# Índice general

# Capítulo 1

# Objetivos

A lo largo de esta sección se mostrará el objetivo principal que debe cumplir el proyecto. Se pretende realizar el estudio detallado de un DSP relacionado con la asignatura Sistemas Electrónicos para el Tratamiento de la Información de la titulación Ingeniero en Electrónica, a lo largo del proyecto se estudiará un DSP y se realizarán enunciados de prácticas con solución orientadas al aprendizaje del dispositivo, con cada práctica se desarrollará un hardware para demostrar el funcionamiento del DSP. Se van a analizar parte de los módulos que contiene el DSP. En concreto este proyecto se centrará en la familia dsPIC30F de Microchip, principalmente se desarrollarán los supuestos prácticos sobre el dsPIC 30f4013.
Este proyecto forma parte de un proyecto conjunto con el proyecto que se titula ESTUDIO Y DESARROLLO DE PRÁCTICAS BASICAS PARA LABORATORIO DE LA FAMILIA DE PROCESADORES DE SEÑAL DIGITAL DE MICROCHIP dsPIC30Fxxx, realizado por mi compañera Isabel González Domínguez y se ha optado por dividirlos teniendo parte de la memoria común, en la que se habla de las herramientas empleadas, el grabador y las características del dsPIC30f y una distinta que trata de los periféricos y las prácticas, estudiando cada una unos periféricos concretos.
Las etapas en las que dividimos las tareas a realizar a lo largo del proyecto son:

- Desarrollo de un grabador compatible con los microcontroladores utilizados actualmente en la asignatura, PIC16f, a través del puerto paralelo del ordenador. Para realizar el esquema del grabador se empleará la herramienta Proteus ISIS que permite la simulación de todos los casos prácticos que se plantearán y se trabajará con la herramienta ARES para realizar el rutado del circuito que permitirá el fresado de la placa, para la realización física de la placa se emplea un programa que genera los vectores que hay que proporcionar a la fresadora empleada para realizar la placa. Se realizará un primer grabador.

- Estudio detallado del DSP, para familiarizarse con los módulos que contiene y su fun-

cionamiento. Para ello se emplearán diversos datasheet proporcionados por el fabricante y libros que ayudarán a entender su funcionamiento.

- Familiarizarse con las herramientas de programación empleadas, se utilizará la herramienta de MPLAB para el desarrollo software de las distintas prácticas que permite ensamblar y compilar los programas así como depurar su funcionamiento.
  Todos los programas se han realizado en lenguaje de alto nivel como C, por lo que se utiliza un compilador de C de la casa HI-TECH compatible con la herramienta MPLAB. Se empleará la herramienta WINPIC800 para grabar los ficheros en el dsPIC, utilizando el grabador diseñado.

- Realización de prácticas de distintos niveles de dificultad, para comprobar el funcionamiento de los distintos módulos de los que dispone el DSP. Cada práctica irá acompañada de un enunciado, orientado a la realización de las prácticas por los alumnos, en el que se especifican todos los requisitos que se tienen que cumplir en el ejercicio propuesto,una posible solución al problema propuesto y una placa que permita demostrar el funcionamiento y que se pueda emplear posteriormente para la corrección de las prácticas realizadas por los alumnos.

- Realización de memoria en la que se incluye una sección dedicada al DSP mencionando las características más importantes y explicando los módulos de los que está compuesto, funcionamiento y configuración, enunciados de las prácticas realizadas con cada uno de los módulos acompañadas de la solución en lenguaje C, así como los circuitos de las distintas placas realizadas, diseñados en Proteus.

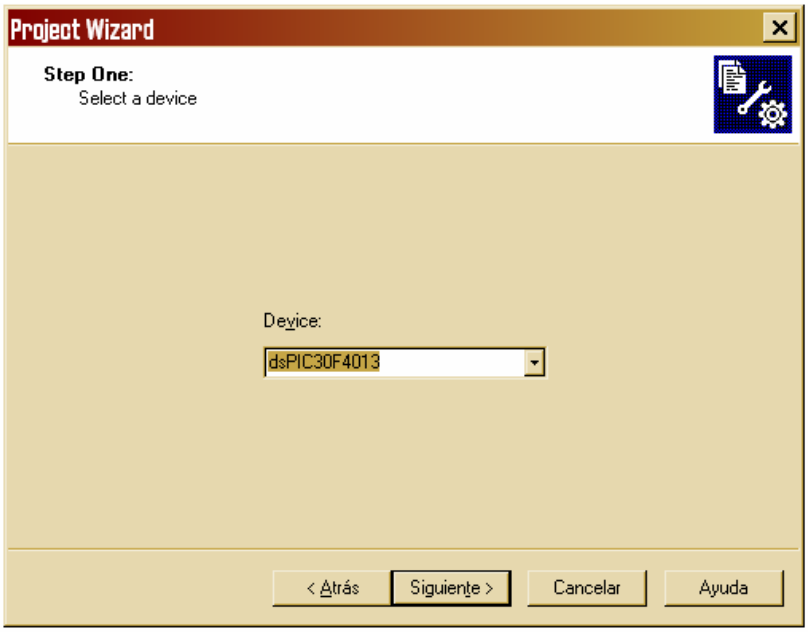
# Capítulo 2

# Herramientas empleadas

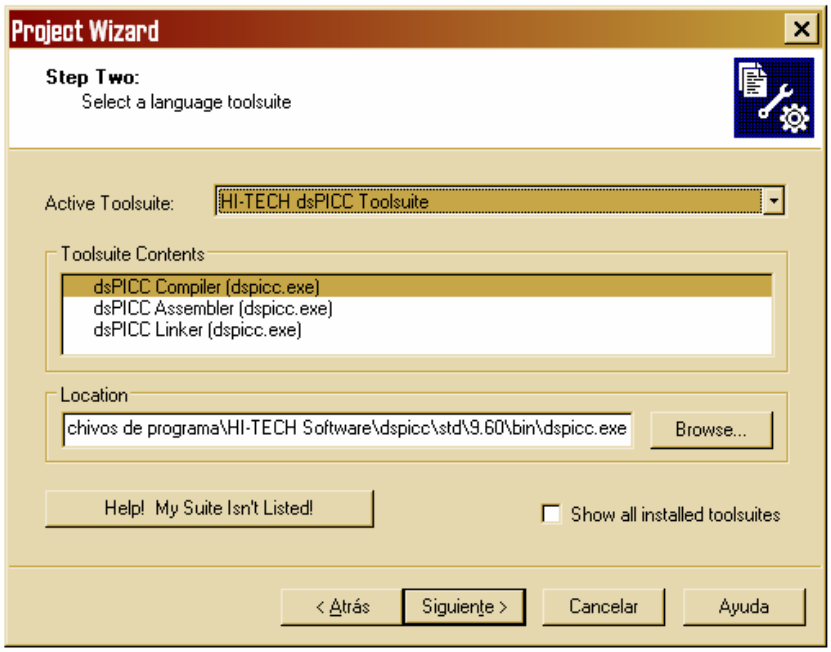
## 2.1. MPLAB.

MPLAB es un editor IDE(entorno de desarrollo integrado) gratuito, destinado a productos de la marca Microchip. Este editor es modular, permite seleccionar los distintos microprocesadores soportados. Es un programa que corre bajo Windows y como tal, presenta las clásicas barras de programa, de menú, de herramientas de estado, etc. El ambiente MPLAB posee editor de texto, compilador y simulación (no en tiempo real).
Los pasos a seguir para crear un proyecto en MPLAB son:

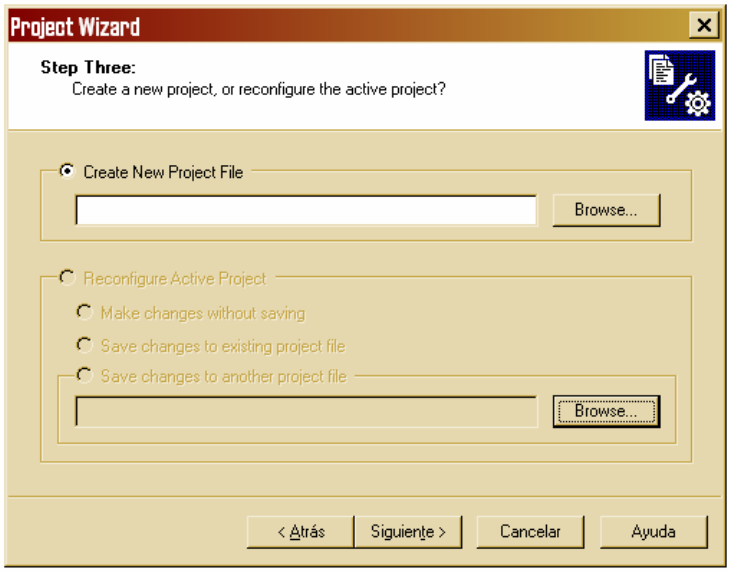
- Desde Project Wizard dentro del menú de Project, aparecerá la siguiente ventana donde se tiene que elegir el dispositivo empleado en este caso es el dsPIC30F4013



- En la siguiente ventana seleccionaremos el compilador, en este caso HI-TECH dsPIC Toolsuite, que es un compilador de lenguaje C.

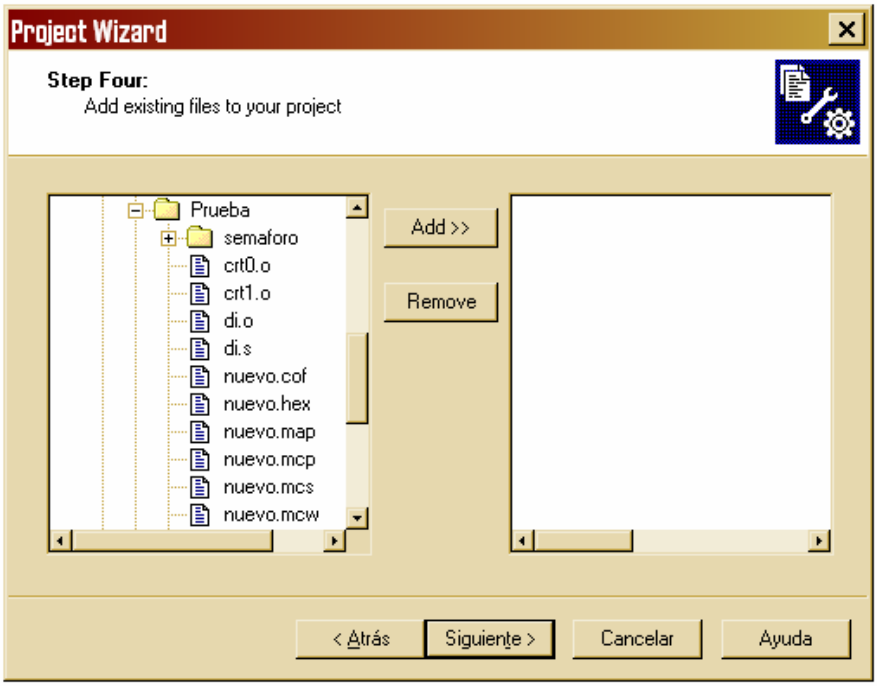



- Después se crea el proyecto indicando el nombre y el directorio.



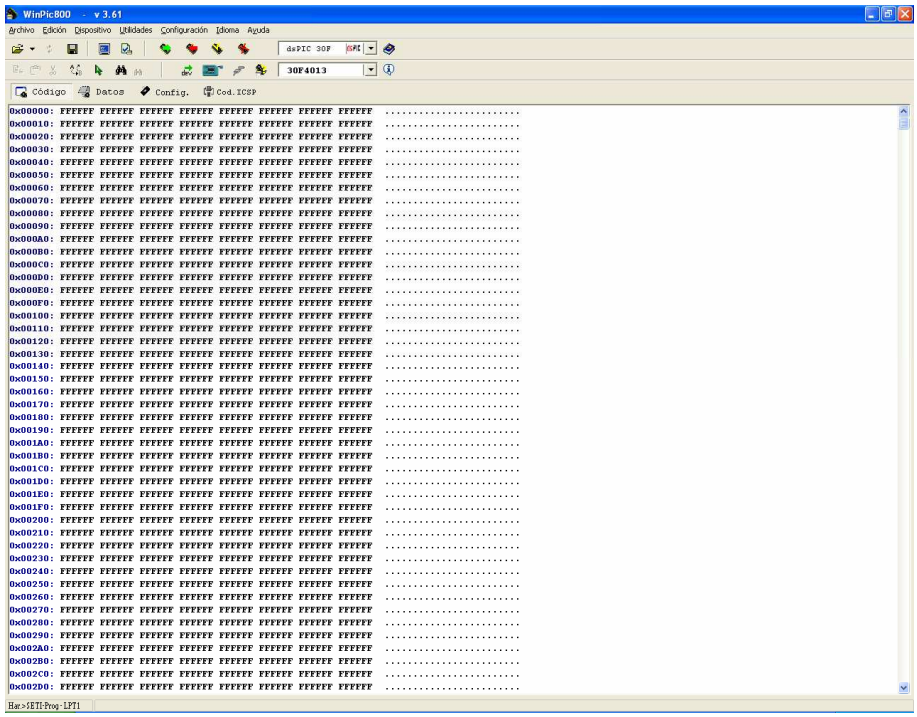
- Si se desean incluir ficheros al proyecto se hace en el siguiente paso, aunque es posible agregar nuevos ficheros una vez creado el proyecto.

  Una vez creado el proyecto se empieza con la programación.

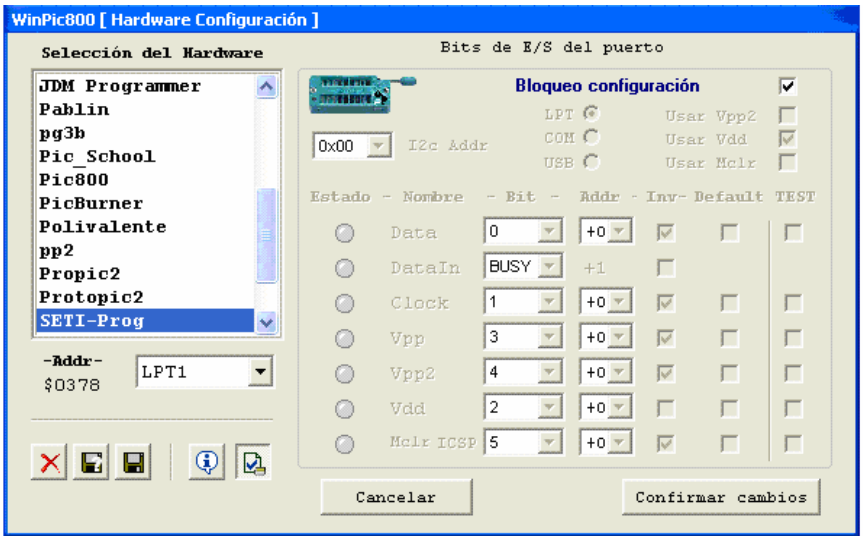


## 2.2. Winpic800.

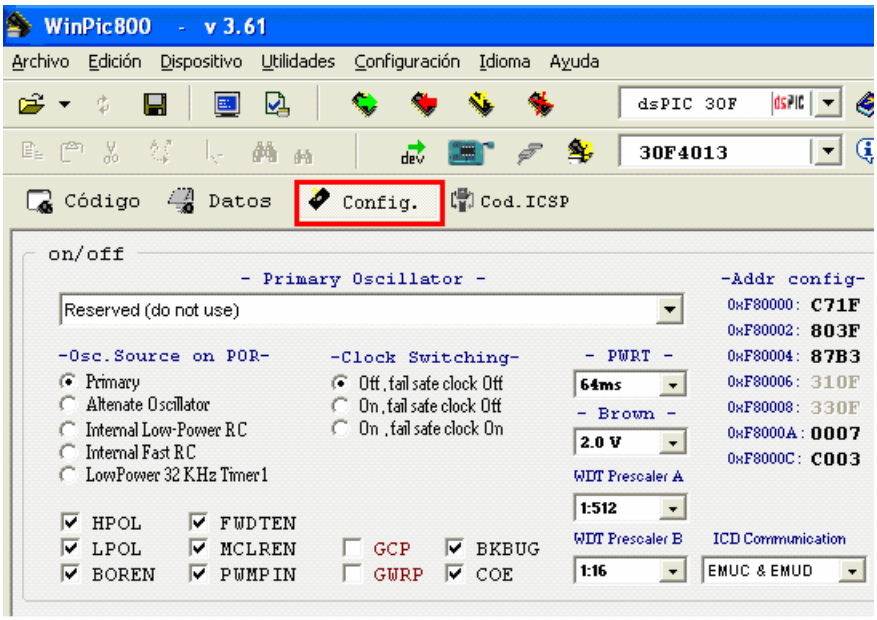
Es el programa empleado para programar el dsPIC empleando el programador via puerto paralelo diseñado en este proyecto.Al abrir el WinPic800 aparece la siguiente pantalla: El

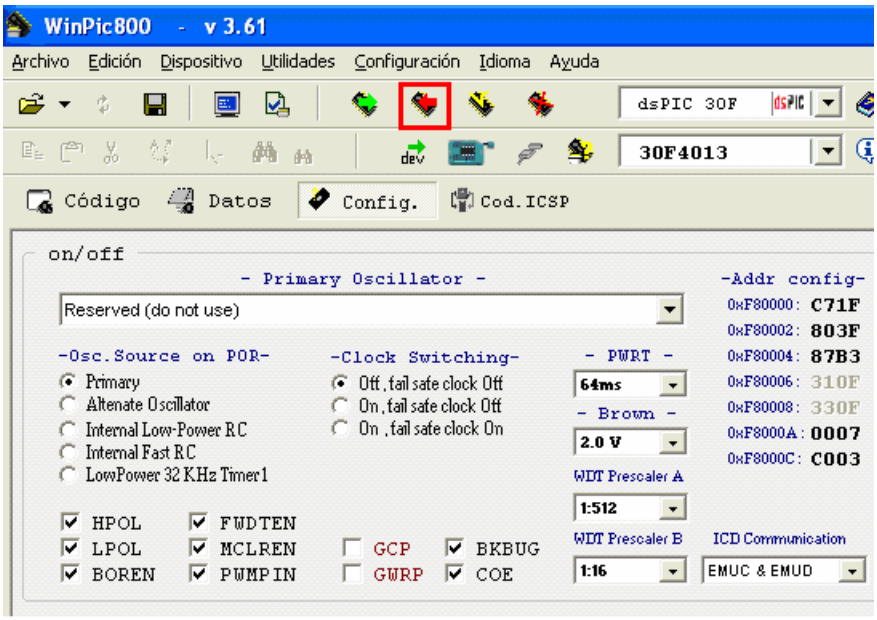



siguiente paso es configurar el programador para que funcione correctamente con el grabador diseñado:



En la ventana anterior también podemos realizar la comprobación del grabador modificando el valor de los distintos pines del puerto paralelo.
Una vez cargado el programa se configura el reloj que empleará el dsPIC, si se habilita o no el perro guardian ... Una vez configuradas las opciones deseadas se procederá a grabar el dsPIC

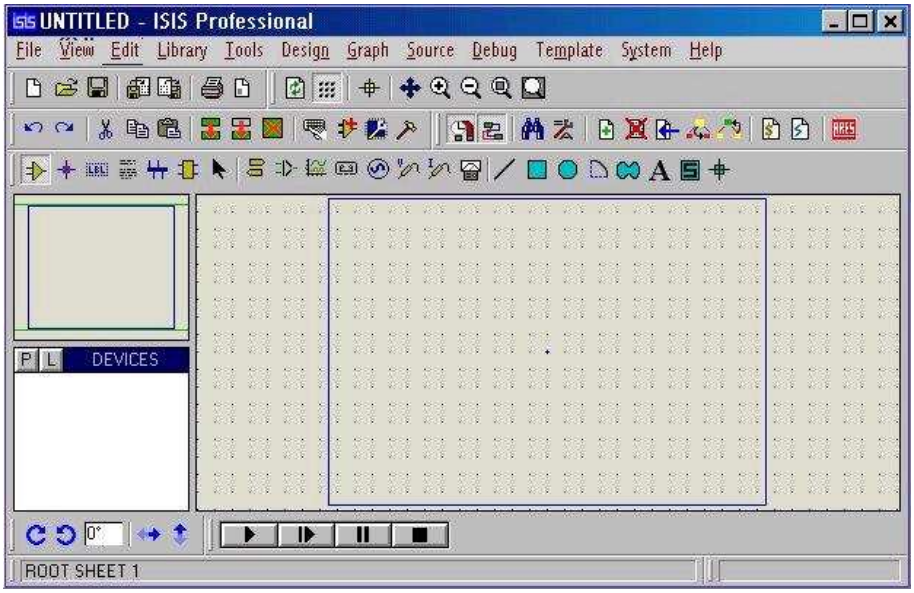


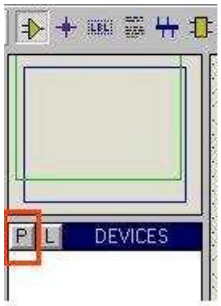
## 2.3. Proteus: Isis y Ares.

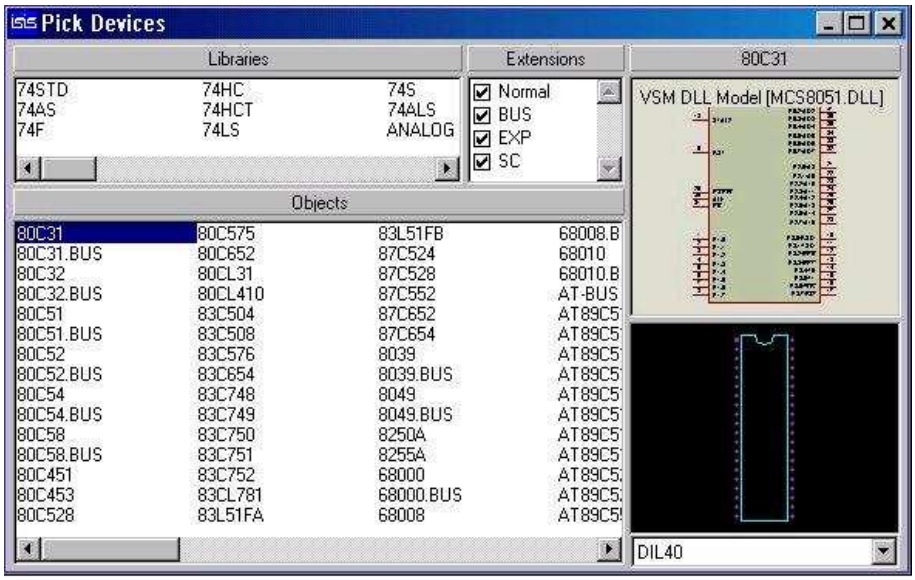
El software de diseño y simulación es Proteus que es una herramienta útil para el desarrollo de aplicaciones analógicas y digitales.
El sistema Proteus consta de dos módulos o programas diferenciados: ISIS que permite el diseño de circuitos empleando un entorno gráfico en el cual es posible colocar los símbolos representativos de los componentes y realizar la simulación de su funcionamiento sin el riesgo de ocasionar daños a los circuitos y ARES que es el módulo dedicado al diseño de placas de circuito impreso (PCBs). El procedimiento de arranque del programa es el siguiente:

- Inicio ⇒ Programas ⇒ Proteus 6 Professional ⇒ ISIS 6 Professional.

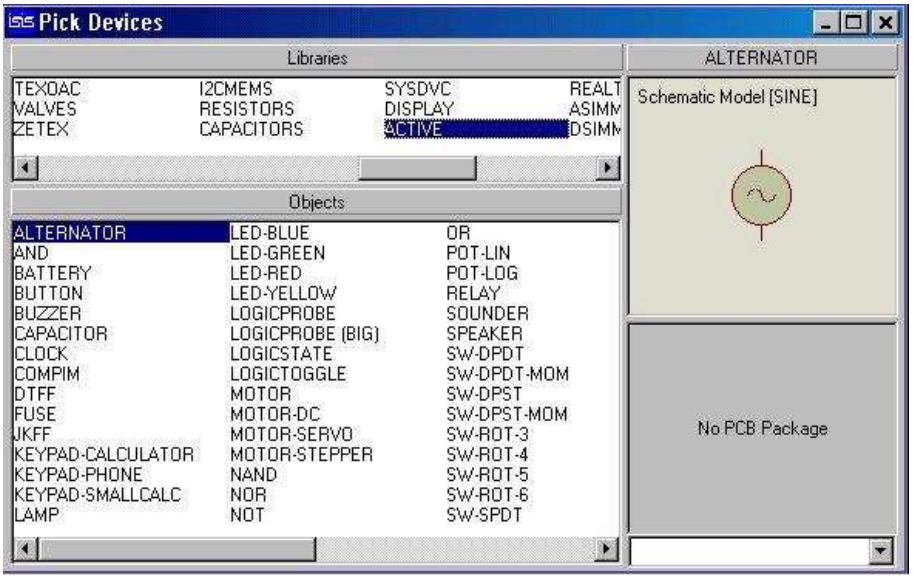



- Dar un click en el botón Pick Devices localizado en la parte izquierda de la pantalla debajo de la pantalla de exploración del diagrama para abrir la ventana en la que se buscan los componentes.



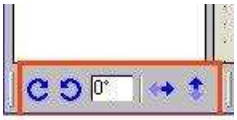



- En la ventana Libraries (Parte superior izquierda) buscar la librería deseada, en la que está el componente, y dar un click sobre ella. Otra opción es buscar el componente deseado en el buscador e incluirlo en el circuito.

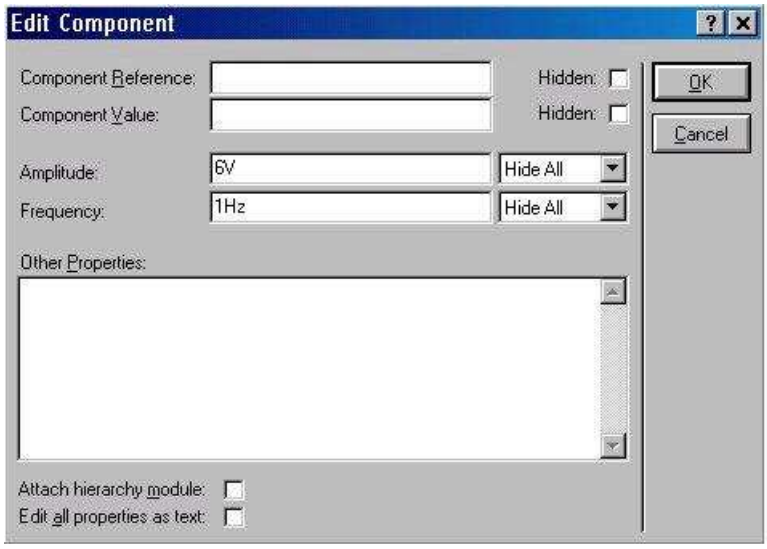



- Se puede observar que en la ventana DEVICES aparece el nombre del componente elegido. Si es el único componente que se va a elegir se puede cerrar la forma Pick Devices, pero si es necesario más de uno, se puede continuar eligiendo los componentes necesarios para el diseño.


- Explorar las funciones de orientación del componente, parte inferior izquierda de la pantalla.



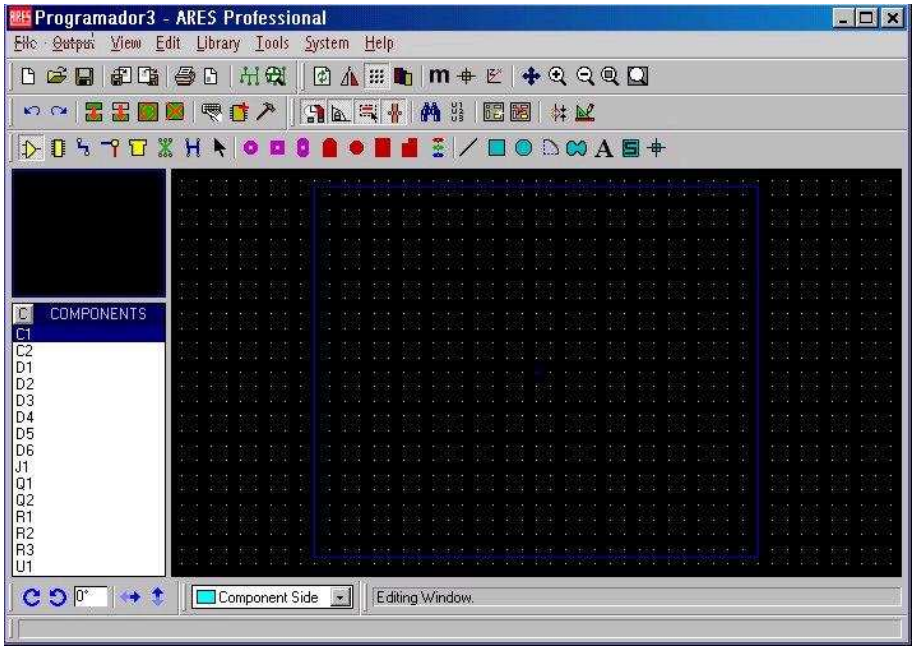

- Configurar los componentes para ello dar un click con el botón derecho sobre el componente y después dar un click con el botón izquierdo para abrir la forma Edit Component. En esta ventana se puede dar el nombre deseado a los componentes.



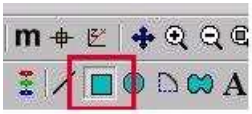

- Una vez buscados todos los componentes del circuito, se realizarán las conexiones y se podrá probar el funcionamiento del circuito pulsando el botón play que se encuentra en la parte inferior de la pantalla.

Una vez creado el circuito se abrirá la herramienta ARES para la creación de la placa de circuito impreso (PCBs). Ir al menú Tools y presionar Netlist Compiler, en el mismo menú presionar Netlist to ARES, o también se puede presionar el icono de ARES que se encuentra en la barra de herramientas. El cual genera el Netlist y lo exporta a ARES. Se abrirá la ventana


de trabajo de ARES Los componentes aparecen del lado izquierdo de la pantalla, es necesario




colocar los componentes dentro de un área que represente el tamaño de la placa que se quiere crear. El proceso de colocación manual sólo es necesario para los componentes que requieren un colocación especial en la placa, ya que los demás componentes se pueden colocar en forma automática usando el Auto Placer. Antes de poner los componentes se realiza la creación del borde que representa el tamaño de la placa a generar.

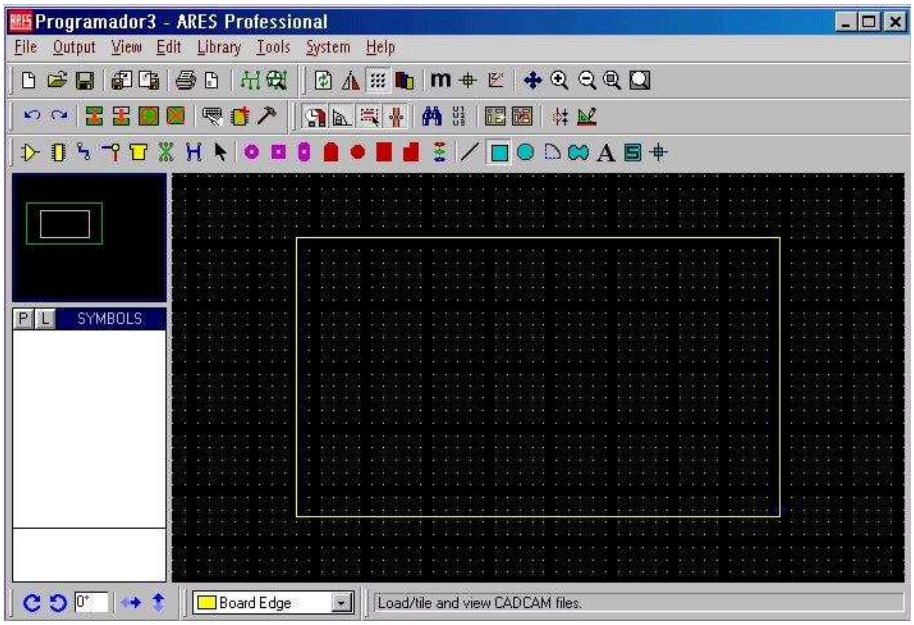


Antes de realizar lo anterior es necesario especificar en que Layer queremos dibujar el rectángulo. Esto se especifica en la parte inferior de la pantalla y se debe seleccionar "Board Edge".

Con esto se logra que el programa ARES pueda identificar el área de trabajo especificado en el

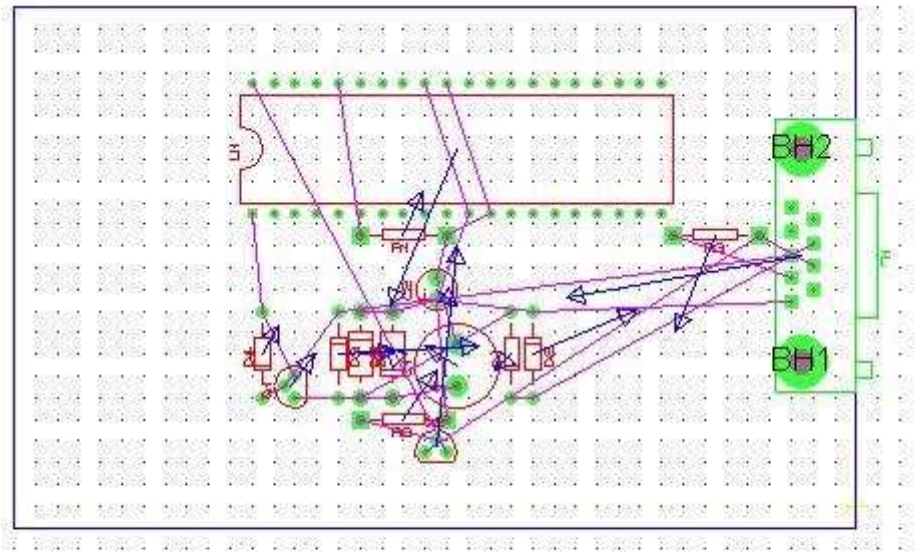

que se tienen que colocar los componentes y en donde se tiene que llevar a cabo el AUTORUTADO.

Después se colocan los componentes en el espacio antes creado o se usa la herramienta de Auto

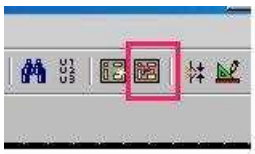


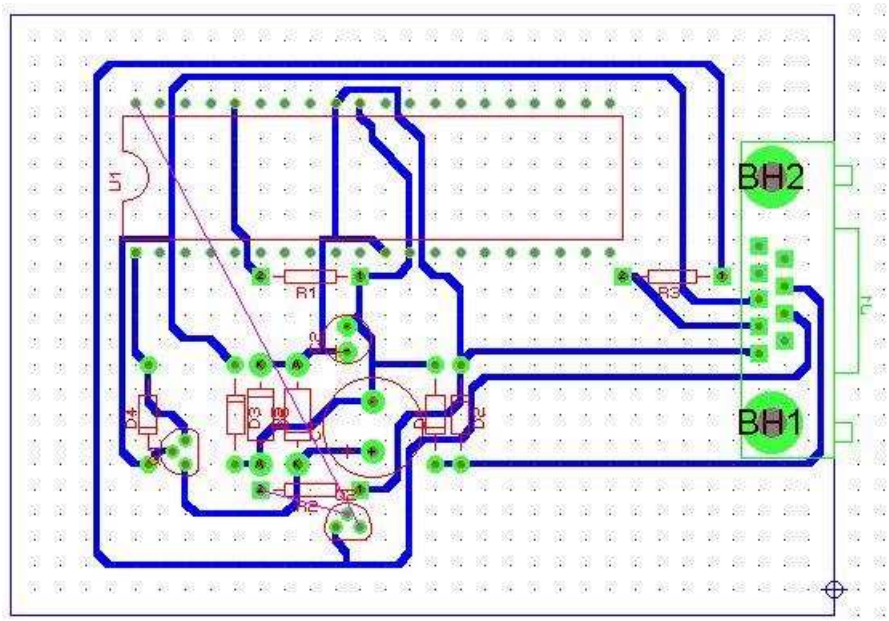

Place para que coloque los componentes, el Auto Placer se activa al dar un click sobre el icono de esta herramienta.

Al usar esta herramienta se obtiene el siguiente resultado: Otra herramienta que facilita




el trabajo es la posibilidad de generar las pistas de forma automática, utilizando el AUTO RUTER. Para ejecutar la AUTO-RUTER se da un click sobre la siguiente herramienta:
Se obtiene el siguiente resultado.






De esta forma se obtiene el rutado de la placa realizada, ahora es necesario emplear el programa que genere los vectores que hay que introducir en la fresadora para que cree la placa. Este programa es un herramienta creada por Jesus M. Mangas Hernández del departamento de Electricidad y Electrónica de la Escuela Técnica Superior de Ingenieros en Telecomunicación de la Universidad de Valladolid y se encarga de generar los vectores necesarios para fresar y taladrar la placa en la que se soldaran los componentes del circuito.

# Capítulo 3

# Diseño de un grabador para la familia dsPIC30F

Se pretende fabricar un grabador para poder almacenar los programas en un dsPIC de la familia 30F, a la vez se desea que sea compatible con el PIC16 utilizado actualmente en la asignatura.
Para comenzar con el desarrollo del grabador se definen una serie de características como por ejemplo que grabe a través del puerto paralelo en vez del puerto serie del ordenador, que se venía utilizando en la asignatura y que permita la conexión directa al puerto paralelo sin cables intermedios. Para la realización de este grabador se parte de un diseño realizado en PROTEUS como se puede observar en la figura 1.1.

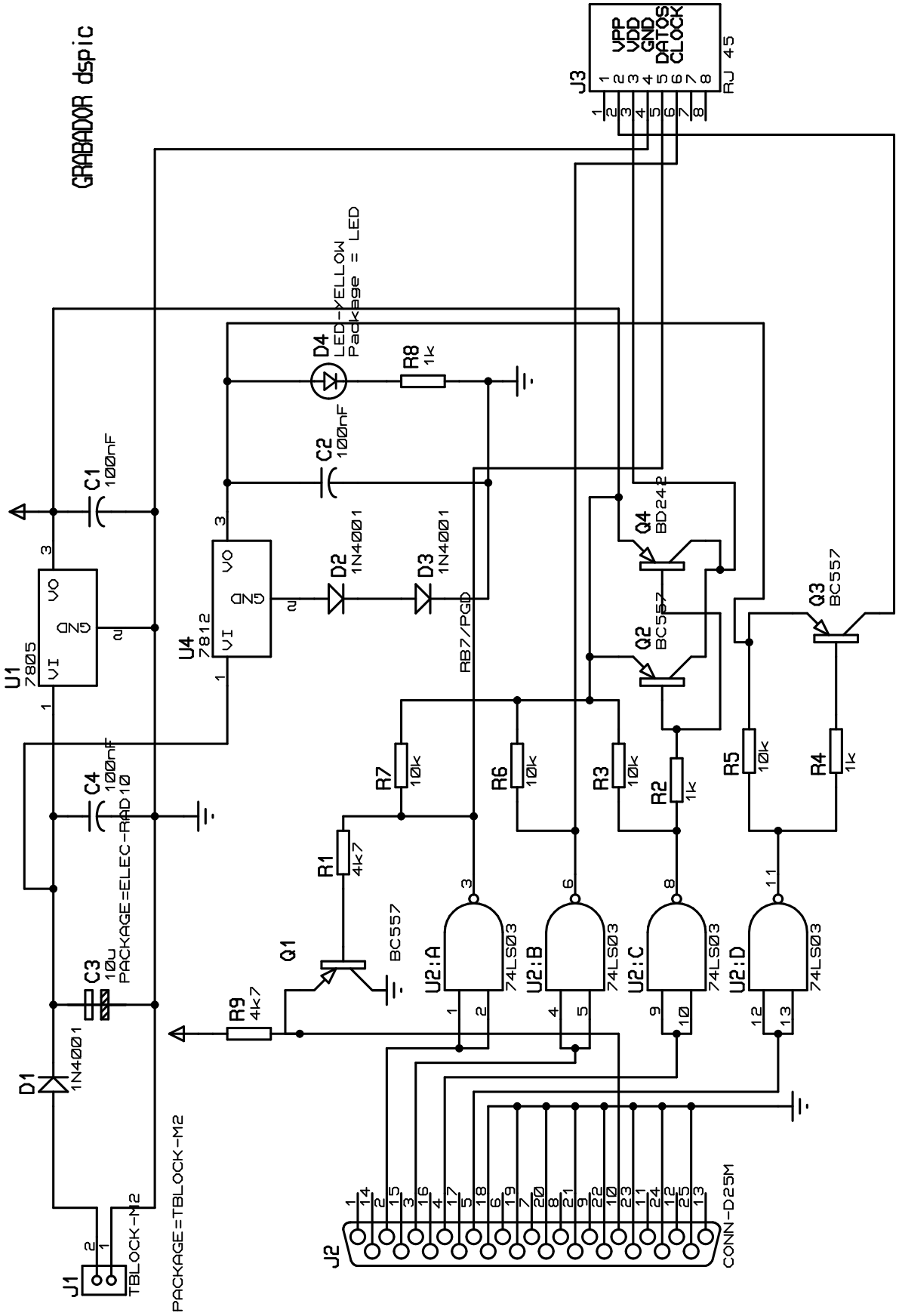



Figura 3.1: Primer diseño de un grabador para la familia dsPIC30F en Proteus.

El siguiente paso a realizar sería el rutado de dicho esquema, para ello se utiliza la herramienta ARES.
En este diseño se tuvo que realizar un nuevo componente, debido a que la librería de Proteus no lo incluía, para el conector RJ45 de 8 patillas. Para ello partiendo de un componente, con patillaje similar, se descompone en partes y se diseñó el conector como se deseaba, asociando a cada patilla el nombre correspondiente, una vez realizado este paso se le asocia un package que pertenece a la huella de dicho elemento y que previamente se realizó con la herramienta ARES y con las medidas exactas del componente, para que en el momento de la colocación de dicho elemento encajara perfectamente.

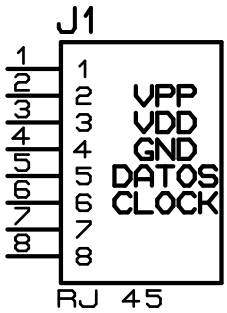



Figura 3.2: Conector RJ45 para Proteus.

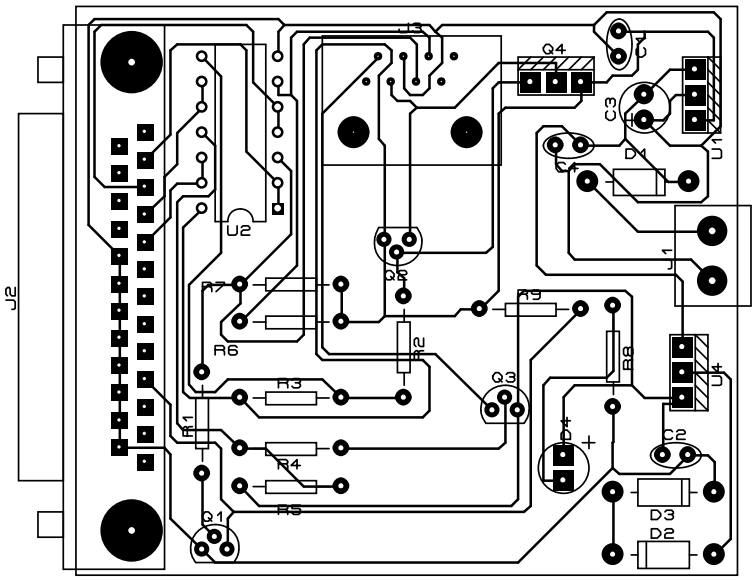



Figura 3.3: Primer diseño de un grabador para la familia dsPIC30F en ARES.

Una vez terminado el rutado se procede a la extracción de los ficheros necesarios para poder utilizar un programa que generará unos ficheros para poder emplear la fresadora disponible en el laboratorio. Para ello una vez que el rutado cumpla todas las reglas de diseño en la pestaña output en la opción CADCAM Output y se elige en la zona de Layers/Artworks: Botton Copper y Drill.

Posteriormente se utiliza el programa CreaFresado en el que primero cargamos el fichero de Botton Copper, luego se pulsa la tecla m para que realice el mirrow y frese de manera correcta y posteriormente a la tecla e, de ensanchar, de esta manera ensancha todas las pistas,luego a la tecla n para que elimine las zonas vacías y por último cargamos el fichero de los agujeros necesarios para colocar todos los componentes.

Se procede al fresado de la pieza para lo cual se prepara la fresadora cargando los ficheros .plt y .ncd en el ordenador que controla dicha fresadora y utilizando el programa que la maneja, que permite introducir dichos ficheros, uno permitirá grabar las pistas y el otro taladrar los agujeros de la placa.

Se coloca una lámina de cobre en la plataforma de la fresadora, bien sujeta para que no se mueva mientras se está fresando, y se coloca la broca necesaria en cada caso. El programa nos permite introducir un offset así como la profundidad necesaria, hay que tener cuidado a la hora de introducir la profundidad porque se puede llegar a partir la broca, conviene comenzar con poca profundidad e ir aumentando ligeramente hasta conseguir la profundidad deseada. Para finalizar se carga el fichero de los agujeros y se termina de taladrar.

Con un polímetro se comprueba que no exista ningún cortocircuito porque en ese momento se pueden repasar las pistas si es necesario.

Una vez creada la placa se procede a soldar los diferentes componentes de dicha placa con el soldador y el estaño disponibles en el laboratorio. Al finalizar es conveniente verificar que no hay ningún cortocircuito debido al estaño empleado y que todas las soldaduras estas bien hechas y hacen contacto de forma correcta.

Solo queda comprobar el funcionamiento del grabador para lo cual se testeará con el programa WINPIC800 y con un polímetro como se detalla a continuación.

Se conecta el grabador al puerto paralelo y a través de este programa vamos a testear todas las señales que llegan al conector RJ45. Para ello se configura el programa como se ve en la imagen y en la columna TEST se cambia el valor de cada pin y se comprueba con el polímetro que las patillas del conector también cambian y con las tensiones correctas, VPP es la tensión de programación y tiene que valer unos 12 voltios, VDD es la tensión de alimentación de 5 voltios, CLK es el reloj y varía entre 0 y 5 voltios al igual que datos. Si alguna de las tensiones no es correcta puede que exista un cortocircuito o una soldadura mal hecha y conviene revisar todas hasta encontrar el fallo y volver a testearlo.


Figura 3.4: Pantalla de pruebas del programa wimpic800.

Una vez realizado este paso se conecta el grabador a una placa de aplicaciones en la que está el dsPIC, configurando esta placa para trabajar con un dsPIc de 40 patas, podemos proceder a grabar nuestro programa. Si lo que se desea es trabajar con un PIC se fabricó de la misma manera un adaptador para el PIC cuyo diagrama es el que se muestra a continuación.

Posteriormente se genera un programa de pruebas que permitiera ver el correcto funcionamiento

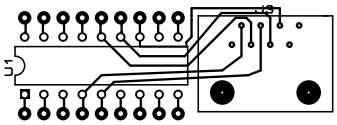


Figura 3.5: Diseño adaptador para un PIC16F88 en Proteus.

del grabador. Después de varias pruebas se observó que había un error en el grabador debido a la falta de unas resistencias y un diodo de protección por lo que se optó por un nuevo diseño del grabador. Repitiendo el proceso anterior con los mismos pasos y comprobando su funcionamiento el grabador utilizado en este proyecto es el que se muestra a continuación.

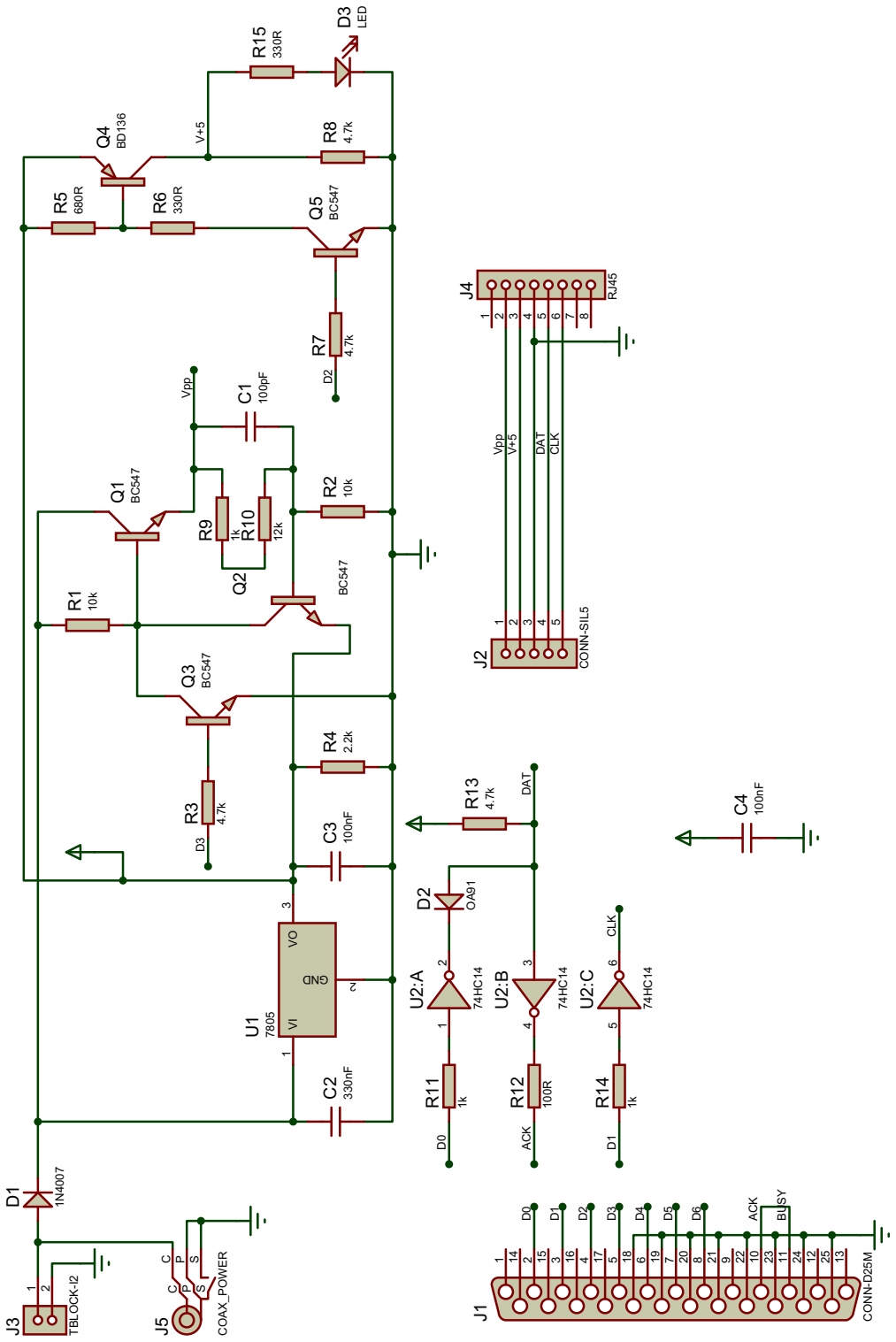



Figura 3.6: Diseño final de un grabador para la familia dsPIC30F en Proteus.

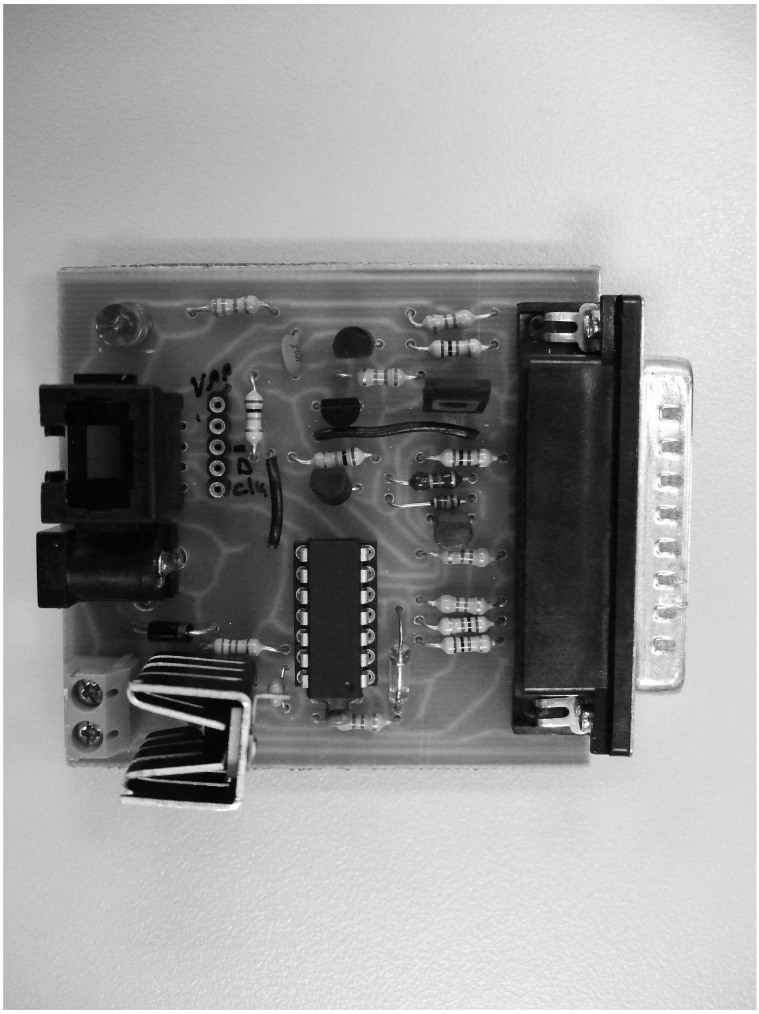



Figura 3.7: Placa del grabador.

