
2887                 unsigned                       : 1;
            volatile unsigned     RXRB0             : 1;
            volatile unsigned     DLC               : 4;
    } C1RX1DLCbits @ 0x374;

2892 static volatile unsigned int  C1RX1B1           @ 0x376;
    static volatile unsigned int  C1RX1B2           @ 0x378;
    static volatile unsigned int  C1RX1B3           @ 0x37A;
    static volatile unsigned int  C1RX1B4           @ 0x37C;
    static volatile unsigned int  C1RX1CON          @ 0x37E;
2897 static volatile bit           C1RX1_FILHIT0     @ ((unsigned)&C1RX1CON*8)+0;
    static volatile bit           C1RX1_FILHIT1     @ ((unsigned)&C1RX1CON*8)+1;
    static volatile bit           C1RX1_FILHIT2     @ ((unsigned)&C1RX1CON*8)+2;
    static volatile bit           C1RX1_RXRTRRO     @ ((unsigned)&C1RX1CON*8)+3;
    static volatile bit           C1RX1_RXFUL       @ ((unsigned)&C1RX1CON*8)+7;
2902 /* Microchip compatible bit field */
    static volatile struct {
                     unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
2907                 unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
2912        volatile unsigned     RXFUL             : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
                     unsigned                       : 1;
            volatile unsigned     RXRTRRO           : 1;
2917        volatile unsigned     FILHIT            : 3;
    } C1RX1CONbits @ 0x37E;

    static volatile unsigned int  C1RX0SID          @ 0x380;
    static volatile bit           C1RX0_RXIDE       @ ((unsigned)&C1RX0SID*8)+0;
2922 static volatile bit           C1RX0_SRR         @ ((unsigned)&C1RX0SID*8)+1;
    static volatile bit           C1RX0_SID0        @ ((unsigned)&C1RX0SID*8)+2;
    static volatile bit           C1RX0_SID1        @ ((unsigned)&C1RX0SID*8)+3;
    static volatile bit           C1RX0_SID2        @ ((unsigned)&C1RX0SID*8)+4;
    static volatile bit           C1RX0_SID3        @ ((unsigned)&C1RX0SID*8)+5;
2927 static volatile bit           C1RX0_SID4        @ ((unsigned)&C1RX0SID*8)+6;
    static volatile bit           C1RX0_SID5        @ ((unsigned)&C1RX0SID*8)+7;
    static volatile bit           C1RX0_SID6        @ ((unsigned)&C1RX0SID*8)+8;
    static volatile bit           C1RX0_SID7        @ ((unsigned)&C1RX0SID*8)+9;
    static volatile bit           C1RX0_SID8        @ ((unsigned)&C1RX0SID*8)+10;
2932 static volatile bit           C1RX0_SID9        @ ((unsigned)&C1RX0SID*8)+11;
    static volatile bit           C1RX0_SID10       @ ((unsigned)&C1RX0SID*8)+12;
    /* Microchip compatible bit field */
    static volatile struct {
                     unsigned                       : 1;
2937                 unsigned                       : 1;
                     unsigned                       : 1;
```

```
            volatile unsigned       SID10               : 1;
            volatile unsigned       SID9                : 1;
            volatile unsigned       SID8                : 1;
2942        volatile unsigned       SID7                : 1;
            volatile unsigned       SID6                : 1;
            volatile unsigned       SID5                : 1;
            volatile unsigned       SID4                : 1;
            volatile unsigned       SID3                : 1;
2947        volatile unsigned       SID2                : 1;
            volatile unsigned       SID1                : 1;
            volatile unsigned       SID0                : 1;
            volatile unsigned       SRR                 : 1;
            volatile unsigned       RXIDE               : 1;
2952  } C1RX0SIDbits @ 0x380;