# Capítulo 4

# Estudio detallado de la familia dsPIC30F

## 4.1. Presentación de los dsPIC.

### 4.1.1. ¿Qué son los dsPIC?.

Los dsPIC, también conocidos como DSC: Controlador Digital de Señal, son un potente microcontrolador de 16 bit a los que se le han añadido las principales capacidades de los DSP. Poseen conjuntamente los recursos de los microcontroladores de 16 bit y las principales características de los DSP.
Los dsPIC ofrecen todo lo que se puede esperar de un microcontrolador: velocidad, potencia, manejo de interrupciones, amplio campo de funciones periféricas analógicas y digitales, opciones de reloj, perro guardián, reloj de tiempo real..., a un precio similar al de los microcontroladores.

### 4.1.2. Principales características de los dsPIC.

- CPU de alto rendimiento: Núcleo RISC con arquitectura Harvard modificada.
- Bus de datos de 16 bits y de instrucciones de 24 bits.
- Repertorio de 84 instrucciones optimizadas para el lenguaje C.
- 16 registros de propósito general.
- 2 acumuladores de 40 bits con opciones de redondeo o saturación.
- Modos complejos de direccionamiento indirecto: circular y de bit inverso.

- Controlador de interrupciones: Latencia de 5 ciclos.
- Hasta 45 fuentes de interrupción, 5 externas.
- 7 niveles de prioridad para las interrupciones programables.
- 4 excepciones especiales.
- Entradas y salidas digitales.
- Memorias: FLASH de programa de hasta 144 KB, EEPROM de datos de hasta 4 KB y SRAM de datos de hasta 8 KB.

## 4.2. CPU.

### 4.2.1. Introducción.

La CPU del dsPIC30F es de 16 bits (datos) con arquitectura Harvard modificada y con un amplio conjunto de instrucciones, entre ellas instrucciones de DSP. La CPU tiene palabras de instrucción de 24 bits, con un campo de código de operación de longitud variable. El contador de programa (PC) es de 24 bits y puede direccionar hasta 4M x 24 bits de memoria de programa de usuario. Todas las instrucciones se ejecutan en un único ciclo de reloj, con la excepción de que las instrucciones para cambiar el flujo del programa, las que mueven dobles palabras y la tabla de instrucciones. Soporta lazos de programa con las instrucciones DO y REPEAT, que se pueden interrumpir en cualquier momento.
Los dispositivos dsPIC30F tiene 16 registros de trabajo de 16 bits en modelo de programación. Cada uno de los registros de trabajo puede actuar como un dato, dirección o indexado a una dirección. El registro de trabajo W15 funciona como un puntero de pila software para interrupciones y llamadas a rutinas.
El conjunto de instrucciones dsPIC30F se divide en dos clases de instrucciones, las instrucciones MCU y las DSP. Estas dos clases de instrucción se integran a la perfección en la arquitectura y se ejecutan en un único ciclo de ejecución. El conjunto de instrucciones incluye modos de direccionamiento y están diseñados para una compilación eficiente en C.
Se puede acceder al espacio de datos como 32K de palabras o 64 Kbytes y se divide en dos bloques, X e Y. Cada bloque de memoria tiene su propia Unidad Generadora de Direcciones (AGU) independiente. Las clases de instrucciones MUC operan únicamente a través de la memoria X y algunas instrucciones de DSP operan a través de las AGUs X e Y.

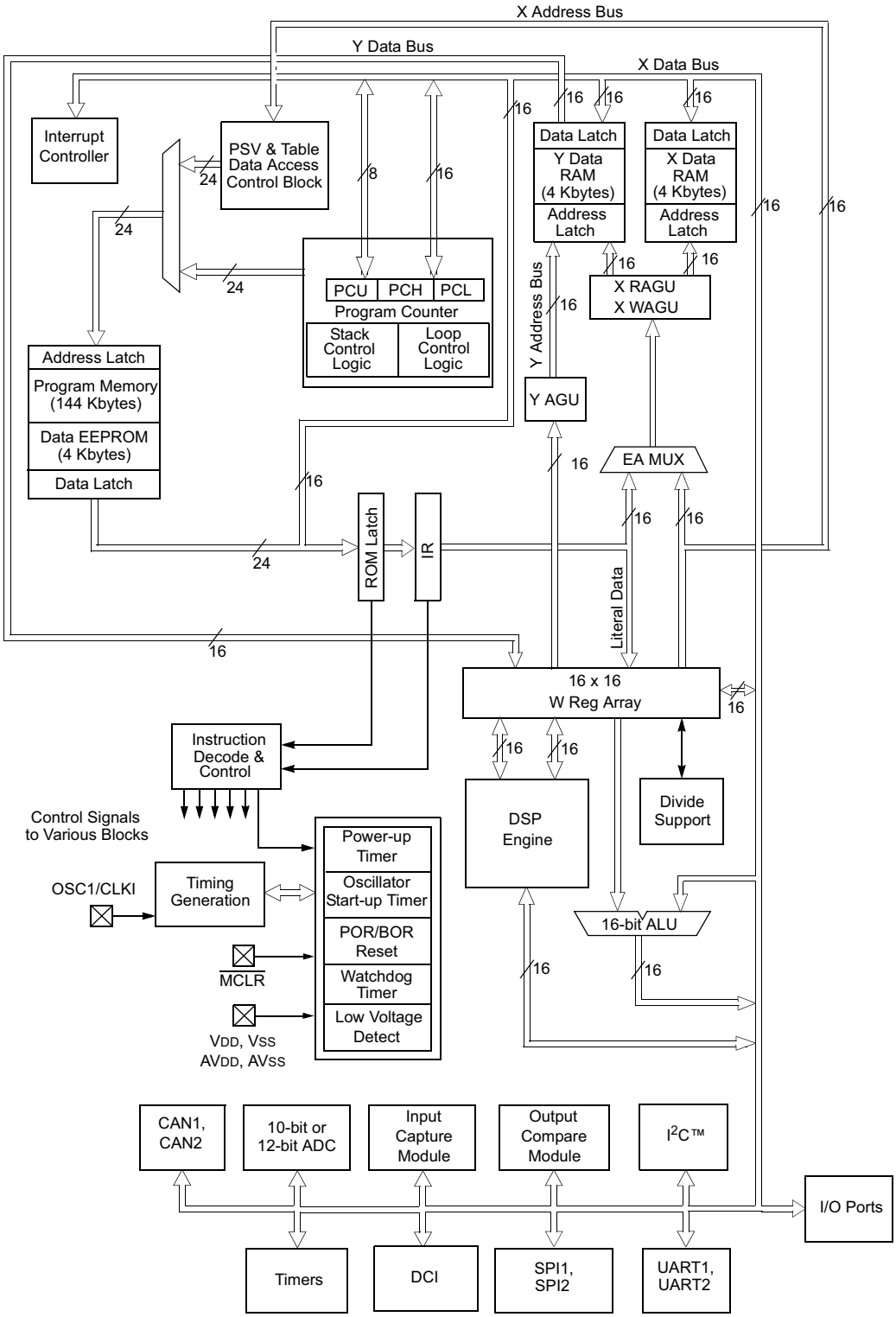



Figura 4.1: Diagrama de bloques del núcleo de la CPU.

### 4.2.2. Modelo de programación.

- **INTRODUCCIÓN**

  En las tabla se muestran los registros del modelo de programación, todos los registros

están mapeados en memoria y pueden manipularse mediante las instrucciones.

| Register(s) Name | Description |
|---|---|
| W0 through W15 | Working register array |
| ACCA, ACCB | 40-bit DSP Accumulators |
| PC | 23-bit Program Counter |
| SR | ALU and DSP Engine Status register |
| SPLIM | Stack Pointer Limit Value register |
| TBLPAG | Table Memory Page Address register |
| PSVPAG | Program Space Visibility Page Address register |
| RCOUNT | REPEAT Loop Count register |
| DCOUNT | DO Loop Count register |
| DOSTART | DO Loop Start Address register |
| DOEND | DO Loop End Address register |
| CORCON | Contains DSP Engine and DO Loop control bits |

Figura 4.2: Tabla con los registros del modelo del programador, la dirección que ocupan.

- **EL BANCO DE REGISTROS DE TRABAJO W**

  Está compuesto por 16 registros de 16 bits cada uno, que pueden almacenar datos, direcciones o desplazamientos de direcciones, dependiendo la instrucción empleada se determina la misión del registro W que referencia:

  - Instrucciones de registros de trabajo: Usan los registros de trabajo como datos o como direcciones, que apuntan a posiciones de memoria.
  - Instrucciones de posiciones de memoria de datos: Manejan un operando de la memoria, que está direccionado por el contenido del código OP y el contenido de W0, cuando no se especifica otro operando. En este caso W0 es tratado como WREG.

  Los registros W están mapeados en la memoria de datos, por lo es posible acceder a cada uno de ellos usando una instrucción de posición de memoria.

```
Ejemplo 1
MOV 0x0004, W8          ;es equivalente a la instrucción MOV W2,W8.
```

  Las instrucciones que manejan operandos de tamaño byte sólo afectan al byte de menos peso de ese registro.
  A continuación se describe cada uno de los registros:

  - **W0/WREG**: Es el registro de trabajo por defecto. Es el único que puede ser utilizado en instrucciones de posiciones de memoria de datos.

- **W1-W3**: Junto con el W0 estos registros disponen de un registro sombra, lo que hace que sean los registros más usados como acumuladores para las instrucciones DSP.
- **W4-W7**: Se usan habitualmente como registros de operandos para las instrucciones DSP.
- **W8-W11**: Normalmente se usan como registros de direcciones para las instrucciones DSP.
- **W12**: Se utiliza en los direccionamientos para fijar un desplazamiento.
- **W13**: Es usado en la post-escritura MAC.
  Para direccionar este valor existen dos métodos:
  - Directo, W13:
    El contenido del acumulador (ACCxH¡31:16¿) no usado en ese momento se escribe en W13.
  - Indirecto con post- incremento, [W13]+=2:
    El contenido del acumulador no usado en ese momento, se escribe en la dirección apuntada por W13. Posteriormente, la dirección de W13 se incrementa en dos.
- **W14**: Puntero del marco de pila.
- **W13**: Puntero de la pila de los dsPIC30F.

**Registros sombra**
Los registros sombra acompañan a registros concretos y no son accesibles directamente. Los hay de dos tipos:

- Los registros sombra que son utilizados por las instrucciones PUSH.S y POP.S.
- Los registros sombra que son utilizados por la instrucción DO.

Los registros sombra del primer grupo son los que están asociados a las instrucciones PUSH.S y POP.S, que se encargan de salvar y restaurar los valores de los registros implicados en las funciones de LLAMADA A SUBRUTINA o ATENCIÓN A RUTINA DE INTERRUPCIÓN.
Los registros que cuentan con registros sombra asociados para las instrucciones PUSH.S y POP.S son: W0-W3 y SR( sólo los bits N,OV,Z,C,DC).

```
Ejemplo 2
PUSH.S          ;salva los registros W0-W3 y el de estado (SR).
```

```
MOV #0x03,W0     ;W0 = 3.
ADD RAM100       ;suma W0 al contenido de RAM100.
BTSC SR,#Z       ;compara con cero, para ver si el resultado de la suma es cero.
BSET Flags,\#lsZero     ;Si es 0, Flag=1.
POP.S            ;restaura los registros W0-W3 y el de estado.
RETURN
```

Existen además tres registros sombra (DOSTART, DOEND, DCOUNT) que son salvados automáticamente cuando se ejecuta la instrucción D0, y permiten anidar automáticamente dos bucles.

- **EL PUNTERO DE PILA.**
  La pila de los dsPIC30F se maneja por software con las instrucciones PUSH y POP, además de funcionar automáticamente en las llamadas a subrutinas, en las interrupciones y en los retornos.
  El registro W15 apunta a la cima de la pila, Stack Pointer SP, y su valor se modifica de forma automática an las llamadas a subrutinas, retornos y atención a las excepciones e interrupciones. Tras un RESET en registro W15 toma el valor de 0x0800.

  - **Funcionamiento del puntero de pila W15.**
    El W15 apunta a la primera palabra disponible de la pila y decrementa su valor en las lecturas de pila efectuadas con la instrucción POP, mientras que post-incrementa su valor en las escrituras de pilas realizadas mediante PUSH.

    Ejemplo 3

    ```
    1. PUSH W0      ;Se escribe el valor de W0 en la posición apuntada por W15.
       Esta instrucción es equivalente a:
       MOV W0,[W15++]

    2. POP W0       ;La palabra apuntada por W15 se carga en W0.
       Esta instrucción es equivalente a:
       MOV[--W15],W0
    ```

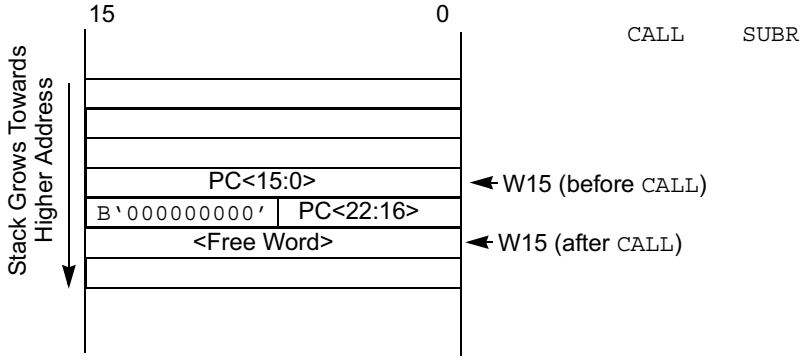



Figura 4.3: Ejemplo de uso de la pila.

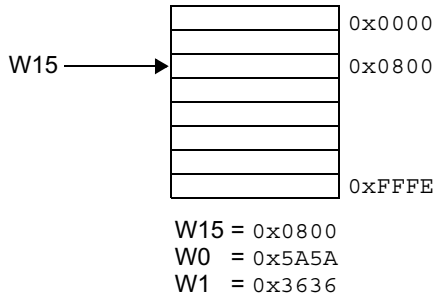



Figura 4.4: Puntero de pila después del reset.

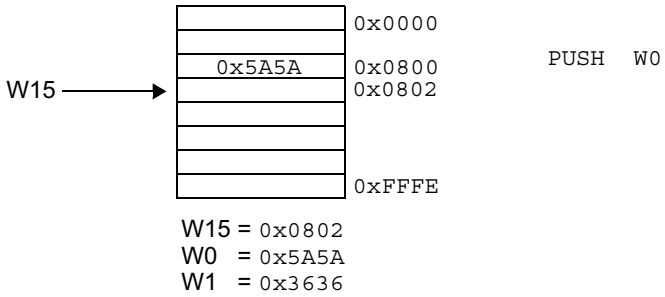



Figura 4.5: Puntero de pila después del primer PUSH.

- **Puntero del marco de pila.**
  Un marco es una sección de la pila definida por el usuario para ser usada por una subrutina. W14 en el registro especial de trabajo que se usa como puntero de marco con las instrucciones LNK(vincular) y ULNK(desvincular).

- **Desbordamiento del puntero de pila.**
  Existe un registro limite de la pila (SPLIM), este consta de 16 bits en el que el de menos peso es cero para mantener la alineación en todas las operaciones de pila. La detección de desbordamiento se habilita cuando se escribe una palabra en SPLIM

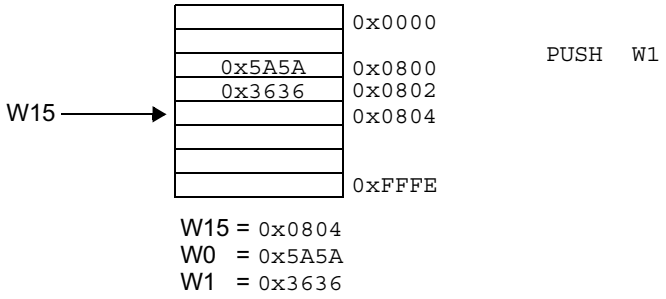



Figura 4.6: Puntero de pila después del segundo PUSH.

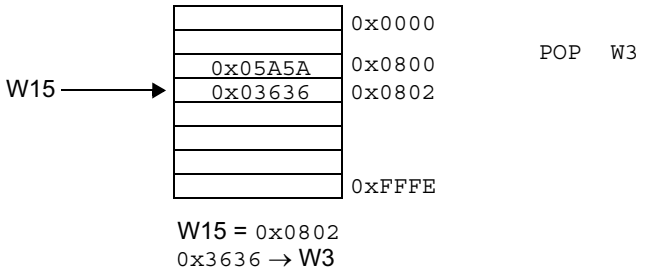



Figura 4.7: Puntero de pila después de un POP.

y sólo puede desactivarse mediante un RESET, una vez habilitada se comparan las direcciones efectivas empleadas por W15 con el valor de SPLIM generando una excepción de error si se supera el valor de ese registro.

- **LOS REGISTROS DE LA CPU.**

  A continuación se describe el conjunto de los registros que completan la CPU.

  - **El registro de Estado (SR).**

    Es un registro de 16 bits que se divide en dos partes: el byte más significativo SRH que monotoriza el estado del DSP contiene los bits de estado del sumador/restador DSP, el bit de estado de bucle activo D0 (DA) y el señalizador de acarreo en el cuarto bit (DC,) y el byte de menos peso SRL que informa del estado del microcontrolador, contienen los señalizadores de la ALU (N,OV,Z y C), los tres bits de prioridad de interrupciones y el bit de bucle activo REPEAT (RA).

    Cuando se procesa una excepción SRL se concatena con el byte de más peso del registro PC para completar una palabra que se salva en la pila.

    En la siguiente figura vemos el contenido del registro SR:

| Upper Byte: | | | | | | | |
|---|---|---|---|---|---|---|---|
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| OA | OB | SA | SB | OAB | SAB | DA | DC |
| bit 15 | | | | | | | bit 8 |

| Lower Byte: | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

Figura 4.8: Registro de estado SR.

- **OA:** Desbordamiento del acumulador A en los bits de guarda.
- **OB:** Desbordamiento del acumulador A en los bits de guarda.
- **SA:** Saturación en el acumulador A.
- **SB:** Saturación en el acumulador B.
- **OAB:** Desbordamiento del acumulador A o B.
- **SAB:** Saturación en el acumulador A o B.
- **DA:** Bucle DO activo.
- **DC:** Acarreo de tipo BCD.
- **IPL0-IPL2:** Nivel de prioridad de las interrupciones:
  - 111 : Nivel de prioridad de interrupción 7 (15 si IPL3 = 1), interrupciones de usuario desactivadas.
  - 110: Nivel de prioridad de interrupción 6 (14 si IPL3 = 1).
  - 101: Nivel de prioridad de interrupción 5 (13 si IPL3 = 1).
  - 100: Nivel de prioridad de interrupción 4 (12 si IPL3 = 1).
  - 011: Nivel de prioridad de interrupción 3 (11 si IPL3 = 1).
  - 010: Nivel de prioridad de interrupción 2 (10 si IPL3 = 1).
  - 001: Nivel de prioridad de interrupción 1 (9 si IPL3 = 1).
  - 000: Nivel de prioridad de interrupción 0 (8 si IPL3 = 1).
- **RA:** Bucle REPEAT activo.
- **N:** Resultado de la operación negativo.
- **OV:** Desbordamiento en la operación.
- **Z:** Resultado de la operación cero.
- **C:** Acarreo del bit de más peso del resultado.

- **El registro de control del núcleo (CORCON.**

  Contiene los bits que controlan las operaciones de multiplicación DSP y bucle DO, además de contener el bit IPL3 que concatenado con los IPL0-IPL2 del SR proporcionan el nivel de prioridad de las interrupciones.
  En la siguiente figura vemos el contenido del registro CORCON:

  La información que nos proporciona cada uno de estos bits es:

| **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 |

Figura 4.9: Registro CORCON.

  - **US:** Cuando es 1 la multiplicación DSP es sin signo, sino es con signo.
  - **EDT:** Terminación rápida del bucle DO.
  - **DL0-DL2:** Número de bucles DO anidados.
  - **SATA:** Saturación automática del acumulador A(SATA =1), sino desbordamiento.
  - **SATB:** Saturación automática del acumulador B(SATB =1), sino desbordamiento.
  - **SATDW:** Saturación automática para la escritura el la memoria de datos.
  - **ACCSAT:** Selección de supersaturación o saturación normal.
  - **IPL3:** Forma el nivel de prioridad de la interrupción junto con IPL0-IPL2.
  - **PSV:** Habilitador de la visualización del espacio de programa en el de datos.
  - **RND:** Selección del modo de redondeo. Tradicional ( RND = 1) o convergente.
  - **IF:** Modo entero (IF=1) o fraccionado de multiplicación.

- **OTROS REGISTROS DE CONTROL DE LA CPU.**

  **TBLPAG:** Registro de dirección de página de tabla de memoria.
  Contiene los 8 bits más significativos de la dirección de memoria de programa

durante las operaciones de lectura y escritura de tabla. Las instrucciones de tabla son empleadas para transferir datos entre la memoria de programa y la memoria de datos.

**PSVPAG:** Registro de dirección de página de visibilidad del espacio del programa.
La visibilidad del espacio del programan permite al usuario mapear una sección de 32 KB de memoria de programa en los 32 KB superiores de la memoria de datos, lo que permite el acceso transparente de constantes con las instrucciones del dsPIC30F. Este registro selecciona la región de la memoria de programa que se mapeada en el espacio de datos.

**MODCON:** Registro de control de direccionamiento modular.
Este registro se utiliza para activar y configurar el direccionamiento modular.

**XMODSTR,XMODEND:** Registros de direcciones de comienzo y final del módulo X.

**YMODSTR,YMODEND:** Registros de direcciones de comienzo y final del módulo Y.

**XBREV:** Registro de módulo de acarreo invertido.
Determina el tamaño de la memoria usada para el direccionamiento por acarreo invertido.

**Acumulador A y B:**
Los acumuladores son registros de 40 bits usados en las instrucciones DSP para realizar operaciones matemáticas y de desplazamientos.

**Contador de Programa PC:**
Registro de 23 bits, el bit de menos peso siempre esta a 0 para mantener el alineamiento, por lo que se incrementa de dos en dos unidades.

**RCOUNT:**
Registro de 16 bits que contiene el contador de bucles REPEAT. El bucle se ejecuta RCOUNT + 1 veces.

**DCOUNT:**
Registro de 16 bits que contiene el contador de bucles DO. El bucle se ejecuta DCOUNT + 1 veces.

**DOSTART:**
Contiene la dirección de comienzo de un bucle DO.

**DOEND:**
Contiene la dirección fin de un bucle DO, la dirección de la última instrucción del bucle DO.

**DISICNT:**

Es un registro que utiliza la instrucción DISI, que inhabilita las interrupciones con prioridad comprendida entre 1 y 6, el número de ciclos que contiene DISICNT.

### 4.2.3. Unidad Aritmético Lógica ALU.

- **ARQUITECTURA INTERNA.**

  El motor DSP es un conjunto de potentes recursos físicos de tipo matemático que proporciona al procesador la posibilidad de realizar operaciones especiales a gran velocidad. No es posible realizar de forma simultánea una operación del motor DSP y una instrucción MCU, pero si pueden utilizarse de forma simultánea el motor DSP con la ALU de la MCU.

  En la figura 4.10 vemos el motor DSP.

  Los elementos que conforman el motor DSP son los siguientes:

  - Multiplicador de alta velocidad de 17 x 17 bits.
  - Registro desplazamiento de +/- 16 bits con 40 bits de longitud.
  - Sumador/Restador de 40 bits.
  - Dos Acumuladores de 40 bits.
  - Módulo de redondeo lógico.
  - Módulo de saturación lógica.

  Los datos de entrada del motor DSP pueden proceder de tres fuentes:

  - Del bando de registros W (W4-W7) desde los buses de datos X e Y, para las instrucciones tipo MAC.
  - Del bus de datos X para el resto de instrucciones DSP.
  - Del bus de datos X para todas las instrucciones MUC que utilicen el registro de desplazamiento.

  Los resultados del motor DSP se depositan en alguno de los siguientes destinos:

  - El acumulador definido en la instrucción.
  - El bus de datos X para la escritura del acumulador, usándose la dirección almacenada en W13.

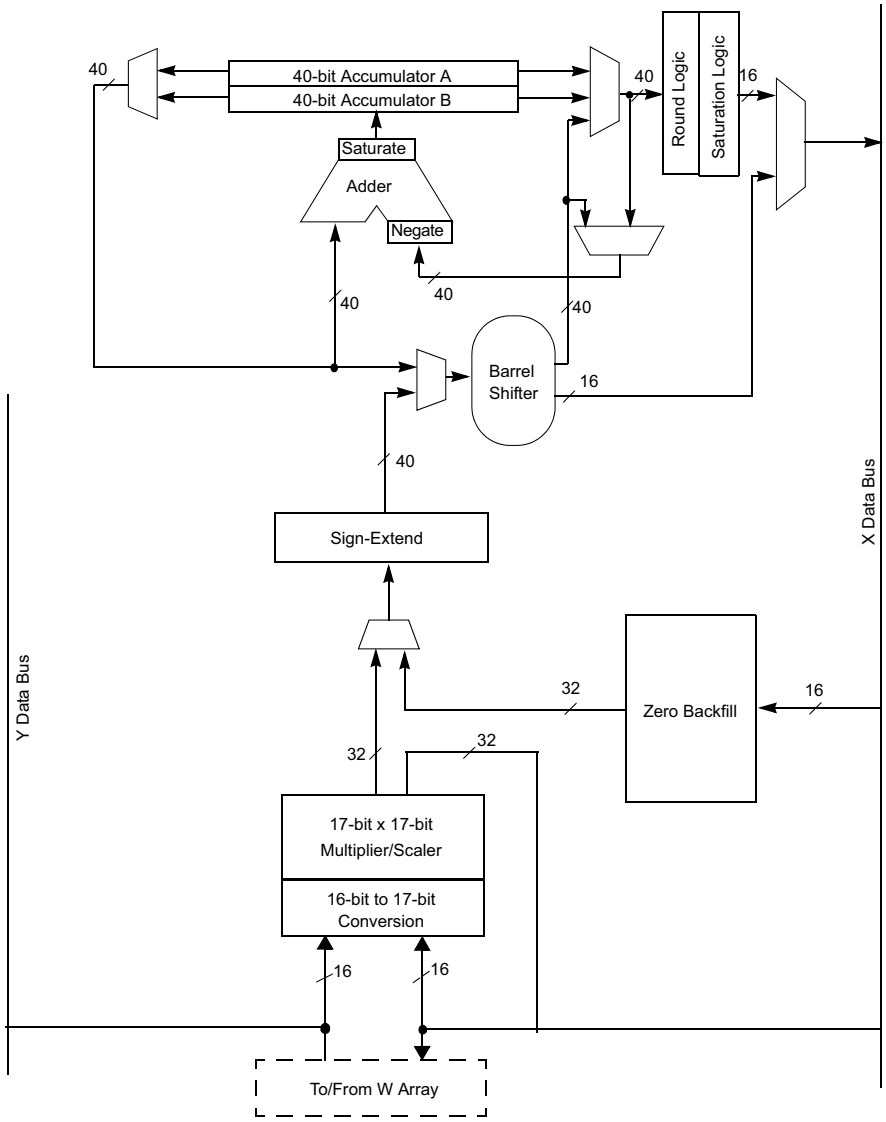



Figura 4.10: Arquitectura interna del motor DSP.

- El bus de datos X para todas las instrucciones MCU que usan el registro desplazamiento.

- **EL MULTIPLICADOR.**

  Se trata de un multiplicador hardware muy rápido que admite operandos de 17 bits y que puede operar con signo y si él para proporcionar un resultados de 32 bits. Este dispositivo va a ser compartido por la AlU de la MCU y el motor DSP.

  Los operandos de entrada de 16 bits se convierten en 17 al extenderse el signo si existe, en caso contrario se pondrá un cero.

  El bit IF del registro CORCON determina si la operación se realiza en formato entero

o en fraccionario. En las instrucciones MCU siempre se realiza en formato entero. En las operaciones con números fraccionarios el multiplicado desplaza un bit a la izquierda, borrando el bit de menos peso del resultado.
El bit US del registro CORCON determina cuándo las instrucciones DSP multiplican con o sin signo. Si US = 1 los operando de entrada se consideran sin signo.

2.1. Instrucciones del multiplicador para el DSP.

| | |
|---|---|
| MAC | a = a + b·c |
| MSC | a = a - b·c |
| MPY | a = b·c |
| MPY.N | a = -b·c |
| ED | a = $(b - c)^2$ |
| EDAC | a = a + $(b - c)^2$ |

2.2. Instrucciones del multiplicador para la MCU.

| | |
|---|---|
| MLU/MLU.U | Multiplicación de dos enteros sin signo |
| MUL.S | Multiplicación de dos enteros con signo |
| MUL.SU/MUL.US | Multiplicación de un entero con signo y otro sin signo |

- **SUMADOR/RESTADOR.**
  Los dos acumuladores disponen de un Sumador/Restador de 40 bits que puede seleccionar a uno de ellos como fuente con pre-acumulación o como destino con post-acumulación. En las operaciones de suma de acumuladores (ADD) y carga (LAC) los datos que van a a ser acumulados o cargados pueden, opcionalmente, ser escalados en el registro de desplazamiento antes de acumularse.
  El Sumador/Restador puede negar uno de los operandos de entrada para cambiar el signo del resultado, esto se usa en las instrucciones MSC y MPY.N.
  El Sumador/Restador dispone de un módulo de saturación que controla la saturación del dato del acumulador cuando está activo dicho módulo.

  - Bits de estado para los acumuladores.
    - OA ⇒ Desbordamiento del acumulador A en los bits de guarda.
    - OB ⇒ Desbordamiento del acumulador B en los bits de guarda.
    - SA ⇒ Saturación en el acumulador A.
    - SB ⇒ Saturación en el acumulador B.
    - OAB ⇒ Operación lógica OR de OA y OB.

- SAB ⇒ Operación lógica OR de SA y SB.

OA y OB son de sólo lectura y se modifican cada vez que los datos pasan por el Sumador/Restador. Cuando están a 1 indican sobrepasamiento en los bits de guarda, pero no es valor catastrófico porque dichos bits de guarda (32:39) preservan el valor del dato acumulado.Cuando estos bits están a 1 generan una excepción de error aritmético.
Los bits SA y SB pueden ponerse a uno cada vez que los datos pasan por el módulo Sumador/Restador y permanecerán con ese valor hasta que el usuario los borra ( bits pegajosos). Estos bits indican que se ha sobrepasado el rango máximo permitido por el acumulador. Si la saturación está activa se saturará su valor y en caso contrario estos bits indican un sobrepasamiento catastrófico al destruirse el bit de signo del acumulador.
Las instrucciones MCU no afectaran al valor de estos bits de estado.

- Saturación en la escritura del espacio de datos.

  El dispositivo presenta tres modos de sobrepasamiento y saturación:

  - Saturación del bit 39 del acumulador. Ocurre cuando la unidad lógica de saturación carga en el acumulador el valor positivo 0x7FFFFFFFFF o el negativo 0x8FFFFFFFFF, entonces el bit SA o SB se pondrán a 1. Este modo de saturación es útil para extender el rango dinámico del acumulador.
  - Saturación del bit 31 del acumulador. Ocurre cuando la unidad lógica de saturación carga en el acumulador el valor positivo 0x007FFFFFFF o el negativo 0x008FFFFFFF, entonces el bit SA o SB se pondrán a 1. En este modo los bits de guarda se usan para extender el signo del valor de acumulador.
  - Sobrepasamiento catastrófico del acumulador. Si el bit SATA y/o SATB están a cero no se lleva a cabo ningún tipo de saturación en el acumulador y éste podrá llegar al sobrepasamiento destruyendo en bit de signo.

- Post-escritura del acumulador.

  Las instrucciones MAC y MSC tienen la opción de escribir en versión redondeada del acumulador, que no es el destino de la operación actual, en la memoria de datos. Esto se realiza a través del bus de datos X sobre el espacio de direcciones de los buses X e Y.
  Hay dos modos de direccionamiento para la post-escritura del acumulador:

  - El resultado redondeado se escribe en el registro W13 como fraccionario.
  - El resultado redondeado se escribe en la dirección a la que apunta W13 como fraccionario y después W13 se incrementa.

- **REDONDEO LÓGICO.**
  La unidad de redondeo lógico puede llevar a cabo funciones de redondeo convencionales (parciales) o convergentes (imparciales) y su comportamiento esta determinado por el bit RND del registro CORCON.
  La forma de trabajo de los dos modos es la siguiente:

  - **Convencional:** Si el bit 15 del acumulador es un 1 (ACCL $\geq$ 0x8000) se incrementa el valor de ACCH, que es la parte alta del acumulador (16:31), si este bit es un cero se queda como está el acumulador. El resultado de este algoritmo es que tras una serie de operaciones aleatorias de redondeo el valor sera ligeramente mayor.
  - **Convergente:** Opera de la misma forma que el redondeo convencional, sólo difiere cuando ACC es 0x8000, bit 15 a 1 y el resto a 0, en este caso si el bit 16 del ACC es 1 se incrementa en una unidad el ACCH y en caso contrario (bit 16 a 0) el ACCH se deja como está.Como el estado del bit 16 es aleatorio esto eliminará cualquier redondeo parcial que pudiera acumularse.

  Las instrucciones SAC y SAC.R almacenan las versiones truncadas y redondeadas, respectivamente, del valor del acumulador en la memoria de datos a través del bus de datos X.

- **REGISTRO DE DESPLAZAMIENTO.**
  El registro de desplazamiento del motor DSP es de 40 bits, siendo capaz de desplazar aritméticamente a izquierdas o derechas hasta 16 bits en un mismo ciclo. Este módulo puede ser empleado tanto por instrucciones DSP como MCU a las que se les pasarán resultados de 40 y 16 bits respectivamente.
  El desplazamiento requiere un valor binario con signo que determine tanto el número de bits a desplazar como la dirección en la que se desplazará el operando. Si el valor es positivo se desplazará a la derecha, si es negativo a la izquierda y si es cero no se modifica el operando.
  Las instrucciones del registro desplazamiento para el DSP son:

  - ASR $\Rightarrow$ Desplazamiento aritmético a la derecha de una dirección de memoria de datos.

- LSR ⇒ Desplazamiento lógico a la derecha de una dirección de memoria de datos.
- SL ⇒ Desplazamiento a la izquierda de una dirección de memoria de datos.
- SAC ⇒ Almacenamiento del acumulador DSP con desplazamiento opcional.
- SFTAC ⇒ Desplazamiento del acumulador DSP.

- **SELECCIÓN DEL MODO DE FUNCIONAMIENTO DEL MOTOR DSP.**
  Las múltiples características del motor DSP expuestas en los apartados anteriores pueden seleccionarse a través del Registro de Configuración del Núcleo de la CPU (CORCON):

  - Multiplicación fraccional o entera.
  - Redondeo convencional o convergente.
  - Activación/Desactivación de la saturación automática para ACCA.
  - Activación/Desactivación de la saturación automática para ACCB.
  - Activación/Desactivación de la saturación automática para la escritura en la memoria de datos.
  - Selección del modo de saturación del acumulador.

- **EXCEPCIONES DE MOTOR DSP.**
  El tratamiento a aplicar cuando se genera una excepción provocada por una instrucción DSP se selecciona a través del Registro de Control de Interrupciones (INTCON1), con las siguientes posibilidades:

  - Habilitación del sobrepasamiento en ACCA utilizando OVATE (bit 10).
  - Habilitación del sobrepasamiento en ACCB utilizando OVBTE (bit 9).
  - Habilitación del sobrepasamiento catastrófico del ACCA y/o ACCB utilizando COVTE (bit 8).

  También se provocará una excepción de error aritmético cuando se quiera desplazar un número de bits mayor que el permitido (16) a través de SFTAC.

### 4.2.4. Fundamentos de las instrucciones DSP.

- **INTRODUCCIÓN:**

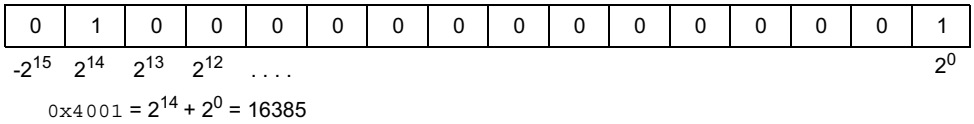
  El repertorio de los procesadores dsPIC consta de 84 instrucciones de las que 19 son instrucciones específicas DSP, que caracterizan como procesadores tipo DSP a los miembros de esta familia. Todas las instrucciones tienen el tamaño de una palabra de la memoria de programa que consta de 24 bits. Excepto las instrucciones CALL, DO y GOTO que ocupan dos palabras al contener sus códigos operandos con muchos bits.

  La mayor parte de las instrucciones se ejecutan en un solo ciclo(33 ns o 30 MIPS).

- **FORMATO DE LOS DATOS DSP:**

  Los dsPIC30F soportan datos de tipo entero y de tipo fraccionario. Los enteros se representan como valores en complemento a 2 y con signo, siendo el bit de más peso el que define al signo. El rango que cubren los números enteros representados con N bits está comprendido entre $-2^{N-1} : 2^{N-1}$

  Ejemplo de representación de datos de tipo entero con N=16:

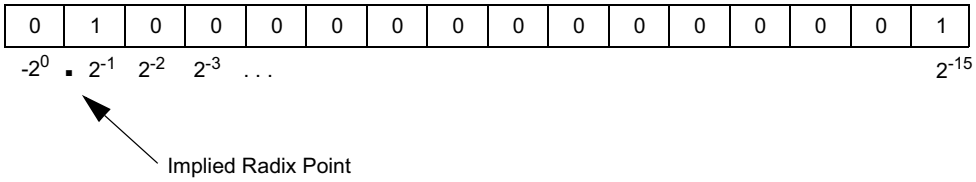



$$0x4001 = 2^{14} + 2^0 = 16384 + 1 = 16385$$

  Así los enteros de 16 bits llegan desde (-32.768) y (32.767).Cuando se trata de 32 bits el rango está comprendido entre(-2.147.483.648)y(2.147.483.647).

  Los números fraccionarios también se representan como valores con signo en complemento a 2, pero tienen el punto o coma fraccionaria(separa la parte entera de la fraccional) implícito detrás del signo que es de más peso.Cuando N=16, el número fraccionario se presenta como de formato 1.15 o Q15, que significa que el primer 1 es el número de bits usados para representar la parte entera y el 15 los bits que representan la parte fraccionaria. Así, para un dato fraccionario de N bits el rango está comprendido entre $-1,0 : (1 - 2^{1-N})$.

  Ejemplo de representación de datos de tipo fraccionario con N=16:




$$0xC002 = -2^{15} + 2^{14} + 2^0 = -32768 + 16384 + 2 = -16382$$

Las instrucciones DSP que realizan operaciones matemáticas y de desplazamiento usan los acumuladores ACCA y ACCB de 40 bits cada uno. Estos acumuladores se dividen en tres registros, uno con los 8 bits de más peso, ACCxU, otro de 16 bits, ACCxH, y otro con los 16 bits de menos peso que se denomina ACCxL como se puede ver:

| ACCxU[39:32] | ACCxH[31:16] | ACCxL[15:0] |
|---|---|---|

Para adaptar el tamaño de los acumuladores de 40 bits al de los buses de datos X e Y de 16 bits, las instrucciones de Carga(LAC)y Almacenamiento(SAC.R) de los acumuladores sólo emplean los 16 bits correspondientes al registro ACCxH.
Cuando se trabaja en el modo de saturación normal, con 31 bits, sólo se utilizan los dos registros de 16 bits de los acumuladores ACCxH:ACCxL, mientras que el bit de signo se extiende por el registro ACCxU.
En el modo de super saturación se emplea todo el rango de los acumuladores. Este modo se activa poniendo el bit ACCSAT(CORCON[4])=1.

- **ASIGNACIÓN DE LOS REGISTROS DE TRABAJO:**
  El procesador de la familia dsPIC30F dispone de un banco de 16 registros de trabajo(W0-W15) cada uno de 16 bits. Los microcontroladores PIC convencionales sólo disponían de un registro de trabajo WREG. Para que exista una compatibilidad con estos se toma por defecto a W0 como WREG. Si en una instrucción no se especifica un registro se toma por defecto este.
  El W0 también se asigna para contener el cociente de una operación de división, mientras que W1 se reserva para soportar el resto de la división.
  Los registros W2 y W3 soportan la palabra de menos y de más peso, respectivamente, de una operación aritmética de multiplicación. Ejemplo:

MUL0X100 [(0x100)xW0 W3:W2]
ANTES WO=0x7705 DESPUÉS W0=0x7705
ANTES W2=0x1235 DESPUÉS W2=0xDEA9

ANTES W3=0x100 DESPUÉS W3=0x00885
ANTES (0X100)=0x1255 DESPUÉS (0X100)=0x125

La instrucción DSP por excelencia es MAC, que se emplea en la resolución de algoritmos DSP habituales. La operación MAC consiste en multiplicar el contenido de dos registros de trabajo, extender el producto a un tamaño de 40 bits y sumar o almacenar el resultado en cualquiera de los dos acumuladores. Los operandos deben residir en los registros W4-W7 indistintamente y cuando se usa un solo registro como único operando, se eleva al cuadrado su valor y se acumula. La instrucción MAC W4xW5, A realiza la siguiente operación ACCA = ACCA + W4xW5.
Además la instrucción MAC puede realizar una prebúsqueda de datos en los espacios X e Y, dependiendo de los registros que se empleen en el direccionamiento indirecto para apuntar a la dirección efectiva del operando. W8 y W9 especifican que la prebúsqueda se realiza en el espacio X, y W10 y W11 en el espacio Y. W12 en este también es posible el direccionamiento indirecto como offset y W13 el destino de MAC con post-escritura. W14 y W15 se utilizan para trabajar con la pila.
W0: Registro por defecto WREG y cociente de división.
W1: Resto de división.
W2: Palabra de menos peso del producto MUL.
W3: Palabra de más peso del producto MUL.
W4: Operando MAC.
W5: Operando MAC.
W6: Operando MAC.
W7: Operando MAC.
W8: Puntero de prebúsqueda dirección (memoria X) en MAC.
W9: Puntero de prebúsqueda dirección (memoria X) en MAC.
W10: Puntero de prebúsqueda dirección (memoria Y) en MAC.
W11: Puntero de prebúsqueda dirección (memoria Y) en MAC.
W12: Offset o desplazamiento de prebúsqueda en MAC.
W13: Destino de MAC con post-escritura.
W14: Puntero de trama de pila.
W15: Puntero de pila.

### 4.2.5. División.

El dsPIC30F soporta los siguientes tipos de divisiones:

- DIVF: 16/16 división de fraccionarles con signo.
- DIV.SD: 32/16 división con signo.
- DIV.UD: 32/16 división sin signo.
- DIV.SW: 16/16 división con signo.
- DIV.UW: 16/16 división sin signo.

El resultado de todas las operaciones de división es almacenado en W0 y el resto en W1. El divisor de 16 bits puede almacenarse en cualquier registro W y el dividendo se almacenara en un registro W cuando es de 16 bits o en dos registros W adyacentes cuando es de 32 bits. Todas las instrucciones de división deben ejecutarse 18 veces con un bucle REPEAT, programado por el usuario, por lo que la instrucción de división tarda 19 ciclos de instrucción en ejecutarse. El flujo de la división es interrumpible como cualquier lazo REPEAT, el usuario será el responsable de guardar los datos en pila en la ISR(Rutina de atención a la interrupción). Los valores intermedios almacenados en los registros W en la operación de división no tienen ningún significado para el usuario, aunque si son importantes para el hardware de división.

### 4.2.6. Flujo de ejecución de instrucciones.

La mayoría de las instrucciones en el dsPIC30F ocupan una palabra y se ejecutan en un ciclo de instrucción. Sin embargo, algunas instrucciones necesitan 2 o 3 ciclos de instrucción para ejecutarse, por lo que hay diferentes tipos de flujo de ejecución de instrucciones en la arquitectura del dsPIC, que se describen a continuación:

- **INSTRUCCIONES DE UNA PALABRA, UN CICLO DE INSTRUCCIÓN:** La mayoría de las instrucciones tardan un ciclo de instrucción en ejecutarse, el flujo de ejecución de estas instrucciones se muestran en la siguiente figura.

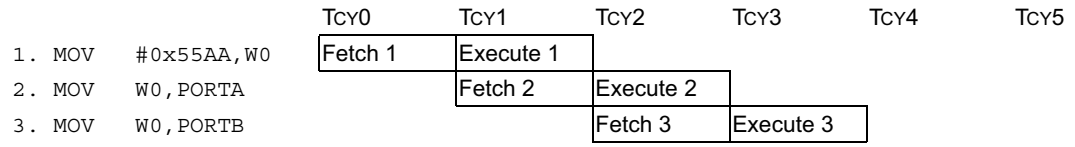



Figura 4.11: Flujo de instrucción: 1 palabra, 1 ciclo.

- **INSTRUCCIONES DE UNA PALABRA, DOS CICLOS DE INSTRUCCIÓN:**

  La única instrucción de este tipo es la MOV.D, carga una doble palabra y se requieren dos ciclos para ejecutar la instrucción, el flujo de ejecución de estas instrucciones se muestran en la siguiente figura.

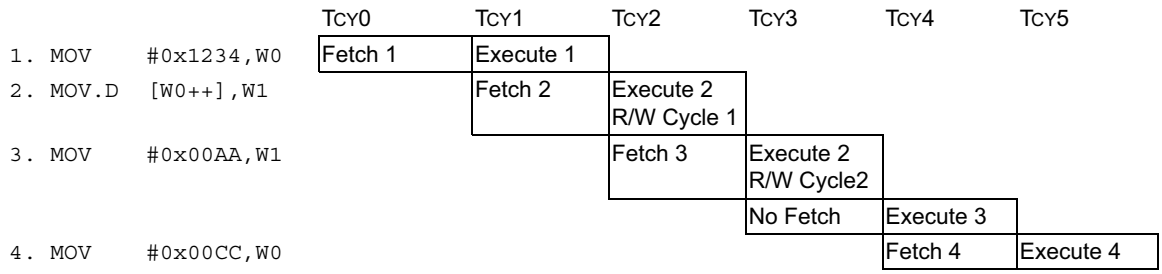



Figura 4.12: Flujo de instrucción: 1 palabra, 2 ciclos.

- **INSTRUCCIONES DE UNA PALABRA, DOS O TRES CICLOS DE INSTRUCCIÓN, CAMBIOS EN EL FLUJO DE EJECUCIÓN:**

  Estas instrucciones incluyen saltos condicionales o incondicionales y llamadas a subrutinas. Son instrucciones que cambian el contenido del PC y necesitan dos ciclos de instrucción para ejecutarse, el flujo de ejecución de estas instrucciones se muestran en la siguiente figura.

  Se necesitan 3 ciclos de instrucción cuando se ejecuta un salto que necesita dos palabras

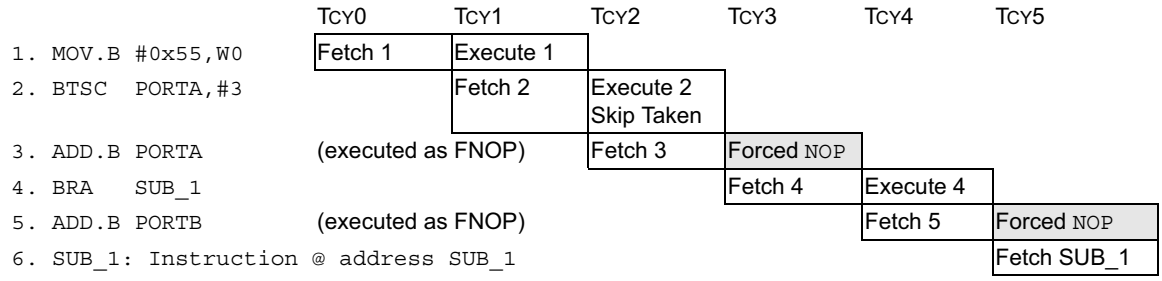



Figura 4.13: Flujo de instrucción: 1 palabra, 2 ciclos, cambios en el flujo del programa.

  para ejecutarse, el flujo de ejecución de estas instrucciones se muestran en la siguiente figura.

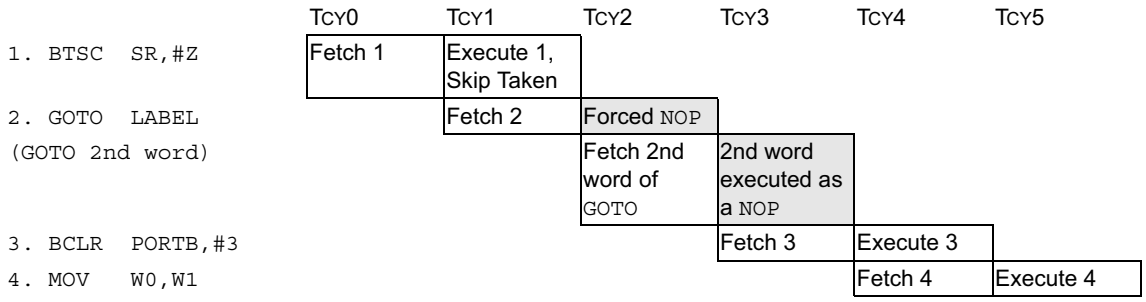



Figura 4.14: Flujo de instrucción: 1 palabra, 3 ciclos, salto de 2 palabras.

- **INSTRUCCIONES DE UNA PALABRA, TRES CICLOS DE INSTRUCCIÓN (RETFIE, RETURN, RETLW):**

  Las instrucciones para retornar de las subrutinas, o de las ISR necesitan tres ciclos de instrucción para ejecutarse, el flujo de ejecución de estas instrucciones se muestran en la siguiente figura:

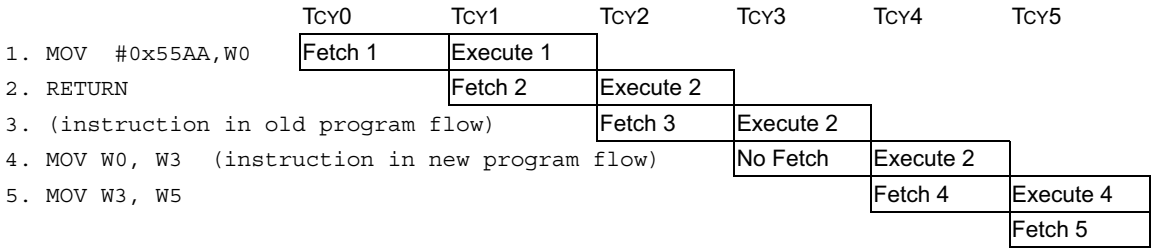



Figura 4.15: Flujo de instrucción: 1 palabra, 3 ciclos (RETFIE, RETURN, RETLW).

- **INSTRUCCIONES DE LECTURA/ESCRITURA EN TABLAS:**

  Hay instrucciones que suspenden el acceso al ciclo de escritura/lectura de la memoria de programa. La instrucción de búsqueda mientras se accede a tablas se salta un ciclo y se ejecuta el ciclo siguiente, tal como se muestra en la figura:

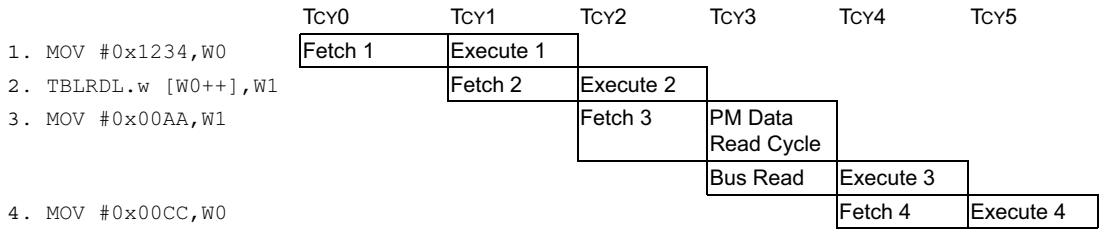



Figura 4.16: Flujo de instrucción operaciones de tabla.