      static volatile unsigned int  C1RX0EID            @ 0x382;
      static volatile unsigned int  C1RX0DLC            @ 0x384;
      static volatile bit           C1RX0_DLC0          @ ((unsigned)&C1RX0DLC*8)+0;
2957  static volatile bit           C1RX0_DLC1          @ ((unsigned)&C1RX0DLC*8)+1;
      static volatile bit           C1RX0_DLC2          @ ((unsigned)&C1RX0DLC*8)+2;
      static volatile bit           C1RX0_DLC3          @ ((unsigned)&C1RX0DLC*8)+3;
      static volatile bit           C1RX0_RXRB0         @ ((unsigned)&C1RX0DLC*8)+4;
      static volatile bit           C1RX0_RXRB1         @ ((unsigned)&C1RX0DLC*8)+8;
2962  static volatile bit           C1RX0_RXRTR         @ ((unsigned)&C1RX0DLC*8)+9;
      static volatile bit           C1RX0_EID0          @ ((unsigned)&C1RX0DLC*8)+10;
      static volatile bit           C1RX0_EID1          @ ((unsigned)&C1RX0DLC*8)+11;
      static volatile bit           C1RX0_EID2          @ ((unsigned)&C1RX0DLC*8)+12;
      static volatile bit           C1RX0_EID3          @ ((unsigned)&C1RX0DLC*8)+13;
2967  static volatile bit           C1RX0_EID4          @ ((unsigned)&C1RX0DLC*8)+14;
      static volatile bit           C1RX0_EID5          @ ((unsigned)&C1RX0DLC*8)+15;
      /* Microchip compatible bit field */
      static volatile struct {
            volatile unsigned       EID5                : 1;
2972        volatile unsigned       EID4                : 1;
            volatile unsigned       EID3                : 1;
            volatile unsigned       EID2                : 1;
            volatile unsigned       EID1                : 1;
            volatile unsigned       EID0                : 1;
2977        volatile unsigned       RXRTR               : 1;
            volatile unsigned       RXRB1               : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
2982        volatile unsigned       RXRB0               : 1;
            volatile unsigned       DLC                 : 4;
      } C1RX0DLCbits @ 0x384;

      static volatile unsigned int  C1RX0B1             @ 0x386;
2987  static volatile unsigned int  C1RX0B2             @ 0x388;
      static volatile unsigned int  C1RX0B3             @ 0x38A;
      static volatile unsigned int  C1RX0B4             @ 0x38C;
      static volatile unsigned int  C1RX0CON            @ 0x38E;
      static volatile bit           C1RX0_FILHIT        @ ((unsigned)&C1RX0CON*8)+0;
2992  static volatile bit           C1RX0_JTOFF         @ ((unsigned)&C1RX0CON*8)+1;
      static volatile bit           C1RX0_RXB0DBEN      @ ((unsigned)&C1RX0CON*8)+2;
      static volatile bit           C1RX0_RXRTRRO       @ ((unsigned)&C1RX0CON*8)+3;
      static volatile bit           C1RX0_RXFUL         @ ((unsigned)&C1RX0CON*8)+7;
      /* Microchip compatible bit field */
2997  static volatile struct {
                     unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
3002                 unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
            volatile unsigned       RXFUL               : 1;
3007                 unsigned                           : 1;
                     unsigned                           : 1;
                     unsigned                           : 1;
            volatile unsigned       RXRTRRO             : 1;
            volatile unsigned       RXB0DBEN            : 1;
3012        volatile unsigned       JTOFF               : 1;
            volatile unsigned       FILHIT              : 1;
      } C1RX0CONbits @ 0x38E;