- **INSTRUCCIONES DE DOS PALABRAS, DOS CICLOS DE INSTRUCCIÓN:**

  Hay instrucciones que ocupan dos palabras de memoria, por lo que se necesita acceder dos veces a memoria para buscar la instrucción, en la siguiente figura aparece un flujo de instrucción en el que se refleja este ejemplo:

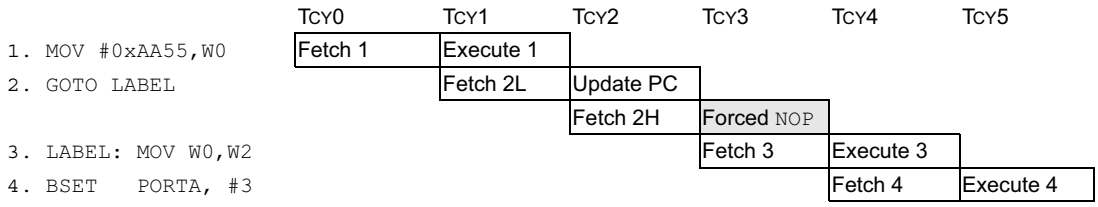



Figura 4.17: Flujo de instrucción: 2 palabras, 2 ciclos.

- **CONFLICTOS EN ACCESO A MEMORIA:**

  Hay instrucciones que su ejecución debe retrasarse debido a dependencias entre operaciones de lectura y escritura en memoria, y se tiene que insertar un ciclo adicional para resolver este conflicto.

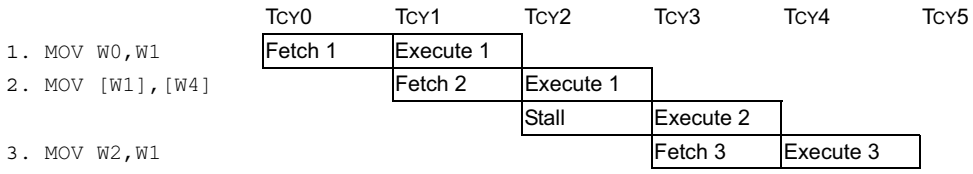



Figura 4.18: Conflictos en acceso a memoria.

### 4.2.7. Construcción de bucles.

En los dsPIC se dispone de instrucciones DO y REPEAT para la construcción de bucles de control. Con la instrucción REPEAT se crea un bucle con una única instrucción y con el DO uno de múltiples instrucciones. Ambas instrucciones usan los bits de control del registro de estado de la CPU SR para modificar el modo de operación de la CPU.

- **BUCLE REPEAT**

  Hace que la siguiente instrucción se repita un número de veces indicado en el registro RCOUNT más uno, leyéndola una única vez de memoria, se puede usar el valor de un registro W para indicar el número de repeticiones ( W+1) o el valor de un literal junto

con la instrucción.

Para hacer un bucle REPEAT podemos:

- REPEAT lit14 ; RCOUNT = lit14
- REPEAT Wn ; RCOUNT = Wn

| | TCY0 | TCY1 | TCY2 | TCY3 | TCY4 | TCY5 |
|---|---|---|---|---|---|---|
| `1.REPEAT #0x2` | Fetch 1 | Execute 1 | | | | |
| `2.MAC W4*W5,A,[W8]+=2,W4` | | Fetch 2 | Execute 2 | | | |
| | | | No Fetch | Execute 2 | | |
| | | | | No Fetch | Execute 2 | |
| `3.BSET PORTA,#3` | | | | | Fetch 3 | Execute 3 |

| | | | | | | |
|---|---|---|---|---|---|---|
| PC (at end of instruction) | PC | PC+2 | PC+2 | PC+2 | PC+4 | PC+6 |
| RCOUNT (at end of instruction) | X | 2 | 1 | 0 | 0 | 0 |
| RA (at end of instruction) | 0 | 1 | 1 | 0 | 0 | 0 |

Figura 4.19: Flujo de ejecución del la instrucción REPEAT.

El registro RCOUNT se inicializa con la instrucción REPEAT, y se decrementa cada vez que se ejecuta la instrucción a repetir. La instrucción de REPEAT pone a uno el bit RA (SR[4]) si el valor de RCOUNT es distinto de cero.

El valor del PC no se incrementara hasta que RCOUNT sea nulo.

Una instrucción REPEAT puede ser interrumpida en cualquier instante y el valor del bit RA, en concreto el byte SRL, se apilará y posteriormente se borrará el valor de este bit para ejecutar la ISR correspondiente, una vez ejecutada la ISR se recupera el valor del SRL y se sigue con la ejecución del bucle. Hay instrucciones que no pueden ponerse después de un REPEAT, como:

- Instrucciones de control del flujo de programa, como saltos, llamadas a subrutinas...
- Instrucciones DO o REPEAT.
- DISI, ULNK, LNK, PWRSAV y RESET.
- MOV.D

- **DO**

Con la instrucción DO se ejecutaran un conjunto de instrucciones encerradas entre el comando DO y el END LOOP un número de veces sin tener que leer las instrucciones todas las veces de memoria, sólo se leen la primera vez que se ejecutan. Se puede usar el

valor de un registro W para indicar el número de repeticiones ( W+1) o el valor de un literal junto con la instrucción. El número de veces que se repite el bucle es DCOUNT + 1. Para hacer un bucle DO podemos:

- DO lit14,LOOP END ; DCOUNT = lit14
- DO Wn,LOOP END ; DCOUNT = Wn[13:0]

Dentro del bucle REPEAT puede haber saltos, el orden de ejecución no tiene por que ser secuencial.
El número de instrucciones que se pueden incluir en un bucle DO está limitado. Los registro DOSTART y DOEND almacenan el valor de la dirección donde empieza y termina el bucle, el bit de estado DA (SR[9]) indica que se está ejecutando un bucle DO. Cuando el PC llega al valor DOEND el registro DCOUNT se decrementa y si este es distinto de cero el PC se carga con DOSTAR.
Pueden anidarse varios bucles DO, los bits DL[2:0] del CORCON indican el nivel del bucle DO que se está ejecutando, cuando se ejecuta un DO dentro de un bucle DO los registros DOCOUNT ,DOSTART y DOEND se guardan en los registros sombra para almacenarse los valores de este nuevo bucle.
Una instrucción DO puede ser interrumpida en cualquier instante y el valor del bit DA se apilará y posteriormente se borrará el valor de este bit para ejecutar la ISR correspondiente, una vez ejecutada la ISR se recupera el valor del SRL y se sigue con la ejecución del bucle. Si en la ISR hay un bucle DO es necesario apilar el valor de los registros DOCOUNT ,DOSTART y DOEND antes de almacenar los nuevos valores, antes de salir de la ISR estos registros deben ser sacados de la pila.
Los bucles DO tienen las siguientes características:

- Se tiene que elegir la última instrucción del bucle.
- Se tiene que elegir la longitud del bucle.
- La dirección de la última instrucción está en DOEND.

Todos los bucles DO tienen que tener al menos dos instrucciones. El registro DOEND no se puede leer por el usuario en la instrucción seguida por la instrucción DO. Las dos últimas instrucciones del bucle no pueden modificar:

- Los bits de prioridad de la CPU IPL (SR [7:5]).
- Bits de habilitación del las interrupciones de los periféricos contenidos en los registro IEC0, IEC1 y IEC2.

- Bits de prioridad del las interrupciones de los periféricos contenidos en los registro IPC0 - IPC11.

Para garantizar el correcto funcionamiento del bucle deben respetarse estás restricciones.

### 4.2.8. Dependencias de los registros de dirección.

Los dsPIC soportan la lectura y la escritura de los operandos y resultados de las instrucciones MCU. La dirección efectiva (EA) calculada por la AGU, y la escritura/lectura en memoria cada una tardará un ciclo de instrucción en ejecutarse. Esto causa un overlap en las operaciones de escritura y lectura, las operaciones de lectura después de escritura (RAW) solo pueden ejecutarse después de instrucciones BOUNDARIES. RAW se detectan por la CPU. Si un registro W se

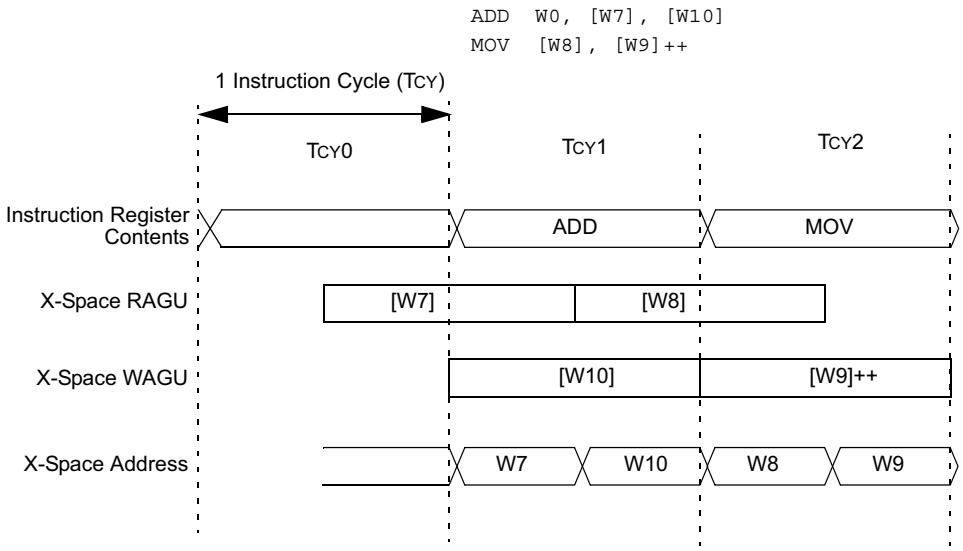



Figura 4.20: Diagrama de acceso al espacio de datos.

usa en como destino en una operación de escritura y el registro W va a ser leído en la siguiente instrucción buscada, se tienen que aplicar las siguientes reglas:

- Si la instrucción actual no modifica el contenido del registro Wn no se parará.
- Si la dirección del dato leído no usa Wn tampoco se parará su ejecución.

En cada ciclo de instrucción el dsPIC comprobará las dependencias RAW. Si no se cumplen las condiciones descritas la CPU añadirá un ciclo extra antes de buscar la instrucción siguiente. En la tabla se muestra las dependencias RAW: Si se detecta una dependencia de datos durante una instrucción RAW el dsPIC realiza una parada de instrucción, durante la cual realizará los siguientes eventos:

| Destination Addressing Mode using Wn | Source Addressing Mode using Wn | Status | Examples (Wn = W2) |
|---|---|---|---|
| Direct | Direct | Allowed | `ADD.w W0, W1, W2`<br>`MOV.w W2, W3` |
| Direct | Indirect | Stall | `ADD.w W0, W1, W2`<br>`MOV.w [W2], W3` |
| Direct | Indirect with modification | Stall | `ADD.w W0, W1, W2`<br>`MOV.w [W2++], W3` |
| Indirect | Direct | Allowed | `ADD.w W0, W1, [W2]`<br>`MOV.w W2, W3` |
| Indirect | Indirect | Allowed | `ADD.w W0, W1, [W2]`<br>`MOV.w [W2], W3` |
| Indirect | Indirect with modification | Allowed | `ADD.w W0, W1, [W2]`<br>`MOV.w [W2++], W3` |
| Indirect with modification | Direct | Allowed | `ADD.w W0, W1, [W2++]`<br>`MOV.w W2, W3` |
| Indirect | Indirect | Stall | `ADD.w W0, W1, [W2]`<br>`MOV.w [W2], W3`<br>`; W2=0x0004 (mapped W2)` |
| Indirect | Indirect with modification | Stall | `ADD.w W0, W1, [W2]`<br>`MOV.w [W2++], W3`<br>`; W2=0x0004 (mapped W2)` |
| Indirect with modification | Indirect | Stall | `ADD.w W0, W1, [W2++]`<br>`MOV.w [W2], W3` |
| Indirect with modification | Indirect with modification | Stall | `ADD.w W0, W1, [W2++]`<br>`MOV.w [W2++], W3` |

Figura 4.21: Dependencias RAW.

- La instrucción en ejecución, de la anterior instrucción, termina su ejecución.
- El espacio de datos no se direccionará hasta después de la parada.
- El incremento del PC se pospone hasta después de la parada.
- La búsqueda de futuras instrucciones también se pospone.

### 4.2.9. Mapa de Registros.

En las siguientes tablas se muestra el mapa de registros, la posición que ocupan en memoria y el valor que toman sus bits después del reset. En las posiciones que pone una x significa que no están inicializados.

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 0000 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 of 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 of ACCB<39> | | | | | | | | ACCBU | | | | | | | | 0000 0000 0000 0000 |
| PCL | 002E | PCL | | | | | | | | | | | | | | | 0 | 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 | | | | | | | | | | | | | | | | xxxx xxxx xxxx xxxx |
| DCOUNT | 0038 | DCOUNT | | | | | | | | | | | | | | | | xxxx xxxx xxxx xxxx |
| DOSTARTL | 003A | DOSTARTL | | | | | | | | | | | | | | | 0 | xxxx xxxx xxxx xxx0 |
| DOSTARTH | 003C | — | — | — | — | — | — | — | — | — | — | DOSTARTH | | | | | | 0000 0000 00xx xxxx |
| DOENDL | 003E | DOENDL | | | | | | | | | | | | | | | 0 | xxxx xxxx xxxx xxx0 |
| DOENDH | 0040 | — | — | — | — | — | — | — | — | — | — | DOENDH | | | | | | 0000 0000 00xx xxxx |
| 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 | DL<1:0> | | 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 | XMODSRT<15:0> | | | | | | | | | | | | | | | 0 | xxxx xxxx xxxx xxx0 |

Figura 4.22: Tabla con los registros del modelo del programador, la dirección que ocupan, la asignación de sus bits y el valor que toman después de un RESET.

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| XMODEND | 004A | XMODEND<15:0> | | | | | | | | | | | | | | | 1 | xxxx xxxx xxxx xxx1 |
| YMODSRT | 004C | YMODSRT<15:0> | | | | | | | | | | | | | | | 0 | xxxx xxxx xxxx xxx0 |
| YMODEND | 004E | YMODEND<15:0> | | | | | | | | | | | | | | | 1 | xxxx xxxx xxxx xxx1 |
| XBREV | 0050 | BREN | XBREV<14:0> | | | | | | | | | | | | | | | xxxx xxxx xxxx xxxx |
| DISICNT | 0052 | — | — | DISICNT<13:0> | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| Reserved | 0054 - 007E | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |

Figura 4.23: Tabla con los registros del modelo del programador, la dirección que ocupan, la asignación de sus bits y el valor que toman después de un RESET.

## 4.3. La memoria de datos.

### 4.3.1. Estructura de la memoria de datos.

El tamaño de los datos que manejan los dsPIC30F es de 16 bits. Esta memoria se divide en dos espacios X e Y. Las instrucciones tipo DSP pueden acceder independientemente a los dos espacios, mientras que las instrucciones convencionales contemplan un solo espacio de datos lineal de 64KB o 32K palabras que comienzan en las direcciones pares. Para acceder a los espacios de datos se emplean Unidades Generadoras de Direcciones (AGU) y buses de datos independientes.

Tenemos una zona de RAM de DATOS X y la RAM de DATOS Y. Cada zona tiene sus propios buses para su direccionamiento y para la transferencia de datos. La capacidad de estos espacios depende de cada modelo específico de dsPIC30F.

Las direcciones inferiores del espacio de memoria están destinadas a contener registros de

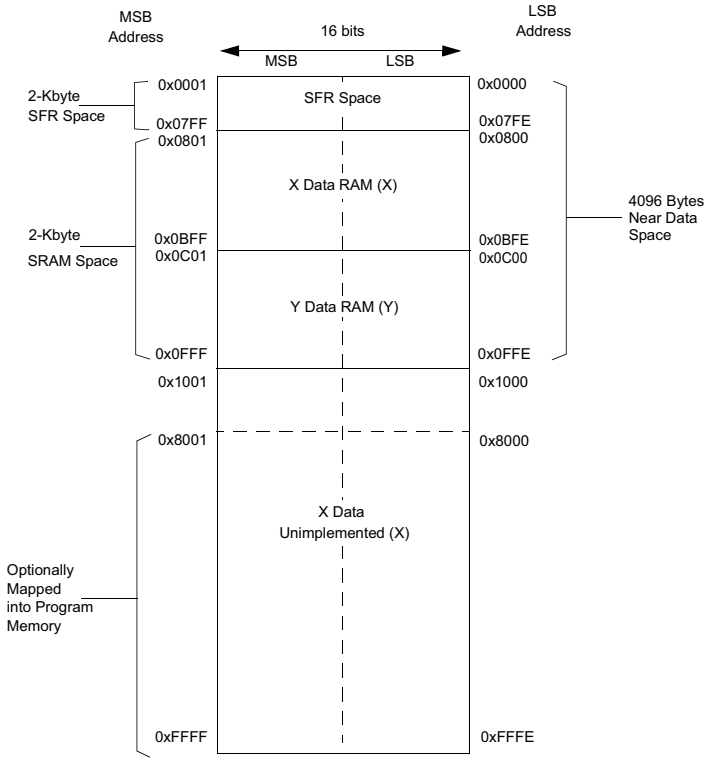



Figura 4.24: Mapa de espacio de la memoria datos para los dispositivos dsPIC30F.

control y de estado específicos (SFR), que sirven para gobernar el comportamiento del dsPIC, así como el de los recursos que integran el chip.

La RAM de datos comienza realmente en la dirección 0x0800 y se divide en dos bloques: Datos X y Datos Y. Para operaciones de escritura de datos los espacios X e Y siempre son accedidos

como si se tratase de un único espacio lineal. Por el contrario, para operaciones de lectura de datos los espacios X e Y pueden ser accedidos independientemente por las instrucciones DSP. Si se trata de lectura de datos de instrucciones normales, los espacios X e Y conforman un único espacio de datos lineal.

Como se ve en la figura posterior las instrucciones MCU de lectura y escritura y las DSP de escritura, consideran a todo el espacio de la memoria de datos como espacio X, incluyendo en él el espacio Y. El espacio Y solo puede ser accedido directamente con las instrucciones DSP y admite el direccionamiento modular. Las instrucciones convencionales acceden al espacio Y mediante el espacio X.

La zona inicial de hasta un máximo de 8KB(0x0000-0x1FFF) de la memoria de datos se denomina, Memoria de Datos Cercana, y a ella se pueden acceder con los 13 bits del campo de la dirección absoluta con todas las instrucciones del tipo de registros de archivo. Según el dispositivo que se trate la capacidad que tiene implementada la memoria cercana varía.

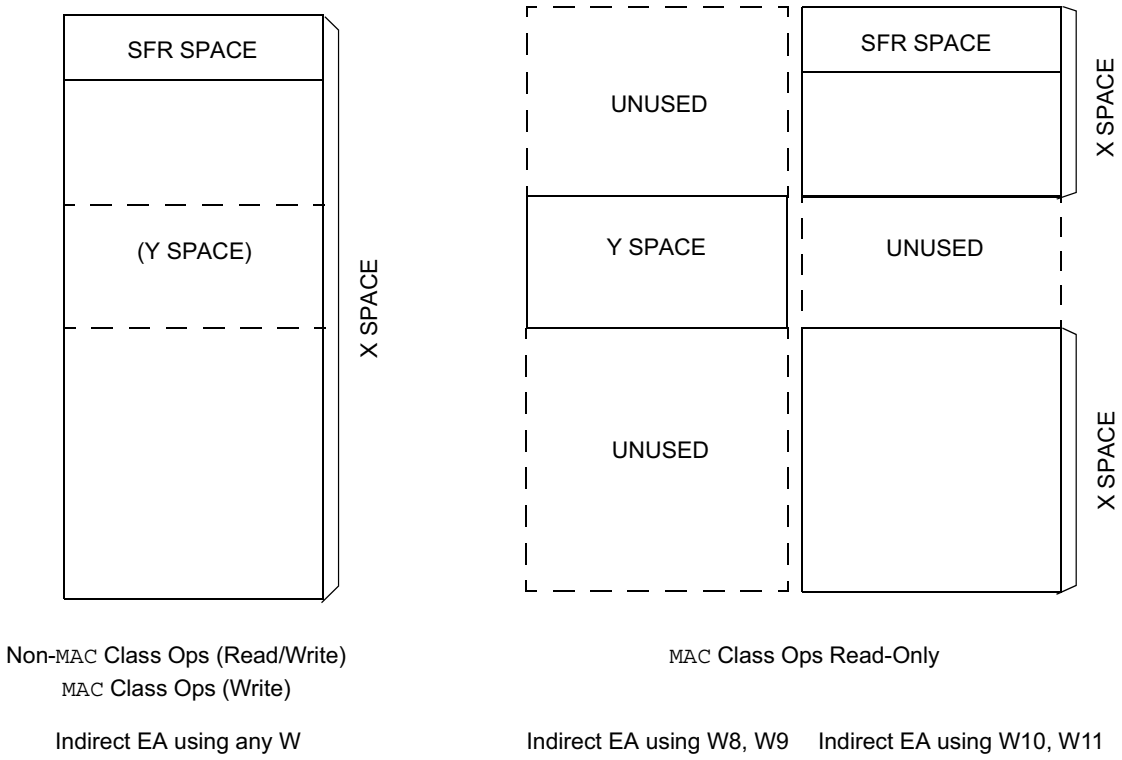



Figura 4.25: Distintas posibilidades de acceso al espacio de la memoria de datos según se trate de instrucciones DSP de lectura de dos operandos y las restantes.

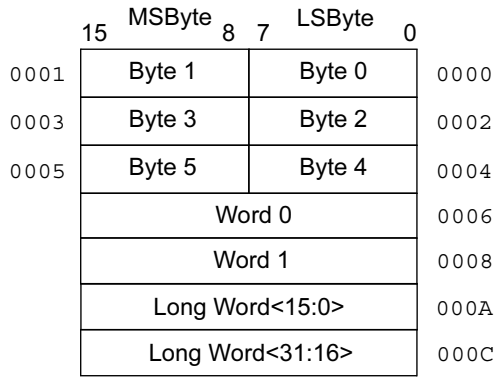



Figura 4.26: Ejemplo de datos alineados.

### 4.3.2. Unidades de generación de direcciones AGU.

Como se comentó anteriormente los dsPIC disponen de una Unidad de Generación de direcciones X y otra Y, llamadas AGU X y AGU Y, las cuales generan direcciones efectivas (EA) que abarcan el rango de 64KB de la memoria de datos.
La AGU X la emplean todas las instrucciones y admite todos los modos de direccionamiento. Consta de una AGU de lectura (RAGU X) y otra de escritura (WAGU X) que funcionan de forma independiente. LA AGU Y soporta la lectura de datos en el espacio de memoria Y. Nunca se utiliza para la escritura de datos.

### 4.3.3. El alineamiento de los datos.

Se conoce con alineamiento de datos en la memoria la condición que debe de cumplir la dirección de los mismos de ser múltiplo del número de bytes que compone su tamaño.
Las instrucciones de los dsPIC30F trabajan con datos de tamaño palabra de 16 bits (2 bytes). En las operaciones con palabras, el bit de menos peso de la dirección del espacio de la memoria de datos se ignora. Los datos se alinean en formato little-endian en el cual el byte de menos peso ocupa la dirección par, haciendo valer 0 al bit de menos peso de esa dirección (LSB=0), mientras que el byte de más peso de la palabra ocupa la dirección impar. Cuando el dato es de 8 bits de menos peso del bus de datos. También el direccionamiento con autoincremento o autodecremento suma o resta 2 unidades cuando se maneja datos de tamaño palabra.

### 4.3.4. Direccionamiento modular(circular).

El objetivo es eliminar la necesidad de software para llevar a cabo las comprobaciones de los límites de las direcciones de datos cuando se ejecutan bucles iterativos como ocurre en muchos algoritmos DSP.
Cualquier registro W, excepto W15, puede ser seleccionado como el puntero del buffer circular. El hardware para el direccionamiento modular realiza comprobaciones del límite en las direcciones contenidas en el registro W seleccionado y ajusta automáticamente el valor del puntero al límite del buffer cuando se requiere.
El direccionamiento modular de los dsPIC30F puede trabajar tanto en el espacio de datos como en de el programa puesto que el mecanismo del puntero es el mismo para ambos. Un buffer circular puede estar soportado en el espacio de datos X y en el espacio de datos Y. A medida que se van leyendo los datos en un espacio, se van escribiendo en el otro.
El funcionamiento de este direccionamiento consiste en ir recorriendo una a una las direcciones, en sentido ascendente o descendente, donde se encuentran los datos hasta llegar a la dirección límite del buffer, momento en el cual se vuelve al principio de este para recorrerlo de nuevo y así sucesivamente. Se usan dos punteros: el puntero X que se encarga de escribir, y el puntero Y que lee los datos.
La longitud del buffer de datos circular puede tener un tamaño de 32K palabras. El buffer modular soporta datos de tamaño byte o palabra. Sin embargo, el módulo lógico sólo realiza comprobaciones de límite de direcciones de tamaño de palabra, por lo que la longitud en bytes del buffer modular ha de ser par. Además, los buffers modulares de tamaño no pueden ser implementados utilizando la AGU Y porque el acceso a bytes no está soportado en el bus de datos de la memoria Y.

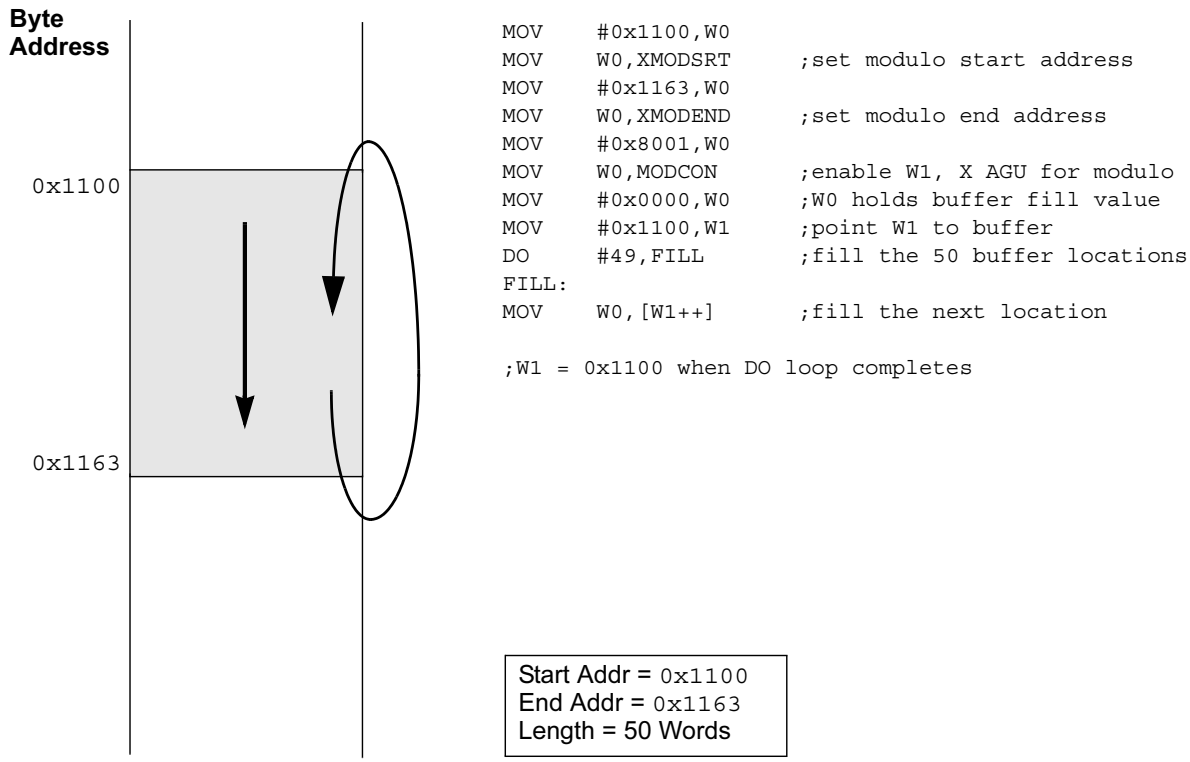



Figura 4.27: Ejemplo de la operación de direccionamiento circular del buffer con incremento.

- **Selección de la dirección modular de comienzo y fin.**

Cuatro registros de direcciones están disponibles para especificar las direcciones de comienzo y fin del buffer modular:

1. XMODSTR: Registro de la dirección modular de comienzo AGU X.
2. XMODEND: Registro de la dirección modular de fin AGU X.
3. YMODSTR: Registro de la dirección modular de comienzo AGU Y.
4. YMODEND: Registro de la dirección modular de fin AGU Y.

La dirección de comienzo del buffer modular debe estar localizada en el byte par de la dirección del límite. El LSB de los registros XMODSTR y YMODSTR está ajustado a '0' para asegurar una correcta dirección modular de comienzo. La dirección final de un buffer modular debe estar localizada en un byte impar de la dirección del límite. El LSB de los registros XMODEND y YMODSTR está ajustado a 1 para asegurar una correcta dirección modular de fin.

| Dirección final = Dirección inicio + Longitud del buffer - 1 |
|---|
| Dirección inicio = Dirección final - Longitud del buffer + 1 |

### 4.3.5. Direccionamiento por inversión del acarreo bit-reverse.

El direccionamiento por inversión del acarreo o bit-reverse simplifica la reordenación de algunos datos para los algoritmos usados en la FFT. Es soportado solamente por WAGU X y consiste en crear una imagen o espejo del puntero a una dirección construida intercambiando la posición de los bits alrededor del punto central del valor binario,como se muestra en la figura. Un ejemplo de una secuencia de inversión del acarreo para el campo de dirección de 4 bits se indica en la tabla posterior.

El direccionamiento por inversión del acarreo es soportado únicamente por WAGU X y es

| Normal Address | | | | | Bit-Reversed Address | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A3 | A2 | A1 | A0 | Decimal | A3 | A2 | A1 | A0 | Decimal |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 8 |
| 0 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 0 | 4 |
| 0 | 0 | 1 | 1 | 3 | 1 | 1 | 0 | 0 | 12 |
| 0 | 1 | 0 | 0 | 4 | 0 | 0 | 1 | 0 | 2 |
| 0 | 1 | 0 | 1 | 5 | 1 | 0 | 1 | 0 | 10 |
| 0 | 1 | 1 | 0 | 6 | 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 | 1 | 1 | 1 | 0 | 14 |
| 1 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 9 | 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | 10 | 0 | 1 | 0 | 1 | 5 |
| 1 | 0 | 1 | 1 | 11 | 1 | 1 | 0 | 1 | 13 |
| 1 | 1 | 0 | 0 | 12 | 0 | 0 | 1 | 1 | 3 |
| 1 | 1 | 0 | 1 | 13 | 1 | 0 | 1 | 1 | 11 |
| 1 | 1 | 1 | 0 | 14 | 0 | 1 | 1 | 1 | 7 |
| 1 | 1 | 1 | 1 | 15 | 1 | 1 | 1 | 1 | 15 |

Figura 4.28: Secuencia de direccionamiento por inversión del acarreo.

controlado por los registros de funciones especiales MODCON y XBREV.

Cuando está habilitado,el hardware del direccionamiento por inversión del acarreo generará direcciones solamente cuando el registro indirecto con modos de direccionamiento con Pre o Post incremento son usados ([Wn++],[++Wn]). Además, las direcciones de inversión del acarreo son generadas sólo por instrucciones modo palabra. No funcionará para otros modos de direccionamiento o instrucciones modo byte (serán generadas direcciones normales).

El direccionamiento modular y el direccionamiento por inversión del acarreo pueden ser habilitados simultáneamente usando el mismo registro W, pero la operación de direccionamiento por inversión del acarreo se adelantará para escribir datos cuando se habilite. Por ejemplo, las siguientes condiciones de instalación asignarán el mismo registro W para el direccionamiento

modular y por inversión del acarreo:

- El direccionamiento modular X es habilitado (XMODEN=1).
- El direccionamiento por inversión del acarreo es habilitado (BREN=1).
- W1 asignado al direccionamiento modular o circular(XWM[3:0]=0001).
- W1 asignado al direccionamiento por inversión del acarreo(BWM[3:0]=0001).

Para la lectura de datos que usa W1 como puntero, ocurrirá una comprobación de los límites de la dirección modular. Para la escritura de datos usando W1 como puntero destino, el hardware de inversión de acarreo corregirá W1 para reordenar los datos.
Si el direccionamiento de los datos por inversión del acarreo ya ha sido habilitado activando el bit BREN(XBREV[15]), entonces una escritura del registro XBREV no debe ser seguida por una operación de lectura indirecta usando el registro W, designado como puntero de dirección de inversión del acarreo.
El valor cargado en el registro XBREV es una constante que define indirectamente el tamaño del buffer de datos de inversión del acarreo. Los valores del modificador XB usados como buffers de inversión del acarreo comunes son presentados en la siguiente tabla. Sólo los valores del modificador mostrados en esta tabla producirán secuencias de direcciones válidas.

| Buffer Size (Words) | XB<14:0> Bit-Reversed Address Modifier Value* |
|---|---|
| 32768 | 0x4000 |
| 16384 | 0x2000 |
| 8192 | 0x1000 |
| 4096 | 0x0800 |
| 2048 | 0x0400 |
| 1024 | 0x0200 |
| 512 | 0x0100 |
| 256 | 0x0080 |
| 128 | 0x0040 |
| 64 | 0x0020 |
| 32 | 0x0010 |
| 16 | 0x0008 |
| 8 | 0x0004 |
| 4 | 0x0002 |
| 2 | 0x0001 |

*Modifier values for buffer sizes greater than 1024 words will exceed the available data memory on the dsPIC30F4011/4012 devices.

Figura 4.29: Valores del modificador para el direccionamiento por inversión del acarreo.

### 4.3.6. Descripción de registros de control.

Los siguientes registros son los usados para el direccionamiento modular y el de inversión del acarreo:

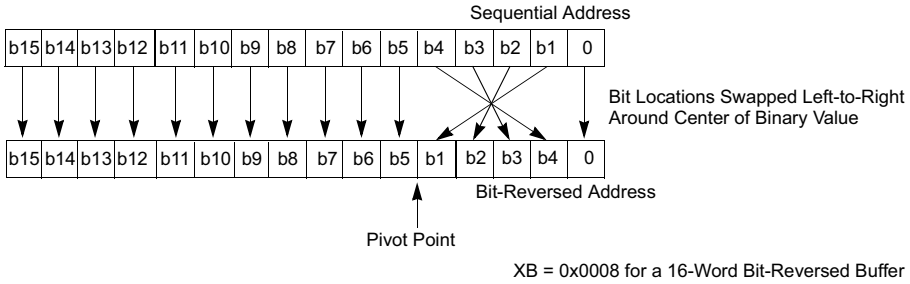



Figura 4.30: Modificación de la dirección por inversión de acarreo para un buffer de 16 palabras.

- MODCON: Registro de control del direccionamiento modular.

| **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 |

Figura 4.31: Registro MODCON.

- XMODEN: Bit de habilitación de direccionamiento modular WAGU X y RAGU X. Se activa con 1.
- YMODEN: Bit de habilitación direccionamiento modular AGU Y.
- BWM[3:0]: Bits de direccionamiento de bit invertido (X WAGU).
  1111 = Direccionamiento de bit invertido deshabilitado. 1110 = W14 seleccionado para el direccionamiento de bit invertido.
  1101 = W13 seleccionado para el direccionamiento de bit invertido...
- YWM[3:0]: Bits de direccionamiento modular (Y WAGU).
  1111 = Direccionamiento modular deshabilitado. 1110 = W10 seleccionado para el direccionamiento modular.
  1101 = W11 seleccionado para el direccionamiento modular...
  El resto de las combinaciones están reservadas y no podrán ser usadas.
- XWM[3:0]: Bits de direccionamiento modular (X WAGU).
  1111 = Direccionamiento modular deshabilitado. 1110 = W14 seleccionado para el

direccionamiento modular.

1101 = W13 seleccionado para el direccionamiento modular...

- XMODSTR: Registro de la dirección modular de comienzo AGU X.

| 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 |
| XS<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-0 |
| XS<7:1> | | | | | | | 0 |
| bit 7 | | | | | | | bit 0 |

Figura 4.32: Registro XMODSTR.

- XMODEND: Registro de la dirección modular de fin AGU X.

| 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 |
| XE<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-1 |
| XE<7:1> | | | | | | | 1 |
| bit 7 | | | | | | | bit 0 |

Figura 4.33: Registro XMODEND.

- YMODSTR: Registro de la dirección modular de comienzo AGU Y.
- YMODEND: Registro de la dirección modular de fin AGU Y.

| 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 |
| YS<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-0 |
| YS<7:1> | | | | | | | 0 |
| bit 7 | | | | | | | bit 0 |

Figura 4.34: Registro YMODSTR.

| 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 |
| YE<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-1 |
| YE<7:1> | | | | | | | 1 |
| bit 7 | | | | | | | bit 0 |

Figura 4.35: Registro YMODEND.

- XBREV: Registro de control de direccionamiento por inversión del acarreo AGU X.

| 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 |

Figura 4.36: Registro XBREV.

- BREN: Bit de habilitación (X AGU) de direccionamiento de bit invertido. Con 1 se habilita.
- XB[14:0]:Bits del modificador de bit invertido.
  0x4000 = buffer de 32768 palabras.
  0x2000 = buffer de 16384palabras.

0x1000 = buffer de 8192 palabras.

0x0800 = buffer de 4096 palabras.

0x0400 = buffer de 2048 palabras.

0x0200 = buffer de 1024 palabras.

0x0100 = buffer de 512 palabras.

0x0080 = buffer de 256 palabras.

0x0040 = buffer de 128 palabras.

0x0020 = buffer de 64 palabras.

0x0010 = buffer de 32 palabras.

0x0008 = buffer de 16 palabras.

0x0004 = buffer de 8 palabras.

0x0002 = buffer de 4 palabras.

0x0001 = buffer de 2 palabras.

## 4.4. La memoria de programa.

### 4.4.1. Mapa de direcciones.

El espacio que los dispositivos dsPIC30F tienen reservado para la memoria de programa alcanza un máximo de 4M posiciones de 24 bits cada una.
El mapa de memoria de programa está dividido en dos espacios:

- Espacio de programa de usuario. Contiene el vector RESET, la tabla de los vectores de interrupción, la memoria de programa y la memoria EEPOROM de datos.
- Espacio de configuración. Contiene los bits de configuración no volátiles que selecciona el comportamiento de los diversos recursos y las posiciones ID del dispositivo.

En la figura vemos un ejemplo de espacio de memoria de programa típico.
Existen tres métodos de acceso al espacio de la memoria de programa:

- A través del Contador de Programa (PC) de 23 bits.
- Mediante las instrucciones de lectura y escritura de Tabla (TBLRD y TBLWT).
- Mapeando un segmento de 32 Kbytes de la memoria de programa en el espacio de direcciones de la memoria de datos.

### 4.4.2. El contador de programa.

El Contador de Programa (PC) dispone de 23 bits y se incrementa de 2 en 2 manteniendo su bit menos significativo a cero para mantener el alineamiento de la memoria.
El bit de menos peso del PC se utiliza para seleccionar el octeto cuando se accede a la memoria de programa desde el espacio de datos usando la Visibilidad del Espacio de Programa o las instrucciones de Tabla.

### 4.4.3. Acceso a los datos desde la memoria de programa

Existen dos métodos para transferir datos entre la memoria de programa y la de datos, uno usando las instrucciones de tabla y otro el mapeo de páginas de programas en la memoria de datos.

- Instrucciones de Tabla. Permiten mover datos de tamaño byte o palabra entre el espacio de programas y el de datos. Las instrucciones para la lectura leen desde la memoria de programa hacia la de datos.
  Hay cuatro instrucciones de Tabla:

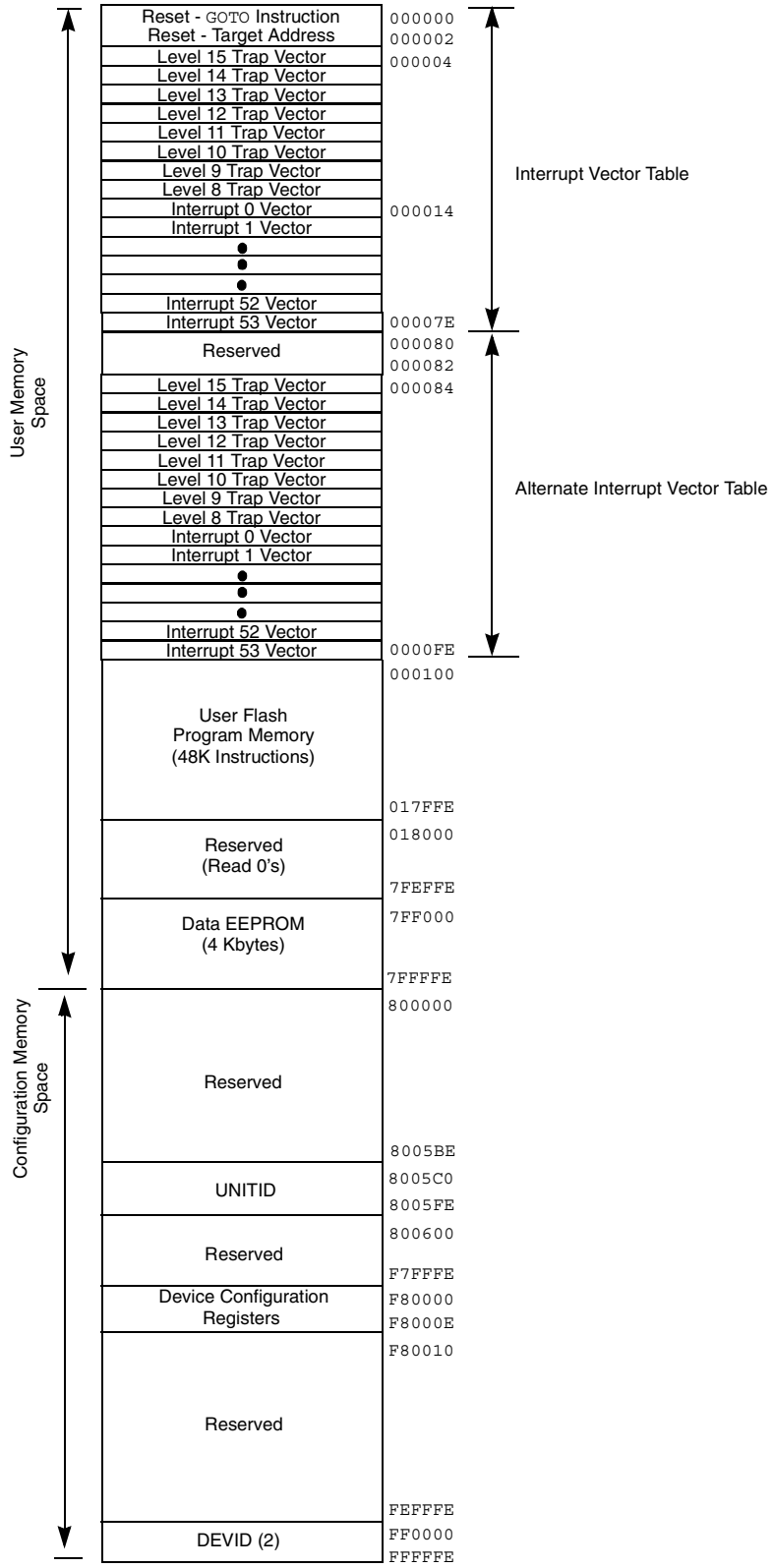



Figura 4.37: Mapa del espacio de memoria de programa.

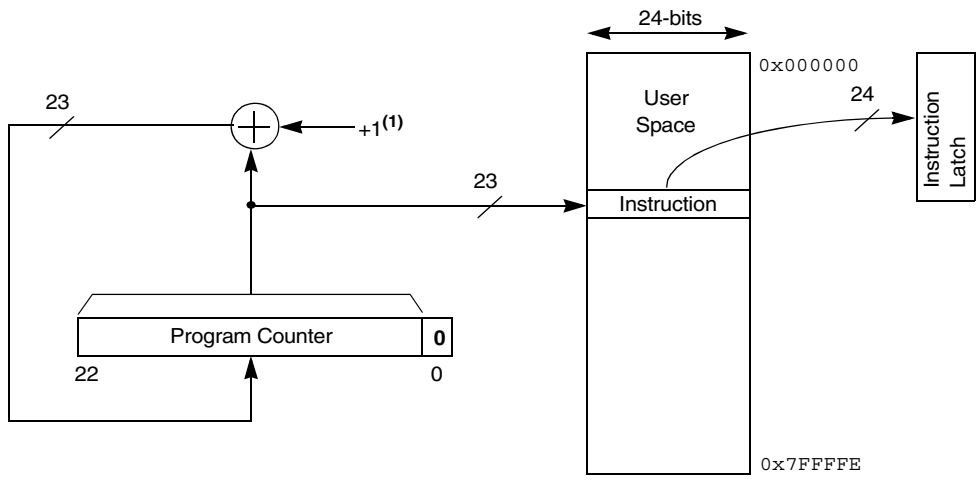



Figura 4.38: Se mantiene el bit de menos peso del PC a 0 sumando 2, que es lo mismo que sumar uno a los 22 bits de más peso.

- **TBLRDL:** Lectura de la palabra baja de Tabla.
- **TBLWTL:** Escritura de la palabra baja de Tabla.
- **TBLRDH:** Lectura de la palabra alta de Tabla.
- **TBLWTH:** Escritura de la palabra alta de Tabla.

Con las instrucciones de Tabla la memoria de programa se comporta como un espacio en el que cada posición ocupa dos palabras de 16 bits.
Para entender el concepto de parte alta y baja de la Tabla se debe tener en cuanta que la memoria de programa por cada dos direcciones del PC se guarda una instrucción de 24 bits. Como cada dirección hace referencia a una palabra de 16 bits, la dirección alta de la Tabla correspondiente a los 16 bits de más peso y la dirección baja contiene los 16 bits de menos peso. Los 8 bits de más peso de la dirección alta no son válidos y siempre se leen cero. Cada dirección del espacio de programa de 24 bits ocupa los 16 bits de la palabra, sobrando el byte de menos peso.

- Generación de direcciones de Tabla. Las instrucciones de Tabla concatenan los 16 bits de un registro W con los 8 del Registro de Página (TBLPAG), de esta forma se pueden manejar 32 Kpalabras como página de tabla.

- Acceso a las palabras de memoria de programa. Con las instrucciones TBLRDL y TBLWTL se acceden a las palabras bajas de la memoria de programa. Cuando accedemos a palabras el bit de menos peso del la dirección, del registro W, está siempre a cero, pero cuando accedemos a un byte este bit determina cual de los dos bytes de la palabra es accedido,

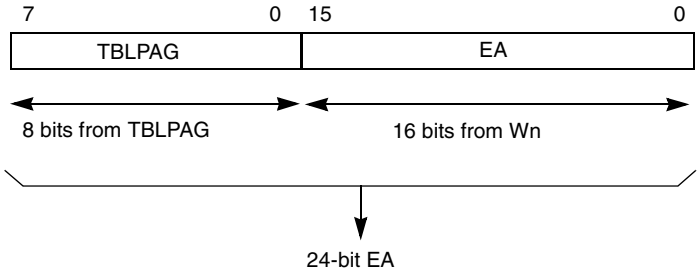



Figura 4.39: Dirección efectiva de 24 bits.

si está a 1 es el byte de más peso y sino el de menos peso. Las instrucciones que acceden

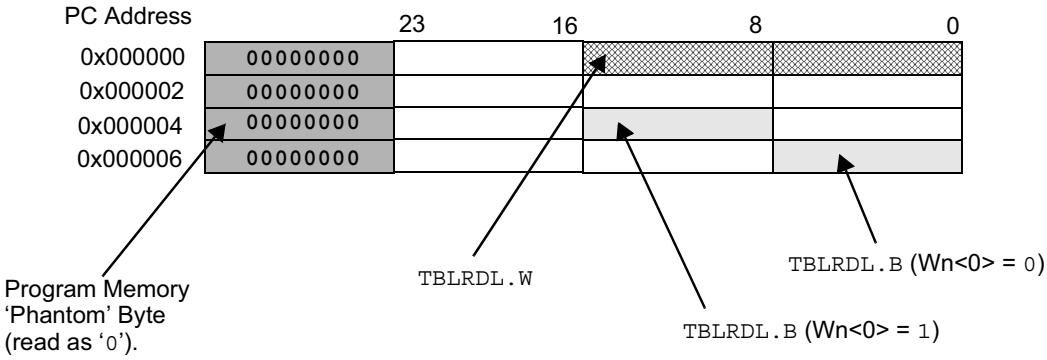



Figura 4.40: Accesos a bytes o a palabras.

a la palabra alta son TBLRDH para acceder a toda la palabra y TBLWTF para acceder al byte de menos peso W[0]=0 o al de más peso cuando W[0]=1.

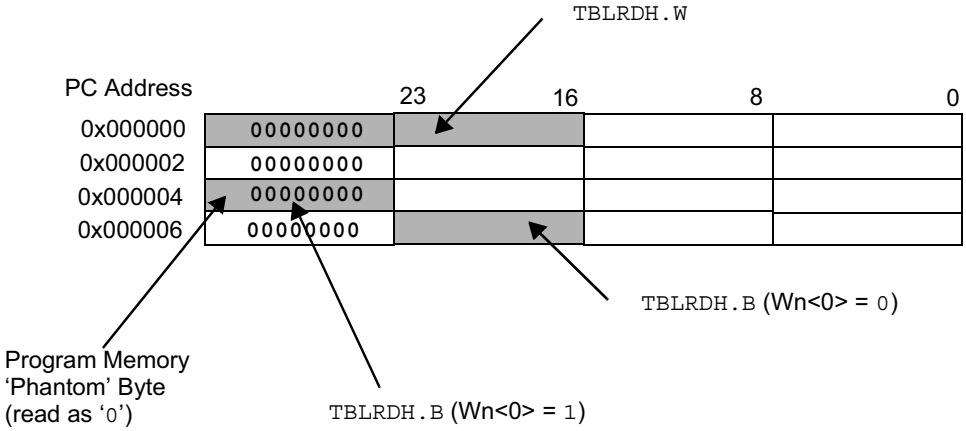



Figura 4.41: Acceso a bytes o a palabras altas.

### 4.4.4. Visibilidad del espacio de programa para el espacio de datos.

Para activar la visibilidad del espacio de programas (PSV) se pone el bit CORCON[PSV] = 1, a partir de este momento cualquier dirección de la mitad superior del mapa de memoria se podrá mapear en una dirección de la memoria de programa. Con este procedimiento no se puede acceder al byte más significativo de la dirección de memoria, sólo las instrucciones de tabla pueden acceder a este byte. La dirección en este modo se obtiene con los 15 bit menos

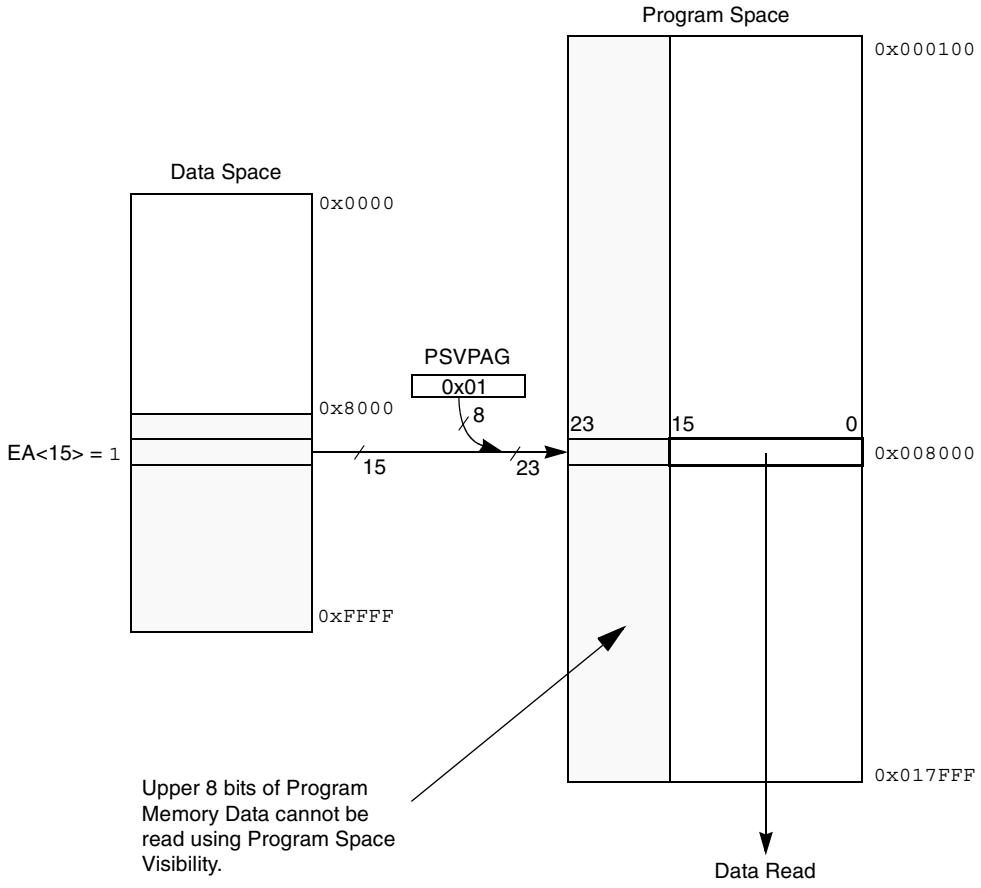



Figura 4.42: Acceso a la memoria con PSV.

significativos de un registro W, el bit de más peso indica si es un acceso PSV (W[15]=1) o un acceso normal cuando esta a cero, los 8 bits restantes para obtener la dirección son el contenido de PSVPAG. Las instrucciones que usan PSV necesitan dos ciclos para completar su ejecución excepto las MAC con datos de prebúsqueda y las MOV.