      static volatile unsigned int  C1CTRL              @ 0x390;
```

```
3017 static volatile bit           C1_ICOD0          @ ((unsigned)&C1CTRL*8)+1;
     static volatile bit           C1_ICOD1          @ ((unsigned)&C1CTRL*8)+2;
     static volatile bit           C1_ICOD2          @ ((unsigned)&C1CTRL*8)+3;
     static volatile bit           C1_OPMODE0        @ ((unsigned)&C1CTRL*8)+5;
     static volatile bit           C1_OPMODE1        @ ((unsigned)&C1CTRL*8)+6;
3022 static volatile bit           C1_OPMODE2        @ ((unsigned)&C1CTRL*8)+7;
     static volatile bit           C1_REQOP0         @ ((unsigned)&C1CTRL*8)+8;
     static volatile bit           C1_REQOP1         @ ((unsigned)&C1CTRL*8)+9;
     static volatile bit           C1_REQOP2         @ ((unsigned)&C1CTRL*8)+10;
     static volatile bit           C1_CANCKS         @ ((unsigned)&C1CTRL*8)+11;
3027 static volatile bit           C1_ABAT           @ ((unsigned)&C1CTRL*8)+12;
     static volatile bit           C1_CANSIDL        @ ((unsigned)&C1CTRL*8)+13;
     static volatile bit           C1_CANFRZ         @ ((unsigned)&C1CTRL*8)+14;
     static volatile bit           C1_CANCAP         @ ((unsigned)&C1CTRL*8)+15;
     /* Microchip compatible bit field */
3032 static volatile struct {
             volatile unsigned     CANCAP            : 1;
             volatile unsigned     CANFRZ            : 1;
             volatile unsigned     CANSIDL           : 1;
             volatile unsigned     ABAT              : 1;
3037         volatile unsigned     CANCKS            : 1;
             volatile unsigned     REQOP             : 3;
             volatile unsigned     OPMODE            : 3;
                      unsigned                       : 1;
             volatile unsigned     ICOD              : 3;
3042                  unsigned                       : 1;
     } C1CTRLbits @ 0x390;

     static volatile unsigned int  C1CFG1            @ 0x392;
     static volatile bit           C1CFG1_BRP0       @ ((unsigned)&C1CFG1*8)+0;
3047 static volatile bit           C1CFG1_BRP1       @ ((unsigned)&C1CFG1*8)+1;
     static volatile bit           C1CFG1_BRP2       @ ((unsigned)&C1CFG1*8)+2;
     static volatile bit           C1CFG1_BRP3       @ ((unsigned)&C1CFG1*8)+3;
     static volatile bit           C1CFG1_BRP4       @ ((unsigned)&C1CFG1*8)+4;
     static volatile bit           C1CFG1_BRP5       @ ((unsigned)&C1CFG1*8)+5;
3052 static volatile bit           C1CFG1_SJW0       @ ((unsigned)&C1CFG1*8)+6;
     static volatile bit           C1CFG1_SJW1       @ ((unsigned)&C1CFG1*8)+7;
     /* Microchip compatible bit field */
     static volatile struct {
                      unsigned                       : 1;
3057                  unsigned                       : 1;
                      unsigned                       : 1;
                      unsigned                       : 1;
                      unsigned                       : 1;
                      unsigned                       : 1;
3062                  unsigned                       : 1;
                      unsigned                       : 1;
             volatile unsigned     SJW               : 2;
             volatile unsigned     BRP               : 6;
     } C1CFG1bits @ 0x392;
3067
     static volatile unsigned int  C1CFG2            @ 0x394;
     static volatile bit           C1CFG2_PRSEG0     @ ((unsigned)&C1CFG2*8)+0;
     static volatile bit           C1CFG2_PRSEG1     @ ((unsigned)&C1CFG2*8)+1;
     static volatile bit           C1CFG2_PRSEG2     @ ((unsigned)&C1CFG2*8)+2;
3072 static volatile bit           C1CFG2_SEG1PH0    @ ((unsigned)&C1CFG2*8)+3;
     static volatile bit           C1CFG2_SEG1PH1    @ ((unsigned)&C1CFG2*8)+4;
     static volatile bit           C1CFG2_SEG1PH2    @ ((unsigned)&C1CFG2*8)+5;
     static volatile bit           C1CFG2_SAM        @ ((unsigned)&C1CFG2*8)+6;
     static volatile bit           C1CFG2_SEG2PHTS   @ ((unsigned)&C1CFG2*8)+7;
3077 static volatile bit           C1CFG2_SEG2PH0    @ ((unsigned)&C1CFG2*8)+8;
     static volatile bit           C1CFG2_SEG2PH1    @ ((unsigned)&C1CFG2*8)+9;
     static volatile bit           C1CFG2_SEG2PH2    @ ((unsigned)&C1CFG2*8)+10;
     static volatile bit           C1CFG2_WAKFIL     @ ((unsigned)&C1CFG2*8)+14;
     /* Microchip compatible bit field */
3082 static volatile struct {
                      unsigned                       : 1;
             volatile unsigned     WAKFIL            : 1;
                      unsigned                       : 1;
                      unsigned                       : 1;
3087                  unsigned                       : 1;
             volatile unsigned     SEG2PH            : 3;
             volatile unsigned     SEG2PHTS          : 1;
             volatile unsigned     SAM               : 1;
             volatile unsigned     SEG1PH            : 3;
3092         volatile unsigned     PRSEG             : 3;
     } C1CFG2bits @ 0x394;
```