Las instrucciones que usan PSV con el bucle REPEAT eliminan los ciclos de instrucción extra.

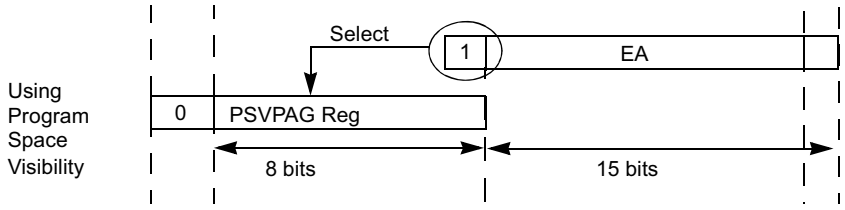



Figura 4.43: Dirección efectiva en modo PSV.

### 4.4.5. Escritura de la memoria de programa.

Los dsPIC30F disponen de una memora FLASH que contienen el código ejecutable, los métodos para grabar la memoria:

- Autoprogramación en tiempo de ejecución (RTSP).
- Programación serie en circuito.

# 4.5. Puertos de Entrada Salida.

## 4.5.1. Introducción.

Todos los puertos, excepto los de $V_{DD}$ ,$V_{SS}$, MCLR y OSC1/CLKI, son compartidos por los periféricos y los pines de entrada/salida de propósito general. Los puertos de entrada/salida de propósito general permiten a los dsPIC controlar otros dispositivos. La mayoría de los pines están multiplexados con funciones alternativas, la multiplexación depende de las características del periférico o de la variante de dispositivo empleada. En general cuando el periférico está funcionando el pin de propósito general no es usado. En la figura se muestra el diagrama de bloques de un puerto de entrada/salida típico, en el que no se incluye ningún periférico integrado.

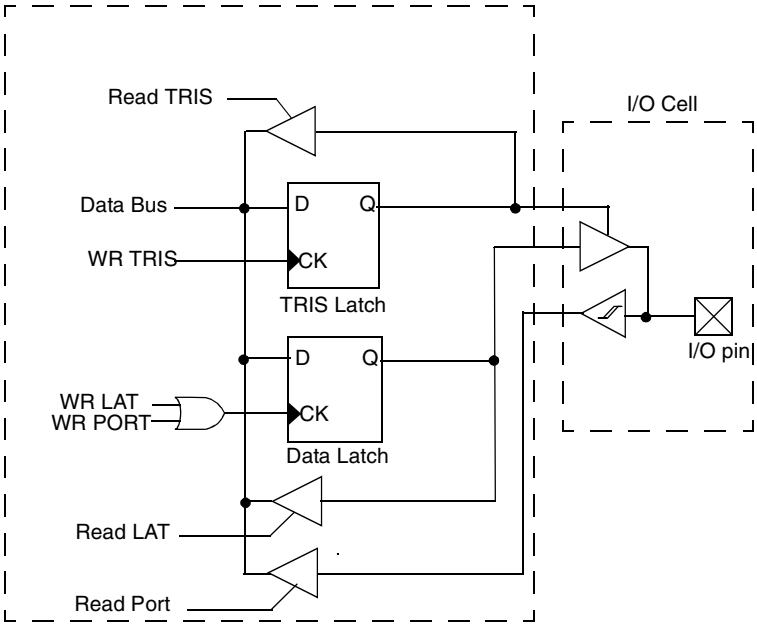



Figura 4.44: Diagrama de bloques de un puerto I/O.

## 4.5.2. Registros de control de los puertos Entrada/Salida.

Todos los puertos de entrada/salida tiene asociados tres registros para definir la función del puerto, la letra x denota la letra del puerto en particular. Estos registros son:

- **TRISx**

  Determina cuando un pin asociado a un puerto es entrada o salida. Cuando el bit de este registro correspondiente al pin está a 1 es que el pin es una entrada y si es un 0 es una salida. Todos los pines se configuran como entrada después de un Reset.

- **PORTx**
  El dato en el puerto de entrada/salida está asociado directamente con este registro, leer el valor de este registro es como leer el valor de los pines del puerto y escribir en este registro es como escribir en el cerrojo asociado a la salida.

- **LATx**
  Es el registro asociado a los pines de entrada/salida que elimina los posibles problemas que pueden ocurrir con las instrucciones de lectura-modificación-escritura. Escribir en el registro LATx tiene el mismo efecto que escribir en PORTx, la única diferencia es que leer el PORTx lee el valor de los pines directamente y leer el LATx lee el valor del cerrojo (latch) asociado el pin.

### 4.5.3. Periféricos multiplexados.

Cuando un periférico multiplexado con una entrada/salida de propósito general se habilita el pin es controlado por el periférico. La salida del periférico y la señal del registro Latch van a un par de multiplexores, que seleccionan cuando la señal de salida proviene del puerto de entrada/salida o del periférico.

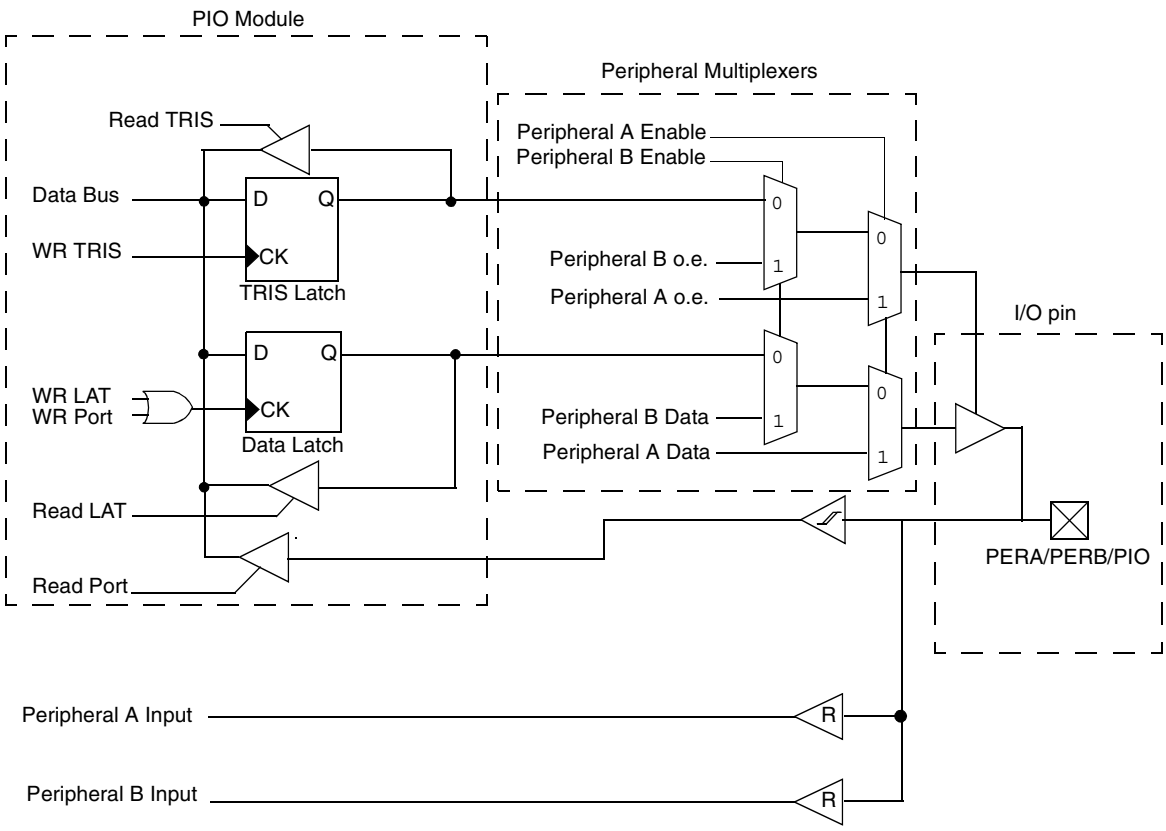



Figura 4.45: Diagrama de bloques de un puerto compartido.

### 4.5.4. Configuración de entradas analógicas.

El uso de los registros ADPCFG y TRIS sirven para controlar funcionamiento de los pines A / D del puerto. Los pines del puerto que son patillas de entrada analógicas deberán tener su correspondiente TRIS bit a uno (entrada) y el bit correspondiente del registro ADPCFG a cero.

Los pines configurados como entradas digitales no se convertirán a una entrada analógica.

### 4.5.5. Descripción de los puertos.

En la siguiente figura se muestra una tabla en la que muestran los pines controlados por los periféricos.

Posteriormente se muestran todos los pines de los que dispone el dsPIC con el que trabajamos.

| Peripheral Module "Enabled State" | Peripheral Pins | TRISx - Pin Output Control | PORTx - Pin Read |
|---|---|---|---|
| SPI™ (x = 1 or 2) | SDOx | <DISSDO = 1>, User Settable<br><DISSDO = 0>, Module Control | Yes |
| | SDIx | User Settable | Yes |
| | SCKx | Module Control | Yes |
| | SSx | <SSEN = 0>, User Settable<br><SSEN = 1>, Module Control | Yes |
| UART (x = 1 or 2) | UxRX | Module Control | Yes |
| | UxTX | <UTXEN = 0>, User Settable<br><UTXEN = 1>, Module Control | Yes |
| $I^2C$™ | SCL | Module Control | Yes |
| | SDA | Module Control | Yes |
| Input Change Notice | CN0 - CN23 | User Settable | Yes |
| Input Capture | IC1 - IC8 | User Settable | Yes |
| Output Compare | OC1 - OC8 | Module Control | Yes |
| Data Converter Interface | COFS | Module Control | Yes |
| | CSCK | Module Control | Yes |
| | CSDI | Module Control | Yes |
| | CSDO | Module Control | Yes |
| Motor Control PWM | PWMx | Module Control | Yes |
| | FLTA/B | User Settable | Yes |
| QEI | QEA | Module Control (QEI mode)<br>User Settable (16-bit Timer mode) | Yes |
| | QEB | Module Control (QEI mode)<br>User Settable (16-bit Timer mode) | Yes |
| | INDX | Module Control (QEIM<2:0> = 100 or 110)<br>User Settable in all other modes | Yes |
| CAN (x = 1 or 2) | CxRX | Module Control | Yes |
| | CxTX | Module Control | Yes |
| INTx | INT0 - INT5 | User Settable | Yes |

Figura 4.46: Pines controlados por los periféricos.

### 4.5.6. Pines de notificación de cambio CN.

Los entradas de notificación de cambio (CN) permiten a los dsPIC generar peticiones de interrupción en espera de un cambio de estando en los pines seleccionados. Se pueden seleccionar más de 24 pines de entrada para generar interrupciones por cambio de estado.

- **Registros de control CN**

  Hay cuatro registros de control asociados con el módulo de CN. Los registros CNEN1 y CNEN2 contienen la habilitación de las interrupciones CNxIE donde 'x' denota el número de pines de entrada NC. Si CNxIE = 1 habilita la interrupción para el cambio de estado de la entrada correspondiente.

  En la figura se muestran los registros CNEN1 y CNEN2. Los registros CNPU1 y CNPU2

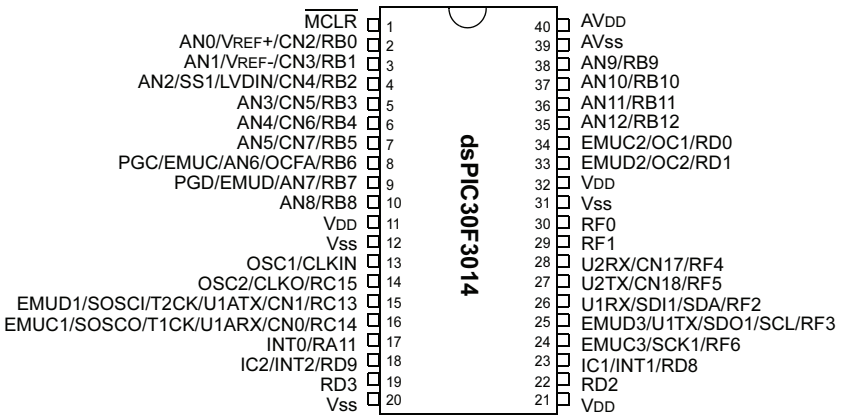



Figura 4.47: Pines del dsPIC.

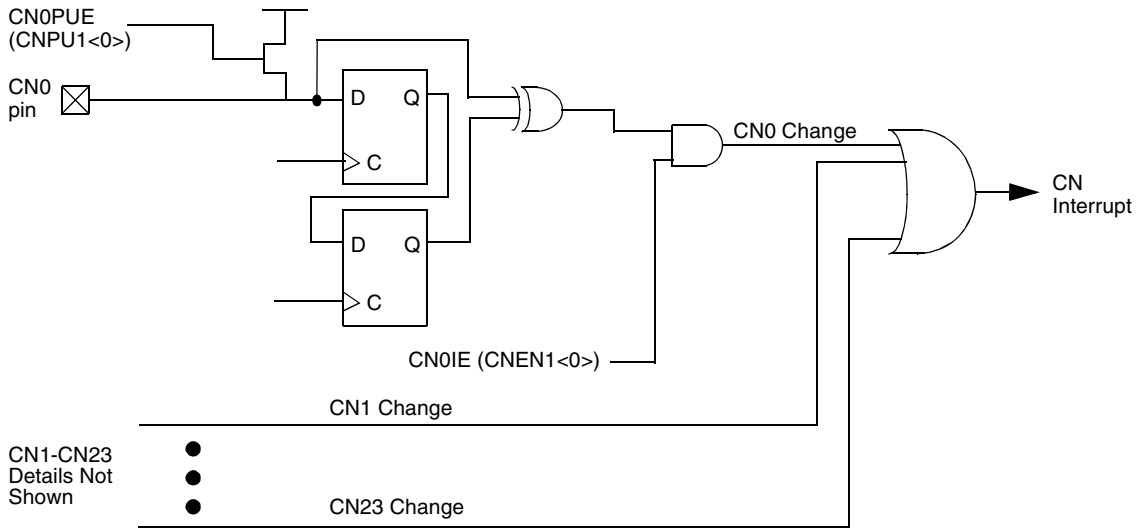



Figura 4.48: Hardware de pin CN.

| **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 |

Figura 4.49: Registro CNEN1.

| 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 |

Figura 4.50: Registro CNEN2.

contienen los bits control CNxPUE, que habilitan o no el pull-up conectado a cada una de las entradas.El pull-up interno elimina la necesidad de poner resistencias externas cuando se conecta un botón o teclado al dsPIC.

En la figura se muestran los registros CNPU1 y CNPU2.

| 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 |

Figura 4.51: Registro CNPU1.

| 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 |

Figura 4.52: Registro CNPU2.

### 4.5.7. Operación en modo Sleep e Idle de los pines CN.

El modulo CN continua funcionando durante los modos de operación Idle y Sleep. Si un pin CN cambia de estado el bit CNIF se activará y si el CNIE (IEC0<15>) está activo el dispositivo despertará de estos modos.
Si la prioridad de la interrupción es mayor que la de la CPU en ese instante el dispositivo se ejecutará la rutina de atención a la interrupción y si la prioridad es menor que la de la CPU se ejecutará la instrucción siguiente al sleep o idle.

## 4.6. Interrupciones y Excepciones.

### 4.6.1. Conceptos generales.

Las interrupciones y excepciones son acontecimientos que provocan la desviación del flujo de control de la CPU, que abandona el programa principal y pasa a ejecutar una rutina que atiende la causa que la ha originado.
Las interrupciones son provocadas generalmente por acontecimientos externos. Las excepciones son desviaciones del flujo de control provocadas automáticamente como secuencia de alguna anomalía en la CPU producida y detectada en el desarrollo del programa en curso de ejecución.
Para manejar las interrupciones y las excepciones los dsPIC30F disponen de una Tabla de Vectores de Interrupción llamada IVT.
Los dsPIC30F disponen de un vector de interrupción por cada uno de los tipos de interrupción o excepción. La tabla IVT tiene 62 entradas distintas, es decir, 62 vectores para atender cada una de las distintas interrupciones.
Además de la IVT, se dispone de una Tabla Alternativa de Vectores de Interrupción AIVT, que es similar a la IVT.
Una característica que destaca en el control de interrupciones de los dsPIC es el uso de prioridades en el manejo de las interrupciones.

### 4.6.2. Tabla de vectores de interrupción y excepción IVT.

Cada tipo de interrupción está asociado a una de las 62 entradas de la tabla IVT. Esta tabla reside en la memoria de programa, comenzando en la dirección 00000004H. 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 pone en marcha cuando la CPU atiende una interrupción.

La tabla alternativa AIVT se encuentra en la memoria de programa justo detrás de la IVT. Se accede a ella mediante el bit ALTIVT que cuando está a 1, las interrupciones y excepciones usarán la tabla de vectores alternativa.
En la siguiente figura se representan las diferentes excepciones e interrupciones que soportan los dsPIC30F.

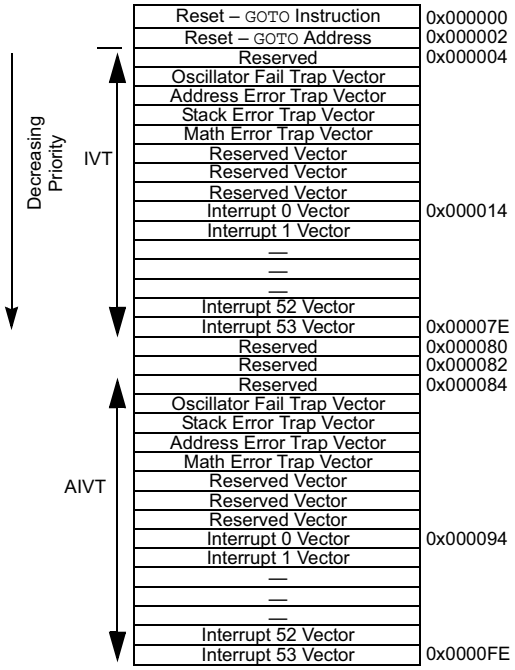



Figura 4.53: Mapa de la tabla de vectores de interrupción(IVT) y de la tabla alternativa(AIVT)

### 4.6.3. Niveles de prioridades.

1. **Prioridades de la CPU.** Las excepciones y las interrupciones admiten 16 niveles de prioridad, del nivel 0 al nivel 15.

   Para poder inicializar el proceso de 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. 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: Interrupciones externas(periféricos). La CPU tiene uno de estos niveles.
   - Del 8 al nivel 15: Excepciones internas, no mascarables, 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. El nivel de prioridad de la CPU se indica en los bits de

| Vector Number | IVT Address | AIVT Address | Interrupt Source |
|---|---|---|---|
| 8 | 0x000014 | 0x000094 | INT0 – External Interrupt 0 |
| 9 | 0x000016 | 0x000096 | IC1 – Input Capture 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 |

Figura 4.54: Tabla de vectores de interrupción

estado:IPL[2:0] del registro SR[7:5] y el IPL3 del registro CORCON[3]. 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 IPL3 nos indica si la interrupción es mascarable o no. Si IPL3 está activo, significa que una excepción está en proceso.

2. **Prioridades de las interrupciones.** Cada interrupción externa es asignada a uno de los 7 niveles de prioridades por el usuario.Estas prioridades empiezan en el nivel 1(nivel más bajo) y acaban en el 7(nivel más alto).
Además de los 7 niveles, cada fuente de interrupción tiene un orden de prioridad natural dado por la IVT.
El motivo de haber dos formas de asignar niveles de prioridades se debe a que el usuario puede asignar a distintas fuentes de interrupción niveles de prioridades iguales, cuando estas interrupciones se produzcan, en el mismo instante, la CPU no sabrá a cuál atender, y entonces entran en juego las prioridades naturales, en este caso se atendería la que mayor nivel de prioridad natural posee. La prioridad natural viene dada por el orden en la tabla de vectores de interrupción, cuanto menor sea el número de vector mayor prioridad tendrá.

### 4.6.4. Tipos de excepciones.

Las excepciones se ejecutan siempre y tienen la función de avisar al usuario que se ha producido una operación errónea y que debe corregirla, tanto a nivel software como hardware. Si el usuario no corrige el error, el vector de excepción determinado se cargará con la dirección de la rutina de software y reiniciará la máquina. La rutina de excepción podrá corregir la causa de la excepción antes de la reinicialización.
Existen cuatro fuentes de excepción, agrupadas en excepciones software y excepciones hardware:

- **Excepciones software:**
  - *Fallo de pila.*
    Existe un registro límite de pila, SPLIM. Cuando se escribe SPLIM se activa la detección del sobrepasamiento de la pila y cuando se intenta escribir en una dirección que está fuera del límite se provoca una excepción.
    Las direcciones efectivas,EA, se comparan al valor contenido en SPLIM. Si son mayores que el límite se lanza una excepción de pila, ya que se está intentado direccionar

fuera del espacio permitido.
Este tipo de excepción se detecta por la activación del bit STKERR del INTCON1[2].

- *Error aritmético.*
  Son excepciones causadas por :
  - Sobrepasamiento del acumulador A.
  - Sobrepasamiento del acumulador B.
  - Sobrepasamiento catastrófico del acumulador.
  - División por 0.
  - El desplazamiento del acumulador(SPTAC) excede en +/- 16 bits.

- **Excepciones hardware:**
  La diferencia entre las excepciones software y hardware consiste en que las últimas fuerzan la atención de la CPU antes que la instrucción culpable de la excepción haya finalizado su ejecución, a diferencia de las excepciones software que permiten que la instrucción finalice. Tipos:
  - Error en el direccionamiento, prioridad 13.
  - Fallo del oscilador, prioridad 14.

Cuando ocurre una excepción hardware la máquina se reinicia automáticamente y el bit de estado TRAPR (RCON[15]) se pone a 1.
Las excepciones sólo se detectarán cuando ya han ocurrido. La instrucción que provoca la excepción tiene dos opciones de finalización:

- Acabar la instrucción antes de lanzar la excepción, si se trata de una excepción software.
- No se la permite acabar en excepciones hardware.

### 4.6.5. Manejo de interrupciones.

#### Funcionamiento de una interrupción.

Las instrucciones son atendidas en cada ciclo de instrucción. Cuando hay una petición de interrupción IRQ, pendiente se señaliza poniendo el bit del flag a 1 en el registro IFS. Esta petición será atendida si el bit IECx del registro de permiso correspondiente está a 1. Al atender la petición de interrupción ninguna instrucción abortará, sino que la que estaba en proceso cuando se atendió la interrupción será completada cuando la interrupción termine.

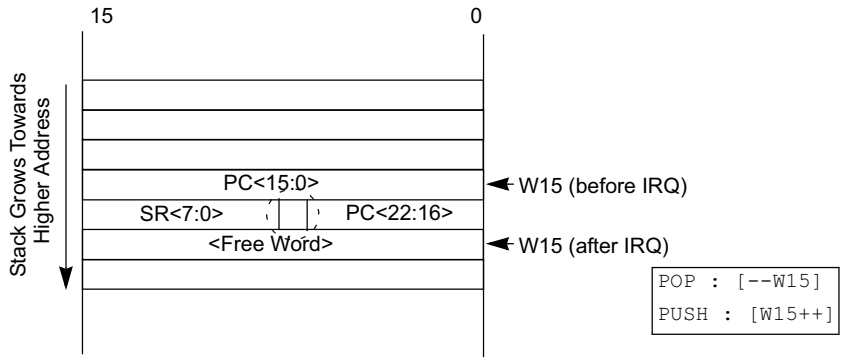



Figura 4.55: Datos almacenados en la atención de una interrupción.

Al atender a las interrupciones se evalúan los niveles de prioridad. Si hay una IRQ pendiente 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 IPL3.

## Interrupciones anidadas.

Las interrupciones son anidables. 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 se produce un conflicto porque esta no puede ser atendida hasta que la de mayor prioridad no termine.
Si el bit NSTDIS del registro INTCON1 está a 1, las interrupciones anidadas se habilitan y entonces se fuerza a la CPU al nivel, IPL=111, descartandose todas las interrupciones de menor nivel al 7.

## Despertar la CPU del estado SLEEP o IDLE.

Las interrupciones pueden despertar al 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 continuara la ejecución comenzando con la instrucción siguiente a la que provocó el estado SLEEP o IDLE en la CPU.

Mediante la instrucción DISI se puede deshabilitar las interrupciones de nivel 1-6.

### 4.6.6. Tiempo de proceso de las interrupciones.

El proceso de una interrupción necesita cuatro ciclos:
$1^{er}$ciclo: Se activa el flag de estado.
$2^{o}$ciclo: Se salva el contenido del PC y de SRL en buffers temporales para poder recuperarlos después.
$3^{o}$ciclo: Se carga el PC con el valor de la tabla de vectores de interrupciones correspondiente a la interrupción que se va a atender.
$4^{o}$ciclo: Se carga el PC con el valor de la dirección de la rutina de atención a la interrupción.
El retorno de interrupción se lleva a cabo por medio de la instrucción RETFIE. Se necesitan tres ciclos de instrucción:
$1^{er}$ciclo: Se cargan en la pila los 8 bits más significativos del PC y todos los del SRL.
$2^{o}$ciclo: Se cargan en la pila los 16 bits menos significativos del PC.
$3^{o}$ciclo: Se usa para buscar la dirección de la instrucción.

### 4.6.7. Registros de control de interrupciones.

Los registros que controlan las interrupciones y su estado se dividen en seis grupos:
Registros INTCON1 e INTCON2 que controlan las funciones globales de las interrupciones.

- INTCON1: Contiene los flags y bits de control de las excepciones y el bit NSTDIS que habilita/deshabilita el anidamiento de interrupciones.

| **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 |
| NSTDIS | OVAERR | OVBERR | COVAERR | COVBERR | OVATE | OVBTE | COVTE |
| bit 15 | | | | | | | bit 8 |

| **Lower Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/W-0-0 | R/W-0-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 |
| SFTACERR | DIV0ERR | — | MATHERR | ADDRERR | STKERR | OSCFAIL | — |
| bit 7 | | | | | | | bit 0 |

Figura 4.56: Estructura del registro INTCON1(Interrupt Control Register 1).

- NSTDIS: Bit que deshabilita las interrupciones anidadas. Estando a 1 se deshabilitan.

- OVATE: Habilita la excepción de sobrepasamiento del acumulador. Se habilita poniendo a 1.
- OVBTE: Habilita la excepción de sobrepasamiento del acumulador B. Con 1 se habilita.
- COVTE: Habilita la excepción de sobrepasamiento catastrófica. Con 1 la excepción de de sobrepasamiento catastrófica del acumulador A o B queda habilitada.
- MATHERR: Bit de estado de error aritmético. Si este flag está activado, indicará que ha ocurrido una excepción motivada por algún error aritmético.
- ADDRERR: Bit de estado de error de direccionamiento. Un 1 significa que ha ocurrido una excepción de error de direccionamiento.
- STKERR: Bit de estado de error de pila. Estando este bit a 1 indica que se ha producido una excepción de error en la pila.
- OSCFAIL: Bit de estado de fallo de oscilador. Si este bit está a 0 aún no habrá ocurrido ninguna excepción de error del oscilador.

- INTCON2: Contiene los bits para el control de las interrupciones externas y el bit que activa la AIVT.

| **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 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
| — | — | — | — | — | INT2EP | INT1EP | INT0EP |
| bit 7 | | | | | | | bit 0 |

Figura 4.57: Estructura del registro INTCON2(Interrupt Control Register 2).

- ALTIVT: Bit que habilita la tabla de vectores de interrupción alternativa. Si está a 1 se utiliza la AIVT. Estando a 0 la IVT.
- DISI: Bit de estado de la instrucción DISI. Si se pone a 1, se ha ejecutado dicha instrucción.

- INT4EP: Interrupción externa número 4. Un 1 habilita la patita de interrupción.
- INT3EP: Interrupción externa número 3. Se activa con 1.
- INT2EP: Interrupción externa número 2. Con 1 se permiten las interrupciones por esta patita.
- INT1EP: Interrupción externa número 1. Las interrupciones por esta patita son aceptadas si está a 1.
- INT0EP: Interrupción externa número 0. Con 1 queda habilitada la patita.

- 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 registro IPL que indica el nivel de prioridad de la CPU.

| **Upper Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| OA | OB | SA | SB | OAB | SAB | DA | DC |
| bit 15 | | | | | | | bit 8 |

| **Lower Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

Figura 4.58: Estructura del registro SR(Status Register).

- IPL[2:0]: Bits que indican el nivel de prioridad de la CPU.
  111 = Nivel de prioridad 7.
  110 = Nivel de prioridad 6.
  101 = Nivel de prioridad 5.
  100 = Nivel de prioridad 4.
  011 = Nivel de prioridad 3.
  010 = Nivel de prioridad 2.
  001 = Nivel de prioridad 1.
  000 = Nivel de prioridad 0.
  Estos bits están concatenados con IPL[3] (registro CORCON[3]) para formar el nivel de prioridad de la CPU.
  Estos bits son leídos solo cuando NSTDIS = 1 (INTCON1[15]).

- Registro CORCON: Contiene el bit de estado IPL3 con el que se identifica el nivel de prioridad de la CPU.

| 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 |

Figura 4.59: Estructura del registro CORCON(Core Control Register).

- IPL3: Bit que marca el nivel de prioridad de la CPU.
  1 = Nivel de prioridad de interrupción de la CPU mayor que 7.
  0 = Nivel de prioridad de interrupción de la CPU menor o igual que 7.

- IFSx: Registro de Estado del flag de interrupción. Contiene todos los señalizadores de petición de interrupciones. Cada fuente de interrupción tendrá un bit de estado, que se activará desde la fuente de interrupción y se desactivará por medio de software. La x es el número de registro: IFS0,IFS1 y IFS2. Los flags se han activado si está a 1, si es 0 no ha ocurrido el evento que marca ese flag.

| Upper Byte: | | | | | | | |
|---|---|---|---|---|---|---|---|
| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| — | MI2CIF | SI2CIF | NVMIF | ADIF | U1TXIF | U1RXIF | SPI1IF |
| bit 15 | | | | | | | bit 8 |

| Lower Byte: | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| T3IF | T2IF | OC2IF | — | T1IF | OC1IF | IC1IF | INT0IF |
| bit 7 | | | | | | | bit 0 |

Figura 4.60: Registro IFS0(Interrupt Flag Status Register 0).

- CNIF: Flag de notificación de un cambio de estado en una entrada que provoca interrupción.
- MI2CIF: Para señalizar que se ha producido una colisión en el bus I2C se emplea este Flag.
- SI2CIF: Una vez completada la transferencia mediante el bus I2C se produce la activación de este flag.

- NVMIF: La activación de este flag indica que se ha completado la escritura en la memoria no-volátil.
- ADIF: Indica que ya ha finalizado la conversión A/D.
- U1TXIF: Para poder señalizar que se está produciendo la transmisión de comunicación UART1.
- U1RXIF: La recepción de la comunicación UART1 se indica a través de este flag.
- SPI1IF: Mediante este flag se señaliza la interrupción en el SPI1.
- T3IF: El Timer3 podrá producir una interrupción que detecta este bit.
- T2IF: El Timer2 podrá producir una interrupción que detecta este bit.
- OC2IF: Este bit se emplea para señalizar la salida del comparador 2.
- IC2IF: Bit empleado para activar el flag de la entrada del módulo de captura 2.
- T1IF: Interrupción del Timer 1.
- OC1IF: Salida del comparador 1.
- IC1IF: Entrada del módulo de captura1.
- INT0IF: Interrupción externa número 0.

| **Upper Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | U-0 | U0 | U-0 |
| AC3IF | AC2IF | AC1IF | — | CNIF | — | — | — |
| bit 15 | | | | | | | bit 8 |

| **Lower Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| — | PWM4IF | PWM3IF | PWM2IF | PWM1IF | PSEMIF | INT2IF | INT1IF |
| bit 7 | | | | | | | bit 0 |

Figura 4.61: Registro IFS1(Interrupt Flag Status Register 1).

- IC6IF: Este flag indica la activación en la entrada 6 del módulo de Captura.
- IC5IF: Este flag indica la activación en la entrada 5 del módulo de Captura.
- IC4IF: Este flag indica la activación en la entrada 4 del módulo de Captura.
- IC3IF: Este flag indica la activación en la entrada 3 del módulo de Captura.
- C1IF: Se emplea para indicar que se ha producido una interrupción en el CAN1 (Combinado).

- SPI2IF: Flag que se activa al producirse una interrupción en el SPI2.
- U2TXIF: Se activa al producirse la transmisión del UART2.
- U2RXIF: Señalización de recepción UART2.
- INT2IF: Bit empleado para indicar que se ha provocado una interrupción externa número 2
- T5IF: Interrupción debida al Timer5.
- T4IF: Interrupción debida al Timer4.
- OC4IF: Cuando el comparador activa su salida 4 esta se indica por medio de este bit.
- OC3IF: Cuando el comparador activa su salida 3 esta se indica por medio de este bit.
- IC8IF: Si en el módulo de Captura se activa la entrada 8 se pone a 1 este flag.
- IC7IF: Si en el módulo de Captura se activa la entrada 7 se pone a 1 este flag.
- INT1IF: Flag que se activa al producirse la interrupción externa número 1.

| **Upper Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
| — | — | — | — | — | ADCP5IF | ADCP4IF | ADCP3IF |
| bit 15 | | | | | | | bit 8 |

| **Lower Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 |
| ADCP2IF | ADCP1IF | ADCP0IF | — | — | — | — | AC4IF |
| bit 7 | | | | | | | bit 0 |

Figura 4.62: Registro IFS2(Interrupt Flag Status Register 2).

- FLTBIF: FAULT B.
- FLTAIF: FAULT A.
- LVDIF: Cuando se ha detectado un voltaje bajo se pone a 1 este flag.
- DCIIF: Indica la interrupción provocada por el Conversor de datos.
- QEIIF: Flag del codificador de cuadratura.
- PWMIF: Modulación de Anchura de Pulsos para control de motores.
- C2IF: CAN2(Combinado).

- INT4IF: Flag activo cuando se produce una interrupción externa número 4.
- INT3IF: Flag activo cuando se produce una interrupción externa número 3.
- OC8IF: Salida 8 del comparador.
- OC7IF: Salida 7 del comparador.
- OC6IF: Salida 6 del comparador.
- OC5IF: Salida 5 del comparador.

- IECx: Registros de control de permiso de interrupciones. Contienen los bits de habilitación de las interrupciones. La x es el número de registro: IEC0,IEC1 y IEC2.

| **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 |

Figura 4.63: Registro IEC0(Registro 0 de control de habilitación de interrupciones).

- CNIE: Para habilitar la interrupción por cambio de estado en una entrada.
- MI2CIE: Para habilitar el control de colisión en el bus I2C.
- SI2CIE: Bit empleado para que cuando se dé una transferencia en el I2C y esta haya sido completada se produzca una interrupción.
- NVMIE: Bit que posibilita que cuando se haya finalizado la escritura en la memoria no volátil, se dé una interrupción.
- ADIE: Si este bit está a 1, se habilitará la interrupción de la conversión A/D.
- U1TXIE: Habilita la interrupción del transmisor UART1.
- U1RXIE: El receptor UART1 podrá provocar una interrupción si este bit vale 1.
- SP1IE: Se habilita la interrupción provocada por el SPI1.
- T3IE: El temporizador 3 podrá generar una interrupción dependiendo del valor de este bit. Se habilita con 1.
- T2IE: Timer 2 podrá provocar una interrupción siempre y cuando este bit contenga 1.

- OC2IE: Cuando se produzca una señal de salida en el comparador 2 se podrá generar una interrupción dependiendo del valor de este bit. Un 1 lo habilita.
- IC2IE: Se habilita la interrupción provocada por la entrada 2 en el módulo de Captura.
- T1IE: Se podrá producir una interrupción debido al Timer 1 si este bit está activado.
- OC1IE: Entrada 1 en el módulo de captura.
- IC1IE: Entrada 1 en el módulo de captura.
- INT0IE: La interrupción externa número 0 podrá provocar una interrupción cuando está activado este bit.

| **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 |

Figura 4.64: Registro IEC1(Registro 1 de control de habilitación de interrupciones).

- IC6IE: Si este bit contiene un uno, se habilitará la interrupción provocada por la entrada 6 en el módulo de captura.
- IC5IE: Entrada 5 del módulo de Captura.
- IC4IE: Entrada 4 del módulo de Captura.
- IC3IE: Entrada 3 del módulo de Captura.
- C1IE: Mediante este bit se posibilita la interrupción del CAN1 (Combinado).
- SPI2IE: Se habilita la interrupción provocada por el SPI2.
- U2TXIE: A través de este bit se habilita o deshabilita la interrupción debido al transmisor UART2.
- U2RXIE: Mediante este bit se da la posibilidad de habilitar o no la interrupción que provoca el receptor UART2.
- INT2IE: Bit empleado para la habilitación de la interrupción externa número 2.

- T5IE: La interrupción provocada por un temporizador, concretamente el Timer 5, podrá estar habilitada o no según el valor de este bit.
- T4IE: La interrupción provocado por un temporizador, concretamente el Timer 4, podrá estar habilitado o no según el valor de este bit.
- OC4IE: Salida 4 del comparador.
- OC3IE: Salida 3 del Comparador.
- IC8IE: Entrada 8 del módulo de Captura.
- IC7IE: Entrada 7 del módulo de Captura.
- INT1IE: Bit empleado para la habilitación de la interrupción externa número 1.

| **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 |

Figura 4.65: Registro IEC2(Registro 2 de control de habilitación de interrupciones).

- FLTBIE: FAULT B.
- FLTAIE: FAULT A.
- LVDIE: Si en algún momento se detecta un voltaje bajo se podrá producir una interrupción.
- DCIIE: El Conversor de datos produce una interrupción si este bit está habilitado.
- QEIIE: Permiso para provocar interrupción el Codificador de Cuadratura.
- PWMIE: La modulación de anchura de pulsos podrá provocar la ejecución de la interrupción si este bit se encuentra habilitado.
- C2IE: Mediante este bit se habilita la ejecución de la rutina de interrupción provocado por el CAN2 (comparador).
- INT4IE: Bit empleado para habilitar la interrupción externa número 4.
- INT3IE: La interrupción externa puerta 3 podrá provocar la ejecución de la rutina de interrupción, siempre y cuando este bit valga 1.
- OC8IE: Salida 8 del comparador.
- OC7IE: Salida 7 del comparador.
- OC6IE: Salida 6 del comparador.
- OC5IE: Salida 5 del comparador.

- IPCx: Registros de control de prioridades de interrupciones. Se usan para almacenar el nivel de prioridad de cada fuente de interrupción.

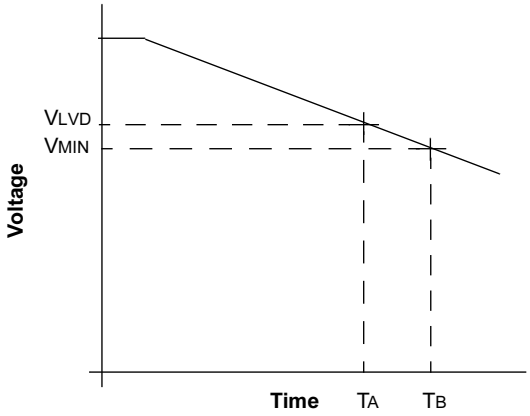
| SFR Name | ADR | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INTCON1 | 0080 | NSTDIS | — | — | — | — | OVATE | OVBTE | COVTE | — | — | — | MATHERR | ADDRERR | STKERR | OSCFAIL | — | 0000 0000 0000 0000 |
| INTCON2 | 0082 | ALTIVT | DISI | — | — | — | — | — | — | — | — | — | INT4EP | INT3EP | INT2EP | INT1EP | INT0EP | 0000 0000 0000 0000 |
| IFT0IF | 0084 | CNIF | MI2CIF | SI2CIF | NVMIF | ADIF | U1TXIF | U1RXIF | SPI1IF | T3IF | T2IF | OC2IF | IC2IF | T1IF | OC1IF | IC1IF | INT0 | 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 |
| 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 | — | 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> | | | — | SPI2IP<2:0> | | | — | U2TXIP<2:0> | | | — | U2RXIP<2:0> | | | 0100 0100 0100 0100 |
| IPC7 | 00A2 | — | IC6IP<2:0> | | | — | IC5IP<2:0> | | | — | IC4IP<2:0> | | | — | IC3IP<2:0> | | | 0100 0100 0100 0100 |
| IPC8 | 00A4 | — | OC8IP<2:0> | | | — | OC7IP<2:0> | | | — | OC6IP<2:0> | | | — | OC5IP<2:0> | | | 0100 0100 0100 0100 |
| IPC9 | 00A6 | — | PWMIP<2:0> | | | — | C2IP<2:0> | | | — | INT41IP<2:0> | | | — | INT3IP<2:0> | | | 0100 0100 0100 0100 |
| IPC10 | 00A8 | — | FLTAIP<2:0> | | | — | LVDIP<2:0> | | | — | DCIIP<2:0> | | | — | QEIIP<2:0> | | | 0100 0100 0100 0100 |
| IPC11 | 00AA | — | — | — | — | — | — | — | — | — | — | — | — | — | FLTBIP<2:0> | | | 0000 0000 0000 0100 |

**Note:** All interrupt sources and their associated control bits may not be available on a particular device. Refer to the device data sheet for details.

Figura 4.66: Mapa de registros.

## 4.7. Módulo de detección de bajo voltaje.

### 4.7.1. Introducción.




**Legend:**
VLVD = LVD trip point
VMIN = Minimum valid device operating voltage

Figura 4.67: Relación voltaje/tiempo del dispositivo controlado por el LVD.

El módulo de detección de voltaje bajo es muy útil en aplicaciones que funcionan con baterías. Mientras la batería alimenta el sistema se va descargando siendo la función del LVD la que detecta cuando el voltaje de la batería baja de un umbral que impide el correcto funcionamiento.
Este módulo utiliza una referencia interna de voltaje para la comparación. Este umbral es programable durante la ejecución de la aplicación.
En esta figura se muestra una posible curva de voltaje de la batería del dispositivo en una aplicación. Con el transcurso del tiempo el nivel de batería disminuye. Cuando su valor iguala a $V_{LVD}$, la lógica del módulo LVD genera una interrupción. Esto ocurre en el instante de tiempo $T_A$. La aplicación software debe dejar de funcionar, ya que el nivel de tensión no es suficiente para que el sistema permanezca activo. En el instante $T_B$ se produce el valor de tensión mínimo válido para trabajar, por lo que el tiempo total para desconectar el sistema es $T_B - T_A$.
Tiene un comparador que usa una tensión de referencia generada internamente. Cuando el voltaje del dispositivo es menor que el voltaje de referencia el bit LVDIF se pone a 1. El multiplexor selecciona uno de entre todos los valores de tensión posibles, pudiendo ser uno de ellos el introducido por la patita LVDIN. La lógica del módulo de detección de voltaje bajo no generará una interrupción mientras la salida del comparador, LVDIF, no se active. Si se habilita el módulo de Detección de Voltaje Bajo, realizará sus funciones incluso funcionando en modo

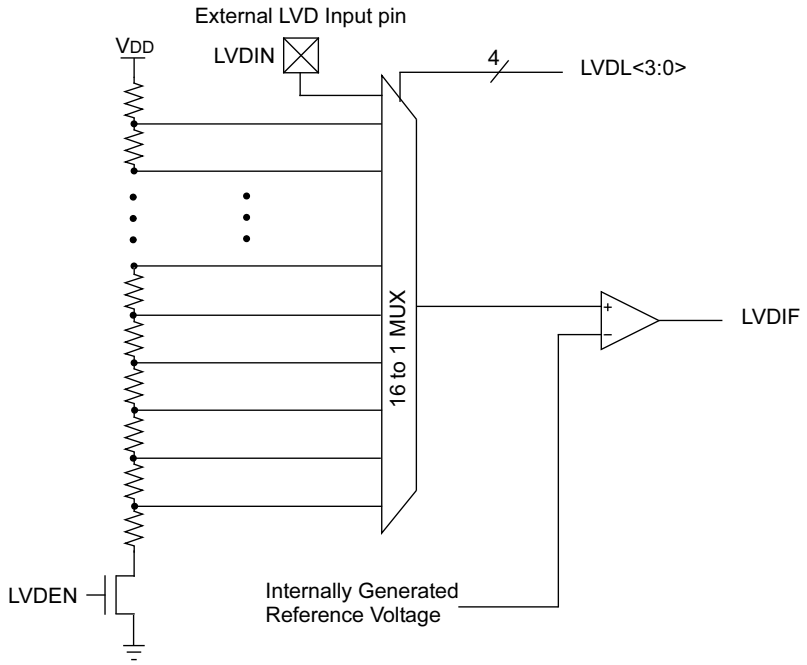



Figura 4.68: Diagrama de bloques del módulo LVD.

Sleep o Idle.

### 4.7.2. Bits de Control del módulo LVD.

| **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 |

Figura 4.69: Registro RCON.

- BGST: Bandgap Stable bit.
  1 = El bandgap está estabilizado.
  0 = El bandgap no estabilizado (puede que las interrupciones LVD deshabilitadas).
- LVDEN: Bit de habilitación del módulo LVD.
  1 = Habilitado LVD.

0 = Deshabilitado.

- LVDL[3:0] Bits que marcan los límites de tensión.
  1111 = Entrada externa a LVD desde LVDIN.
  1110 = 4.6V
  1101 = 4.1V
  1100 = 4.1V
  1011 = 3.9V
  1010 = 3.7V
  1001 = 3.6V
  1000 = 3.4V
  0111 = 3.1V
  0110 = 2.9V
  0101 = 2.8V (por defecto)
  0100 = 2.6V
  0011 = 2.5V
  0010 = 2.3V
  0001 = 2.1V
  0000 = 1.9V

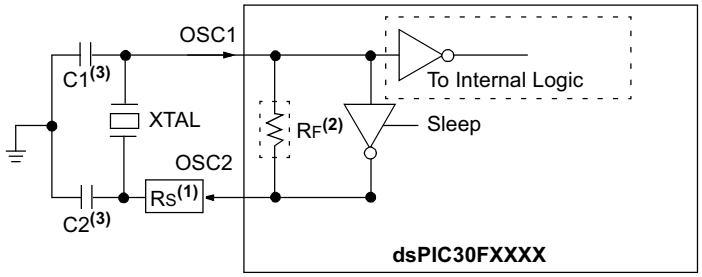
# 4.8. Oscilador.

## 4.8.1. El sistema oscilador.

Es el encargado de proporcionar la señal de reloj principal y todas las auxiliares que alimentan a los recursos del dispositivo. Para realizar esta función se dispone de tres osciladores primarios, un oscilador secundario, dos osciladores internos y un oscilador externo. También se dispone de un circuito PLL para elevar la frecuencia interna.
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.
- Selección entre varias fuentes de reloj.
- Un postscaler programable.
- Un monitor de seguridad ante fallo del reloj(FSCM) que detecta posibles fallos.
- Un registro de control de opciones de reloj(OSCON).
- Bits de configuración no volátiles para la selección del oscilador principal.

  - **Osciladores primarios:**
    Existen tres tipos:




**Note 1:** A series resistor, Rs, may be required for AT strip cut crystals.
**2:** The internal feedback resistor, RF, is typically in the range of 2 to 10 MΩ.
**3:** See **Section 7.7 "Determining Best Values for Crystals, Clock Mode, C1, C2 and Rs"**.

Figura 4.70: Oscilador Primario.

    - XTL: Se ha diseñado para trabajar con un cristal de cuarzo o resonador cerámico para un rango de frecuencias comprendido entre 200KHz y 4MHz.

- XT: Utiliza un cristal o resonador y trabaja a frecuencias comprendidas entre 4MHz y 10MHz.
- HS: Utiliza sólo cristal de cuarzo y cubre la banda de frecuencias comprendida 10MHz y 25MHz.

Todos los osciladores primarios utilizan las patitas OSC1 y OSC2.

| 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 |
| 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 | 8 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:** The dsPIC30F maximum operating frequency of 120 MHz must be met.
**2:** LP oscillator can be conveniently shared as a system clock, as well as a Real-Time Clock for Timer1.
**3:** Requires external R and C. Frequency operation up to 4 MHz.

Figura 4.71: Modos de funcionamiento del oscilador.

- **Oscilador secundario:**
  El oscilador secundario LP está diseñado para funcionar a baja potencia y se basa en un resonador de cristal o cerámico de 32KHz, siendo SOSC1 y SOSC2 las patillas que se utilizan. Este oscilador también puede controlar el temporizador Timer1 para aplicaciones en tiempo real.
- **Osciladores internos:**
  Existen dos osciladores internos. El FRC (Fast RC) trabaja a 8 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 512KHz. 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 único oscilador externo disponible (EXTRC) trabaja a frecuencias que llegan a los 4MHz. Utiliza una resistencia y un condensador externos conectados a la patita OSC1, la cual también puede conectarse a una señal de reloj externa(MODO EC).

### 4.8.2. Registros de control.

A través de los siguientes registros podemos configurar el oscilador.

| File 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 | All Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FOSCSEL | F80006 | — | — | — | — | — | — | — | — | — | — | — | — | — | FNOSC<2:0> | | | |
| FOSC | F80008 | — | — | — | — | — | — | — | — | FCKSM<1:0> | | FRANGE | — | — | OSCIOFNC | POSCMD<1:0> | | |

**Legend:** x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Figura 4.72: Registro de configuración del oscilador.

| File 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 | All Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OSCCON | 0742 | — | COSC<2:0> | | | — | NOSC<2:0> | | | CLKLOCK | — | LOCK | PRCDEN | CF | TSEQEN | — | OSWEN | 7700(1) |
| OSCTUN | 0748 | TSEQ3<3:0> | | | | TSEQ2<3:0> | | | | TSEQ1<3:0> | | | | TUN<3:0> | | | | 0000 |
| OSCTUN2 | 0746 | TSEQ7<3:0> | | | | TSEQ6<3:0> | | | | TSEQ5<3:0> | | | | TSEQ4<3:0> | | | | 0000 |
| LFSR | 0748 | — | LFSR<14:0> | | | | | | | | | | | | | | | 0000 |

**Legend:** x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.
**Note 1:** OSCCON register Reset values are dependent on the FOSCSEL Configuration bits and type of Reset.

Figura 4.73: Registro de control del oscilador.