```
    static volatile unsigned int  C1INTF            @ 0x396;
    static volatile bit           C1_RXB0IF         @ ((unsigned)&C1INTF*8)+0;
3097 static volatile bit           C1_RXB1IF         @ ((unsigned)&C1INTF*8)+1;
    static volatile bit           C1_TXB0IF         @ ((unsigned)&C1INTF*8)+2;
    static volatile bit           C1_TXB1IF         @ ((unsigned)&C1INTF*8)+3;
    static volatile bit           C1_TXB2IF         @ ((unsigned)&C1INTF*8)+4;
    static volatile bit           C1_ERRIF          @ ((unsigned)&C1INTF*8)+5;
3102 static volatile bit           C1_WAKIF          @ ((unsigned)&C1INTF*8)+6;
    static volatile bit           C1_IRXIF          @ ((unsigned)&C1INTF*8)+7;
    static volatile bit           C1_EWARN          @ ((unsigned)&C1INTF*8)+8;
    static volatile bit           C1_RXWARN         @ ((unsigned)&C1INTF*8)+9;
    static volatile bit           C1_TXWARN         @ ((unsigned)&C1INTF*8)+10;
3107 static volatile bit           C1_RXBP           @ ((unsigned)&C1INTF*8)+11;
    static volatile bit           C1_TXBP           @ ((unsigned)&C1INTF*8)+12;
    static volatile bit           C1_TXBO           @ ((unsigned)&C1INTF*8)+13;
    static volatile bit           C1_RXB1OVFL       @ ((unsigned)&C1INTF*8)+14;
    static volatile bit           C1_RXB0OVFL       @ ((unsigned)&C1INTF*8)+15;
3112 /* Microchip compatible bit field */
    static volatile struct {
            volatile unsigned     RXB0OVFL          : 1;
            volatile unsigned     RXB1OVFL          : 1;
            volatile unsigned     TXBO              : 1;
3117        volatile unsigned     TXBP              : 1;
            volatile unsigned     RXBP              : 1;
            volatile unsigned     TXWARN            : 1;
            volatile unsigned     RXWARN            : 1;
            volatile unsigned     EWARN             : 1;
3122        volatile unsigned     IRXIF             : 1;
            volatile unsigned     WAKIF             : 1;
            volatile unsigned     ERRIF             : 1;
            volatile unsigned     TXB2IF            : 1;
            volatile unsigned     TXB1IF            : 1;
3127        volatile unsigned     TXB0IF            : 1;
            volatile unsigned     RXB1IF            : 1;
            volatile unsigned     RXB0IF            : 1;
    } C1INTFbits @ 0x396;

3132 static volatile unsigned int  C1INTE            @ 0x398;
    static volatile bit           C1_RXB0IE         @ ((unsigned)&C1INTE*8)+0;
    static volatile bit           C1_RXB1IE         @ ((unsigned)&C1INTE*8)+1;
    static volatile bit           C1_TXB0IE         @ ((unsigned)&C1INTE*8)+2;
    static volatile bit           C1_TXB1IE         @ ((unsigned)&C1INTE*8)+3;
3137 static volatile bit           C1_TXB2IE         @ ((unsigned)&C1INTE*8)+4;
    static volatile bit           C1_ERRIE          @ ((unsigned)&C1INTE*8)+5;
    static volatile bit           C1_WAKIE          @ ((unsigned)&C1INTE*8)+6;
    static volatile bit           C1_IRXIE          @ ((unsigned)&C1INTE*8)+7;
    /* Microchip compatible bit field */
3142 static volatile struct {
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
3147                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
            volatile unsigned     IRXIE             : 1;
3152        volatile unsigned     WAKIE             : 1;
            volatile unsigned     ERRIE             : 1;
            volatile unsigned     TXB2IE            : 1;
            volatile unsigned     TXB1IE            : 1;
            volatile unsigned     TXB0IE            : 1;
3157        volatile unsigned     RXB1IE            : 1;
            volatile unsigned     RXB0IE            : 1;
    } C1INTEbits @ 0x398;

    static volatile unsigned int  C1EC              @ 0x39A;
3162 static volatile bit           C1_REC0           @ ((unsigned)&C1EC*8)+0;
    static volatile bit           C1_REC1           @ ((unsigned)&C1EC*8)+1;
    static volatile bit           C1_REC2           @ ((unsigned)&C1EC*8)+2;
    static volatile bit           C1_REC3           @ ((unsigned)&C1EC*8)+3;
    static volatile bit           C1_REC4           @ ((unsigned)&C1EC*8)+4;
3167 static volatile bit           C1_REC5           @ ((unsigned)&C1EC*8)+5;
    static volatile bit           C1_REC6           @ ((unsigned)&C1EC*8)+6;
    static volatile bit           C1_REC7           @ ((unsigned)&C1EC*8)+7;
    static volatile bit           C1_TEC0           @ ((unsigned)&C1EC*8)+8;
    static volatile bit           C1_TEC1           @ ((unsigned)&C1EC*8)+9;
3172 static volatile bit           C1_TEC2           @ ((unsigned)&C1EC*8)+10;
```

```
    static volatile bit           C1_TEC3           @ ((unsigned)&C1EC*8)+11;
    static volatile bit           C1_TEC4           @ ((unsigned)&C1EC*8)+12;
    static volatile bit           C1_TEC5           @ ((unsigned)&C1EC*8)+13;
    static volatile bit           C1_TEC6           @ ((unsigned)&C1EC*8)+14;
3177 static volatile bit           C1_TEC7           @ ((unsigned)&C1EC*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
           volatile unsigned      TEC               : 8;
           volatile unsigned      REC               : 8;
3182 } C1ECbits @ 0x39A;

    static volatile unsigned int  RCON              @ 0x740;
    static volatile bit           POR               @ ((unsigned)&RCON*8)+0;
    static volatile bit           BOR               @ ((unsigned)&RCON*8)+1;
3187 static volatile bit           IDLE              @ ((unsigned)&RCON*8)+2;
    static volatile bit           SLEEP             @ ((unsigned)&RCON*8)+3;
    static volatile bit           WDTO              @ ((unsigned)&RCON*8)+4;
    static volatile bit           SWDTEN            @ ((unsigned)&RCON*8)+5;
    static volatile bit           SWR               @ ((unsigned)&RCON*8)+6;
3192 static volatile bit           EXTR              @ ((unsigned)&RCON*8)+7;
    static volatile bit           IOPUWR            @ ((unsigned)&RCON*8)+14;
    static volatile bit           TRAPR             @ ((unsigned)&RCON*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
3197        volatile unsigned      TRAPR             : 1;
           volatile unsigned      IOPUWR            : 1;
                  unsigned                         : 1;
                  unsigned                         : 1;
                  unsigned                         : 1;
3202              unsigned                         : 1;
                  unsigned                         : 1;
                  unsigned                         : 1;
           volatile unsigned      EXTR              : 1;
           volatile unsigned      SWR               : 1;
3207       volatile unsigned      SWDTEN            : 1;
           volatile unsigned      WDTO              : 1;
           volatile unsigned      SLEEP             : 1;
           volatile unsigned      IDLE              : 1;
           volatile unsigned      BOR               : 1;
3212       volatile unsigned      POR               : 1;
    } RCONbits @ 0x740;