### 4.8.3. Diagrama por bloques del sistema oscilador.

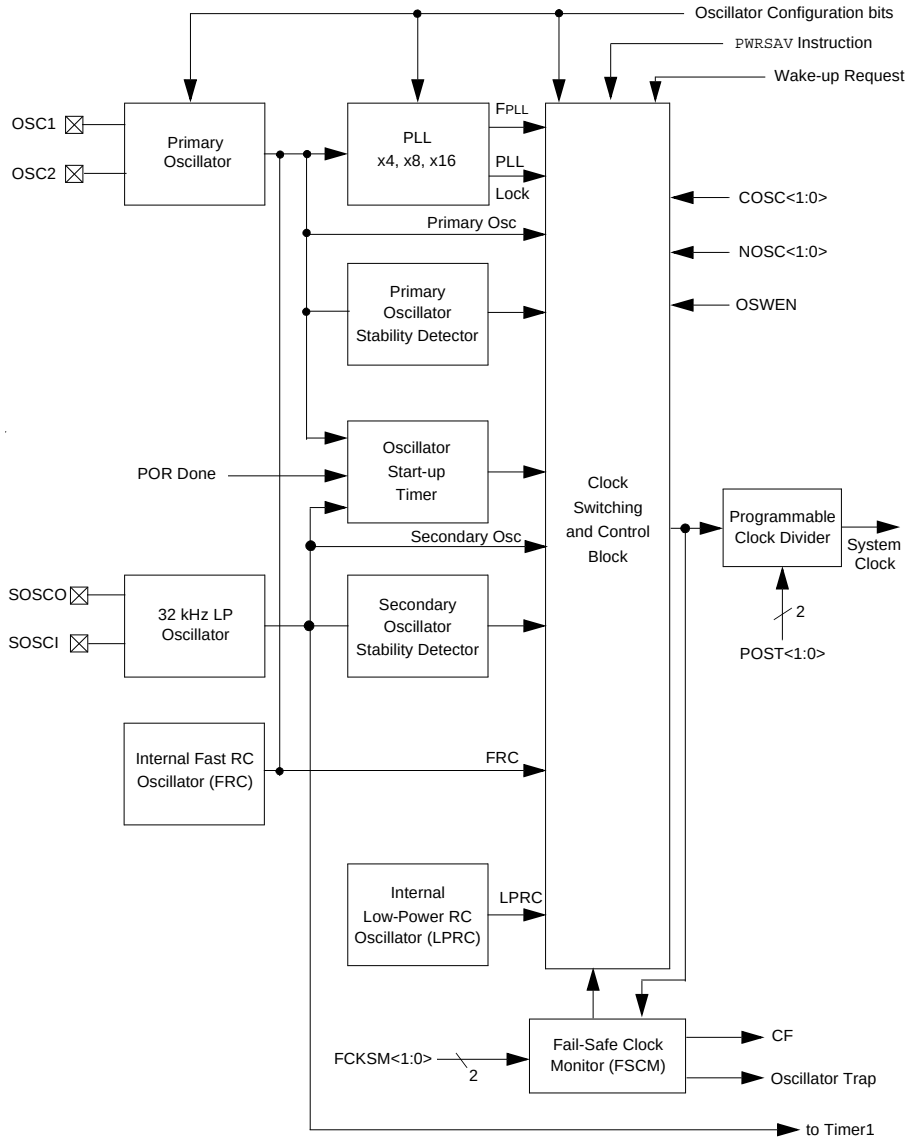



Figura 4.74: Diagrama de bloques del sistema oscilador.

# 4.9. Temporizadores.

## 4.9.1. Introducción.

La familia dsPIC30F dispone de un número variable de temporizadores de 16 bits. Estos temporizadores son designados como Timer1, Timer2,...etc. 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 algunos de ellos se pueden agrupar para formar un temporizador de 32 bits.

## 4.9.2. Temporizador Tipo A.

Al menos uno de los temporizadores será de tipo A (Timer 1, típicamente). Características:

- Puede operar con un oscilador de bajo consumo de 32 kHz. Esto le permite ser utilizado como un reloj en tiempo real (RTC).
- Puede funcionar en modo asíncrono desde una fuente de reloj externa.

Los bits que forman este registro son:

- TON: Activación del temporizador.
  1 = Temporizador activado.
  0 = Temporizador desactivado.
- TSDIL: Detención del temporizador en modo Idle.
  1 = El temporizador se detiene en modo Idle.
  0 = EL temporizador continua en modo Idle.

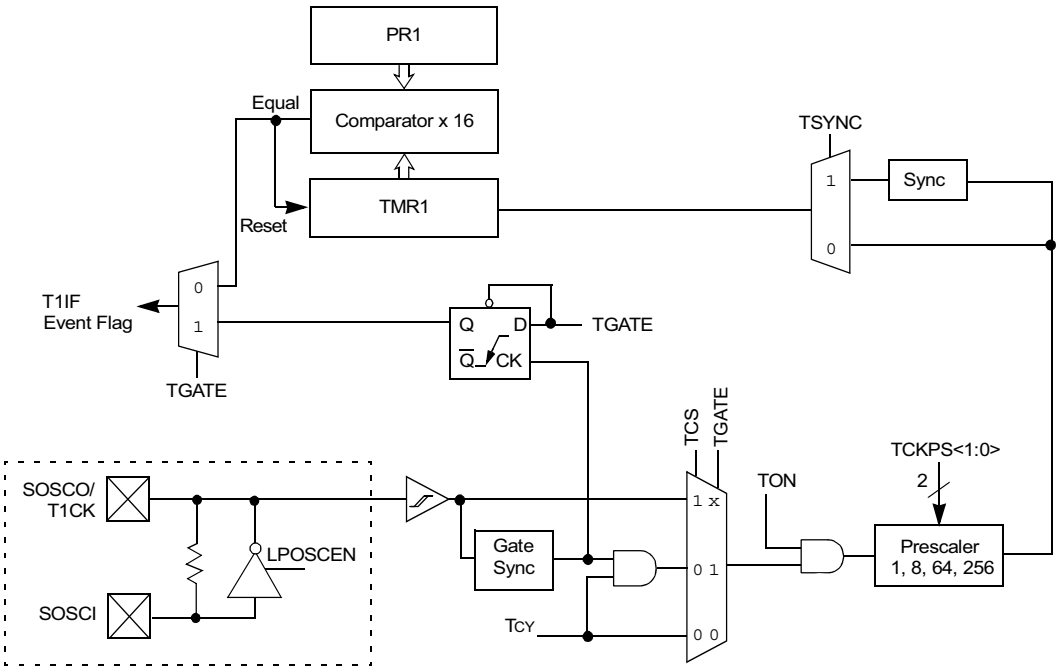



Figura 4.75: Diagrama de bloques del temporizador Tipo A.

- TGATE: Habilitación de modo de disparo por acumulación de tiempo.
  1 = Habilitado.
  0 = Deshabilitado.
  En caso en el que esté habilitado el bit TCS deberá tomar el valor 0 ya que en caso contrario TGATE será leído como 0 siempre.

- TCKPS[1:0]: Bit para el prescaler:
  11 = 1:256
  10 = 1:64
  01 = 1:8
  00 = 1:1

- TSYNC: Sincronización con señal externa de reloj.
  Cuando TCS=1.
  1 = Hay sincronización.
  0 = No hay sincronización.
  Cuando TCS=0.
  Este bit será ignorado.

- TCS: Selección de la fuente de reloj.
  1 = Reloj externo aplicado a la patilla TxCK.

| 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 |

Figura 4.76: Registro TxCON.

0 = Reloj interno(Fosc/4).

### 4.9.3. Temporizador Tipo B.

Los temporizadores 2 y 4 si están presentes son de este tipo. Características:

- 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.

El diagrama de bloques de este tipo de temporizador es: En la siguiente figura se aprecia el

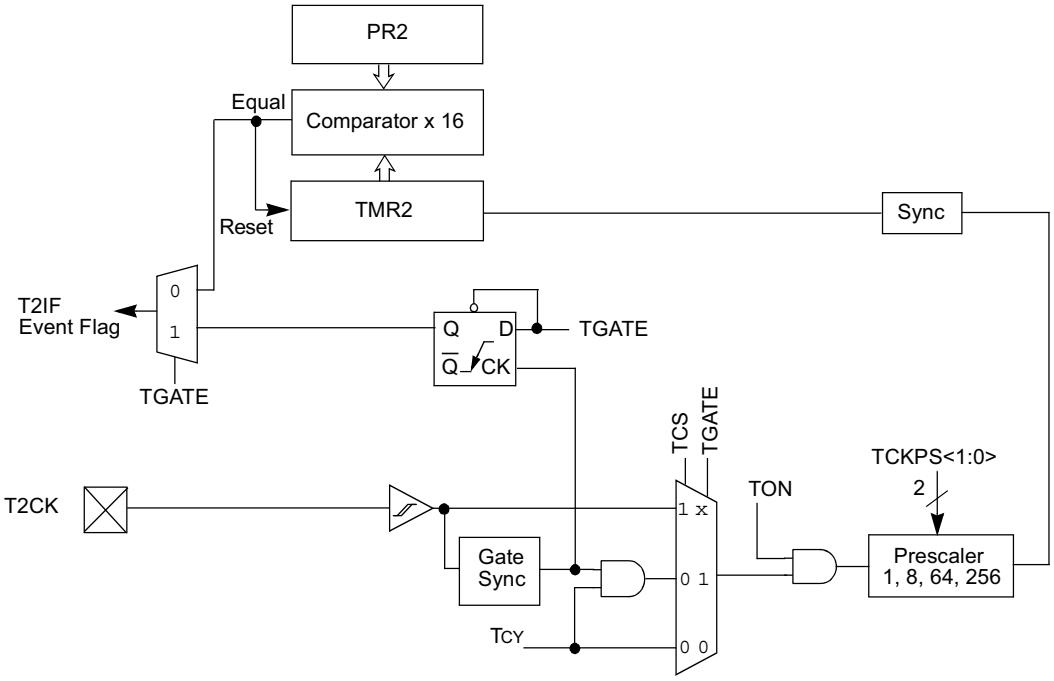



Figura 4.77: Diagrama de bloques del temporizador Tipo B.

| 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 |

Figura 4.78: Registro TxCON.

registro TxCON para este tipo de temporizadores: Los bits más significativos son:

- TON: Activación del temporizador.
  Cuando T32 = 1(Modo de Temporizador de 32 bits).
  1 = Temporizador de 32 bits activado.
  0 = Temporizador de 32 bits desactivado.
  Cuando T32 = 0(Modo de Temporizador de 16 bits).
  1 = Temporizador de 16 bits activado.
  0 = Temporizador de 16 bits desactivado.

- TSDIL: Detención del temporizador en modo Idle.
  1 = El temporizador se detiene en modo Idle.
  0 = EL temporizador continua en modo Idle.

- TGATE: Habilitación de modo de disparo por acumulación de tiempo.
  1 = Habilitado.
  0 = Deshabilitado.
  En caso en el que esté habilitado el bit TCS deberá tomar el valor 0 ya que en caso contrario TGATE será leído como 0 siempre.

- TCKPS[1:0]: Bit para el prescaler:
  11 = 1:256
  10 = 1:64
  01 = 1:8
  00 = 1:1

- T32: Selección del modo temporizador de 32 bits.
  1 = TMRx y TMry forman un temporizador de 32 bits.
  0 = TMRx y TMry temporizadores de 16 bits por separado.

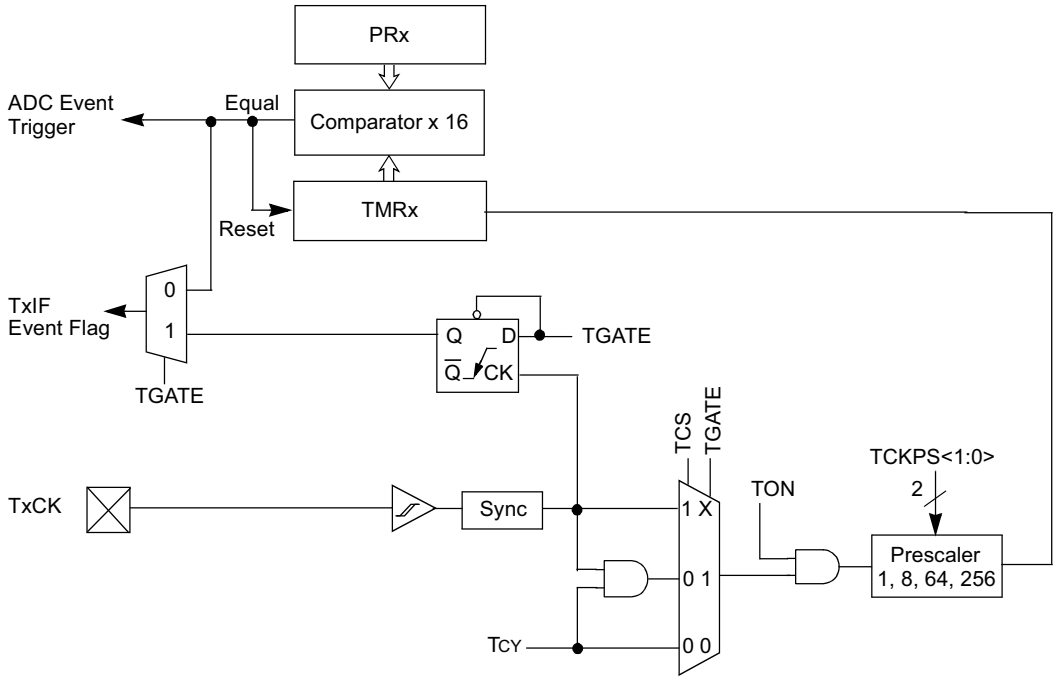
- TCS: Selección de la fuente de reloj.
  1 = Reloj externo aplicado a la patilla TxCK.
  0 = Reloj interno(Fosc/4).

### 4.9.4. Temporizador Tipo C.

El diagrama de bloques de este tipo de temporizador es: Los temporizadores 3 y 5 si están presentes son de este tipo. Características:

- Puede concatenarse con un temporizador tipo B para formar un temporizador de 32 bits.
- En un dispositivo dado, al menos un temporizador tipo C tiene la capacidad de disparo (trigger) en una conversión analógica/digital (A/D).

El diagrama de bloques se muestra en la figura posterior:




**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.

Figura 4.79: Diagrama de bloques del temporizador Tipo C.

| 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 |

Figura 4.80: Registro TxCON.

A continuación se puede ver el registro que controla este temporizador con los bits que lo forman:

- TON: Activación del temporizador.
  1 = Temporizador de 16 bits activado.
  0 = Temporizador de 16 bits desactivado.

- TSDIL: Detención del temporizador en modo Idle.
  1 = El temporizador se detiene en modo Idle.
  0 = EL temporizador continua en modo Idle.

- TGATE: Habilitación de modo de disparo por acumulación de tiempo.
  1 = Habilitado.
  0 = Deshabilitado.
  En caso en el que esté habilitado el bit TCS deberá tomar el valor 0 ya que en caso contrario TGATE será leído como 0 siempre.

- TCKPS[1:0]: Bit para el prescaler:
  11 = 1:256
  10 = 1:64
  01 = 1:8
  00 = 1:1

- TCS: Selección de la fuente de reloj
  1 = Reloj externo aplicado a la patilla TxCK.
  0 = Reloj interno(Fosc/4).

### 4.9.5. Modos de funcionamiento.

Los temporizadores tienen la posibilidad de funcionar en cuatro modos diferentes. 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.**
  Los tres tipos de temporizadores antes mencionados puedes trabajar en este modo. Para ello, la señal de reloj utilizada será la del reloj interno del sistema(Fosc/4). Una vez que el temporizador es incrementado se irá incrementando una unidad por cada ciclo de instrucción cuando la configuración del divisor de frecuencias de 1:1.

- **Modo contador síncrono.**
  En este modo pueden funcionar los tres tipos de temporizadores. TCs debe estar a 1, los incrementos del temporizador vienen dados por los pulsos en la patita TxCK.
  En el caso de los temporizadores tipo A además TSYNC deberá igualmente tomar el valor 1 para realzar la sincronización con la señal de reloj externa. En el caso de los otros temporizadores no será necesario hacer ningún tipo de sincronización.
  Para llevar a cabo la sincronización entre el reloj de instrucciones del dispositivo y la señal externa de reloj, se muestrea dicha señal externa en dos ocasiones por cada ciclo de instrucción.
  Un temporizador que este trabajando en modo contador síncrono no funcionará dentro del modo reposo(SLEEP) ya que los circuitos de sincronización se desconectan cuando el dispositivo funciona en dicho modo.

- **Modo contador asíncrono.**
  Este modo solo es posible en temporizadores tipo A. La señal de reloj utilizada será la misma que en el modo contador síncrono, es decir, la externa. En este caso el valor del bit TSYNC es 0 por lo que no se llevará a cabo la sincronización entre el reloj del dispositivo y la señal externa del reloj, de esta forma el temporizador se incrementa de manera asíncrona respecto al reloj interno del dispositivo.
  El incremento asíncrono presenta las ventajas de que el temporizador sigue funcionando cuando el dispositivo se encuentra en reposo y que se puede generar una interrupción que saque al dispositivo de dicho estado. Hay que mencionar que al operar el temporizador con el dispositivo en modo reposo pueden producirse resultados no deseados.
  En vez de tomar como señal externa los pulsos introducidos por la patita TxCK, se puede utilizar el LP (Low Power) u oscilador secundario de 32KHz para aplicaciones en tiempo

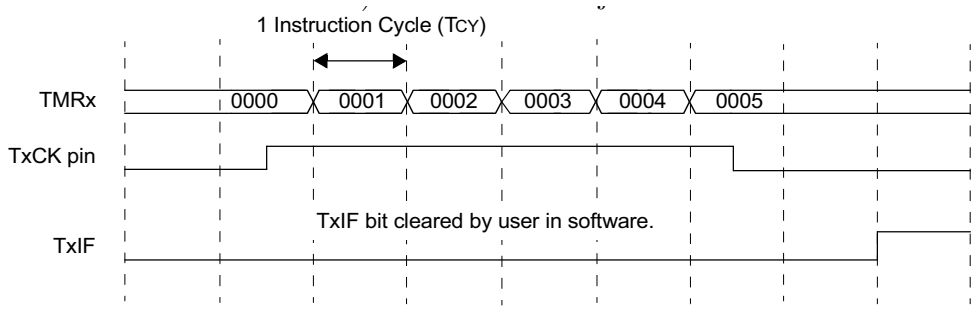



Figura 4.81: Cronograma del modo de contaje por disparo por acumulación de tiempo.

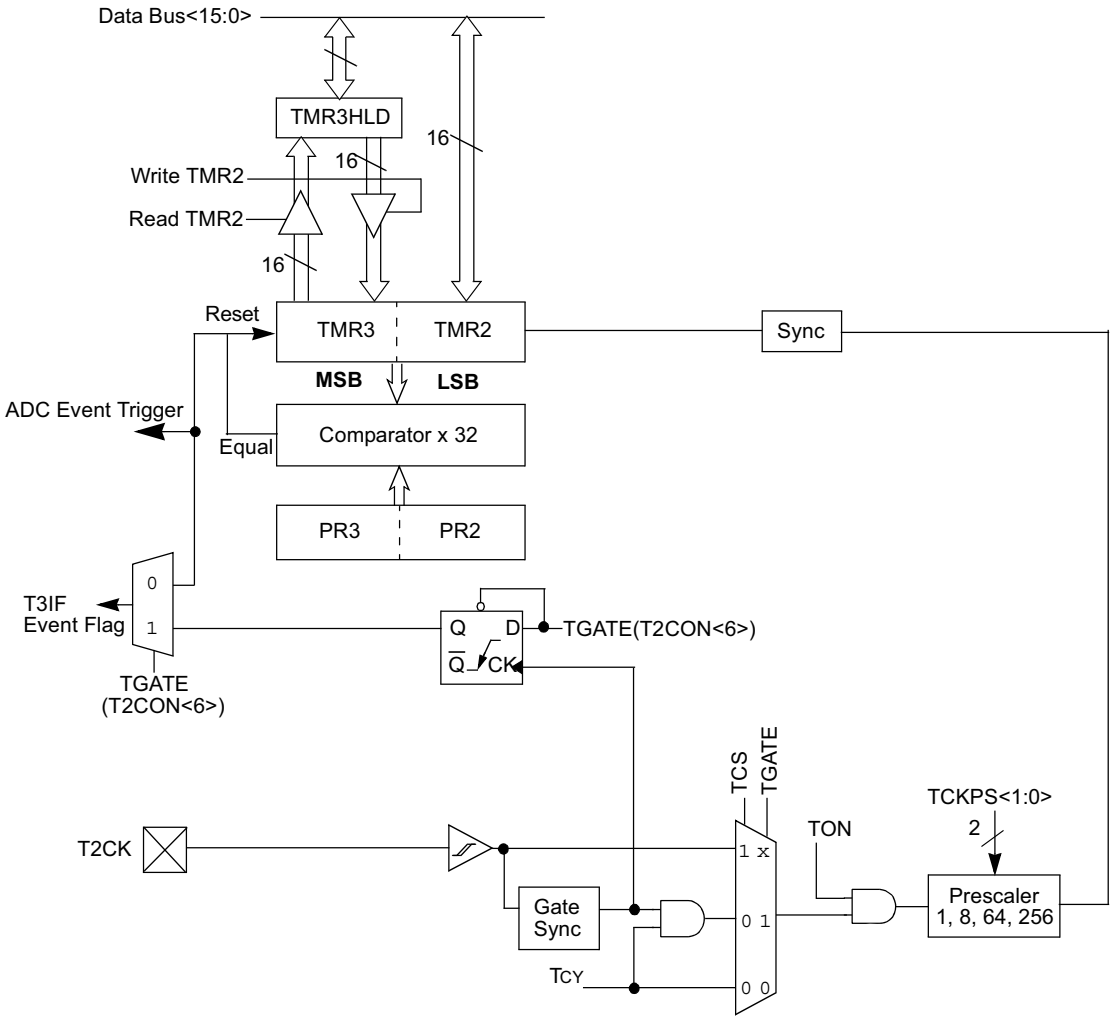
real.

- **Modo contaje por disparo por acumulación de tiempo "Gated Time Accumulation".**
  Este modo de funcionamiento permite que el registro contador del temporizador se incremente en función de la duración de los pulsos aplicados en la patita TxCK y utiliza la señal de reloj interna. La fuente de reloj del temporizador se deriva del reloj interno del sistema.
  El proceso comienza con un flanco ascendente en la patita TxCK y mientras el estado de la señal sea alto, el contador se incrementa hasta que se produzca una coincidencia de periodos o cambie el estado de la patita TxCK a bajo.
  Un cambio de estado de alto a bajo en la patita TxCK pondrá el flag TxIF a 1. Dependiendo del flanco en le que se produzca, el flag se testeará uno o dos ciclos de instrucción después del flanco descendente en la patita TxCK.
  Un flanco descendente detendrá el incremento del contador pero no pondrá a cero su valor. Si el usuario desea que que en el próximo flanco ascendente el contador comience desde 0, deberá borrar su valor. Además, un flanco descendente genera una interrupción. Para que este modo sea posible, el bit de control TGATE debe estar activo y el bit TCS a 0. Y por supuesto, el bit TON deberá ser habilitado para la puesta en marcha.

### 4.9.6. Temporizadores de 32 bits.

Para formar temporizadores de 32 bits se deben concatenar un temporizador tipo B y otro tipo C. Una vez combinados, los 16 bits del temporizador tipo C forman la parte alta del nuevo temporizador, mientras que los 16 bits correspondientes al temporizador de tipo B forman la parte baja.

Los bits de control que se encargan de gobernar el nuevo temporizador son los asociados al




**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.

Figura 4.82: Estructura interna del temporizador de 32 bits.

temporizador tipo B, por los que bits del registro TxCON del temporizador tipo C no tienen ningún efecto sobre el nuevo temporizador.

Para el control de las interrupciones se utilizan los bits (TxIE, TxIF y TxIP[2:0]) del temporizador tipo C.

Si se asume que el TIMER3 es de tipo C y el TIMER2 de tipo B, la configuración de un temporizador de 32 bits es la siguiente:

- TON=1.
- T32=1.
- TCKPS[1:0]=Bits para el prescaler de TMR2.

- T3IE: Bit empleado para la habilitación de las interrupciones del temporizador de 32 bits.
- T3IF: Flag de estado de interrupciones.
- T3IP[2:0]: Nivel de prioridad de interrupciones para el temporizador de 32 bits.

Los temporizadores de 32 bits pueden funcionar en tres modos diferentes:

- Temporizador síncrono.
- Contador síncrono.
- Contaje disparo por acumulación de tiempo.

Tanto el modo Contador síncrono como el Contaje disparo por acumulación de tiempo tienen el mismo funcionamiento que los temporizadores de 16 bits.

# Capítulo 5

# Periféricos de la familia dsPIC30F

## 5.1. Interfaz Periférica Serie (SPI).

### 5.1.1. Introducción.

La interfaz serie periférica (SPI) es un módulo de interfaz serie síncrono útil para comunicarse con otros periféricos o dispositivos microcontrolador. Estos dispositivos periféricos pueden ser memorias serie EEPROMs, registros de desplazamiento, controladores de pantalla, convertidores A/D, etc. El módulo SPI es compatible con las interfaces SPI y SIOP de Motorola.
Dependiendo de la variante, la familia dsPIC30F ofrece uno o dos módulos SPI, SPI1 y SPI2 que son funcionalmente idénticos. El módulo SPI2 está disponible en la mayoría de los dsPIC con un número alto de pines(64 o más), mientras que el módulo SPI1 está disponible en todos los dispositivos.

El puerto SPI consta de los siguientes Registros de Función Especial(SFR):

- SPIxBUF: Dirección en el espacio SFR que se utiliza para almacenar los datos que deben transmitirse y los datos que se reciben. Esta dirección es compartida por los registros SPIxTXB y SPIxRXB.
- SPIxCON: Registro de control que configura el módulo para los diversos modos de operación.
- SPIxSTAT: Registro de estado que indica el estado del módulo SPI.

Además, hay un registro de 16 bits registro de desplazamiento, SPIxSR, que no está mapeado en memoria. Se utiliza para transmitir y recibir los datos del puerto SPI. El registro mapeado en memoria SPIxBUF,almacena los datos recibidos o los que se van a transmitir. Internamente el registro SPIxBUF se compone de dos registros separados de 16 bits - SPIxTXB y SPIxRXB.

El buffer de datos recibidos SPIxRXB, y el buffer de transmisión SPIxTXB.
Si un usuario escribe datos, para transmitirlos, en el registro SPIxBUF, los datos internamente se escribe en el registro SPIxTXB. Del mismo modo, cuando el usuario lee los datos recibidos de SPIxBUF, los datos internamente se leen del registro SPIxRXB. Transmisión y recepción se producen simultáneamente.
El SPI consta de los siguientes cuatro pines el:

- SDIx: entrada serie de datos.
- SDOx: salida serie de los datos.
- SCKx: reloj de entrada o salida.
- SX: pulso de entrada o salida que activa el esclavo, a nivel bajo, o envía la trama de sincronización.

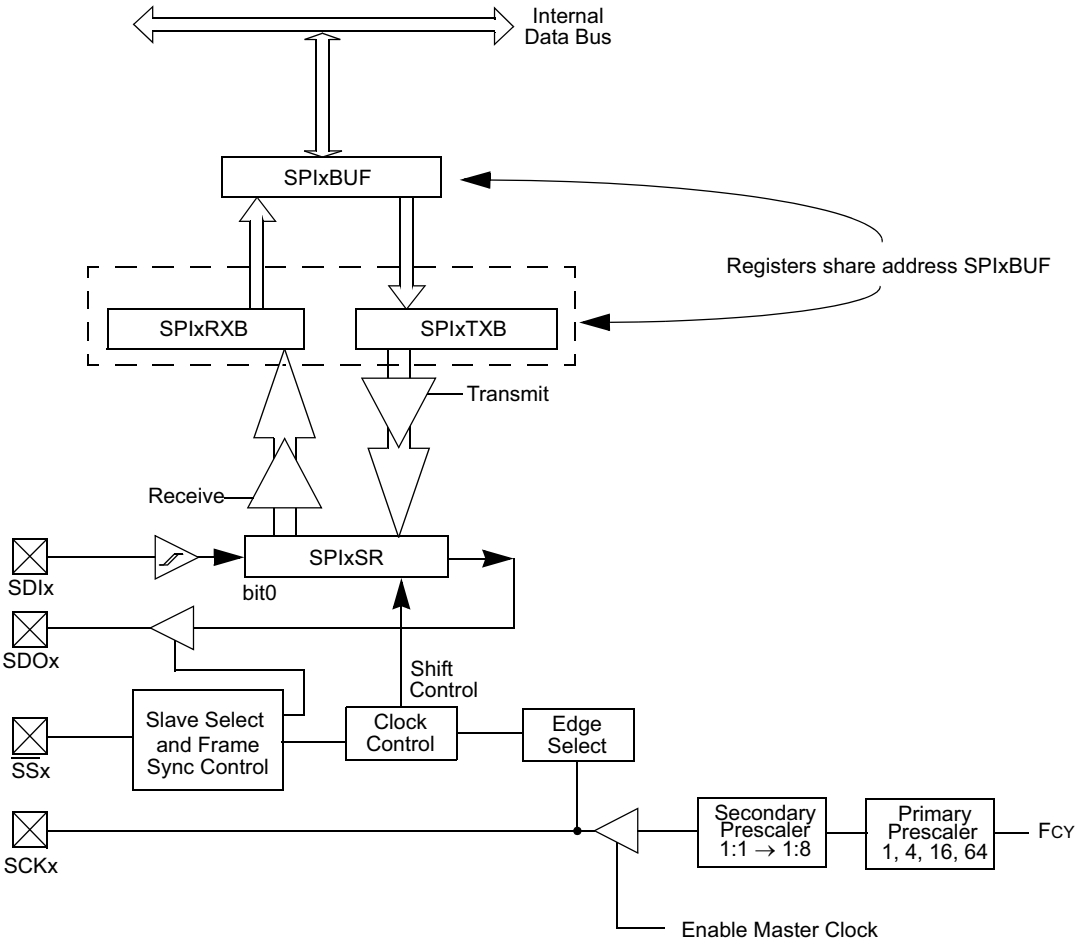



Figura 5.1: Diagrama de bloques del módulo SPI.

### 5.1.2. Registros de estado y de control.

**SPIxSTAT**

| **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 |

Figura 5.2: Registro de estado del módulo SPIx.

- SPIEN: bit de activación el módulo SPI:

  1 = Habilita el módulo SPI y configura los pines SCKx, SDOx, SDIx y SSX como patillas asociadas al SPI.

  0 = Desactiva el módulo.

- SPISIDL: Bit de stop en modo Idle:

  1 = El módulo deja de funcionar cuando el dispositivo entra en modo de reposo, Idle.

  0 = Continuar el funcionamiento del módulo en modo Idle.

- SPIROV: Bit de desbordamiento en la recepción:

  1 = Un nuevo byte / palabra recibido y descartado. El software del usuario no ha leído el dato anterior del registro SPIxBUF.

  0 = No se ha producido desbordamiento

- SPITBF: Bit de estado del buffer de transmisión del SPI:

  1 = La transmisión no se han iniciado todavía, está lleno el SPIxTXB.

  0 = la transmisión ha comenzado, SPIxTXB está vacío.

  Se pone automáticamente a uno cuando la CPU escribe en SPIxBUF (SPIxTXB) y se borra automáticamente cuando el módulo SPI transfiere el dato desde SPIxTXB al SPIxSR.

- SPIRBF: Bit de estado del buffer de recepción del SPI lleno:

1 = Recepción completa, SPIxRXB está lleno.

0 = Recepción no completa, SPIxRXB está vacío.

Se pone automáticamente a uno cuando el SPIx transfiere un dato desde SPIxSR a SPIxRXB y se borra automáticamente cuando el programa lee el registro SPIxBUF (SPIxRXB).

**SPIxCON**

| 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 |

Figura 5.3: Registro de control del módulo SPIx.

- FRMEN: bit de habilitación de trama de apoyo.

  1 = Trama de apoyo al SPI habilitada.

  0 = Trama de apoyo al SPI deshabilitada.

- SPIFSD: Bit de control de la dirección del pulso de sincronismo en SSx.

  1 = la trama de sincronismo es entrada (esclavo).

  0 = la trama de sincronismo es una salida (maestro).

- ISSDO: Desactivar el pin SDOx.

  1 = El pin SDOx no se utiliza por módulo SPI. El pin es controlada por los registros asociados al puerto.

  0 = El pin SDOx es controlado por el módulo SPI.

- MODE16: Selecciona el tamaño byte o palabra para la transmisión.

  1 = La comunicación es de palabras (16 bits).

0 = La comunicación es de byte (8 bits).

- SMP: SPI fase de muestreo de los bits de entrada.

  Modo maestro:

  1 = Los datos de entrada se muestrean al final del dato de salida.

  0 = Los datos de entrada se muestrean en el medio del dato de salida.

  Modo esclavo:

  Este bit debe borrarse al usar el SPI en modo esclavo.

- CKE: bit de selección del flanco del reloj del SPI.

  1 = Los datos de salida cambian en las transiciones de reloj de activo a reposo.

  0 = Los datos de salida cambian en las transiciones de reloj de reposo a activo.

  Nota: El bit CKE bit no se utiliza cuando FRMEN = 1, el usuario debe programar este bit a'0' en este caso.

- SEN: bit de selección de modo esclavo:

  1 = El pin SS es utilizado para el modo esclavo.

  0 = EL pin SS no utilizados por el módulo SPI.

- CKP: Bit de selección de la polaridad del reloj:

  1 = Estado de reposo del reloj es a nivel alto; estado activo es a nivel bajo.

  0 = Estado de reposo del reloj es a nivel bajo; estado activo es a nivel alto.

- MSTEN: Bit de habilitación del modo maestro.

  1 = Modo maestro.

  1 = modo esclavo.

- SPRE[2:0]: Bits que establecen la preescala secundaria (Modo Maestro): Desde 1:1 hasta 8:1.

  111 = Preescala secundaria 1:1.

  110 = Preescala secundaria 2:1.

  ...

000 = Preescala secundaria 8:1.

- PPRE¡1:0¿: Bit de selección del la preescala primaria (Modo Maestro):

  11 = Preescala primaria 1:1.

  10 = Preescala primaria 4:1.

  01 = Preescala primaria 16:1.

  01 = Preescala primaria 16:1.

### 5.1.3. Modos de operación.

El módulo SPI tiene modos de funcionamiento flexibles, que se dividen en las siguientes subsecciones:

- Datos de transmisión / recepción de 8 y 16 bits.
- Modos maestro y esclavo.
- Modos Framed SPI.

1. **Datos de 8 y 16 bits.**
   El bit de control MODE16 (SPIxCON [10]), selecciona al módulo para los modos de 8 bits o 16 bits. La funcionalidad será la misma para cada modalidad, excepto el número de bits que se reciben o transmiten. Por otra parte, cabe señalar lo siguiente en este contexto:

   - El módulo se reinicia cuando el valor del bit MODE16 (SPIxCON [10]) es cambiado. Consecuentemente, este bit no debe de ser cambiado mientras esté operando el SPI.
   - Los datos se transmiten en 7 bits del SPIxSR para el modo de operación de 8 bits, y 15 del SPIxSR para el modo de operación de 16 bits. En ambos modos, los datos se envían desde el bit 0 del SPIxSR.
   - Se requieren ocho pulsos de reloj en el pin SCKx para enviar/recibir los datos de 8 bits, mientras que se requieren 16 para los datos de 16 bits.

2. **Modos maestro y esclavo.**

   - **Modo Maestro.** Para configurar el módulo SPI en modo maestro hay que seguir los siguientes pasos:

     - Si se usan las interrupciones:
       - Borrar el flag SPIxIF.

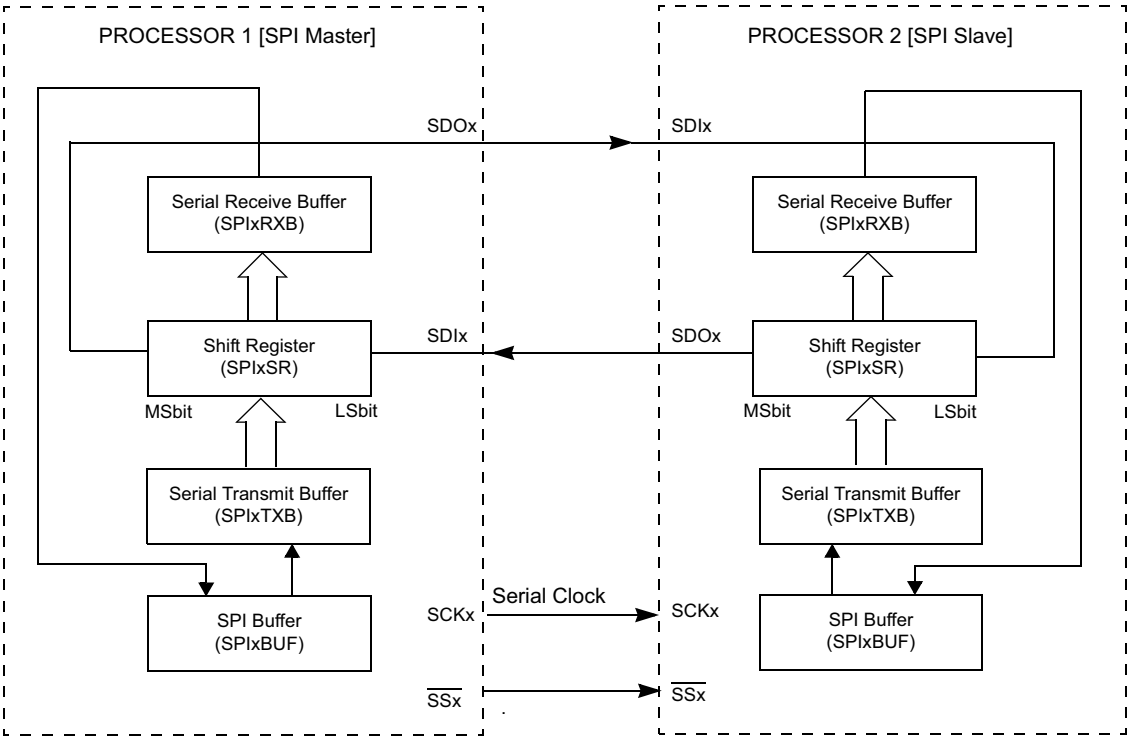



Figura 5.4: Conexión SPI maestro/esclavo.

- Activar el bit SPIxIE, que habilita las interrupciones.
- Asignar la prioridad de la interrupción del SPI, escribiendo en los bits SPIx-IP.

- Activar el modo maestro, MSTEN (SPIxCON [5]) = 1.
- Borrar el bit SPIROV (SPIxSTAT [6]).
- Habilitar el SPI mediante la activación del bit SPIEN (SPIxSTAT [15]).
- Escribir los datos a transmitir en el registro SPIxBUF, la transmisión empezará cuando se escriba el dato en este registro.

En modo Master, el reloj del sistema es preescalado y se usa como reloj de la transmisión serie. El preescalado se configura con los bits PTRES y SPRE. El reloj es producido a través del SCKx hacia el dispositivo esclavo. El reloj solo se genera cuando se transmiten los datos.

Los bits CKP y CKE determinan en que flanco del reloj se transmiten los datos. Los datos trasmitidos se escriben en el registro SPIxBUF y los recibidos se leen en el mismo registro.

En los siguientes pasos se describe la operación del módulo SPI como maestro:

- Una vez que el módulo esté configurado para el modo de funcionamiento maestro

y esté activado, los datos que se quieren transmitir se escriben en el registro SPIxBUF. El bit SPITBF (SPIxSTAT [1]) se activa.

- El contenido de SPIxTXB se pasa al registro de desplazamiento SPIxSR y el bit SPITBF se borra.
- Una serie de 8 / 16 pulsos de reloj transmiten los 8 / 16 bits del dato del registro SPIxSR al pin SDOx y, al mismo tiempo los datos del pin SDIx se escriben en el registro SPIxSR.
- Una vez completada la transferencia, ocurren los siguientes eventos:
  - El plan de interrupción se activa, SPIxIF y se producen una interrupción en el caso de que estén habilitadas (SPIxIE=1). El flag SPIxIF se tiene que borrar por software.
  - El contenido del registro SPIxSR se traslada al registro SPIxRXB cuando la recepción termina.
  - El bit SPIRBF (SPIxSTAT [0]) se activa indicando que el buffer de recepción está lleno, este bit se borra automáticamente al leer el registro SPIxRXB.
- Si el bit SPIRBF está a uno cuando el módulo SPI transfiere los datos del registro SPIxSR al SPIxRXB, el bit SPIROV (SPIxSTAT [6]) se activará indicando la condición de desbordamiento.
- Los datos que han de transmitirse pueden escribirse en SPIxBUF en cualquier momento, siempre que el bit SPITBF sea cero. La escritura puede ocurrir mientras el registro de desplazamiento transmite los datos previamente por escrito, lo que permite la transmisión continua.

- **Modo esclavo.**

  Para configurar el módulo SPI en modo ESCLAVO hay que seguir los siguientes pasos:

  - Borrar el registro SPIxBUF.
  - Si se quieren usar las interrupciones:
    - Borrar el bit SPIxIF.
    - Activar el bit SPIxIE.
    - Asignar la prioridad de la interrupción del SPI, escribiendo en los bits SPIx-IP.
  - Deshabilitar el modo maestro, con lo que habilitas el modo esclavo con MSTEN (SPIxCON [5]) = 0.

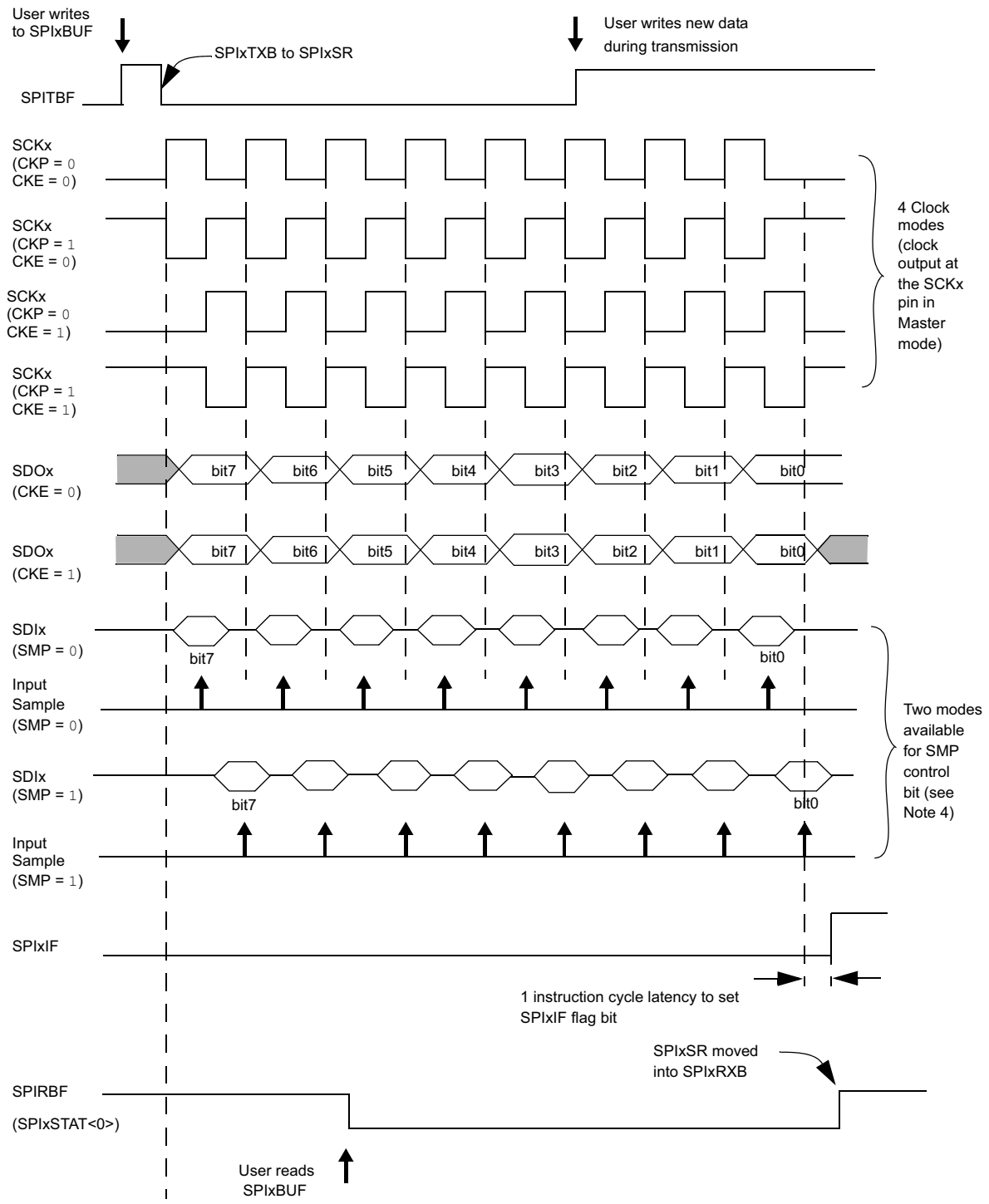



Figura 5.5: Operación en modo maestro.

- Si el bit CKE rd 1, entonces el bit SSEN debe activarse, habilitando así el pin SSx.
- Borrar el bit SMP.
- Borrar el bit SPIROV (SPIxSTAT [6]) y,

- Habilitar el SPI mediante la activación del bit SPIEN (SPIxSTAT [15]).

En modo Esclavo, los datos se transmiten y reciben en los pulsos de un reloj externo aplicado en el pin SCKx . Los bits CKP (SPIxCON [6]) y CKE (SPIxCON [8]) determinan en que flanco del reloj se produce la transmisión de datos. Los datos deben transmitirse y los datos que se reciben son escritos o leídos del registro SPIxBUF. El resto del funcionamiento del módulo es idéntico al de modo Master. Unas características adicionales en el modo esclavo son los siguientes:

**Selección de sincronización del esclavo**: El pin SSX permite el sincronismo en modo esclavo. Si el bit SSEN (SPIxCON [7]) está activado la recepción y la transmisión en modo esclavo sólo se habilita si el pin SSx está a nivel bajo. El pin SSx debe estar funcionando como entrada cuando se trabaja en este modo. Si el bit SSEN está activado y el pin SSx está a nivel alto el pin SDOx no transmite los datos y estará en el estado triestado. Una nueva transmisión comenzara cuando el pin SSx pase a nivel bajo transmitiendo el dato almacenado en el registro SPIcTXB. Si el bit SSEN no está activado el pin SSX no afecta a la operación del SPI en modo esclavo.

**Flag del estado de operación del SPITBF:** La función del bit SPITBF (SPIxSTAT [1]) es algo diferente en el modo de funcionamiento de esclavo. Los siguientes pasos describen la función del SPITBF para diversas configuraciones de operación del modo esclavo:

- Si el bit SSEN (SPIxCON [7]) es cero, el SPITBF se activa cuando se escribe en el registro SPIxBUF. Se borra cuando el módulo transfiere el valor del registro SPIxTXB al SPIxSR.
- Si el bit SSEN (SPIxCON [7]) está activado, el SPITBF se activa cuando el registro SPIxBUF se escribe por el usuario. Sin embargo, se borra únicamente cuando el módulo de SPIx completa la transmisión de datos.

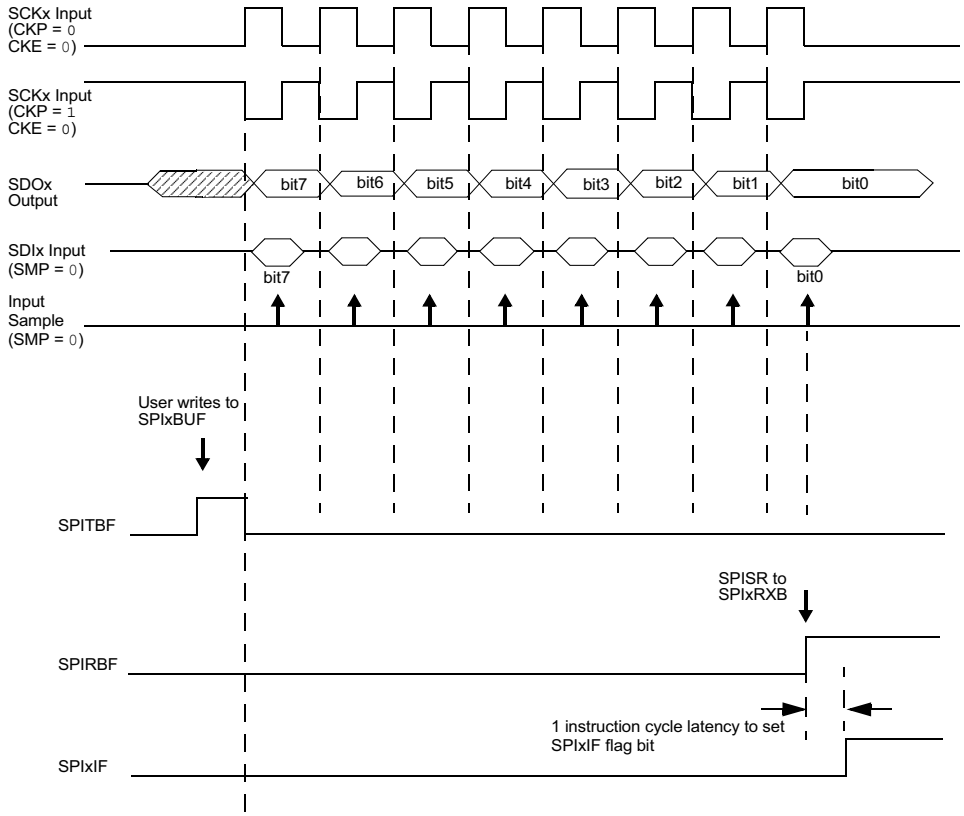



Figura 5.6: Operación en modo esclavo con selección de esclavo desactivado.

3. **Modo trama del SPI.**

El módulo soporta un protocolo SPI de trama tanto operando en modo maestro, como en modo esclavo. Las siguientes funciones se ofrecen en el módulo SPI para apoyar el modo de trama:

- El bit de control, FRMEN (SPIxCON [14]), habilita el modo de trama del SPI y las causas que el pin SSx sea utilizado para enviar la trama de sincronización. El estado del bit SSEN (SPIxCON [7]) se ignora.
- El bit de control SPIFSD (SPIxCON [13]), determina si el pin SSx es entrada o salida (es decir, si el módulo recibe o genera el pulso de sincronismo).
- La trama de sincronización es un pulso a nivel alto durante un ciclo del reloj del SPI.

El módulo SPI cuenta con los siguientes dos modos de trama del SPI:

- Trama en modo maestro: El módulo SPI genera el pulso de sincronización y proporciona este pulso a otros dispositivos en el pin SSx.

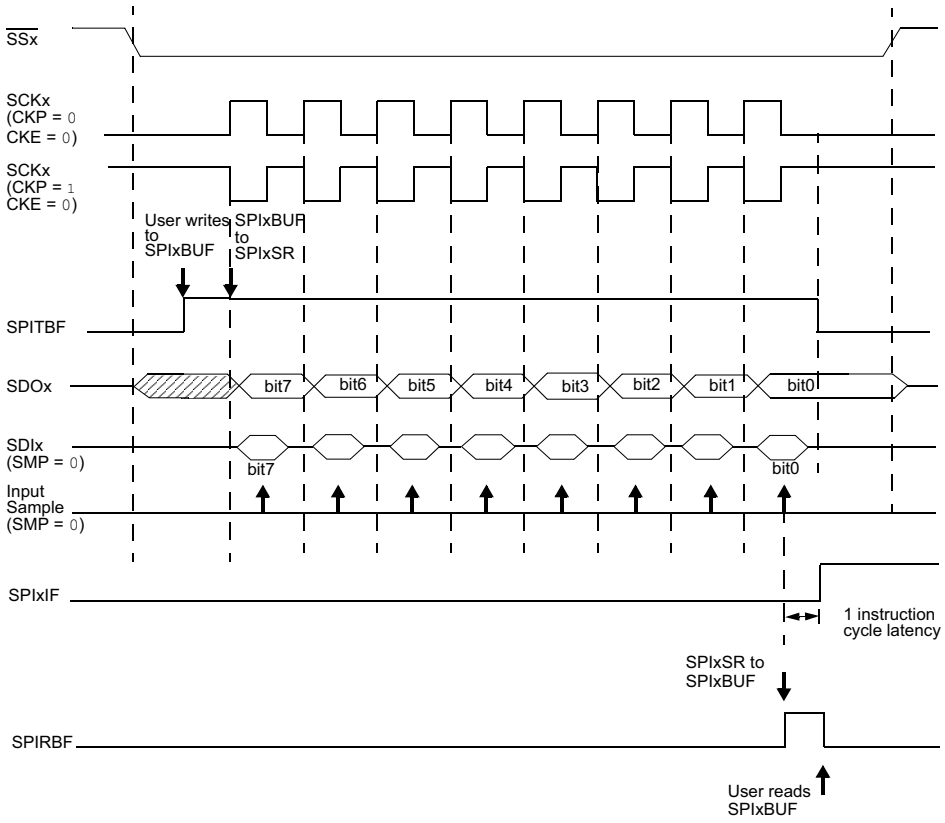



Figura 5.7: Operación en modo esclavo con selección de esclavo activado.

- Trama en modo Esclavo: El módulo SPI utiliza la trama de sincronización recibida por el pin SSx.