    static volatile unsigned int  OSCCON            @ 0x742;
    static volatile bit           OSWEN             @ ((unsigned)&OSCCON*8)+0;
3217 static volatile bit           LPOSCEN           @ ((unsigned)&OSCCON*8)+1;
    static volatile bit           CF                @ ((unsigned)&OSCCON*8)+3;
    static volatile bit           LOCK              @ ((unsigned)&OSCCON*8)+5;
    static volatile bit           POST0             @ ((unsigned)&OSCCON*8)+6;
    static volatile bit           POST1             @ ((unsigned)&OSCCON*8)+7;
3222 static volatile bit           NOSC0             @ ((unsigned)&OSCCON*8)+8;
    static volatile bit           NOSC1             @ ((unsigned)&OSCCON*8)+9;
    static volatile bit           COSC0             @ ((unsigned)&OSCCON*8)+12;
    static volatile bit           COSC1             @ ((unsigned)&OSCCON*8)+13;
    /* Microchip compatible bit field */
3227 static volatile struct {
                  unsigned                         : 1;
                  unsigned                         : 1;
           volatile unsigned      COSC              : 2;
                  unsigned                         : 1;
3232              unsigned                         : 1;
           volatile unsigned      NOSC              : 2;
           volatile unsigned      POST              : 2;
           volatile unsigned      LOCK              : 1;
                  unsigned                         : 1;
3237       volatile unsigned      CF                : 1;
                  unsigned                         : 1;
           volatile unsigned      LPOSCEN           : 1;
           volatile unsigned      OSWEN             : 1;
    } OSCCONbits @ 0x742;
3242
    static volatile unsigned int  NVMCON            @ 0x760;
    static volatile bit           PROGOP0           @ ((unsigned)&NVMCON*8)+0;
    static volatile bit           PROGOP1           @ ((unsigned)&NVMCON*8)+1;
    static volatile bit           PROGOP2           @ ((unsigned)&NVMCON*8)+2;
3247 static volatile bit           PROGOP3           @ ((unsigned)&NVMCON*8)+3;
    static volatile bit           PROGOP4           @ ((unsigned)&NVMCON*8)+4;
    static volatile bit           PROGOP5           @ ((unsigned)&NVMCON*8)+5;
    static volatile bit           PROGOP6           @ ((unsigned)&NVMCON*8)+6;
```

```
    static volatile bit           WRERR             @ ((unsigned)&NVMCON*8)+13;
3252 static volatile bit           WREN              @ ((unsigned)&NVMCON*8)+14;
    static volatile bit           WR                @ ((unsigned)&NVMCON*8)+15;
    /* Microchip compatible bit field */
    static volatile struct {
           volatile unsigned      WR                : 1;
3257       volatile unsigned      WREN              : 1;
           volatile unsigned      WRERR             : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
3262                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
           volatile unsigned      PROGOP            : 7;
    } NVMCONbits @ 0x760;
3267
    static volatile unsigned int  NVMADR            @ 0x762;
    static volatile unsigned int  NVMADRU           @ 0x764;
    static volatile unsigned int  NVMKEY            @ 0x766;
    static volatile unsigned int  PMD1              @ 0x770;
3272 static volatile unsigned int  PMD2              @ 0x772;
    static volatile unsigned int  IPC11             @ 0xAA;
    static volatile bit           FLTBIP0           @ ((unsigned)&IPC11*8)+0;
    static volatile bit           FLTBIP1           @ ((unsigned)&IPC11*8)+1;
    static volatile bit           FLTBIP2           @ ((unsigned)&IPC11*8)+2;
3277 /* Microchip compatible bit field */
    static volatile struct {
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
3282                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
3287                unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
                    unsigned                        : 1;
3292       volatile unsigned      FLTBIP            : 3;
    } IPC11bits @ 0xAA;


    #endif
```

Apéndice D

Palabra de configuración (dsPIC3xF)

La palabra de configuración define el comportamiento de algunas características del circuito.

Se emplea:

Listado D.1:

```
__CONFIG(FOSC,XTPLL16); // Primero el registro de configuracion
                        // Luego la opcion
```

dsPIC30F. Lista de opciones.

Description	Config Register	Symbols
Primary oscillator types	FOSC	ECPLL16, ECPLL8, ECPLL4, ECIO, EC, ERC, ERCIO, XTPLL16, XTPLL8, XTPLL4, XT, HS, XTL
Oscillator select	FOSC	POSC, LP, FRC, LPRC
Oscillator system clock switch	FOSC	CLKSWDIS, CLKSWEN, FSCMDIS, FCSMEN
Watchdog timer enable	FWDT	WDTEN, WDTDIS
Watchdog timer pre-scale select	FWDT	WDTPSA512, WDTPSA64, WDTPSA8, WDTPSA1, WDTPSB1-WDTPSB16
Powerup timer enable	FBORPOR	PWRT64, PWRT16, PWRT4, PWRTDIS
Brown-out reset enable	FBORPOR	BOREN, BORDIS
Brown-out reset voltage	FBORPOR	BORV20, BORV27, BORV42, BORV45
MCLR pin function	FBORPOR	MCLREN, MCLRDIS
Motor control PWM	FBORPOR	PWMBIN, HPOL, LPOL [1]
Code protection	FGS	GCPU, GCPP, GWRU, GWRP

dsPIC33F y PIC24H. Lista de opciones.

Description	Config Register	Symbols
Code protection	FGS	GCPU, GCPP, GWRU, GWRP
Oscillator two-speed startup	FOSCSEL	IESOEN, IESODIS
Temperature protection	FOSCSEL	TEMPDIS, TEMPEN
Initial oscillator source selection	FOSCSEL	FRCPS, LPRC, LP, OSCPLL, OSC, FRCPLL, FRC
Oscillator clock switching modes	FOSC	FCKSMDIS, CLKSWEN, FCKSMEN
OSC2 pin function	FOSC	OSC2OUT, OSC2DIO
Primary oscillator modes	FOSC	POSCDIS, POSCHS, POSCXT, POSCEC
Watchdog timer enable	FWDT	WDTEN, WDTDIS, WINDIS, WINEN
Watchdog timer prescaler	FWDT	WDTPRE128, WDTPRE32
Watchdog timer postscaler	FWDT	WDTPS32768, WDTPS16384, WDTPS8192, WDTPS4096, WDTPS2048, WDTPS1024, WDTPS512, WDTPS256, WDTPS128, WDTPS64, WDTPS32, WDTPS16, WDTPS8, WDTPS4, WDTPS2, WDTPS1
Motor Control[2]	FPOR	PWMPORT, PWMPWM, PWMHPAH, PWMHPAL, PWMLPAH, PWMLPAL
Power-on Reset Timer	FPOR	PWRT128, PWRT64, PWRT32, PWRT16, PWRT8, PWRT4, PWRT2, PWRTDIS

PIC24F. Lista de opciones.

Description	Flash Config Word[3]	Symbols
JTAG port enable bit	FLSHCFGWRD1	JTAGEN, JTAGDIS
Program memory code protection bit	FLSHCFGWRD1	GCPU, GCPP
Code flash write protection bit	FLSHCFGWRD1	GWRPU, GWRPP
Background debugger enable bit	FLSHCFGWRD1	DEBUGDIS, DEBUGEN
Set clip on emulation bit	FLSHCFGWRD1	COEDIS, COEEN
ICS: ICD pin placement select bit	FLSHCFGWRD1	ICDEMU2, ICDEMU1
Watchdog timer enable bit	FLSHCFGWRD1	WDTEN, WDTDIS
Windowed WDT disable bit	FLSHCFGWRD1	WINDIS, WINEN
WDT prescaler ratio select bit	FLSHCFGWRD1	WDTPRE128, WDTPRE32
WDT postscaler select bits	FLSHCFGWRD1	WDTPS32768, WDTPS16384, WDTPS8192, WDTPS4096, WDTPS2048, WDTPS1024, WDTPS512, WDTPS256, WDTPS128, WDTPS64, WDTPS32, WDTPS16, WDTPS8, WDTPS4, WDTPS2, WDTPS1
Internal external switchover bit	FLSHCFGWRD2	IESOEN, IESODIS
Initial oscillator select bits	FLSHCFGWRD2	FRCDIV, LPRC, SOSC, HSECPLL, XTHSEC, FRCPLL, FRC
Clock switching & fail-safe clock monitor	FLSHCFGWRD2	FCKSMDIS, CLKSWEN, FCKSMEN
OSC2 pin configuration bit	FLSHCFGWRD2	CLKO, PORTIO
Primary oscillator configuration bits	FLSHCFGWRD2	POSCDIS, POSCHS, POSCXT, POSCEC

Índice alfabético