El modo de trama del SPI es soportado tanto en modo maestro y como en esclavo. Así, las siguientes cuatro configuraciones están disponibles para el usuario:

- SPI en modo maestro y trama en modo maestro.
- SPI en modo maestro y trama en modo esclavo.
- SPI en modo esclavo y trama en modo maestro.
- SPI en modo esclavo y trama en modo esclavo.

Estos cuatro modos de determinar si el módulo SPIx genera el reloj serie y la trama de sincronización.

### 5.1.4. Frecuencia de reloj del SPI.

En el modo maestro, el reloj del módulo SPI viene del ciclo de instrucción ($T_{CY}$).

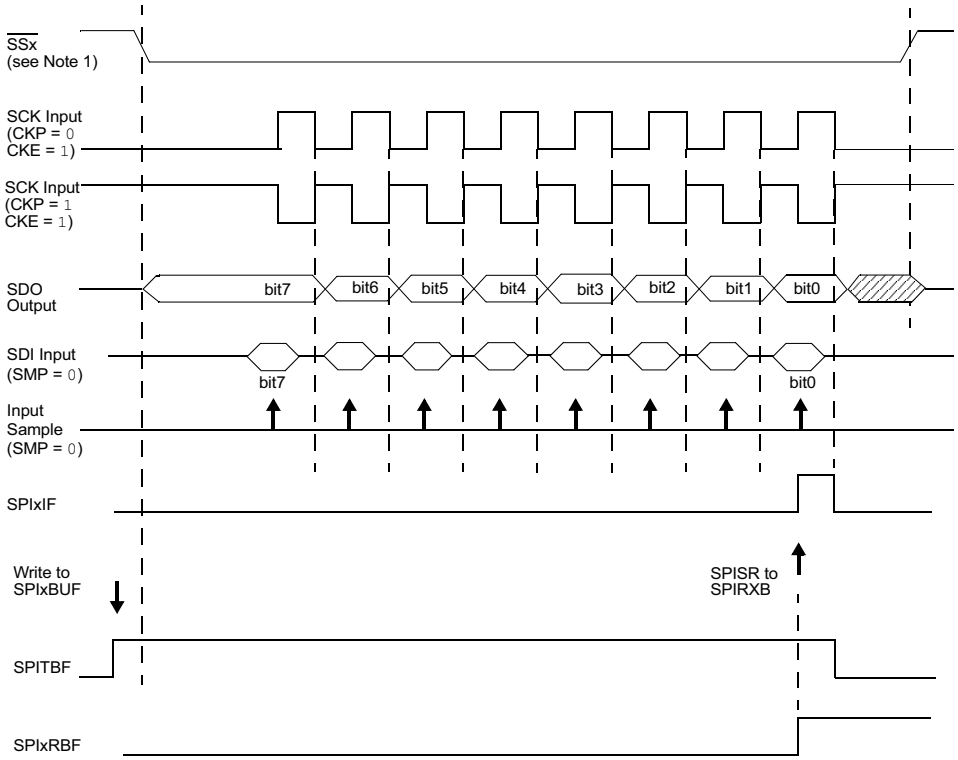



Figura 5.8: Operación en modo esclavo cuando CKE = 1.

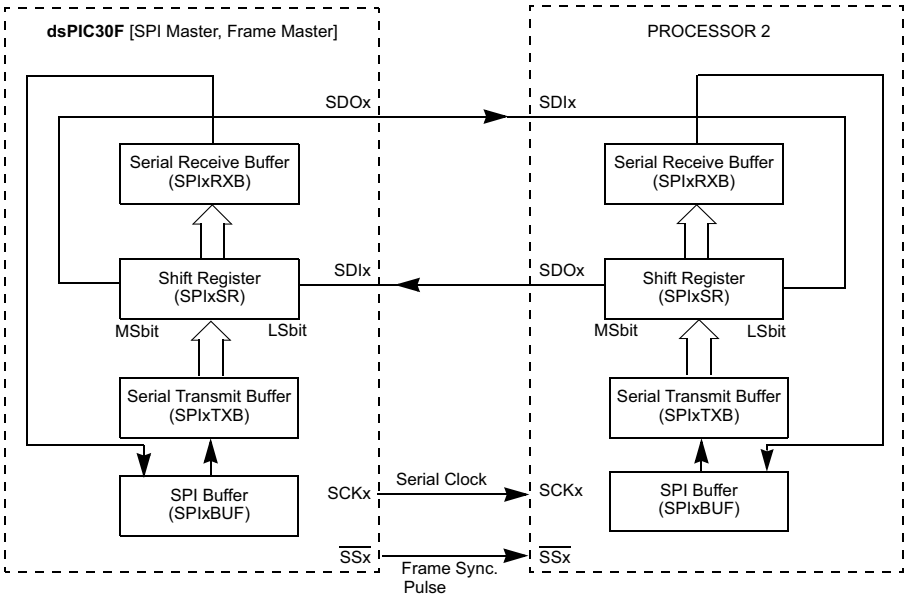



Figura 5.9: Diagrama de conexión SPI en modo maestro y maestro de trama.

A continuación se escala con la primera preescala (especificado por PPRE [1:0] (SPIxCON [1:0])), y después con la secundaria preescala (especificado por SPRE [2:0] (SPIxCON [4:2])).

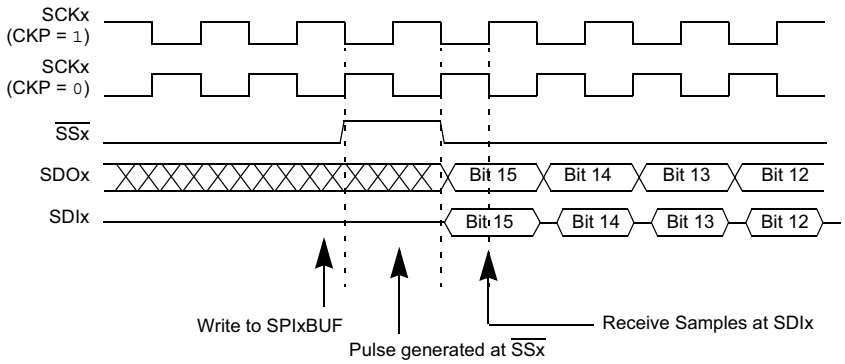



Figura 5.10: Diagrama temporal SPI en modo maestro y maestro de trama.

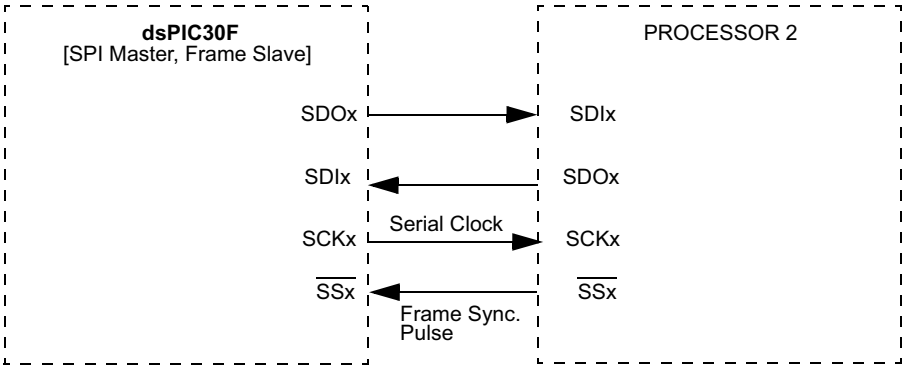



Figura 5.11: Diagrama de conexión SPI en modo maestro y esclavo de trama.

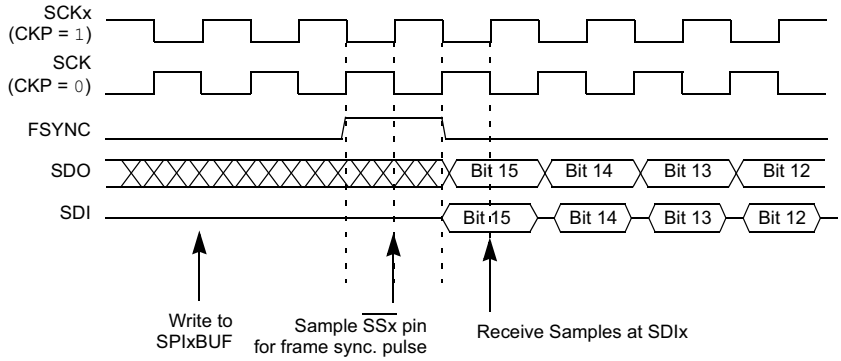



Figura 5.12: Diagrama temporal SPI en modo maestro y esclavo de trama.

Así se obtiene el reloj serie que ofrece el maestro a los dispositivos externos a través del pin SCKx.

En la tabla mostrada a continuación se ven un ejemplo de frecuencias de reloj SPI.

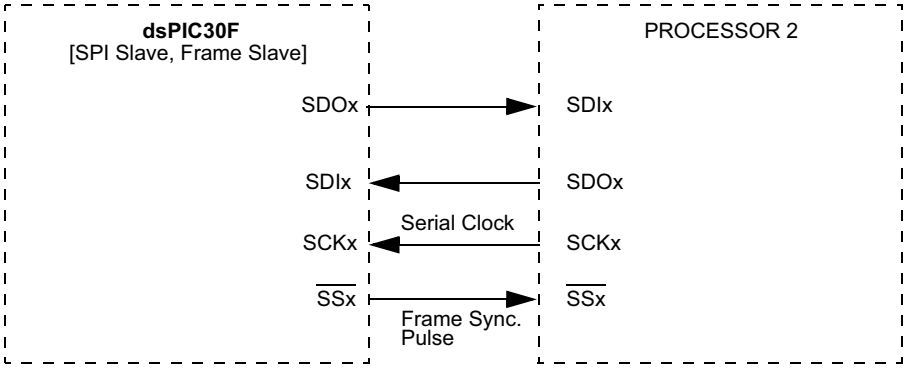



Figura 5.13: Diagrama de conexión SPI en modo esclavo y maestro de trama.

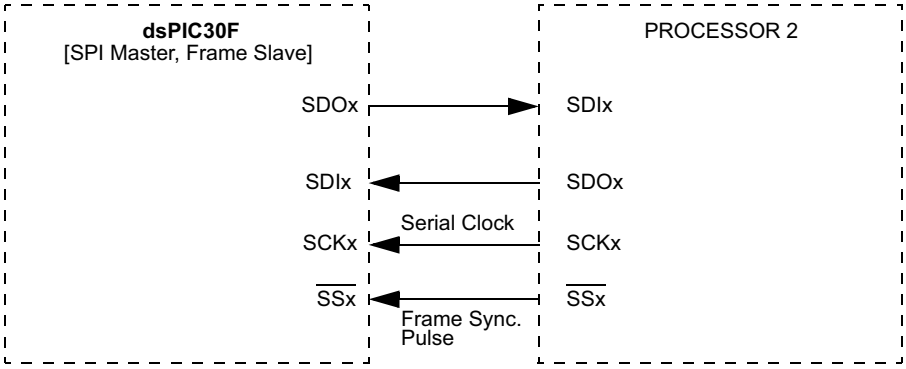



Figura 5.14: Diagrama de conexión SPI en modo esclavo y esclavo de trama.

$$F_{SCK} = \frac{F_{CY}}{\text{Primary Prescaler} \bullet \text{Secondary Prescaler}}$$

| **FCY = 30 MHz** | | **Secondary Prescaler Settings** | | | | |
|---|---|---|---|---|---|---|
| | | 1:1 | 2:1 | 4:1 | 6:1 | 8:1 |
| Primary Prescaler Settings | 1:1 | 30000 | 15000 | 7500 | 5000 | 3750 |
| | 4:1 | 7500 | 3750 | 1875 | 1250 | 938 |
| | 16:1 | 1875 | 938 | 469 | 313 | 234 |
| | 64:1 | 469 | 234 | 117 | 78 | 59 |
| **FCY = 5 MHz** | | | | | | |
| Primary Prescaler Settings | 1:1 | 5000 | 2500 | 1250 | 833 | 625 |
| | 4:1 | 1250 | 625 | 313 | 208 | 156 |
| | 16:1 | 313 | 156 | 78 | 52 | 39 |
| | 64:1 | 78 | 39 | 20 | 13 | 10 |

## 5.2. Módulo $I^2C$.

### 5.2.1. Introducción.

En la siguiente figura se puede apreciar el diagrama de bloques de este módulo. El módulo $I^2C$ se utiliza en las comunicaciones síncronas de tipo serie bien sea con microcontroladores o con periféricos, tales como memorias EEPROM serie, convertidores A/D, displays, registro de desplazamiento, etc. Este módulo puede funcionar de diversas formas:

- Interfaz $I^2C$ para comunicaciones, maestro-esclavo.
- Modo esclavo $I^2C$ con direcciones de 7 a 10 bits.
- Modo maestro $I^2C$ con direcciones de 7 a 10 bits.
- Puerto $I^2C$ bidireccional para sistemas maestro-esclavo.
- Control de la comunicación serie mediante el puerto $I^2C$ para sincronización serie de reloj.
- Operaciones en sistemas multi-maestro. Pueden detectar colisiones de bus y controlar el arbitraje de acceso al mismo.
- Control del Slew Rate para velocidades del bus de 100KHz y 400Khz.

### 5.2.2. Características de funcionamiento.

El hardware implementa todas las funciones del maestro y del esclavo así como el direccionamiento de 7 bits y 10 bits. De esta forma, un módulo $I^2C$ puede funcionar como esclavo o maestro en un mismo bus.
El bus I2C es una interfaz serie de dos hilos. En la siguiente figura se puede ver un esquema de una conexión típica $I^2C$ entre el dispositivo dsPIC30F y una EEPROM 24LC256. La interfaz I2C emplea un protocolo para garantizar una transmisión y recepción fiable de datos. En una comunicación, un dispositivo es el "maestro"que inicia la transferencia en el bus y genera las señales de reloj para permitir la transferencia, mientras que el otro dispositivo actúa como el .esclavo.en respuesta a la transferencia. La línea de reloj, "SCL", es la salida del maestro y entrada para el esclavo, aunque de vez en cuando el esclavo es el motor de la línea SCL. La línea de datos, "SDA", puede ser de salida y entrada de ambos.
Debido a que las líneas SDA y SCL son líneas bidireccionales,SDA y SCL deben tener drenador abierto con el fin de realizar el cableado-AND en el bus.Se utilizan resistencias pull-up para asegurar el nivel alto cuando el dispositivo no está tirando la linea abajo. En el protocolo de

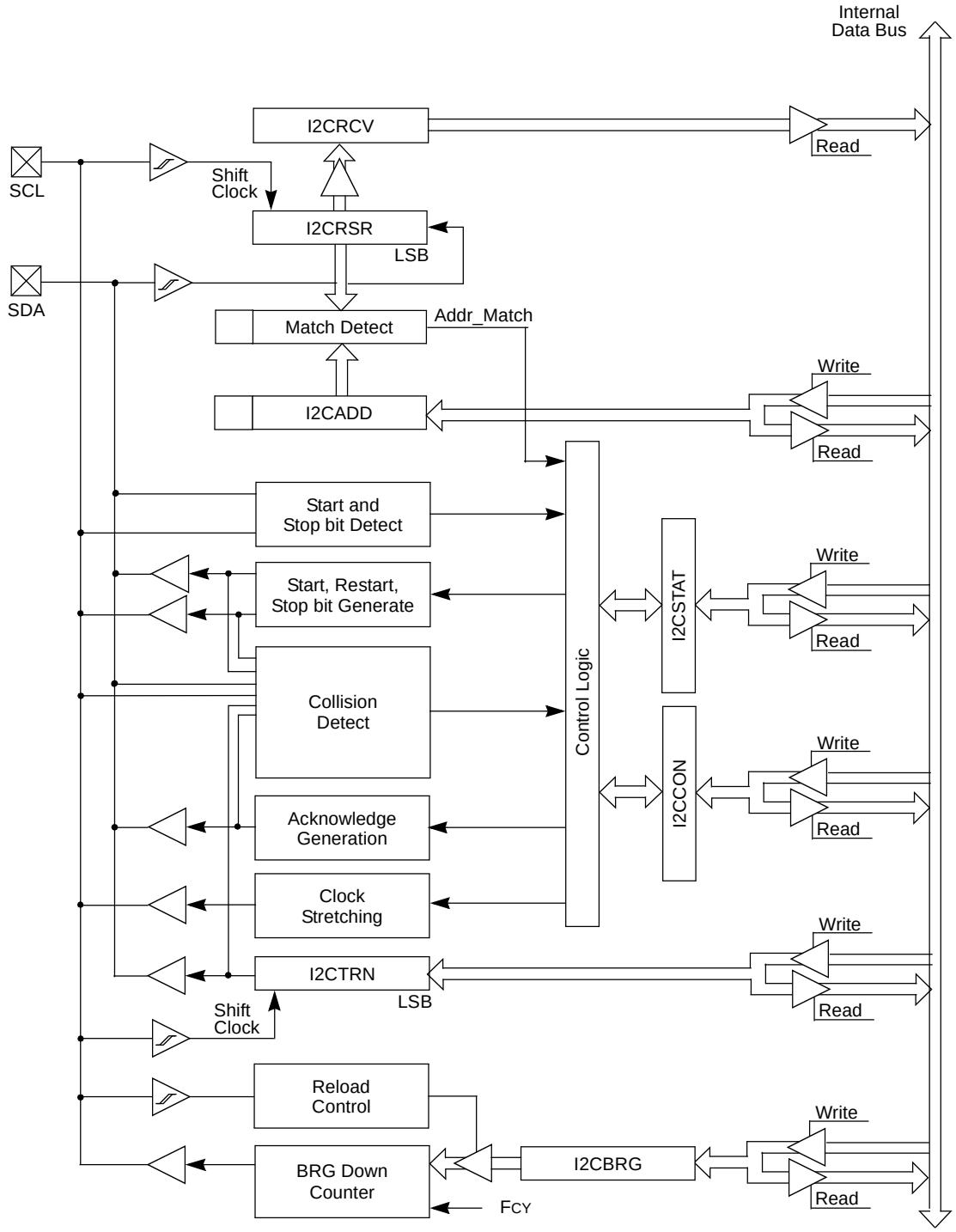



Figura 5.15: Diagrama de bloques del módulo $I^2C$.

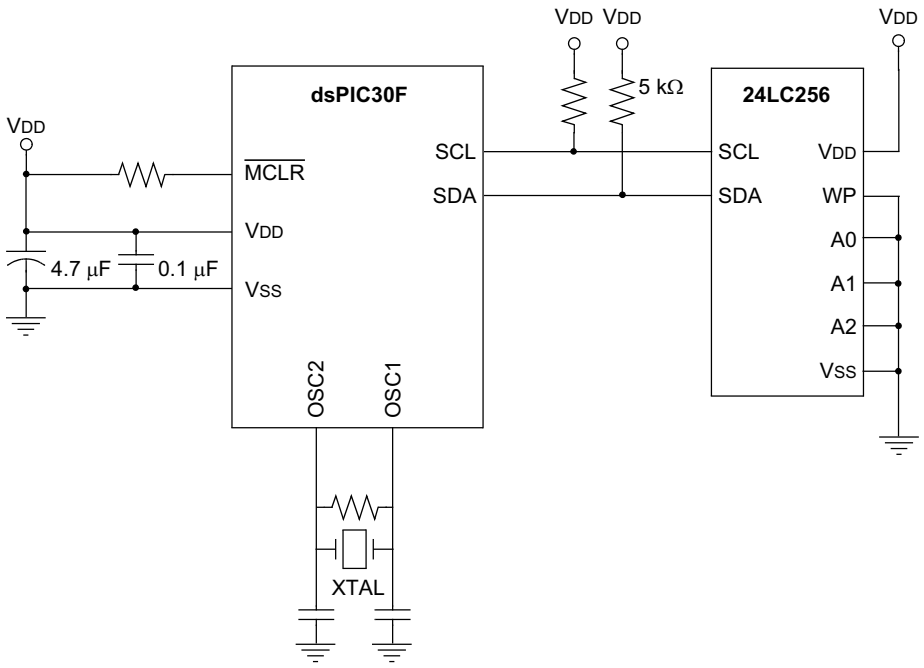



Figura 5.16: Diagrama de bloques de interconexión típica del módulo $I^2C$.

interfaz I2C, cada dispositivo tiene una dirección. Cuando un maestro desea iniciar un transferencia de datos, si es la primera vez transmite la dirección del dispositivo con el que se va a comunicar. Todos los dispositivos escuchan para ver si esa es su dirección. Dentro de esta dirección, el bit'0 'especifica si el maestro desea leer o escribir en el dispositivo esclavo. El maestro y el esclavo están siempre frente a los modos (transmisor / receptor) de operación durante una transferencia de datos. Es decir, puede ser pensado como que operan en cualquiera de estas dos relaciones:

- Maestro-Esclavo y transmisor-receptor.
- Esclavo-transmisor y receptor-Maestro.

En ambos casos, el maestro origina la señal de reloj SCL.

### 5.2.3. Protocolo $I^2C$.

En la siguiente gráfica se puede apreciar el protocolo $I^2C$.

Por defecto las dos líneas, la de datos y la de reloj están en alta, entonces cuando el maestro inicia una transmisión con uno de los esclavos primero envía la condición de start que consiste en bajar SDA, todos los datos deben estar precedidos por la condición de start y seguidamente

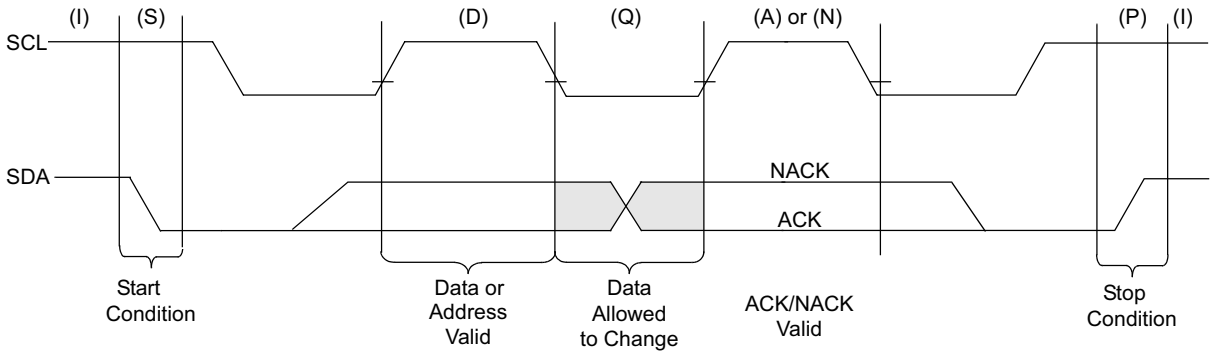



Figura 5.17: Protocolo $I^2C$.

se manda la dirección del esclavo, al ser la primera transmisión, a la vez se está transmitiendo el reloj, se considera dato válido en el ciclo positivo del reloj, un bit por cada ciclo de reloj.Los datos deben ser cambiados durante el periodo de baja de la señal de reloj.

Todos los bytes transmitidos tienen que ser reconocidos ACK o no reconocidos NACK por parte del receptor, este pondrá la linea SDA baja para un ACK o alta para un NACK.

Después de un estado WAIT, una transición de alta a baja de la linea SDA mientras que el reloj está en alta determina la condición repetida de inicio, que permite al maestro cambiar la dirección de bus sin perder el control del bus.

Una transición de baja a alta en la linea SDA cuando la linea SCL está en alta es la condición de stop. Todas las transferencias de datos deben de finalizar con la condición de stop.

Ambas líneas, SDA y SCL pasan a alta después de la condición de Stop y antes de la condición de Start.

Posteriormente se muestra el diagrama en la lectura de una EEPROM a modo de ejemplo: Si

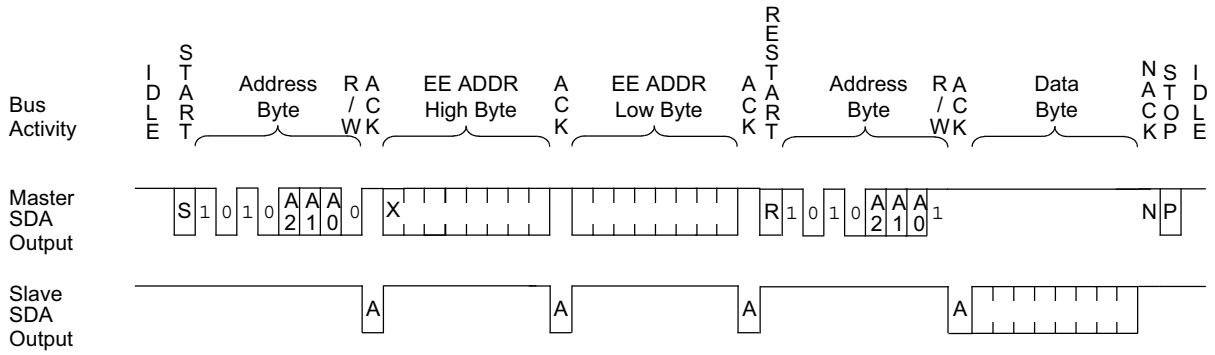



Figura 5.18: Lectura de una EEPROM con protocolo $I^2C$.

el software escribe en I2CTRN cuando la secuencia de Start está en progreso, IWCOL se activa y el contenido del buffer de transmisión es ignorado.

### 5.2.4. Registros de estado y de control.

El módulo I2C tiene seis registros de operación $I^2C$ accesibles por el usuario, como se muestran en la siguiente figura:

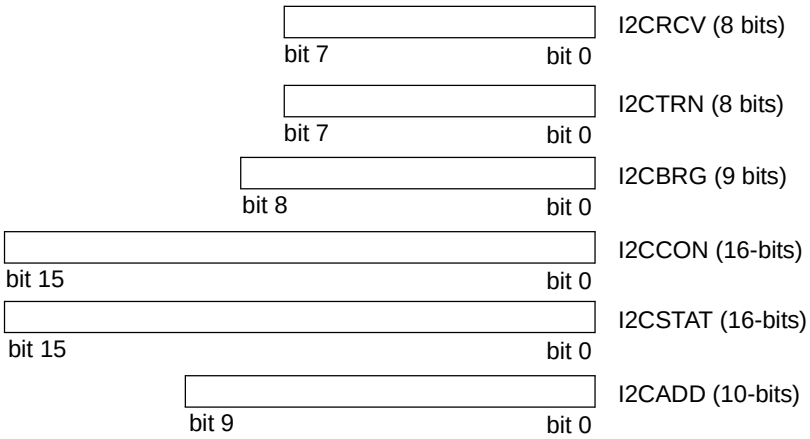



Figura 5.19: Registros del módulo $I^2C$.

- Control de Registro (I2CCON): Este registro permite el control de la operación I2C.

| **Upper Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/W-0 | U-0 | R/W-0 | R/W-1<br>HC | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| I2CEN | — | I2CSIDL | SCLREL | IPMIEN | A10M | DISSLW | SMEN |
| bit 15 | | | | | | | bit 8 |

| **Lower Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/W-0 | R/W-0 | R/W-0 | R/W-0<br>HC | R/W-0<br>HC | R/W-0<br>HC | R/W-0<br>HC | R/W-0<br>HC |
| GCEN | STREN | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| bit 7 | | | | | | | bit 0 |

Figura 5.20: Registro I2CCON.

- I2CEN: Bit de habilitación del módulo.
  1 = Habilita módulo $I^2C$ y configura los pins SDA y SCL como pins de puerto serie.
  0 = Deshabilita el modulo $I^2C$. Todos los pins del módulo $I^2C$ son controlados por las funciones del puerto.
- I2CSIDL: Configura el modo de operación en estado Idle.
  1 = Módulo de operación no continua en modo Idle.
  0 = Módulo de operación continua en modo Idle.

- SCLREL:Bit que controla SCL cuando opera en modo esclavo.
  1 = Libera SCL.
  0 = Mantiene SCL en baja.
- IPMIEN:Bit de habilitación IPMI (Intelligent Platform Management Interface)
  1 = Habilitación del modo de soporte IPMI. Todas las direcciones son reconocidas.
  0 = Modo IPMI no habilitado.
- A10M: Dirección del esclavo de 10 bits.
  1 = I2CADD guarda la dirección del esclavo de 10 bits.
  0 = I2CADD guarda la dirección del esclavo de 7 bits.
- DISSLW: Bit de control de la habilitación del Slew Rate.
  1 = Deshabilitado control.
  0 = Habilitado control.
- SMEN: Bit de niveles de entrada del SMBus.
  1 = Habilita umbrales de pin I / O compatible con la especificación SMBus.
  0 = Desactivar umbrales de entrada SMBus.
- GCEN: Bit de habilitación de llamadas generales cuando opera ene modo esclavo.
  1 = Habilita interrupción cuando una dirección de llamada general es recibida en I2CRSR.
  0 = Dirección de llamada general es deshabilitado.
- STREN: Bit de habilitación de tramo de reloj SCL. Cuando opera en modo esclavo. Utilizado junto al bit SCLREL.
  1 = Activar el software.
  0 = Inhabilitar el software.
- ACKDT :Bit de reconocimiento de datos. Cuando opera en modo maestro. Aplicable durante la recepción del maestro. Valor que será transmitido cuando el software inicia una secuencia de reconocimiento.
  1 = Enviar NACK durante reconocimiento.
  0 = Enviar ACK durante reconocimiento.
- ACKEN: Bit de habilitación de la secuencia de reconocimiento. Cuando opera en modo maestro aplicable durante la recepción.
  1 = Inicia la secuencia de reconocimiento en SDA y SCL, y transmite bit de dato ACKDT. Hardware lo borra al final de la secuencia de reconocimiento.
  0 = Secuencia de reconocimiento no está en progreso.

- RCEN: Bit de habilitación de recepción. Cuando opera como maestro.
  1 = Habilita recepción. Borrado por hardware al final del bit octavo del byte recibido por el maestro.
  0 = Secuencia de recepción no está en progreso.
- PEN: Bit de habilitación de la condición de STOP. Cuando opera en modo maestro.
  1 = Se inicia la condición de stop en los pins SDA y SCL. Borrado por hardware al final de la secuencia de stop.
  0 = Secuencia de stop no está en progreso.
- RSEN: Bit de habilitación de la condición de repetición de Start. Cuando opera en modo maestro.
  1 = Inicia la condición de repetición de start en las líneas SDA y SCL. Borrado por hardware al final de la secuencia.
  0 = Secuencia de repetición Start no está en progreso.
- SEN: Bit de habilitación de la condición de Start. Cuando opera en modo maestro.
  1 = Inicia la condición de Start en las líneas SDA y SCL. Borrado por hardware al final de la secuencia.
  0 = Condición de Start no está en progreso.

- Registro de estado (I2CSTAT): Este registro contiene la condición de flags que indican el estado del módulo $I^2C$ durante la operación.

| **Upper Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| R-0<br>HS, HC | R-0<br>HS, HC | U-0 | U-0 | U-0 | R/C-0<br>HS | R-0<br>HS, HC | R-0<br>HS, HC |
| ACKSTAT | TRSTAT | — | — | — | BCL | GCSTAT | ADD10 |
| bit 15 | | | | | | | bit 8 |

| **Lower Byte:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/C-0<br>HS | R/W-0<br>HS | R-0<br>HS, HC | R/C-0<br>HS, HC | R/C-0<br>HS, HC | R-0<br>HS, HC | R-0<br>HS, HC | R-0<br>HS, HC |
| IWCOL | I2COV | D_A | P | S | R_W | RBF | TBF |
| bit 7 | | | | | | | bit 0 |

Figura 5.21: Registro I2CSTAT.

- ACKSTAT: Bit de estado de reconocimiento. Cuando opera en modo maestro. Aplicable en la operación de transmisión de maestro.
  1 = NACK recibido del esclavo.
  0 = ACK recibido del esclavo.

- TRSTAT: Bit de estado de transmisión. Cuando opera como maestro. Aplicable durante la transmisión del maestro.
  1 = Transmisión del maestro está en progreso(8 bits + ACK).
  0 = Transmisión del maestro no está en progreso.
  Hardware lo activa al comienzo de la transmisión y lo borra al final del reconocimiento del esclavo.
- BCL: Bit de detección de colisión.
  1 = Colisión se ha detectado durante la operación del maestro.
  0 = No hay colisión.
  Hardware lo activa en una detección de colisión.
- GCSTAT: Bit de estado de llamada general.
  1 = Dirección de llamada general ha sido recibida.
  0 = Dirección de llamada general no ha sido recibida.
  Hardware lo borra después de la condición de Stop.
- ADD10: Bit de estado de la dirección de 10 bits.
  1 = dirección de 10 bits ha coincidido.
  0 = Dirección de 10 bits no ha coincidido.
  Se borra al final de la condición de Stop.
- IWCOL: Bit de detección de colisión de escritura.
  1 = Un intento de escribir en el registro I2CTRN ha fracasado porque el módulo I2C está ocupado.
  0 = No colisión.
  Activado por hardware con el bus está ocupado y borrado por software.
- I2COV: Indica desbordamiento.
  1 = Un byte se recibió I2CRCV mientras que el registro mantiene el byte anterior.
  0 = No desbordamiento Hardware lo activa en el intento de transferencia I2CRSR a I2CRCV (borrado por software).
- D-A: Bit Dato/Dirección cuando opera en modo esclavo.
  1 = Indica que el último byte recibido era dato.
  0 = Indica que el último byte recibido era la dirección de un dispositivo.
- P:Bit de Stop.
  1 = Indica que el bit de Stop se ha detectado.
  0 = Bit de Stop no ha sido detectado.
  El hardware lo borra cuando detecta condición de Stop, Start o Repeated Start.

- S: Bit de Start.
  1 = Indica que el bit de Start o de Repeated Start ha sido detectado.
  0 = El bit de Start no ha sido detectado.
  El hardware lo borra cuando detecta condición de Stop,Start o Repeated Start.
- R-W: Bit de información Lectura/Escritura. Cuando opera en modo esclavo.
  1 = Lectura- indica que el dato transferido es salida desde el esclavo.
  0 = Escritura- Indica que el dato transmitido es entrada del esclavo.
  El hardware lo borra después de la recepción del byte de dirección.
- RBF: Bit que indica que el buffer de recepción está lleno.
  1 = Recepción completa, I2CRCV está lleno.
  0 = Recepción no completa, I2CRCV está vacío.
  Hardware lo activa cuando I2CRCV es escrito con el byte recibido y lo borra cuando el software lee I2CRCV.
- TBF: Bit que indica que el buffer de transmisión está lleno.
  1 = Transmisión en progreso, I2CRTN está lleno.
  0 = Transmisión completa, I2CRTN está vacío.
  Hardware lo activa cuando el software escribe en I2CRTN y lo borra al final de la transmisión.

- Registro buffer de recepción (I2CRCV): Este es el registro buffer desde el cual los datos pueden ser leídos. Es un registro de solo lectura.
- Registro de transmisión (I2CTRN): Los bytes que se quieren transmitir se escriben en este registro. Es un registro de lectura y de escritura.
- Registro de dirección(I2CADD): Este registro contiene la dirección del dispositivo esclavo.
- Registro en el que se carga el valor del generador de baudios(I2CBRG):Contiene la velocidad de generador de baudios.

$$I2CBRG = (Fcy/Fscl - Fcy/1,111,111) - 1$$

### 5.2.5. Interrupciones $I^2C$

Este módulo genera dos interrupciones. Una interrupción es asignada para el maestro y la otra para el esclavo. Estas interrupciones activaran el correspondiente flag de interrupción e

| Required System FSCL | FCY | I2CBRG Decimal | I2CBRG HEX | Actual FSCL |
|---|---|---|---|---|
| 100 kHz | 30 MHz | 272 | 0x110 | 100 kHz |
| 100 kHz | 20 MHz | 181 | 0x0B5 | 100 kHz |
| 100 kHz | 1 MHz | 8 | 0x008 | 101 kHz |
| 400 kHz | 10 MHz | 15 | 0x00F | 400 kHz |
| 400 kHz | 5 MHz | 7 | 0x007 | 400 kHz |
| 400 kHz | 1 MHz | 1 | 0x001 | 345 kHz(1) |
| 1 MHz(2) | 20 MHz | 1 | 0x001 | 1 MHz(2) |
| 1 MHz | 1 MHz | 0 | 0x000 (invalid) | 1 MHz |

**Note 1:** This is the closest value to 400 kHz for this value of FCY.
**2:** FCY = 20 MHz is the minimum input clock frequency to have FSCL = 1 MHz.

Figura 5.22: Velocidades de reloj en módulo $I^2C$.

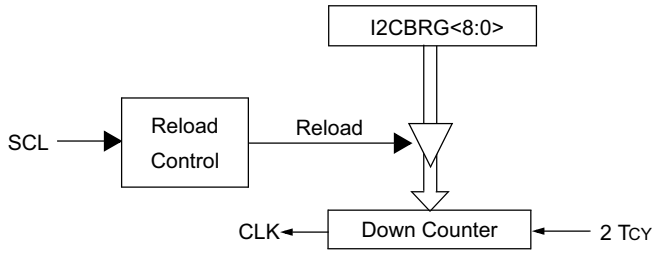



Figura 5.23: Diagrama de bloques del generador de baudios.

interrumpirán el proceso software si la correspondiente interrupción está habilitada y tiene una prioridad lo suficientemente alta. La interrupción del maestro se denomina MI2CIF y se activa en la finalización de una transmisión del maestro. Los siguientes eventos generar la interrupción MI2CIF.

- Condición Start.
- Condición Stop.
- Transferencia de bytes de datos transmitidos / recibidos.
- Transmisión de reconocimiento.
- Repetición de inicio.
- Detección de colisión en el bus.

El flag de interrupción del esclavo es SI2CIF y se activa en la detección de un mensaje dirigido al esclavo.

- Detección de una dirección válida dispositivo (incluyendo convocatoria general).
- Transmisión de datos byte.
- Recepción de datos byte.

## 5.3. Módulo UART

### 5.3.1. Introducción.

El UART (Transmisor Receptor Universal Asíncrono) es un módulo para la comunicación serie asíncrona disponible en los dsPIC30F. Funciona como un sistema de comunicación full-duplex o bidireccional asíncrono que puede adaptarse a magnitud de periféricos, como ordenadores personales RS-232 y RS-485. Principales características:

- Funcionamiento en modo full-duplex con datos de 8 o 9 bits, a través de los pines UxTX y UxRX.
- Posibilidad de trabajar con paridad par, impar o sin paridad.
- Uno o dos bits de STOP.
- Contiene un generador de baudios con una preescala de 16 bits que se encarga de generar la frecuencia de trabajo del módulo.
- Buffer de transmisión con capacidad para 4 caracteres.
- Buffer de recepción con capacidad de 4 caracteres.
- Posibilidad de emplear interrupciones para indicar la finalización de la transmisión o recepción.
- Pines específicos TX y RX para transmitir y recibir.
- Soporta direcciones de 9 bits.

Las transferencias de información se realizan sobre dos líneas UTx y URX, transmisión y recepción respectivamente, saliendo y entrando los bits por dichas líneas al ritmo de una frecuencia controlada internamente por el UART.
Cada palabra o dato de información se envía independientemente de los demás. Suele constar de 8 ó 9 bits y van precedidos por un bit de START y detrás de ellos se coloca un bit de STOP, de acuerdo con las normas del formato estándar NRZ.
El módulo UART está compuesto por tres grandes bloques:

1. Generador de baudios.
2. Transmisor asíncrono.
3. Receptor asíncrono.

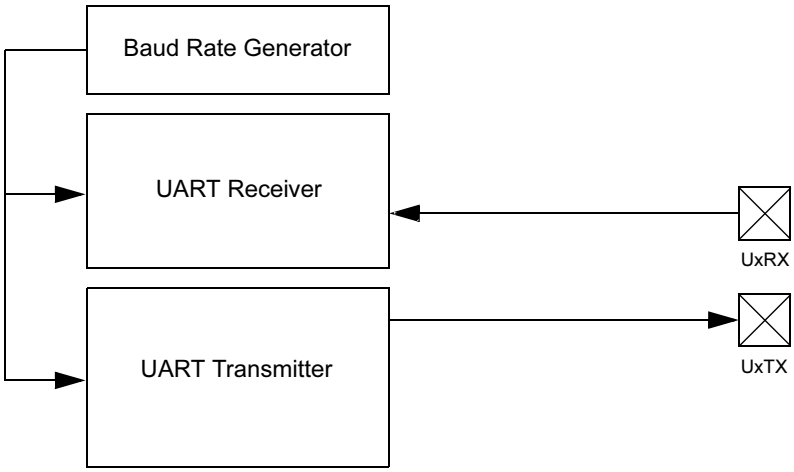



Figura 5.24: Diagrama de bloques del módulo UART.

### 5.3.2. Registros de control y estado del uart.

1. **UxSTA:**Es el registro de control del modo UART:

| **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 |

Figura 5.25: Registro de control del módulo UART.

- UTXISEL: Bit que permite interrumpir la transmisión(se activa con 1).

  1 = Interrupción cuando un carácter se transfiere al registro de desplazamiento de transmisión y como consecuencia, el buffer de transmisión se vacía.

  0 = Interrupción cuando un carácter se transfiere al registro de desplazamiento de transmisión (esto implica que hay por lo menos un carácter en el buffer de transmisión).

- UTXBRK: Bit que indica el modo de trabajo del módulo UART (con 0 transmisión normal).

  1 = UxTX pin está a nivel baja, independientemente del estado del transmisor.

0 = UxTX pin opera normalmente.

- UTXEN: Este bit permite o prohíbe la operación de transmisión.

  1 = El UART transmisor activado, el pin UxTX está controlado por el UART (si UARTEN = 1)

  0 = El UART transmisor deshabilitado, cualquier transmisión se aborta y se reinicia buffer. El pin UxTX está controlado por el puerto.

- UTXBF: Bit de lectura que indica si el buffer de transmisión está lleno o no.

  1 = Buffer de transmisión lleno.

  0 = Buffer de transmisión no está lleno, por lo menos una palabra más de datos se pueden escribir.

- TRMT: Bit de estado del registro de desplazamiento de transmisión

  1 = El registro de desplazamiento de transmisión está vacío y el buffer de transmisión también (la última transmisión ha terminado)

  0 = El registro de desplazamiento de transmisión no está vacío, hay una transmisión en curso o datos en el buffer de transmisión.

- URXISEL[1:0]: Bit de selección de interrupción de la recepción.

  11 = El flag de interrupción se establece cuando el buffer de recepción se llena (es decir, tiene 4 caracteres de datos).

  10 = El flag de interrupción se establece cuando un buffer de recepción está a 3 / 4 de su capacidad (es decir, tiene 3 caracteres de datos).

  0x = El flag de interrupción se establece cuando un carácter se recibe.

- ADDEN: Detección de dirección.

  1 = Modo de detección de dirección habilitado. Sólo se aplica en el modo de 9 bits.

  0 = Modo de detección de dirección deshabilitado.

- RIDLE: Bit de solo lectura que indica el estado de recepción.

  1 = El receptor está inactivo.

  0 = Los datos se está recibiendo.

- PERR: Bit de solo lectura que indica si se ha producido un error de paridad.

  1 = Error de paridad ha sido detectado para el dato actual.

  1 = Error de paridad no detectado.

- FERR: Bit de error de trama.

1 = Trama de error detectada para el dato actual.

0 = Trama de error no ha sido detectada para el dato actual.

- OERR: Bit de error de sobrepasamiento (Si está a 1 hay error).

  1 = Buffer de recepción desbordado.

  0 = Buffer de recepción no desbordado.

- URXDA: Bit de lectura que indica la disponibilidad o no de un dato en el buffer de recepción.

  1 = El buffer de recepción tiene datos disponibles para leer.

  0 = El buffer de recepción está vacío.

2. **UxMODE:** Registro para la configuración del módulo UART.

| **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 |

Figura 5.26: Registro de estado y control del módulo UART.

- UARTEN: bit que activa el UART.

  1 = El UART está habilitado. Los pines asociados el UART están controlados por UART, tal como se define por los bits de control UEN [1:0] y UTXEN.

  0 = El UART está desactivado. Los pines del UART están controlados por puerto correspondiente, LAT, y TRIS bits.

- USIDL: bit de parada en modo Idle.

  1 = Deje de funcionar cuando el dispositivo entra en modo Idle.

  0 = Continuar la operación en modo Idle.

- ALTIO: bit que alterna la selección de I/O del UART.

  1 = El UART se comunica usando los pines de entrada/salida UxATX y UxARX.

  0 = El UART se comunica usando los pines de entrada/salida UxTX y UxRX.

Nota: Las entradas/salidas suplentes del UART no están disponibles en todos los dispositivos. Ver la ficha de datos de dispositivo para más detalles.

- WAKE: Bit para habilitar el Wake-up cuando se detecte el bit de Start durante el modo Sleep.

  1 = Wake-up habilitado.

  0 = Wake-up deshabilitado.

- LPBACK: Bit de selección del modo loopback del UART.

  1 = Habilitar el modo de Loopback.

  0 = Deshabilitar el modo de Loopback.

- ABAUD: Bit que activa el Auto-Baudios.

  1 = El pin UxRX es entrada del módulo de captura de entrada.

  0 = El pin ICx es entrada del módulo de captura de entrada.

- PDSEL [1:0]: bits de selección de paridad y tamaño de datos.

  11 = 9 bits de datos, sin paridad.

  10 = 8 bits de datos, paridad impar.

  01 = 8 bits de datos, paridad par.

  00 = 8 bits de datos, sin paridad.

- STSEL: bit de selección de Stop.

  1 = 2 bits de parada.

  0 = 1 bit de parada.

3. **UxRXREG:** Es el registro de recepción.

| **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 |

Figura 5.27: Registro de recepción del módulo UART.

- URX8: Bit 8 del carácter recibido(en el modo de 9 bits).
- URX [7:0]: Bit del 0 al 7 del carácter recibido.

4. **UxTXREG:** Es el registro de transmisión.

| 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 |

Figura 5.28: Registro de transmisión del módulo UART.

- UTX8: Bit 8 del carácter transmitido(en el modo de 9 bits).
- UTX[7:0]: Bit del 0 al 7 del carácter transmitido.

5. **UxBRG:** Registro de 16 bits, dependiendo de su valor la transferencia se realiza a una frecuencia u otra.

| 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 |

Figura 5.29: Registro de configuración de los baudios del módulo UART.

### 5.3.3. Generador de baudios.

En el protocolo asíncrono RS-232 la frecuencia en baudios(bits por segundo) a la que se realiza la transferencia de información debe de tomar un valor de baudios normalizado(por

ejemplo 9600). Para generar una frecuencia el UART dispone de un generador de frecuencias en baudios, cuyo valor es controlado por el contenido del registro UBRG.

$$frecuenciaenbaudios = Fosc/(16(UBRG + 1))$$

$$UBRG = (Fosc/(16 * frecuenciaenbaudios)) - 1$$

### 5.3.4. Transmisor UART.

El diagrama de bloques del transmisor UART se muestra en la siguiente figura: El corazón

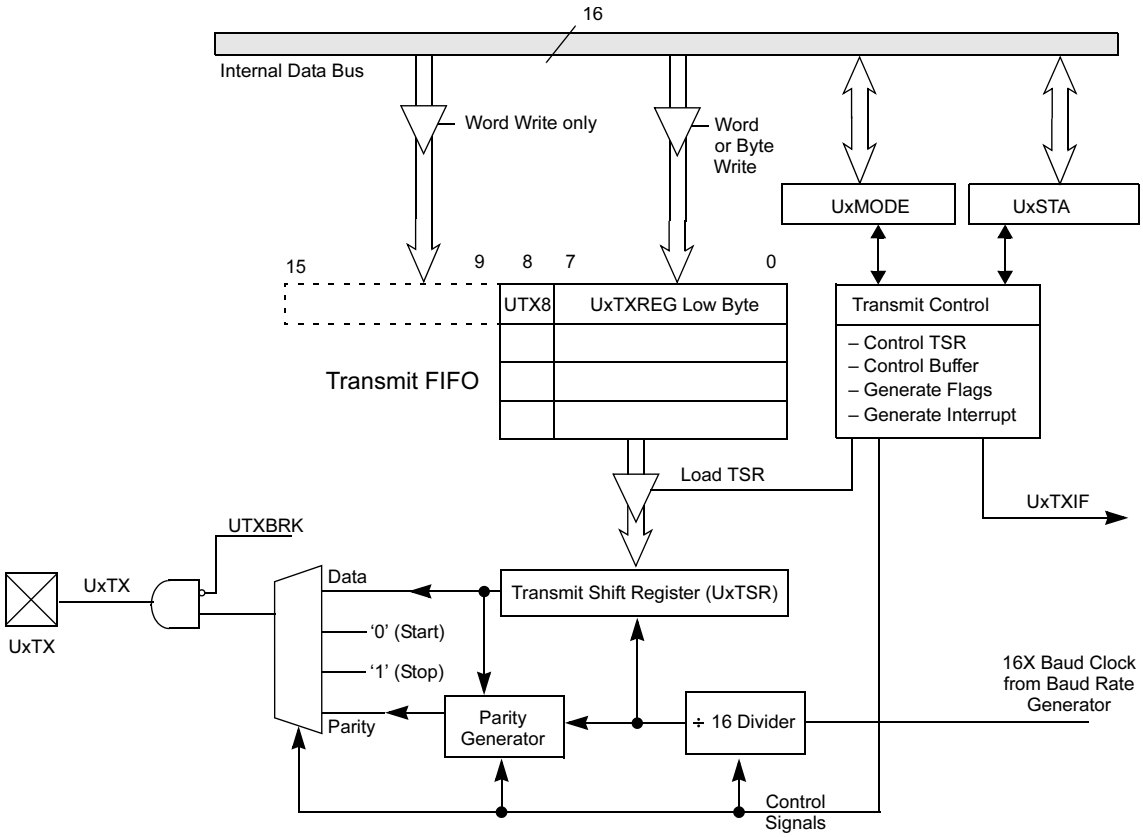



Figura 5.30: Diagrama de bloques de la sección de transmisión del UART.

del transmisor es el registro de desplazamiento ( UxTSR), que obtienen el dato del buffer FIFO UxTXREG, el que es cargado por software. El registro UxTSR no es cargado con el siguiente dato hasta que no se transmita el bit de Stop del dato actual. Tan pronto como el Bit de parada es transmitido, el UxTSR se carga con nuevos datos del registro UxTXREG (si están disponibles).

La transmisión se activa configurando el bit UTXEN (UxSTA [10]). La transmisión real no se produce hasta que el registro UxTXREG ha sido cargado con los datos y el generador de

la velocidad en baudios (UxBRG) ha producido un cambio de reloj. La transmisión también puede iniciarse cargando primero el registro UxTXREG y activando el bit UTXEN después. Normalmente, cuando la transmisión se activa primero, el registro UxTSR está vacío, por lo que una escritura en el registro UxTXREG da lugar a una inmediata transferencia a UxTSR. Borrando el bit UTXEN durante una transmisión provocará el aborto de la transmisión. Como resultado, los pines UxTX pasaran a alta impedancia.
Para seleccionar la trasmisión de datos de 9 bits, los bits PDSEL [1:0] bits (UxMODE [2:1]) deben fijarse a 11 y el noveno bit debe ser escrito en el UTX9 (UxTXREG [8]). Los nueve bits se tienen que escribir a la vez en UxTXREG.

El dato que se desea transmitir por el transmisor UART se escribe en el registro UxTXREG y a continuación se traspasa al registro de desplazamiento UxTSR, que va sacando los bits secuencialmente y a la frecuencia establecida. Además, antes de los bits de información existe un bit de START y después de todos los bits de datos se añaden uno o dos bits de STOP. El receptor UART recibe uno a uno los bits, elimina los dos de control y una vez que los bits de información han llenado al registro de desplazamiento UxRSR, se trasladan automáticamente al registro UxRXREG, donde quedan disponibles para su posterior procesamiento.

Pasos a seguir para realizar una transmisión:

- Iniciar el registro UxBRG con el valor adecuado para la tasa de transmisión. Para mas información mirar el funcionamiento del registro UxBRG.
- Establecer el número de bits de datos, el número de bits de parada y la paridad escribiendo en los bits PDSEL [1:0] (UxMODE [2:1]) y STSEL (UxMODE [0]).
- Si se desea que se produzca una interrupción activa el bit de control UxTXIE en su correspondiente registro de habilitación de interrupción (IEC), y se especifica la prioridad de la interrupción con los bits UxTXIP [2:0]. También, se selecciona el modo de interrupción de transmisión con los bits de control UTXISEL (UxSTA [15]).
- Habilitar el módulo UART UARTEN (UxMODE [15]) =1.
- Habilitar la transmisión UTXEN (UxSTA [10]) =1, que también activaré el bit UxTXIF. El bit UxTXIF debe borrarse por el software de la rutina de servicio a la interrupción de transmisión del UART. El funcionamiento del bit UxTXIF es controlado por el bit de control UTXISEL.

- Cargar los datos en el registro UxTXREG (se inicia la transmisión). Si se quieren transmitir 9 bits se cargará una palabra y si son 8 un byte. Los datos pueden ser cargados en el buffer hasta que el bit de estado UxTXBF bit (UxSTA [9]) sea 1.

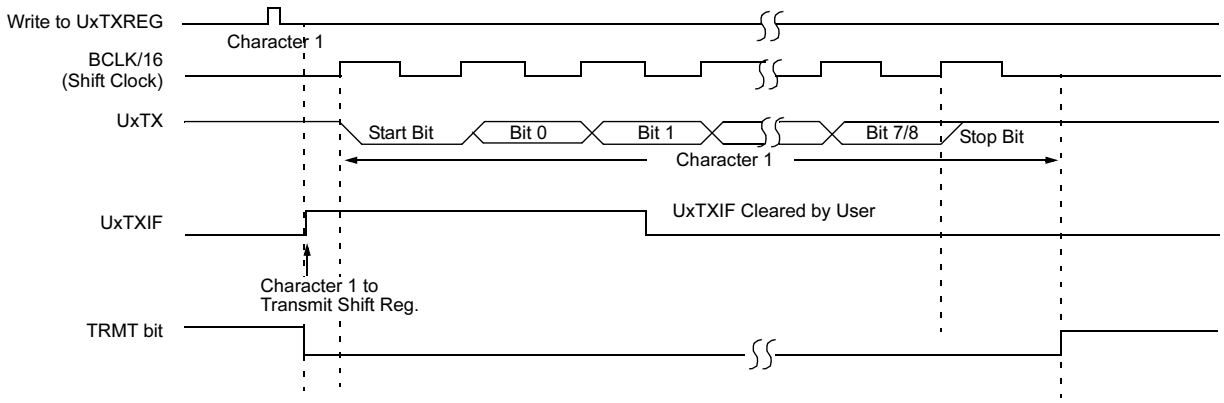



Figura 5.31: Transmisión de 8 o 9 bits.

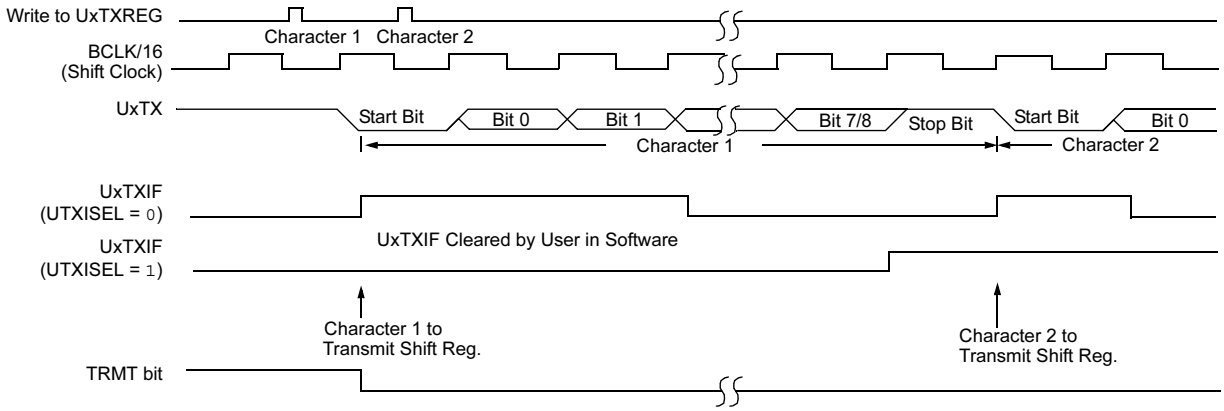



Figura 5.32: Transmisión back to back.

### 5.3.5. Receptor UART.

El receptor diagrama de bloques se muestra en siguiente circuito.

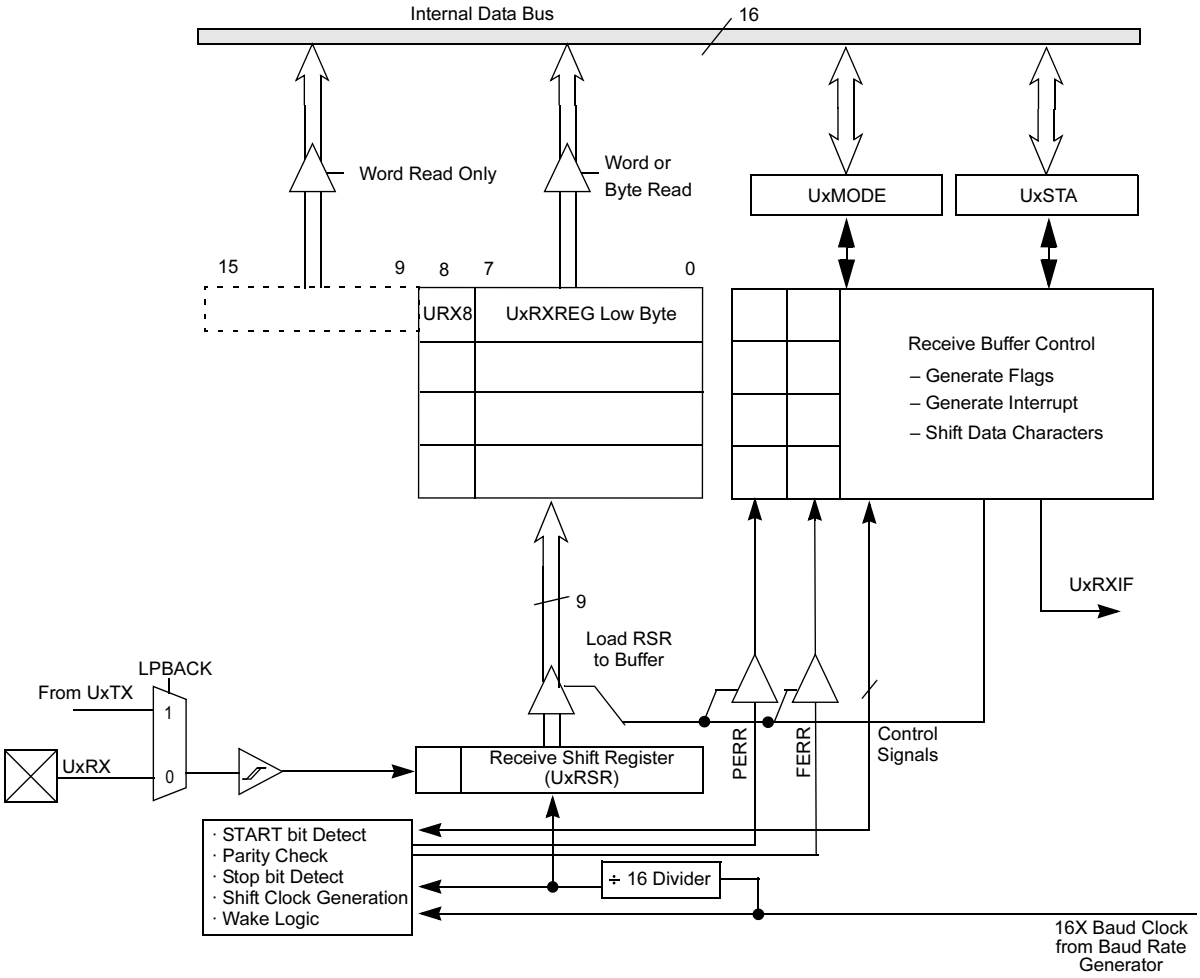



Figura 5.33: Diagrama de bloques de la sección de recepción del UART.

Los datos se reciben en serie bit a bit y se van introduciendo secuencialmente en el registro de desplazamiento UxRSR, que funciona a una frecuencia 16 veces más rápida que la de trabajo. Los datos en el pin UxRX se muestrean tres veces para decidir si es un uno o un dato que está presente en el pin de entrada UxRX.

Cuando un procesador maestro intenta enviar información a uno de los esclavos, primero envía un byte de dirección que identifica al destinatario. El byte de dirección se identifica porque el noveno bit que llega vale 1. Si el bit ADDEN=1, en el esclavo se ignoran todos los bytes de datos. Pero si el noveno bit que se recibe vale 1, quiere decir que se trata de una dirección y el esclavo provocará una interrupción y se transferirá el contenido del registro UxRSR al buffer de recepción. Tras la interrupción el esclavo deberá examinar la dirección y si coincide con la

suya debe poner ADDEN a 0 para poder recibir datos del maestro.
Si ADDEN vale 1, todos los datos son ignorados, el bit de STOP no se carga en el UxRSR, por lo que este hecho no produce interrupción.
Los pasos a seguir en el modo interrupción son los siguientes:

- Iniciar el registro UxBRG con el valor adecuado para la tasa de transmisión. Para mas información mirar el funcionamiento del registro UxBRG.
- Establecer el número de bits de datos, el número de bits de parada y la paridad escribiendo en los bits PDSEL [1:0] (UxMODE [2:1]) y STSEL (UxMODE [0]).
- Si se desea que se produzca una interrupciones activa el bit de control UxRXIE en su correspondiente registro de habilitación de interrupción (IEC), y se especifique la prioridad de la interrupción con los bits UxRXIP [2:0]. También, se selecciona el modo de interrupción de recepción con los bits de control URXISEL (UxSTA [7:6]).
- Habilitar el módulo UART UARTEN (UxMODE [15]) =1.
- Recibir interrupciones dependerá de la configuración empleada en los bits de control URXISEL [1:0]. Si las interrupciones no están habilitadas, el usuario puede consultar el bit URXDA para saber si ha recibido un dato. El bit UxRXIF debe borrarse por el software de la rutina de servicio a la interrupción de recepción del UART.
- Leer del buffer los datos recibidos. Si se reciben 9 bits se debe leer una palabra y si son 8 un byte. El estado del bit URXDA (UxSTA [0]) indicará cuando los datos recibidos están en el buffer.

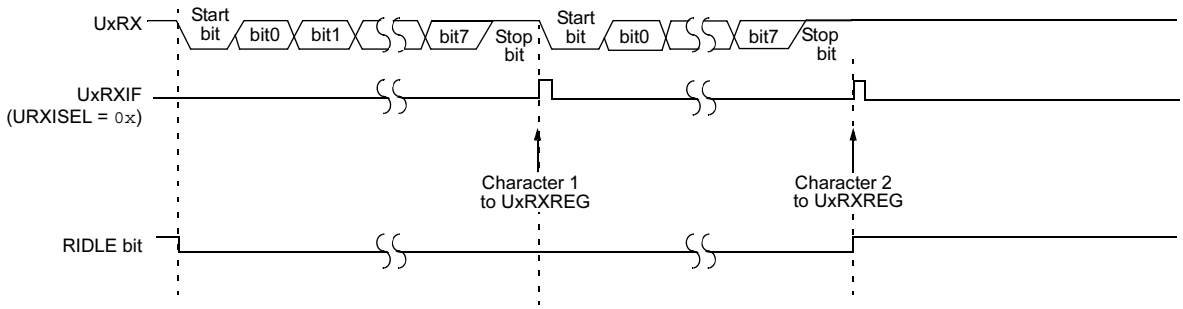



Figura 5.34: Recepción del UART.

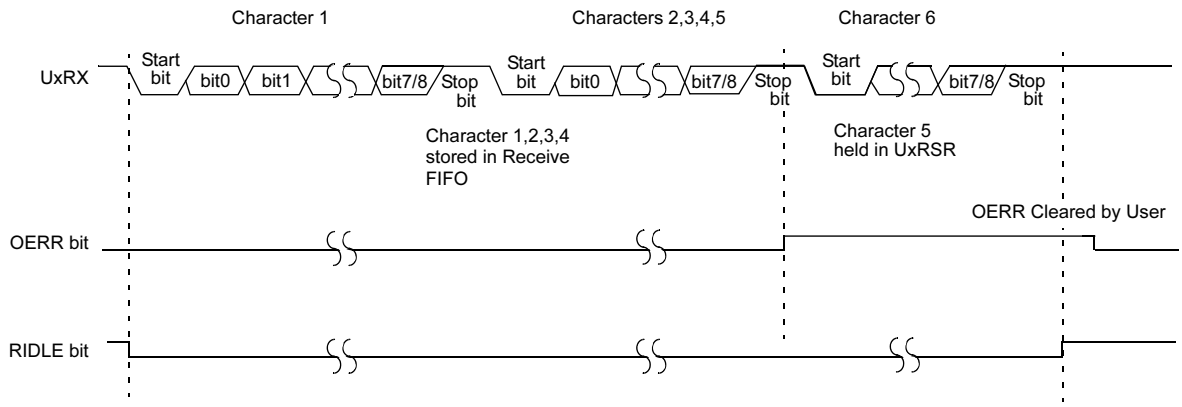



Figura 5.35: Recepción del UART con overrun.

## 5.4. Comparador de Salida.

### 5.4.1. Introducción.

El módulo de comparación de salida resulta muy útil en aplicaciones en las que se requiere trabajar en alguno de estos modos:

- Generación de pulsos de anchura variable.
- Operación simple mediante PWM.

El módulo de comparación de salida compara el valor de un temporizador con el de uno o dos registros, de forma que cuando éstos coincidan se activará un bit que solicita una interrupción. Además tiene la capacidad de generar interrupciones. Según el dispositivo de la familia dsPIC30F que se trate se puede disponer de hasta ocho canales de comparación de salida(OCx),en el caso del dsPIC 30f4013 tiene cuatro canales, que ofrecen las siguientes características operacionales:

- Selección de los temporizadores Timer2 y Timer3.
- Funcionamiento simple del módulo de comparación.
- Funcionamiento dual del módulo.
- Funcionamiento del módulo durante los estados Sleep e Idle de la CPU.
- Posibilidad degenerar interrupciones durante el funcionamiento del módulo.

Todos los canales de comparación de salida son funcionalmente idénticos. Cada uno de ellos puede utilizar un temporizador. La base de tiempo se selecciona con el bit OCTSEL (OCx-CON[3]).

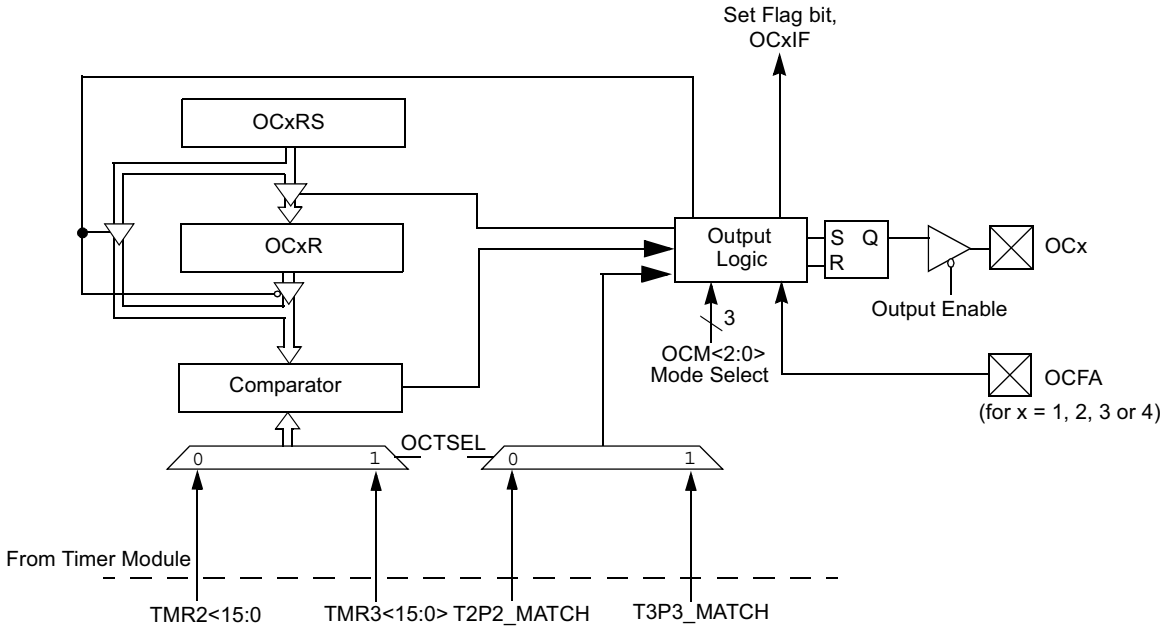



Figura 5.36: Diagrama de bloques del módulo de comparación.

Cuando el módulo de comparación está habilitado, la dirección de la patita de entrada o salida está controlada por el módulo de comparación.
Si este módulo está inhabilitado, devolverá el control de la patita de entrada/salida a la patita LAT y a los bits de control del registro TRIS apropiados. En la figura se marca OCx donde x se refiere a los registros asociados de los canales de comparación de salida(1-4 en el dsPIC 30f4013). La patita OCFA controla los canales OC1 al OC4. Cada canal puede utilizar uno de los dos posibles temporizadores.
Se compara el contenido del registro de forma continua con los valores del temporizador. Cuando ambos valores coinciden pueden suceder una serie de eventos:

- La salida pasa a nivel alto.
- La salida pasa a nivel bajo.
- El estado no varía pero se produce una interrupción.

### 5.4.2. Registros de comparación.

Cada canal de comparación de salida se maneja con los siguientes registros.

- OCxCON: Es el registro de control para el canal x, que presenta el número de canales de comparación de salida.

- OCxR: Es el registro de datos para el canal de comparación de salida x.
- OCxRS: Es el registro de datos secundario para el canal de comparación de salida x.

En la siguiente figura se muestra todos los registros con los bits correspondientes: El registro

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OC1RS | 0180 | Output Compare 1 Secondary Register | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC1R | 0182 | Output Compare 1 Main Register | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC1CON | 0184 | — | — | OCSIDL | — | — | — | — | — | — | — | — | OCFLT | OCTSEL | OCM<2:0> | | | 0000 0000 0000 0000 |
| OC2RS | 0186 | Output Compare 2 Secondary Register | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC2R | 0188 | Output Compare 2 Main Register | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC2CON | 018A | — | — | OCSIDL | — | — | — | — | — | — | — | — | OCFLT | OCTSE | OCM<2:0> | | | 0000 0000 0000 0000 |
| OC3RS* | 018C | Output Compare 3 Secondary Register | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC3R* | 018E | Output Compare 3 Main Register | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC3CON* | 0190 | — | — | OCSIDL | — | — | — | — | — | — | — | — | OCFLT | OCTSEL | OCM<2:0> | | | 0000 0000 0000 0000 |
| OC4RS* | 0192 | Output Compare 4 Secondary Register | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC4R* | 0194 | Output Compare 4 Main Register | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC4CON* | 0196 | — | — | OCSIDL | — | — | — | — | — | — | — | — | OCFLT | OCTSEL | OCM<2:0> | | | 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.

Figura 5.37: Mapa de registros del módulo comparador de salida.

OCxCON está formado por los siguientes bits:

- OCSIDL: Es un bit de control para la desactivación del comparador de salida en modo Idle.
  1 = El comparador se detendrá en modo Idle.
  0 = El comparador seguirá funcionando en modo Idle.
- OCFLT: Indica la condición de fallo de PWM. Solamente se utiliza cuando OCM[2:0]=111
  1 = Fallo en PWM.
  0 = No se ha detectado ningún fallo en PWM.
- OCTSEL: Bit para seleccionar el temporizador.
  1 = Timer3.
  0 = Timer2.
- OCM[2:0]:Tres bits que seleccionan el modo del módulo de comparación. 111 = Modo PWM en OCx con la patita habilitada.
  110 = Modo PWM en OCx con la patita de fallo deshabilitada.
  101 = Inicializa la patita OCx a 0, generando pulsos contiguos a la salida.
  100 = Inicializa la patita OCx a 0, generando un solo pulso a la salida.
  011 = El evento de comparación invierte la patita OCx.
  010 = Si el valor inicial de la patita OCx a 1, la comparación lo cambia a 0.
  001 = Si el valor inicial de la patita OCx a 0, la comparación lo cambia a 1.
  000 = Canal de comparación de salida deshabilitado.

### 5.4.3. Modos de operación.

Cada módulo de comparación tiene tres modos de operación.

- Modo de comparación simple.
- Modo de comparación doble.
  - Pulso simple a la salida.
  - Pulsos continuos a la salida.
- Modo de modulación por anchura de pulsos.
  - Con entrada de protección de fallos.
  - Sin entrada de protección de fallos.

1. **Modo de comparación simple.**
   Cuando los tres bits de control OCM[2:0] toman el valor 001,010,011, el canal de comparación de salida seleccionado es configurado por uno de los tres modos simples de comparación de salida.

   En el modo simple de comparación, el registro OCxR se carga con un valor y este se compara con el registro seleccionado incrementando el registro del contador de tiempo(TMRy).

   Se puede producir uno de los siguientes eventos:

   - La comparación pone el bit OCx a 1 cuando su estado inicial era 0. La interrupción se genera en el mismo evento de comparación simple.
   - La comparación pone el bit OCx a 0 cuando su estado inicial era 1. La interrupción se genera en el mismo evento de comparación simple.
   - La comparación invierte el bit OCx. La inversión es continua y se genera una interrupción por cada inversión que se realice.

2. **Modo de comparación doble.**
   Cuando los bits de control de OCM[2:0] son 100 o 101, el canal del módulo de comparación seleccionado es configurado para uno de los dos modos de comparación doble posibles:

   - Modo con pulso simple a la salida.
   - Modo con pulsos continuos a la salida.

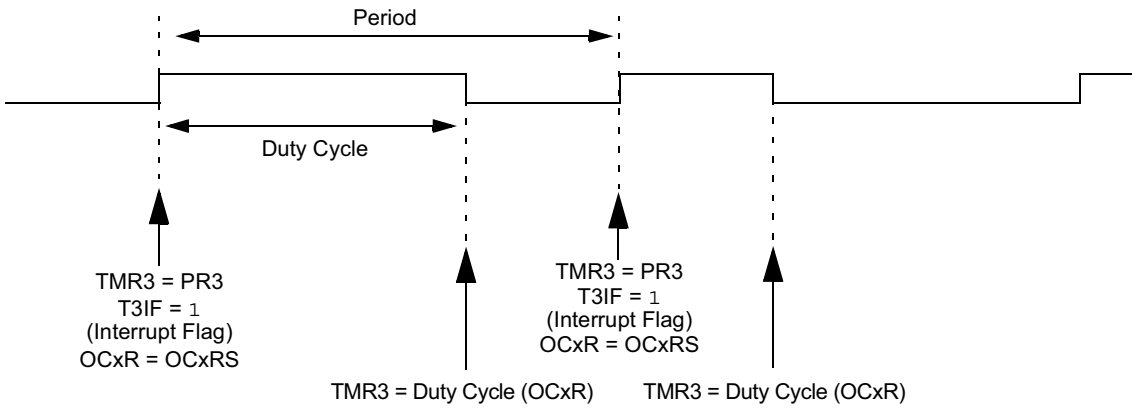



Figura 5.38: Señal PWM.

En el modo de comparación doble, el módulo usa los registros OCxR y OCxRS para comparar los eventos.
El registro OCxR se compara con el Timer(TMRy), y se genera un flanco ascendente en OCx. El registro OCxRS se compara más tarde con el mismo Timer TMRy, produciéndose esta vez un flanco descendente.

3. **Modo de modulación de anchura de pulsos.**
Cuando los bits de control OCM[2:0] son 110 o 111 se selecciona el módulo comparador de salida en modo PWM. Existen dos modos de PWM posibles:

   - PWM sin entrada de protección de fallos.
   - PWM Con entrada de protección de fallos.

   El pin de entrada OCFA se utiliza para el segundo modo PWM.
   En el modo PWM, el registro OCxR contiene el ciclo de trabajo actual y OCxRS es un registro buffer que es modificado por el usuario para actualizar el ciclo de trabajo de la señal PWM. Al final del periodo PWM OCxR es actualizado con el contenido de OCxRS. La finalización de cada periodo de la señal PWM se puede marcar con la interrupción que genera el temporizador que lo marca con el flag TyIF. Los pasos a seguir para configurar este módulo son:

   - Definir el periodo de la señal PWM en el registro PRy.
   - Activar el ciclo de trabajo escribiendo en el registro OCxRS.
   - Escribir en OCxR el ciclo de trabajo inicial.
   - Habilitación de las interrupciones, si se requiere, para el temporizador y modulo comparador. Esta última es requerida si tenemos protección de fallos.

- Configurar el módulo comparador en OCM.
- Activar la preescala del temporizador y activar la base de tiempos TON.

**Periodo PWM.**

Este periodo puede ser calculado siguiendo la siguiente expresión:

$$PWMPeriodo = [(PRy) + 1] * Tcy * (PreescalaTMRy)$$

$$PWMFrequency = 1/[PWMPeriod]$$

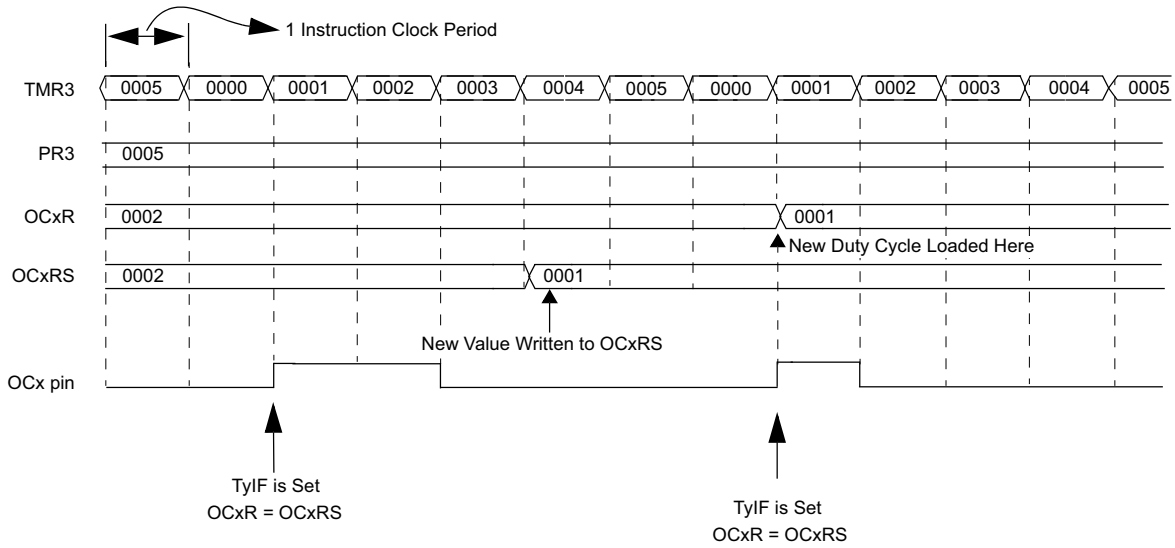
$$TCy = 4/Fosc$$

**Ciclo de trabajo.**

Si en OCxR se carga 0x0000 El ciclo de trabajo es 0, si se cargan valor mayor a PRy el ciclo es 100(por ciento) y si se carga PRy el pin OCx estará en baja una parte del tiempo y la otra en alta.

**Máxima Resolución PWM.**

$$Resolution(bits) = \lg(Fosc/Fpwm)/\lg(2)$$




**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.

Figura 5.39: Temporización de la salida.

### 5.4.4. Módulo comparador de salida y el de ahorro de energía.

- **Modo Sleep.**
  Cuando el dispositivo entra en este modo el sistema de reloj se detiene.
  Durante este tiempo el canal de comparación de salida aplicará a la patita de salida el estado activo que tenía al entrar en el modo sleep. Entonces el módulo parará en este estado. Cuando la CPU vuelva a reactivarse, el módulo de comparación de salida reasumirá la operación.

- **Modo Idle.**
  Cuando el dispositivo entra en este modo, las fuentes de reloj del sistema siguen funcionando y la CPU deja de ejecutar instrucciones.

# 5.5. Módulo de Captura de Entrada.

## 5.5.1. Introducción.

Esta sección describe el módulo de captura de entrada y sus modos operativos. El módulo de captura de entrada se utiliza para capturar el valor de un temporizador de una de las dos bases de tiempo TMR2/TMR3, cuando se produzca un evento a un evento en un pin de entrada. Este módulo es muy útil en aplicaciones en las que se requiere medir frecuencia (Período) o medir pulsos.

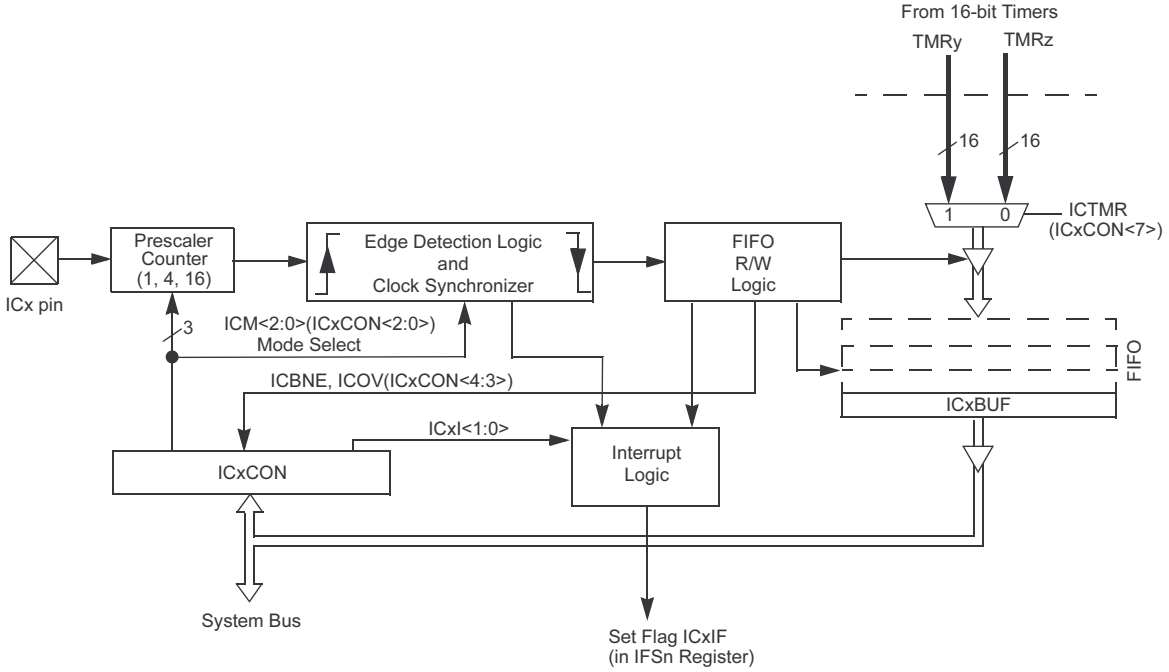



Figura 5.40: módulo de captura de entrada.

El dsPIC30F4013 dispone de 4 entradas con dicho módulo. Todos los canales de entrada de captura son funcionalmente idénticos. En esta sección, una 'x' en el pin o registro da nombre específico de entrada de captura canal. El módulo de captura de entrada tiene varios modos de funcionamiento, que son seleccionados a través del registro ICxCON. Los modos de funcionamiento incluyen: o Captura el valor del temporizador en flanco de bajada de entrada aplicado al pin ICx o Captura el valor del temporizador en flanco de subida aplicados al pin ICx o Captura el valor del temporizador en cada cuarto flanco de subida aplicado al pin ICx o Captura el valor del temporizador en cada 16 flanco de subida aplicado al pin ICx o Captura el valor del temporizador en cada flanco aplicado al pin ICx. El módulo de captura de entrada tiene un buffer FIFO de cuatro niveles. El número de capturas de eventos necesarios para generar una interrupción de la CPU puede ser seleccionado por el usuario.

### 5.5.2. Registros del módulo de captura de entrada.

Cada canal de captura disponible en los dispositivos dsPIC30F tiene los siguientes registros, donde 'x' denota el número del canal captura:

- Registro de control de la captura de entrada.
- ICxBUF: Buffer de captura de entrada.

**ICxCON**

| 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 |

Figura 5.41: Registro ICxCON.

- ICSIDL: Bit de control de stop del módulo de captura de entrada en Idle.
  1 = el módulo de entrada de captura parara cuando la CPU esté en modo Idle.
  0 = el módulo de entrada de captura seguirá funcionando en modo Idle de la CPU.
- ICTMR: Bit de selección del temporizador asociado al módulo de captura de entrada.
  1 = el TMR2 es almacenado en los eventos de captura.
  0 = el TMR3 es almacenado en los eventos de captura.
- ICI: Bits de selección del número de capturas para provocar una interrupción.
  11 = Interrupción en cada cuatro capturas.
  10 = Interrupción en cada tres capturas.
  01 = Interrupción en cada dos capturas.
  00 = Interrupción en cada captura.
- ICOV: Bit que indica desbordamiento en el módulo captura de entrada.
  1 = ha ocurrido un desbordamiento en la captura de entrada.
  0 = no ha ocurrido un desbordamiento en la captura de entrada.

- ICBNE: Bit que indica que el buffer de captura de entrada está vacío (sólo lectura).
  1 = el buffer de captura de entrada no está vacío, por lo menos una captura se puede leer.
  0 = el buffer de captura de entrada está vacío.

- ICM: Bits de selección del la entrada del módulo captura.
  111 = captura de entrada funciona como un pin de interrupción, cuando el dispositivo está en modo Idle.
  110 = no utilizado.
  101 = se captura la entrada cada 16 flancos de subida.
  100 = se captura la entrada cada 4 flancos de subida.
  011 = se captura la entrada cada flanco de subida.
  010 = se captura la entrada cada flanco de bajada de la señal.
  001 = se captura la entrada cada flanco de la señal.
  000 = módulo de entrada de captura apagado.

### 5.5.3. Selección del temporizador.

Cada dsPIC puede tener uno o más canales de captura entrada y cada canal puede seleccionar entre uno de los dos temporizadores de 16. La selección del temporizador se logra mediante el bit de control ICTMR (ICxCON ¡7]). Los temporizadores se pueden configurar mediante la fuente de reloj interno (FOSC / 4), o utilizando un reloj externo aplicado a la pin TxCK.

### 5.5.4. Eventos capturados.

El módulo de captura de entrada captura el valor de los 16 bits de la base de tiempo seleccionado cuando un evento se produce en el pin ICx. Los acontecimientos que pueden ser capturados se enumeran a continuación en tres categorías:

- Modo de captura de eventos simple.
  - Captura el valor del temporizador en cada flanco de bajada de la señal aplicada al pin ICx.
  - Captura el valor del temporizador en cada flanco de subida de la señal aplicada al pin ICx.
- Captura el valor del temporizador en cada flanco de la señal ( ascendente y descendente).
- Preescala de los eventos de captura.

- Captura el valor del temporizador en cada 4º flanco de subida de la señal aplicada al pin ICx.
- Captura el valor del temporizador en cada 16º flanco de subida de la señal aplicada al pin ICx.

Estos modos de captura de entrada se configuran mediante los bits ICM [2:0] (ICxCON [2:0]).

1. Captura de eventos simple.
   El módulo de captura puede capturar el valor del temporizador (TMR2 o TMR3) en los flancos de subida y/o bajada de la entrada aplicada al pin ICx. En estos modos, el preescalado del contador no se utiliza. Vea en la Figura 3 y en la Figura 4 el diagrama de captura de eventos simples.

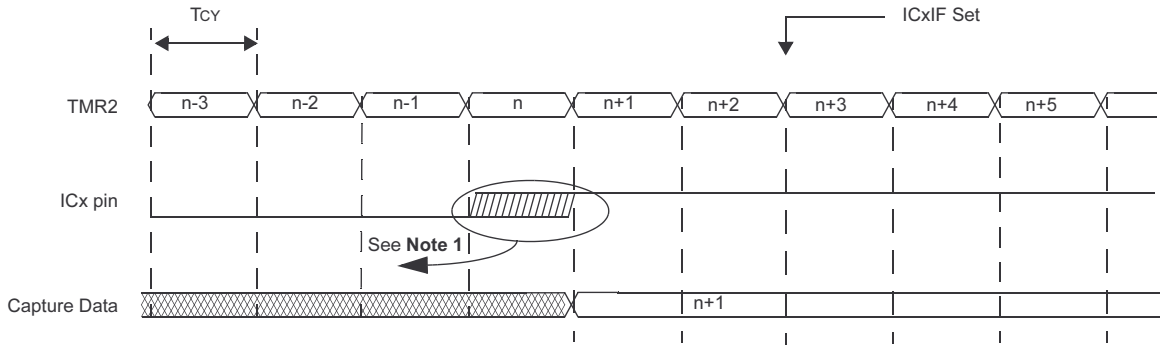



Figura 5.42: Diagrama temporal de evento de captura simple, preescalado 1:1.

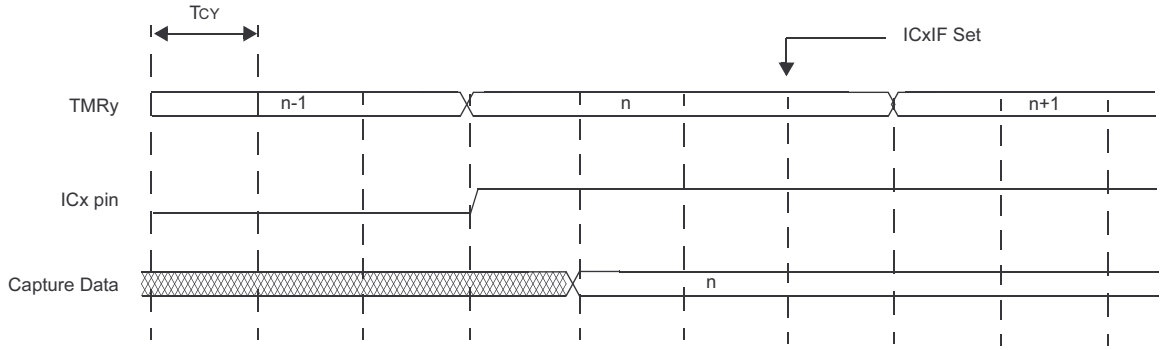



Figura 5.43: Diagrama temporal de evento de captura simple, preescalado 1:4.

Cuando el número de eventos de capturas transcurridos coincida con el número especificado por el ICI [1:0], se producirá una interrupción si están habilitadas, poniéndose a 1 el flag ICxIF dos ciclos después del evento de captura. El valor capturado será el valor que

estuvo presente 1 o 2 ciclos de instrucción pasado el tiempo del evento en el pin ICx. Este tiempo de retraso está relacionado con la instrucción ciclo de reloj y las demoras asociadas con la lógica del módulo de captura de entrada. Si la entrada de reloj es preescalada, el retraso en el valor capturado puede ser eliminado. Vea la Figura 3 y Figura 4 para más detalles.

2. Preescalado en la captura de eventos.
   El módulo de captura tiene dos modos de captura preescalados, que se seleccionan con ICM [2:0] (ICxCON [2:0]) a 100 bits ó 101 . El módulo de captura cuenta cuatro o dieciséis flancos de subida en el pin antes de que se produzca el suceso de captura. El contador de preescalado se incrementa en cada flanco de subida aplicado al pin de captura de entrada. El flanco de subida aplicado al pin sirve como un reloj a un contador. Cuando el contador de preescalado es igual a cuatro o dieciséis (dependiendo del modo seleccionado), se produce la captura del temporizador 2 ciclos de instrucción después del flanco aplicado al pin.

   Cambiar de una preescala a otra puede generar una interrupción ya que el contador de preescalado no será borrado.

   El contador de preescalado se borra cuando:

   - El canal de captura está apagado (es decir, ICM [2:0] ='000').
   - Reset.

   El preescaler contador no se elimina cuando:

   - El usuario cambia de un modo de captura activo a otro.

3. Modo detección de flancos.
   El módulo de captura puede capturar el valor del temporizador al producirse cualquier flanco en el pin de entrada ICx. Este modo se selecciona el modo poniendo los bits ICM [2:0] (ICxCON [2:0]) bits a'001'. Cuando el módulo de captura de entrada está configurado para el detección de flancos:

   - ICxIF se activa en cada flanco de la señal de entrada.
   - ICI no se utiliza, cada flanco produce una interrupción, en el caso de estar habilitadas.
   - No se produce desbordamiento en la captura ICOV (ICxCON [4]).

   Al igual que con el modo de simple captura de eventos, la lógica de captura de entrada detecta y sincroniza los flancos de subida y de bajada de la señal aplicada al pin con la

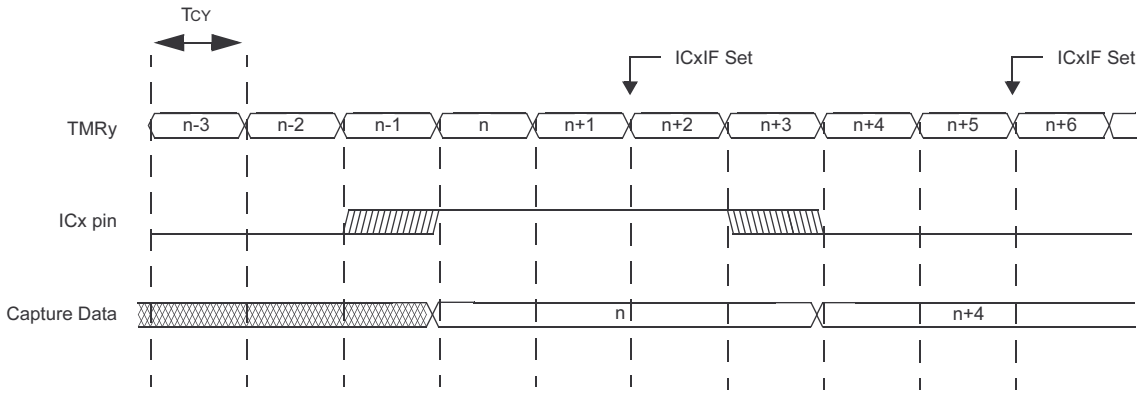



Figura 5.44: Diagrama del modo detección de flancos.

fase del reloj. Cuando se produce el flanco el módulo de captura almacena el valor del temporizador y produce una interrupción lógica, y el flag ICxIF se activa dos ciclos de instrucción después del evento de captura. El temporizador capturado será de 1 ó 2 $T_{CY}$ (ciclos de instrucción) mayor que el instante del evento en el pin ICx.

### 5.5.5. Operación del Buffer de captura.

Cada canal de captura tiene asociado una pila FIFO de cuatro niveles. El registro ICxBUF es el registro visible para el usuario. Cuando el módulo de captura de entrada se reinicia, ICM [2:0] = 000 (ICxCON [2:0]):

- Borra la condición de desbordamiento (pone ICxOV (ICxCON [4]) a'0').
- Borra el bit que indica que el buffer está lleno(ICBNE (ICxCON [3]) a'0').

Al leer el buffer FIFO bajo las siguientes condiciones dará lugar a resultados indeterminados:

- En el caso de que el módulo de captura de entrada se deshabilite y posteriormente se habilite.
- En el caso de que se lea estando el buffer vació.
- Después de un Reset

Hay dos flags de condición de proporcionar el estado del buffer FIFO:

- ICBNE (ICxCON [3]): Buffer de captura de entrada no vació.
- COV (ICxCON [4]): Desbordamiento en la captura de entrada.

### 5.5.6. Interrupciones del módulo de captura de entrada.

El módulo de captura de entrada tiene la capacidad para generar una interrupción al producirse un número de capturas determinado previamente. Un evento de captura se define como una escritura del valor de la base de tiempo en el buffer de captura. Esta interrupción está configurado por el bit control ICI [1:0] (ICxCON [6:5]).

- 11 = Interrupción en cada cuatro capturas.
- 10 = Interrupción en cada tres capturas.
- 01 = Interrupción en cada dos capturas.
- 00 = Interrupción en cada captura.

Cada canal de captura de entrada tiene un flag de estado de interrupción (ICxIF), y un bit que habilita la interrupción (ICxIE) y unos bits de control de la prioridad de la interrupción (ICxIP [2:0]), para más información mirar el tema de interrupciones.

### 5.5.7. Soporte UART Auto-baudios.

El módulo de captura de entrada puede ser utilizado por el módulo UART cuando está funcionando en modo Auto-baudios, ABAUD = 1 (UxMODE [5]). Cuando el bit de control ABAUD está activado el pin RX del UART se conecta internamente al módulo de captura de entrada. El pin I / O asociado con la captura módulo se desconectará. La tasa de baudios se puede determinar midiendo el bit de Start cuando un carácter nulo es recibido. Tenga en cuenta que el módulo de captura debe estar configurado para el modo de detección de flancos para aprovechar la característica auto-baudios. El módulo de captura de entrada usado para cada UART dependerá de la variante de dispositivo dsPIC30F empleado.

### 5.5.8. Operación de la captura de entrada en los estados de ahorro de energía.

- Operación en modo Sleep. Cuando el dispositivo entra en modo sleep, el reloj del sistema es desactivado y el módulo de captura de entrada sólo puede funcionar como una fuente externa de interrupción. Este modo se activa mediante el establecimiento de el control de MCI bits [2:0] ='111'. En este modo, un aumento en el pin de captura hará que el dispositivo se despierte. Si el bit de habilitación de interrupción está activado y la prioridad de la interrupción es la necesaria, una interrupción se generará. En caso de que el módulo de captura se ha configurado para un modo distinto de MCI [2:0] ='111' se

lleva al dispositivo a modo sleep ningún estímulo exterior aplicado el pin despertará al dispositivo.

- Operación en modo Idle. Cuando el dispositivo entra en modo Idle, el reloj del sistema continua funcionando y la CPU detiene la ejecución de código. El bit ICSIDL (ICxCON [13]) selecciona si el módulo se detendrá en modo Idle, o seguirá funcionando en este modo. Si ICSIDL = 0 (ICxCON [13]), el módulo seguirá en funcionamiento cuando el dispositivo esté en modo Idle. Funcionalidad completa del módulo de captura de entrada definida por los bits de control MCI bits [2:0] (ICxCON [2:0]). Se requiere que el temporizador empleado siga funcionando en modo Idle. Si el modo de captura de entrada está configurado para ICM [2:0] ='111', el pin de entrada de captura sólo servirá como un pin de interrupción externa. En este modo, un flanco de subida en el pin de captura despertará el dispositivo del modo Idle. Si los respectivos bits de habilitación de las interrupciones están activadas y la prioridad es mayor a la actual de la CPU se generará una interrupción. Si ICSIDL = 1 (ICxCON [13]), el módulo se detendrá en el modo Idle.

- Despertar al dispositivo del modo Sleep/Idle.

  Un evento de captura de entrada puede generar que el dispositivo se despierte, si el dispositivo está Idle/Sleep o generar una interrupción, si está activado. Independientemente de que el temporizador esté activado o no, la captura de entrada puede sacar al dispositivo de los modos Sleep o Idle cuando se produce una captura si:

  - ICM [2:0] ='111 '(ICxCON [2:0].
  - ICxIE = 1.

  Se produce una interrupción si:

  - ICxIE = 1 y la prioridad es la necesaria.

  Las características de despertar al dispositivo son útiles para añadir un pin de interrupción exterior, cuando el módulo de captura de entrada se usa en este modo se cumple que:

  - El contador de preescala no es utilizado.
  - Los bits ICI [1:0] (ICxCON [6:5]) no son aplicables.

### 5.5.9. Control de los pines de I/O.

Cuando el módulo de captura está activado, el usuario debe configurar la dirección de los pines I / O asociados a este módulo como pines de entrada mediante el TRIS asociado,

ya que la dirección del pin no se establece habilitando este módulo, y todos los periféricos multiplexados con este módulo deben ser deshabilitados.

## 5.6. Convertidor Analógico Digital.

### 5.6.1. Introducción.

El dsPIC30F tiene un módulo convertidor Analógico Digital de 10 bits con las siguientes características:

- Registro de aproximación sucesiva (SAR).
- Velocidad de la conversión de hasta 1Msps.
- Hasta 16 pines de entrada analógica.
- Entradas de tensión de referencia.
- Cuatro amplificadores sample/hold (S/H).
- Muestreo simultaneo de hasta cuatro pines de entrada analógica.
- Modo de muestreo de canales automático.
- Selección de fuente de activación de conversión.
- Buffer de 16 de palabra conversión.
- Cuatro opciones de alineación.
- Operación durante modos de Sleep e Idle.

Los 10 bits convertidor A / D puede tener hasta a 16 pines de entrada analógica, designado AN0-AN15. Además, hay dos pines de entrada analógica para voltaje de referencia externo. Estas entradas de voltaje de referencia puede ser compartida con otros pines de entrada analógica. El número real de pines de entrada analógica y externos de tensión de referencia de entrada dependerá del dispositivo específico dsPIC30F, en el caso del dsPIC30F4013 tenemos 12 entradas analógicas.
Las entradas analógicas se conectan a través de multiplexores a cuatro amplificadores S / H, designado CH0-CH3. Uno, dos, o cuatro amplificadores S / H puede ser activado para la adquisición de datos de entrada. Los multiplexores de entrada pueden conmutarse entre dos conjuntos de entradas analógicas durante las conversiones. Son posibles conversiones simple o diferencial en todos los canales de entrada.

El modo de muestreo puede ser habilitado en CH0 S / H amplificador. Un registro de control especifica que canales de entrada analógica se incluirán en la secuencia de muestreo.

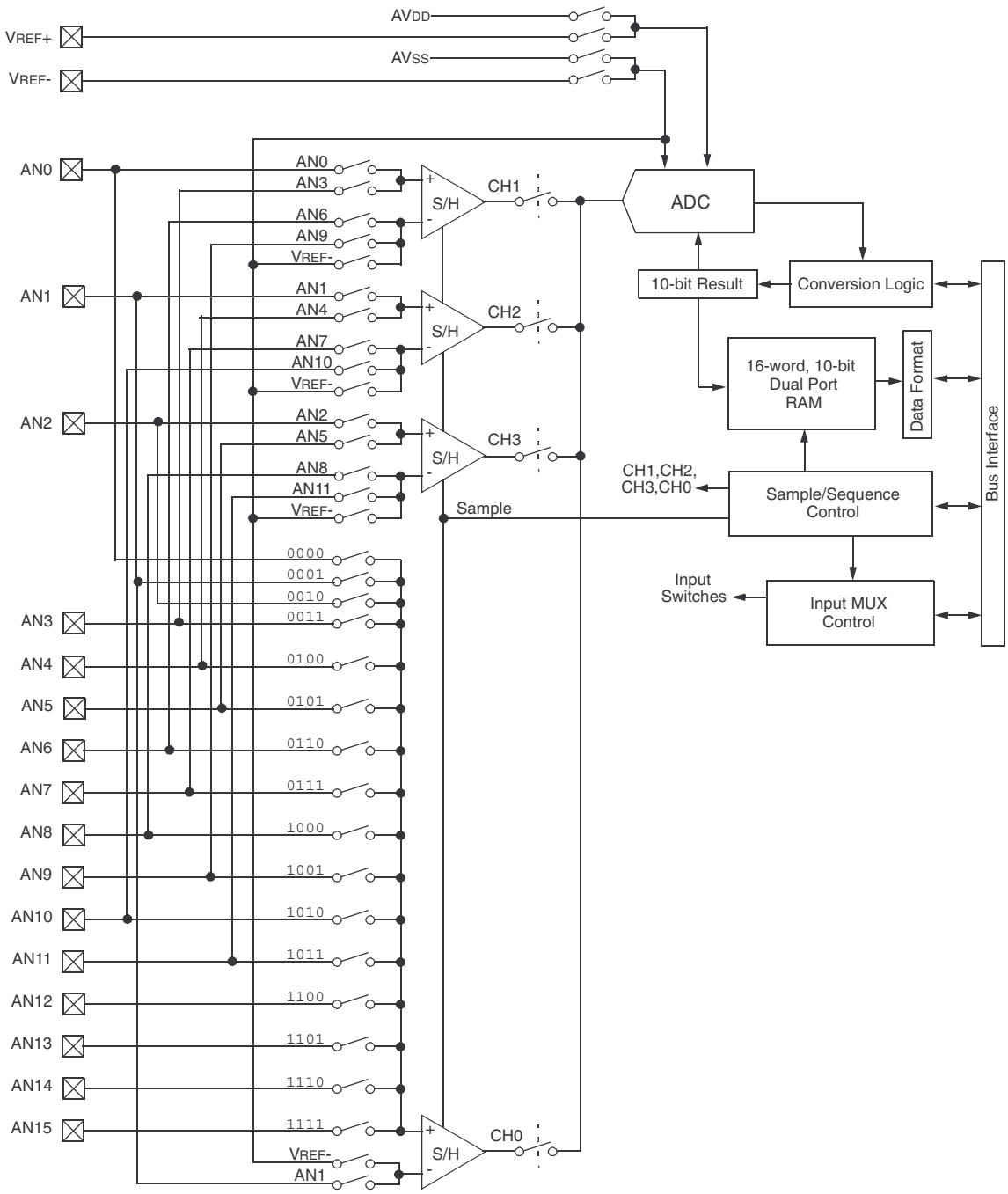



Figura 5.45: Diagrama de bloques de los 10-bit A / D.

El convertidor A / D de 10 bits está conectado a un buffer de 16 palabras. Cada resultado de 10 bits se convierte en uno de los cuatro formatos de salida de 16 bits.

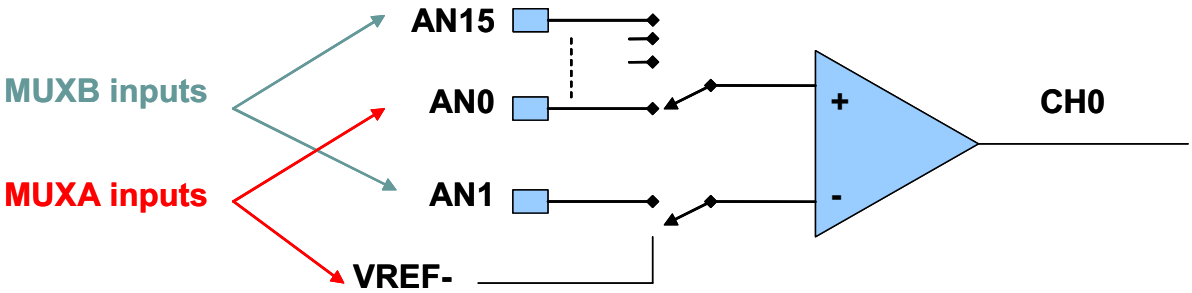



Figura 5.46: MuxB entrada diferencial y MuxA simple.

### 5.6.2. Registros de control.

El convertidor A / D tiene 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 convertidor.
- ADPCFG: A / D Registro de configuración del puerto.
- ADCSSL: Registro de selección de las entradas de muestreo del convertidor A / D.

Los registros ADCON1, ADCON2 y ADCON3 controlan el funcionamiento del convertidor A / D. El registro ADCHS selecciona el pin de entrada a ser conectado al amplificador S /H. El registro ADPCFG configura que pines son entradas analógicas o que pines son I / O digitales. El registro ADCSSL selecciona las entradas para ser muestreadas secuencialmente.

**ADCON1.**

- ADON: Mode de operación.
  1 = Convertidor A/D funcionando.
  0 = Convertidor A/D apagado.
- ADSIDL: Parar en modo Idle.
  1 = Deja de funcionar en modo Idle.
  0 = Continua funcionando en modo Idle.

| 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 | R/W-0 | R/W-0 | R/W-0 HC, HS | R/C-0 HC, HS |
| SSRC<2:0> | | | — | SIMSAM | ASAM | SAMP | DONE |
| bit 7 | | | | | | | bit 0 |

Figura 5.47: ADCON1.

- FORM[1:0]: Formato de los datos de salida.
  11 = Fraccional simple (DOUT = sddd dddd dd00 0000).
  10 = Fraccional (DOUT = dddd dddd dd00 0000).
  01 = Entero simple (DOUT = ssss sssd dddd dddd).
  00 = Entero (DOUT = 0000 00dd dddd dddd).

- SSRC[2:0]: Selecciona la fuente de disparo de la conversión.
  111 = Un contador interno finaliza el muestreo e inicializa la conversión (auto convert).
  110 = Reservado.
  101 = Reservado.
  100 = Reservado.
  011 = Intervalo controlado por el motor PWM finaliza el muestreo e inicializa la conversión.
  010 = GP Timer3 compare finaliza el muestreo e inicializa la conversión.
  001 = Transición en el pin INT0 finaliza el muestreo e inicializa la conversión.
  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 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 esta habilitado.
  0 = Todos los amplificadores sample and hold están deshabilitados.
  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 empezando una nueva conversión. Si se borra este bit no afecta a las operaciones que estén en progreso.

**ADCON2.**

| Upper Byte: | | | | | | | |
|---|---|---|---|---|---|---|---|
| R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
| VCFG<2:0> | | | reserved | — | CSCNA | CHPS<1:0> | |
| 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 |

Figura 5.48: ADCON2.

- 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: Selecciona la entrada de muestreo para el canal CH0+ S/H, para el MUX A pone a uno la entrada del multiplexor.
  1 = Escanea las entradas.
  0 = No escanea las entradas.

- CHPS[1:0]: Bits de 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 SIMSAM (ADCON1[3]) = 0 múltiples canales son muestreados secuencialmente.
  Cuando el bit SMSAM (ADCON1[3]) = 1 múltiples canales son muestreados según CHPS[1:0].

- BUFS: Bit de estado del buffer de salida. Solo válido cuando BUFM = 1 (Dirección dividida en 2x8).
  1 = Posiciones 0x8-0xF del buffer de salida del A/D están llenas, el usuario debe acceder a los datos de las posiciones 0x0-0x7.
  0 = Posiciones 0x0-0x7 del buffer de salida del A/D están llenas, el usuario debe acceder a los datos de las posiciones 0x8-0xF.

- SMPI[3:0]: Bits de selección de interrupciones de la secuencia de muestreo conversión.
  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: Bit de selección del modo del Buffer.
  1 = Buffer configurado como dos buffer de 8 palabras cada una, ADCBUF (15...8), ADCBUF (7...0).
  0 = Buffer configurado como un único buffer de 16 palabras ADCBUF (15...0.).

- ALTS: Bit de selección de una entrada alternativa del modo de muestreo.
  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 Uses MUX A.
  0 = Utiliza siempre el multiplexor MUX A.

**ADCON3.**

| 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 |

Figura 5.49: ADCON3.

- SAMC[4:0]: Indica el tiempo de muestreo automático.
  11111 = 31 $T_{AD}$.
  .....
  00001 = 1 $T_{AD}$.
  00000 = 0 $T_{AD}$ (si sólo se permite realizar conversiones secuencial usando más de un amplificador S / H).

- : 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 = $T_{CY}/2$ o (ADCS[5:0] + 1) = $32T_{CY}$.
  .....
  000001 = $T_{CY}$ /2 o (ADCS[5:0] + 1) = $T_{CY}$.
  000000 = $T_{CY}$ /2 o (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.

- 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 CH8 negativo es la entrada AN8

| 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 |
| CH123NB<1:0> | | CH123SB | CH0NB | CH0SB<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 |
| CH123NA<1:0> | | CH123SA | CH0NA | CH0SA<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

Figura 5.50: ADCHS.

0x = El CH1 negativo es la entrada $V_{REF-}$.
El CH1 negativo es la entrada $V_{REF-}$.
El CH1 negativo es 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 CH8 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 CH8 negativo es la entrada AN8.
0x = El CH1 negativo es la entrada $V_{REF-}$.
El CH1 negativo es la entrada $V_{REF-}$.
El CH1 negativo es 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 CH8 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 MUX A.
  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.

- PCFG[15:0]: Bits de control de la configuración de las patillas de entrada.
  1 = Pines de entrada analógico configuradas en modo digital, las entradas A/D del multiplexor conectadas a $AV_{SS}$.
  0 = Pines de entrada analógico configuradas en modo analógico, muestras del A/D en el pin de tensión.

| 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 |

Figura 5.51: ADPCFG.

**ADCSSL**

Este registro selecciona el orden en el que las entradas serán secuencialmente muestreadas.

| 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 |

Figura 5.52: ADCSSL

- CSSL[15:0]: Bit de selección de la exploración de los pines de entrada.
  1 = Selecciona el canal ANx correspondiente para ser muestreado.
  0 = No selecciona canal ANx.

**BUFFER DE RESULTADOS.**

El módulo contiene una RAM de 16 palabras de doble puerto, llamada ADCBUF, para amortiguar el A / D los resultados. Cada una de las posiciones de buffer se denominan ADCBUF0, ADCBUF1, ADCBUF2, .... ADCBUFE, ADCBUFF.

### 5.6.3. A / D terminología y secuencia de conversión.

El muestreo del pin de entrada analógica se realiza por los amplificadores sample/hold. El convertidor A/D de 10 bits tiene cuatro amplificadores S / H, llamados CH0-CH3. Los canales

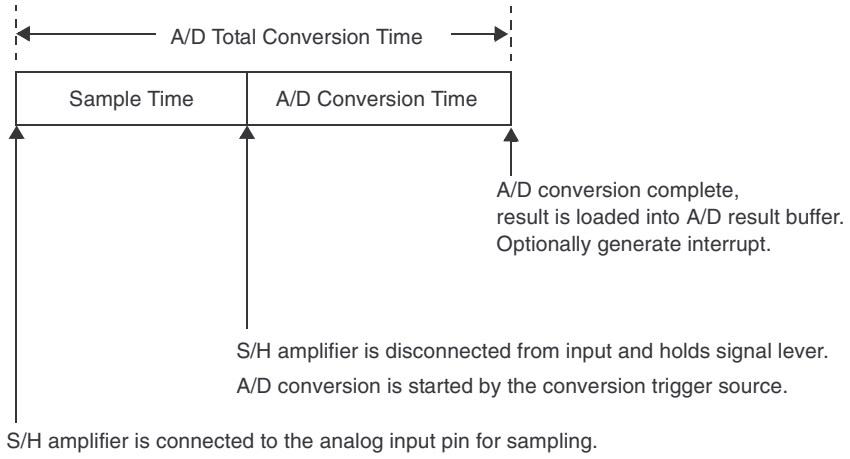



Figura 5.53: secuencia básica de conversión.

del S / H están conectados a los pines de entrada analógica a través de un multiplexor. El multiplexor de entrada analógica está controlado por el registro ADCHS.
Hay dos conjuntos de bits de configuración de los multiplexores en el registro ADCHS con la misma función. Estos dos conjuntos de bits de control permiten programar dos configuraciones del multiplexor de entrada analógica diferentes, las que se llaman multiplexor MUX A y MUX B. El convertidor A / D, opcionalmente, puede cambiar entre la configuración MUX A y MUX B entre las conversiones.El convertidor A / D, opcionalmente, también puede muestrear una serie de entradas analógicas.
El tiempo de muestreo es el tiempo que el amplificador S / H del CAD está conectado a la pin de entrada analógica. La muestra puede comenzar manualmente activando el bit SAMP (ADCON1 [1]) o iniciarse automáticamente por el CAD. El tiempo de muestreo termina manualmente borrando el bit SAMP o automáticamente por la fuente de disparo de la conversión.
El tiempo de conversión es el tiempo necesario para el convertidor A / D para convertir la tensión facilitada por el S/H. El A / D se desconecta del pin de entrada analógica al final del tiempo de muestreo. El CAD requiere un ciclo de reloj A/D ($T_{AD}$) para convertir cada bit del resultado más un ciclo de reloj adicional. Un total de 12 ciclos $TA_{AD}$ son necesarios para realizar la conversión completa.
Cuando el tiempo de conversión se ha completado, el resultado es cargado en uno de 16 registros de resultado del CAD (ADCBUF0... ADCBUFF), 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 CAD 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 . El usuario debe seleccionar un reloj de entrada que no viole las disposiciones mínimas del

$T_{AD}$.
Los convertidores A / D de 10 bits permite muchas opciones para especificar la secuencia de muestro/conversión que puede ser muy simple, empleando un único S/H, o puede emplear varios S/H. El número de S/H utilizado en la secuencia muestreo/ conversión está determinada por el bit control CHPS.
Durante el tiempo total de la conversión el amplificador S/H está conectado al pin de entrada analógica. La conversión se inicia con la fuente de disparo externa. Cuando se finaliza la conversión el resultado es cargado en el buffer de resultado y opcionalmente se genera una interrupción.
Un proceso de muestreo/conversión que utilice varios canales S/H las entradas pueden ser muestreadas simultáneamente, esto está controlado por el bit SIMSAM (ADCON1 [3]). 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 SAMP, o controlado automáticamente por el hardware controlado por el bit ASAM. La fuente de disparo de la conversión termina el tiempo de muestreo y comienza una conversión A / D o una secuencia muestreo/conversión. La fuente de disparo es seleccionada por los bits de control SSRC.
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 SMPI. El número total de conversiones entre las interrupciones no deben exceder el tamaño del buffer.

### 5.6.4. Pasos a seguir para la conversión A/D.

1. Configura el módulo A/D.

   - Configura los pins analógicos, la referencia de tensión y la entrada/salida digital.
     Una patilla se configura como entrada analógica cuando el bit PCFG ( ADPCFG[n]) correspondiente esta a '0'.
   - Seleccione la fuente de referencia del voltaje para que coincida con el rango esperado de entradas analógicas ADCON2 [15:13]
   - Selecciona los canales de entrada
     El canal 0 es el más flexible. El usuario puede elegir hasta 16 bits los CH0SA selec-

cionan las entradas analógicas a dicho canal.

Las entradas especificadas con CH0SA se seleccionan con MUX A y las seleccionadas con CH0SB con MUX B. Cuando el bit ALTS toma el valor 1 el módulo alterna entre las entradas del MUXA en un muestreo y las de MUXB en el siguiente.

En los canales 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.

| CHPS<1:0> | SIMSAM | Sample/Conversion Sequence | # of Sample/ Convert Cycles to Complete | Example |
|---|---|---|---|---|
| 00 | x | Sample CH0, Convert CH0 | 1 | Figure 17-4, Figure 17-5, Figure 17-6, Figure 17-7, Figure 17-10, Figure 17-11, Figure 17-14, Figure 17-15 |
| 01 | 0 | Sample CH0, Convert CH0<br>Sample CH1, Convert CH1 | 2 | |
| 1x | 0 | Sample CH0, Convert CH0<br>Sample CH1, Convert CH1<br>Sample CH2, Convert CH2<br>Sample CH3, Convert CH3 | 4 | Figure 17-9, Figure 17-13, Figure 17-20 |
| 01 | 1 | Sample CH0, CH1 simultaneously<br>Convert CH0<br>Convert CH1 | 1 | Figure 17-18 |
| 1x | 1 | Sample CH0, CH1, CH2, CH3 simultaneously<br>Convert CH0<br>Convert CH1<br>Convert CH2<br>Convert CH3 | 1 | Figure 17-8<br><br>Figure 17-12, Figure 17-16, Figure 17-17, Figure 17-9, |

Figura 5.54: Opciones de control del A/D.

- Determinar cómo se llevará a cabo el muestreo ADCON1 [3] y ADCSSL [15:0].
- Seleccionar la secuencia muestro/conversión adecuada ADCON1 [7:0] y ADCON3 [12:8].
- Seleccionar la forma de conversión resultados se presentan en el buffer ADCON1 [9:8].
- Selecciona la frecuencia del reloj.
- Selecciona el disparo (trigger) del convertidor.

$$T_{AD} = \frac{T_{CY}(ADCS + 1)}{2}$$

$$ADCS = \frac{2\,T_{AD}}{T_{CY}} - 1$$

Figura 5.55: Opciones de control del A/D.

- Habilitar el módulo.
  Cuando el bit AD0N (ADCON1[15]) está activada el módulo está habilitado, en el caso contrario estará deshabilitado.

2. Configura la interrupción de conversor (si es necesario).
   - Borramos el bit ADIF.
   - Selecciona la prioridad de la interrupción.
   - Activa el bit ADIE.
3. Inicia muesteo.
   Activando el bit SAMP (ADCON1[1]) si estamos trabajando en modo manual, cuando el bit ASAM (ADCON1[2]) estamos en modo automático el muestreo se inicia cuando termina la conversión.
4. 4. Espera el tiempo necesario de adquisición de la muestra.
   La conversión se inicia cuando termina el muestreo. Con los bits SSRC se selecciona la fuente de disparo del conversor.
5. Espera a que se complete la conversión.
   - Esperar la interrupción de conversor.
   - Esperar a que se active el bit DONE.
6. Lee el resultado del buffer y borra el bit ADIF si es necesario.

# Capítulo 6

# Prácticas para dsPIC30F

## 6.1. Práctica guiada.

Para familiarizarse con los comandos del dsPIC así como con las herramientas MPLAB y WINPIC800 que se han explicado se pide crear un nuevo proyecto en MPLAB para un dsPIC30F4013 e insertar el código que acompaña a este enunciado, ejecutarlo comprobar que no existe ningún error, grabarlo en el dsPIC y montarlo para comprobar su funcionamiento. Cuando se realice el montaje hay que tener cuidado de no confundir las alimentaciones y colocar en las patillas correctas el cristal de cuarzo de 4MHZ para no estropear el dsPIC.

```
#include <dspic.h>
#include "binario.h"
//Comprobaremos el funcionamiento de los temporizadores
//enciendo y apagando unos LEDs
/* Practica implementada para un cristal de 4Mhz usando el
oscilador primario XT w/PLL 4X -XT crytal oscillator with 4 X PLL */
/* Configuracion empleada en WinPic800*/
int cont;
main()
{
        TRISF = 0x0000; //Definimos el puerto F como salidas
        TRISB = 0xFF00; //Definimos el puerto B la mitad salidas
                        // y la otra entradas
        ADPCFG = 0xffff;// Salidas digitales
        SWDTEN = 0;     //Deshabilitamos el perro guardián
        T2CON = 0X8010; //Habilitamos temporizador.
                        //Caracteristicas consultar datasheet
        TMR2=0;         //Inicializamos el temporizador
        while(1)
        {
                RB1=1;  //Iniciamos con un Led encendido y
                RF0=0;  // Otro apagado
                while ( cont <40)
                {
                        while(TMR2<=0xF424);// Esperamos un tiempo
                        TMR2=0;
                        cont++;
```

```
                }
                cont=0;
                RB1=0;  //Cambiamos estado de los LEDs
                RF0=1;
                while ( cont <40)
                {
                        while(TMR2<=0xF424);// Esperamos un tiempo
                        TMR2=0;
                        cont++;

                }
                cont=0; //Cambiamos estado de los LEDs
                RB1=1;
                RF0=1;
                while ( cont <40)
                {
                        while(TMR2<=0xF424);// Esperamos un tiempo
                        TMR2=0;
                        cont++;

                }
                cont=0;
                RB1=0;  //Cambiamos estado de los LEDs
                RF0=0;
                while ( cont <40)
                {
                        while(TMR2<=0xF424);// Esperamos un tiempo
                        TMR2=0;
                        cont++;

                }
        }
}
```

## 6.2. Práctica: Cruce de semáforos.

Diseñar un sistema de control que pretende regular un cruce de semáforos como se muestra en la siguiente figura: La temporización por cada semáforo será:

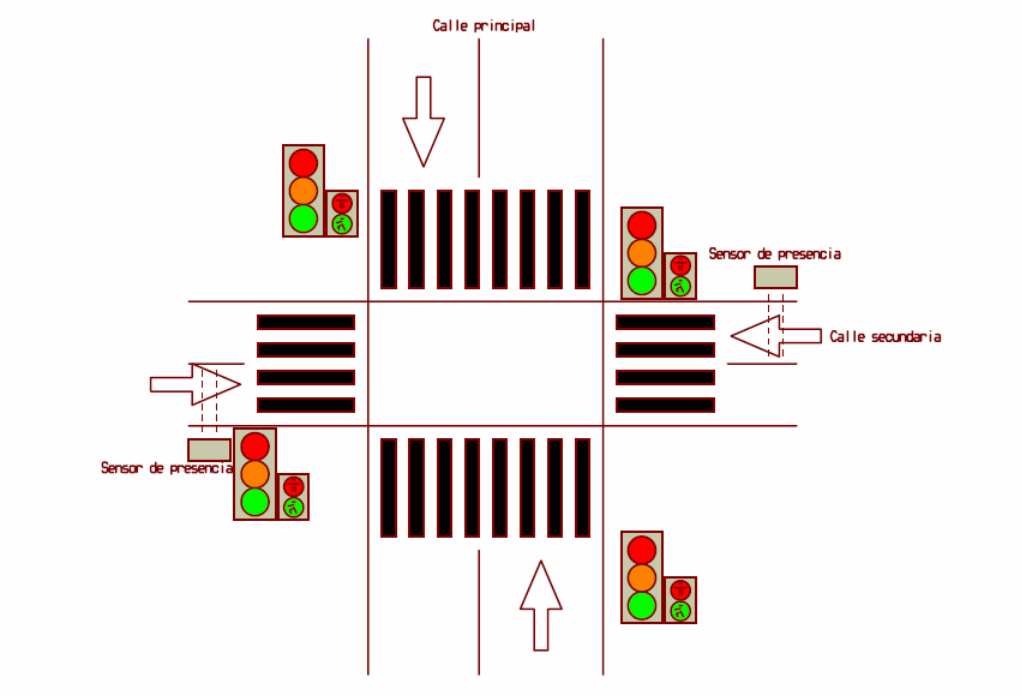



Figura 6.1: Cruce de semáforos.

- La luz verde estará activada durante al menos 25 segundos en la calle principal. Transcurrido este tiempo seguirá en verde hasta que cualquiera de los dos sensores de la calle secundaria indique la presencia de un coche.
- La luz ámbar parpadea durante 5 segundos (cada $\frac{1}{2}$ segundos cambia su estado).
- La luz roja se mantiene encendida durante 30 segundos.
- Cuando la luz verde, o la luz ámbar se encuentren encendidas, la luz roja de peatones debe estar activada.
- Cuando la luz roja se encuentre encendida, la luz verde de paso de peatones debe estar activada, salvo durante los últimos 10 segundos en que debe parpadear con un periodo de 1 segundo.
- En el caso de que ninguno de los dos sensores esté activado la luz verde de los peatones permanecerá parpadeando desde que han pasado los 20 segundos.

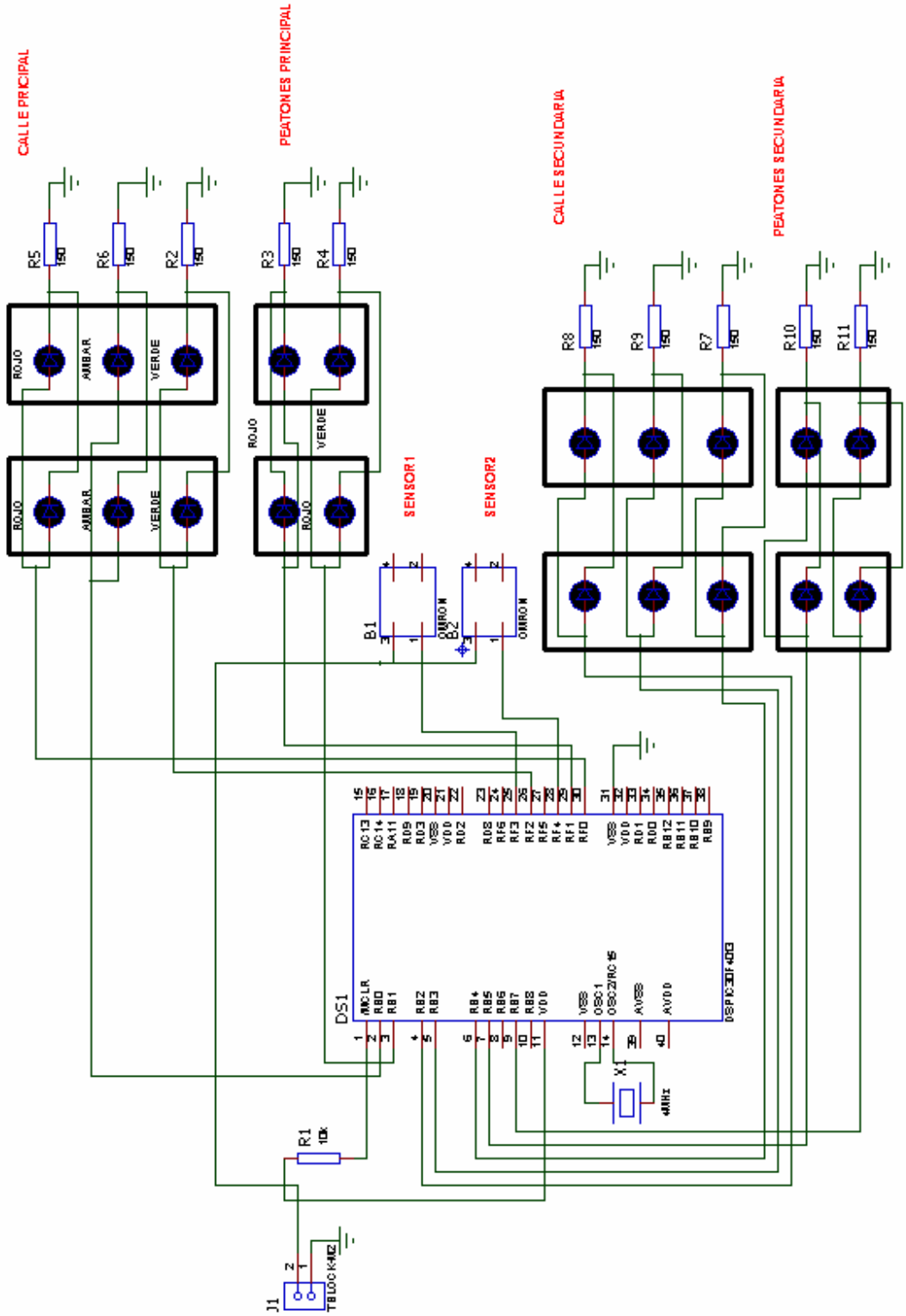



Figura 6.2: Esquema hardware en Proteus de la placa de pruebas.

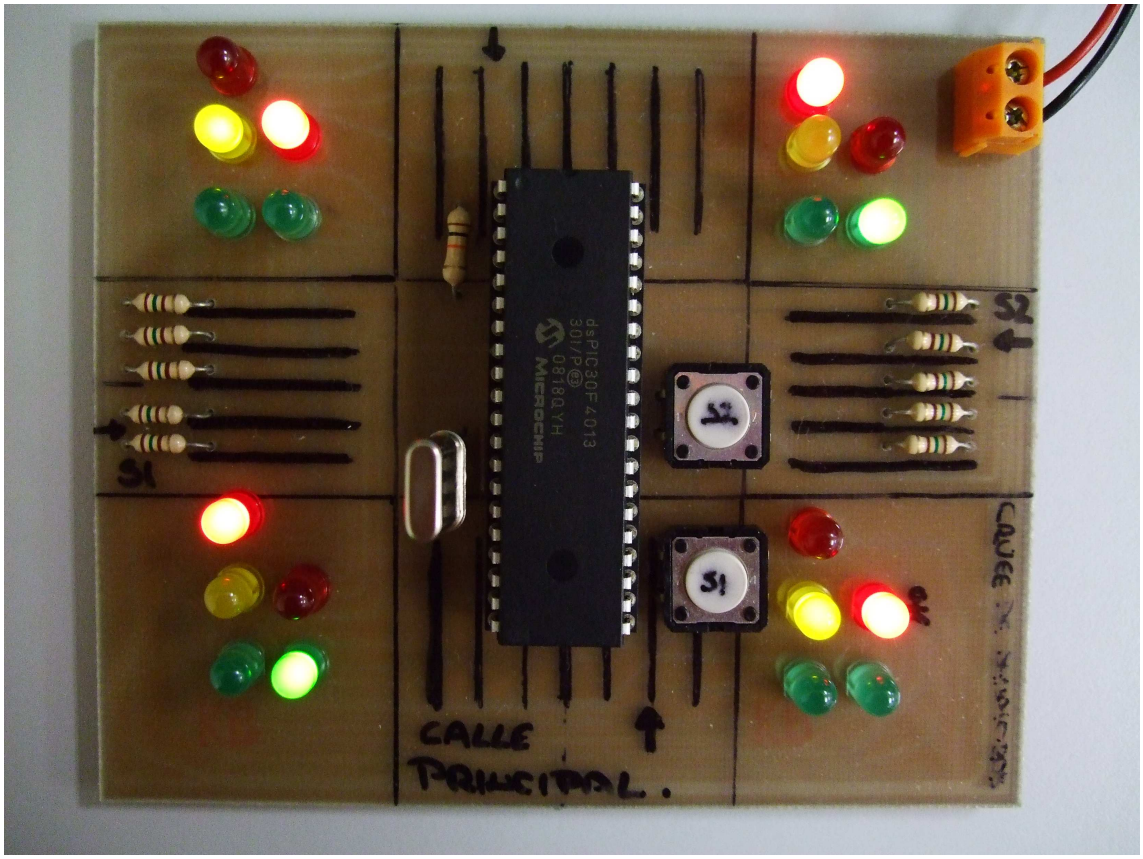



Figura 6.3: Placa de pruebas.

Posteriormente se presenta la solución software de esta práctica en la que se ha utilizado el temporizador 2 disponible en estos dsPIC y se han utilizado los puertos B y F , algunas de sus patillas, para representar las salidas que simbolizan las luces de los semáforos utilizados.

## *Solución: Cruce de Semáforos.*

```
/***     CRUCE DE SEMAFOROS       ***/
/* Practica implementada para un cristal
de 4Mhz usando el oscilador primario
XT w/PLL 4X -XT crytal oscillator with 4 X PLL */
/* Configuración empleada en WinPic800*/
#include <dspic.h>
#include "binario.h"
int cont;// Variable auxiliar
void main()
{
SWDTEN = 0;
         //Deshabilita el perro guardian
TRISF = B11111000;
        /* RF0-RF2 Semáforos calle principal
        RF0 -> Rojo coches
        RF1 -> Rojo peatón calle principal
        RF2 -> Verde coches  */
TRISB = B00000000;
```

```
        /* RB0-RB7 Salidas menos RB6
        RB0 -> Ámbar coches calle principal
        RB1 -> Verde peatón calle principal
        RB2 -> Rojo coches calle secundaria
        RB3 -> Ámbar coches calle secundaria
        RB4 -> Verde coches calle secundaria
        RB5 -> Rojo peatón calle secundaria
        RB7 -> Verde peatón calle secundaria */
ADPCFG = 0XFFFF;
        //Se utiliza las patillas del puertoB como E/S digitales.
T2CON = 0X8000;
        // Ton=1; TCKPS0=0 y TCKPS1=1 Escala 1:64
TMR2=0;
        // Cuenta :  0.5seg = 0.25useg * ( Cuenta * Escala)
        // Escala=64, Cuenta=31250= 0x7A12
cont = 0;
while(1)
{
/*      SITUACIÓN INICIAL       */
        PORTF=B11111110;
         /* RF0 -> Rojo OFF
        RF1 -> Rojo peatón calle principal ON
        RF2 -> Verde ON  */
        PORTB   = B10000100;
        /* RB0 -> Ámbar coches calle principal OFF
        RB1 -> Verde peatón calle principal OFF
        RB2 -> Rojo coches calle secundaria ON
        RB3 -> Ámbar coches calle secundaria OFF
        RB4 -> Verde coches calle secundaria OFF
        RB5 -> Rojo peatón calle secundaria OFF
        RB7 -> Verde peatón calle secundaria ON */
        while (cont<40)
        // Mantengo esta situación durante 20 seg,
        // como cada temporización son 0.5s
        // por lo que esperamos 40 veces
        // a que finalice el temporizador
        {
           while(TMR2<=0x7A12);
           TMR2=0;
           T2TCKPS0=0;
            // Preescala 1:64
           T2TCKPS1=1;
            // este bit se borra con la inicialización del temp2
           cont++;
        }
        while ( cont < 50 ||(!RF3 && !RF4 ))
            // Parpadea el semáforo de
            //los peatones secundario ( verde)
            // Hasta que pasen los 25 seg
        {
           RB7=!RB7;
            // Invierte su valor
           while(TMR2<=0x7A12);
            // Espera a que finalice TMR1
           TMR2=0;
           T2TCKPS0=0;
            // Preescala 1:64
           T2TCKPS1=1;
           cont++;
        }
        cont=0;
        RF2=0;
```

```
        // Apagamos verde coches principal
         while (cont < 10)
         // Hacemos que el ámbar de la calle principal
         //y el verde de peatones de la secundaria parpadeen
         {
                 RB7=!RB7;
                 while(TMR2<=0x7A12);
                 TMR2=0;
                 T2TCKPS0=0;
              // Preescala 1:64
                 T2TCKPS1=1;
                 cont++;
                 RB0=!RB0;
         }
         PORTF=B11111001;
                 /* RF0 -> Rojo ON
                 RF1 -> Peatones rojo principal OFF
                 RF2 -> Verde    OFF */
         PORTB=B00110010;
                 /* RB0 -> Ámbar coche calle principal OFF
                 RB1 -> Verde peatón calle principal ON
                 RB2 -> Rojo coches calle secundaria OFF
                 RB3 -> Ámbar coches calle secundaria OFF
                 RB4 -> Verde coches calle secundaria ON
                 RB5 -> Rojo peatón calle secundaria ON
                 RB7 -> Verde peatón calle secundaria OFF */
         cont=0;
         while ( cont < 40 )
         // Mantengo esta situación durante 20 segundos
         {
                 while(TMR2<=0x7A12);
                 TMR2=0;
                 T2TCKPS0=0;
             // Preescala 1:64
                 T2TCKPS1=1;
                 cont++;
         }
         while ( cont < 50)
         // Parpadea el semáforo de los peatones principal( verde)
         {
                 RB1=!RB1;
                 while(TMR2<=0x7A12);
                 TMR2=0;
                 T2TCKPS0=0;        // Preescala 1:64
                 T2TCKPS1=1;
         cont++;
         }
         cont=0;            // Se hace que el ámbar de la
         RB4=0;             // calle secundaria parpadee y el
         while (cont < 10) // verde de los peatones de la principal
         {       RB1=!RB1;
                 while(TMR2<=0x7a12);
                 TMR2=0;
                 T2TCKPS0=0;        // Preescala 1:64
                 T2TCKPS1=1;
                 cont++;
                 RB3=!RB3;
         }// Se inicializa contador para una nueva iteración.
                 cont=0;
   }
}
```

## 6.3. Práctica: Conexión teclado - display con protocolo $I^2C$

En esta práctica se pide diseñar un controlador para conectar por un lado un teclado matricial 3x4 y obtener por el otro una salida que cumpla el protocolo $I^2C$, esta salida ira a otro dsPIC que transmitirá el dato a un display alfanumérico tipo LM1602. Para ello tendremos que controlar bien el protocolo de comunicaciones con el dsPIC y la temporización necesaria para dicha comunicación.

**TECLADO**

La forma de obtener los caracteres está relacionada con el número de pulsaciones consecutivas de la misma tecla dentro de un intervalo temporal determinado. Así en la figura podemos ver asignación de valores.


Figura 6.4: Teclado 3x4 alfanumérico

- La tecla 1 tendrá como salida los siguientes caracteres: espacio, 1.
- La tecla 2 tendrá como salida: ABC2.
- La tecla 3 los caracteres DEF3.
- La tecla 4: GHI4.
- La tecla 5: JKL5.
- La tecla 6: MNO6.
- La tecla 7: PQRS7.
- La tecla 8: TUV8.

- La tecla 9: WXYZ9.
- El * elimina la pulsación.
- El 0 da 0
- El ♯ de ada por buena la salida pudiendo pasar a la siguiente letra.

Se dará por válida una pulsación si:

- Se pulsa a continuación la tecla de aceptación ♯.
- Ha transcurrido 1 segundo desde la última pulsación.
- Se ha pulsado una tecla diferente a la de la última pulsación y esa tecla no era la de anulación *.

Inicialmente se puede visualizar el resultado con LEDs, y una vez eso sea correcto pasar a implementar el protocolo de comunicaciones. El analizador lógico será de gran ayuda a la hora de analizar el protocolo. No olvidéis seleccionar la frecuencia de muestreo apropiada a la velocidad del protocolo.

**DISPLAY**

Utilizando la información proporcionada se pide que conectemos y actuemos sobre un display alfanumérico. Para ello tendremos que controlar bien el protocolo de comunicaciones con el mismo y la temporización necesaria para dicha comunicación.

Los tiempos de espera necesarios para el correcto funcionamiento del display dependen del fabricante (el reloj interno del display suele funcionar a 240 kHz). Para una escritura normal el tiempo necesario es de 40 microsegundos como máximo. Para la iniciación del LCD o para borrar la pantalla, puede ser de hasta 1.62 ms.

**Para sacar nota**:

Usa todas las prestaciones del display para hacer que aparezcan mensajes en la segunda línea del display y diseñar efectos de aparición de texto con scroll.Existe la posibilidad de conexión con un bus de datos de 4 bits en vez de usar un bus de datos de 8 bits. Implementa las rutinas y el hardware necesario.

Se adjuntan posteriormente los datos necesarios para poder implementar el display.

**PROTOCOLO $I^2C$**

En la gráfica posterior se puede apreciar el protocolo $I^2C$.

Por defecto las dos líneas, la de datos y la de reloj están en alta, entonces cuando el maestro inicia una transmisión con uno de los esclavos primero envía la condición de start que consiste

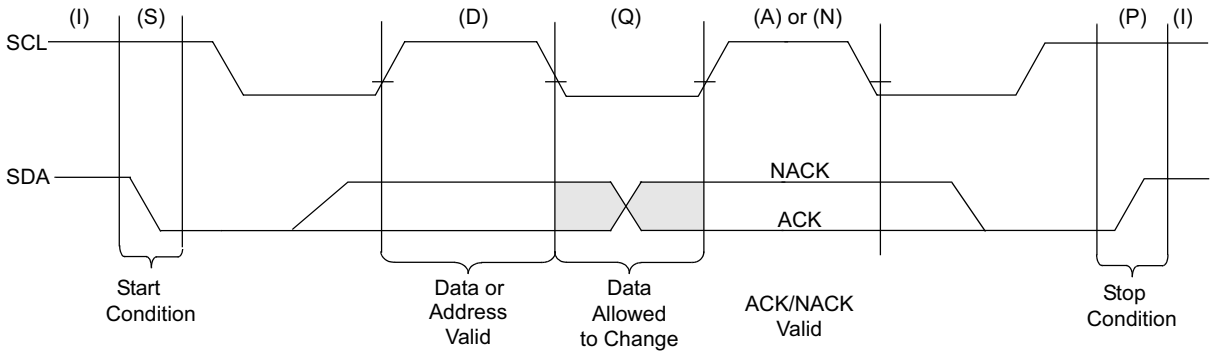



Figura 6.5: Protocolo $I^2C$

en bajar SDA, todos los datos deben estar precedidos por la condición de start y seguidamente se manda la dirección del esclavo, al ser la primera transmisión,a la vez se está transmitiendo el reloj, se considera dato válido en el ciclo positivo del reloj, un bit por cada ciclo de reloj.Los datos deben ser cambiados durante el periodo de baja de la señal de reloj.
Todos los bytes transmitidos tienen que ser reconocidos ACK o no reconocidos NACK por parte del receptor,este pondrá la linea SDA baja para un ACK o alta para un NACK.
Después de un estado WAIT, una transición de alta a baja de la linea SDA mientras que el reloj está en alta determina la condición repetida de inicio, que permite al maestro cambiar la dirección de bus sin perder el control del bus. Una transición de baja a alta en la linea SDA cuando la linea SCL está en alta es la condición de stop. Todas las transferencias de datos deben de finalizar con la condición de stop. Ambas líneas, SDA y SCL pasan a alta después de la condición de Stop y antes de la condición de Start. Si el software escribe en I2CTRN cuando la secuencia de Start está en progreso, IWCOL se activa y el contenido del buffer de transmisión es ignorado.

# *Solución: Módulo $I^2C$ como maestro (TECLADO)*

```
/***TECLADO PROTOCOLO I2C***/
#include <dspic.h>
#include "binario.h"
// Declaración de varibles globales.
volatile unsigned char cont,columna, fila,desplazamiento,pulsado;
//cont -->Variable auxiliar para calcular el desplazamiento para
//       obtener la letra seleccionada.
/*columna --> Variable auxiliar que almacena columna seleccionada
fila --> Variable auxiliar para almacenar la fila seleccionada.
desplazamiento --> Variable auxiliar que indica
                   la letra que es en cada tecla.
pulsado --> Variable auxiliar que indica si se ha pulsado otra tecla.
 */
volatile unsigned char barrido,dato,columna_nueva,fila_nueva,aux,tiempo;
```

```
/* barrido --> almacena el valor de las filas
dato --> Variable en la que se almacena el valor
        final de la letra seleccionada.
columna --> Variable auxiliar que almacena columna
             seleccionada en una nueva pulsacion para
             comparar con la anterior.
fila --> Variable auxiliar para almacenar la fila
        seleccionada en una nueva pulsacion.
aux --> Variable auxiliar que guarda las filas
         para poder comparar.
tiempo --> Variable auxiliar para poder contar 1 segundo.
*/
const unsigned char letras [] = {"␣1␣1-ABC2-DEF3-GHI4
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣-JKL5-MNO6-PQRS7TUV8-WXYZ9␣␣␣␣0000-"};
// En el vector letras se guardan todas las letras y números del
// teclado en grupos de cinco para
// poder repetir las letras si se pulsa una tecla más de cinco veces.
// Se definen l0s nombres de las entradas y salidas con las que
//se trabaja para una mejor identificación.
#define Fila0 RB0
#define Fila1 RB1
#define Fila2 RB3
#define Fila3 RB4
#define Columna0 RB10
#define Columna1 RB2
#define Columna2 RB12

// FUNCIÓN EMPLEADA

// Función que espera un tiempo para evitar rebotes de las entradas

void rebotes(void)
{
        TMR2=0;
        PR2 =1250;
// divido por 4 y programo con XT
        while(!T2IF);
// Tiempo que espera  20 ms
        T2IF  = 0;
// Borro flag
        TMR2=0;
        T2TCKPS0=0;
// Preescala 1:64, este bit se borra con la inicializacion del temp2
        T2TCKPS1=1;
}

//Para introducir un pequeño retardo para que
// le de tiempo a transmitir todos los datos al maestro

void Pausa(void)
{
TMR2=0;
while(TMR2<=5);
TMR2=0;
T2TCKPS0=0;
// Preescala 1:1, este bit se borra con la inicializacion del temp2
T2TCKPS1=0;
}//Fin pausa
//Funcion que se encarga de mandar los datos
//Función encargada de enviar el dato con protocolo I2C
void enviar_dato(unsigned char dato)
{
```

```
        I2C_SEN=1;
//Se envíabit de START, se apaga por hardware
        Pausa();
//Pausa para que envie todos los datos
        I2CTRN=B01110100;
//01110100 Enviamos direccion del esclavo
        while(I2C_TRSTAT==1);
//Espero a que termine de mandar dato y recibir el correcto ACK
        Pausa();
        I2CTRN=B11000000;
//Byte de control
        while(I2C_TRSTAT==1);
//Espero a transmitir y recibir ack
        Pausa();
//Mando dato deseado
        I2CTRN=dato;
        while(I2C_TRSTAT==1);
//Espero a terminar de trasmitir dato y recibir ack
        Pausa();
        I2C_PEN=1;
//Mando bit de stop, borrado por hardware
        while(I2C_PEN==1);
//Espero hasta que lo mande
        Pausa();
}//Fin Enviar_dato

//PROGRAMA PRINCIPAL
void main()
{
        ADPCFG = 0XFFFF;
//Se utiliza las patillas del puertoB como E/S digitales.
        I2CON = 0X8020;
        TRISB=0x001B;
//Configuracion el módulo I2C
        I2C_I2CEN = 1;
//Habilitación del I2C modo maestro
        I2C_I2CSIDL=1;
//Con 0 continua cuando esta en modo IDLE por defecto esta a 0
        I2C_A10M=0;
// Se identifica la direccion de 7 bits(Si es 1 es de 10 bits)
        I2C_IPMIEN=1;
//Acepta todas las direcciones
        I2C_ACKDT=0;
        I2CBRG=8;
//Si la frecuencia de instruccion es de 1MHz Y queremos una frecuencia de
//oscilacion de 100KHz necesitamos 8 en este registro.Consultar datasheet
        I2CCON=I2CCON&0xFFE0;
        TMR2=0;
        while(1)
//Se realiza un barrido en un bucle infinito
{
        columna = 0;
// Inicializa variables
        fila  = 0;
        cont  = 0;
// Mientras no se pulse ninguna tecla
// o se pulse un * sigo haciendo el bucle
        while (( columna == 0) && (fila == 0)||
              (columna ==1) && (fila==4))
        {
                columna = 0;
// Se hace un barrido activando todas las
```

```
//columnas de forma secuencial
//y se comprueba que letra está activada
                if(pulsado == 0)
// Si no se ha pulsado otra tecla entro en el if                    143
                {
                Columna0 = 1;
                Columna1 = 1;
                Columna2 = 1;
// Activo todos las columnas                                        148
            aux = ((PORTB & B00000011 ) |
                        ((PORTB & B00011000) >> 1));
// aux es el valor de las filas
            while ((Fila0==0)&&(Fila1==0)&&
                  (Fila2==0)&&(Fila3==0))                           153
// Espero a que se pulse alguna tecla
            {
                aux = ((PORTB & B00000011 ) |
                            ((PORTB & B00011000) >> 1));
            }                                                       158
// Si se detecta que se pulsa alguna tecla esperamos los rebotes
// y se vuelve a leer y nos quedamos con el ultimo valor leído

                columna = 0;
// Se inicializa para volver a usarlas                              163
                fila = 0;
                rebotes();
// Se llama a la función que espera por los rebotes
// Se vuelve a leer el teclado para quedarnos con el dato
//correcto  hacer el barrido de nuevo.                              168
                Columna0 = 1;
// colunma0 = 1 es la que está activada
                Columna1 = 0;
// las otras columnas desactivadas
                Columna2 = 0;                                       173
                barrido = ((PORTB & B00000011 ) |
                        ((PORTB & B00011000) >> 1));
//Se queda solo con el valor de las filas empleando una máscara
                barrido = 0x0f & (barrido);
// Trabajamos con lógica positiva en las comparaciones por          178
// lo que negamos la variable obtenida.
// Si la que esta activa es la columna cero en la variable
// columna indico que es la primera columna.
                if(barrido != 0)
                {                                                   183
                   columna = 1;
                }
                else     // Sino testeo otra columna
                {
                   Columna0 = 0;   //  Columna 1  = 0 y el resto a 1   188
                   Columna1 = 1;
                   Columna2 = 0;
                   barrido = ((PORTB & B00000011 ) |
                     ((PORTB & B00011000) >> 1));
                   barrido = 0x0f & (barrido);                      193
                   if(barrido != 0)
// Está activa la columna dos lo guardo en la variable columna
                   {
                         columna = 2;
                   }                                                198
                   else
                   {
                         Columna0 = 0;
```

```
                        Columna1 = 0;  // Columna 2= 0 y el resto a 1
                        Columna2 = 1;
                        barrido = ((PORTB & B00000011 ) |
                      ((PORTB & B00011000) >> 1));
                        barrido = 0x0f & (barrido);
                        if(barrido != 0)  // Está activa la columna tres
                        {
                            columna = 3;
                    }
                   }
//Aquí ya se sabe que columna está activada
              }

              fila = barrido;
// Barrido tiene  la fila activada y el resto a 0 Comparamos y buscamos
// que fila es la que esta activada. Para saber que fila está activada
// solo tenemos que ver el valor almacenado en Barrido
// Si fila ya es 1 en el puerto tenemos 00000001 si es 2 -->00000010
// pero si la fila es 3 o 4 no coincide con 00000011 y 00000100
// por lo que se testea lo que tiene que valer
// y guardamos nosotros directamente el valor en la variable
// fila para que coincida con los valores en binario
               if ( fila == B00000100)
               {
                      fila = 3;
               }
               if ( fila == B00001000)
               {
                      fila = 4;
               }
        }     // cierro if del  pulsado == 0
        else  // Si se ha pulsado una nueva guardo
              // las anteriores en columna y fila.
        {
               columna = columna_nueva;
               fila=fila_nueva;
        }
    }
    Columna0 = 1;
    Columna1 = 1;
    Columna2 = 1;  // Activo todos las columnas
    aux = ((PORTB & B00000011 ) | ((PORTB & B00011000) >> 1));
                   // aux es el valor de las filas
   while ((Fila0==1)||(Fila1==1)||(Fila2==1)||(Fila3==1))
                   // Si sigue activada espero a que se levante
                   // el dedo para no contarla como varias pulsaciones.
        {
           aux = ((PORTB & B00000011 ) | ((PORTB & B00011000) >> 1));
        }
        TMR2= 0;
        T2TCKPS0=0;        // Preescala 1:64
        T2TCKPS1=1;
        PR2= 31250;
        pulsado = 0;   //Inicialización
        desplazamiento  = 0;
        tiempo = 0;
        while((tiempo != 2) && !pulsado)
// Si todavía no hemos contado un segundo
//y se pulsa otra tecla entro en el while
        {
//Se repite hacer el barrido y se guarda en
//columna_nueva y fila_nueva igual que antes.
```

```
        Columna0 =1;
            Columna1 = 1;
            Columna2 = 1;         // Activo todos las columnas
        aux = ((PORTB & B00000011 ) | ((PORTB & B00011000) >> 1));
// aux es el valor de las filas                                          268
                while ((Fila0==0)&&(Fila1==0)&&(Fila2==0)&&
                    (Fila3==0) && tiempo !=2)
// Espero a que se pulse alguna tecla
        {
            aux = ((PORTB & B00000011 ) | ((PORTB & B00011000) >> 1));   273
            if(T2IF == 1) // Se espera otro medio segundo
            {
                T2IF = 0;
                PR2= 31250;
                tiempo++;                                                 278
                TMR2=0;
                T2TCKPS0=0;
                T2TCKPS1=1;
            }
        }                                                                 283
        if ( tiempo != 2)
// Si ya paso el segundo no hago el barrido
        {
            rebotes();
// Primero se espera a que pasen los rebotes                             288
//y después leemos la tecla pulsada.
            Columna0 = 1;         // colunma0 = 1
            Columna1 = 0;
            Columna2 = 0;
            barrido = ((PORTB & B00000011 ) |                             293
            ((PORTB & B00011000) >> 1));
//Se queda solo con el valor de las filas
            barrido = 0x0f & (barrido);
            if(barrido != 0)    // Está activa la columna uno
            {                                                             298
                columna_nueva = 1;
            }
            else
            {
                Columna0 = 0;    //  Columna 1 = 1 y el resto a 0         303
                Columna1 = 1;
                Columna2 = 0;
                barrido = ((PORTB & B00000011 ) |
                            ((PORTB & B00011000) >> 1));
                //Se queda solo con el valor de las filas                 308
                barrido = 0x0f & (barrido);
                if(barrido != 0)      // Está activa la columna dos
                {
                   columna_nueva = 2;
                }                                                         313
                else
                {
                    Columna0 = 0;
                    Columna1 = 0;         // Columna 2= 1 y el resto a 0
                    Columna2 = 1;                                         318
                    barrido = ((PORTB & B00000011 ) |
                               ((PORTB & B00011000) >> 1));
                   //Se queda solo con el valor de las filas
                    barrido = 0x0f & (barrido);
                    if(barrido != 0)      // Está activa la columna tres  323
                    {
                        columna_nueva = 3;
```

```
                    }
                            }
                     }
              fila_nueva=barrido;
// Barrido tiene a uno la fila activada
//y el resto a 0
              if ( fila_nueva == B00000100)
              {
                     fila_nueva = 3;
              }
              if ( fila_nueva == B00001000)
              {
                     fila_nueva = 4;
              }

              if ( (fila_nueva != 0) && (columna_nueva != 0))
              {
//  Si pulso una tecla distinta o pulso #
//salgo del bucle de espera, poniendo pulsado = 1
                if( (( fila_nueva!= fila) || (columna_nueva!=columna))||
                  ( fila_nueva == 4  &&  columna_nueva==3))
                {
                    pulsado = 1;
                }
                else // Sino se cumple ninguna de las anteriores
                {
                    desplazamiento++;
// Es la misma tecla e incrementamos
// desplazamiento para pasar a la
// siguiente letra o numero de esa tecla
                    tiempo = 0;
// Variable auxiliar para poder
//contar 1 segundo, cuenta 0.5s en 0,5s 1=2*0.5
                    Columna0 = Columna1 = Columna2 = 1;
// Activo todos las columnas
                    aux = ((PORTB & B00000011 ) |
                             ((PORTB & B00011000) >> 1));
// aux es el valor de las filas
                    while (aux != B00000000)
// Si sigue activada espero a que se levante
// el dedo para no contarla como varias pulsaciones
                    {
                         aux = ((PORTB & B00000011 ) |
                                 ((PORTB & B00011000) >> 1));
                    }
// Ya se levantó el dedo ahora espero por
// los rebotes que ha podido causar esta pulsación
                    TMR2=0;
                    T2TCKPS0=0;
                    T2TCKPS1=1;
                    PR2= 31250;
                }
            }
            if(T2IF == 1)        // Se espera otro medio segundo
            {
                T2IF = 0;
                tiempo++;
                TMR2=0;
                T2TCKPS0=0;
                T2TCKPS1=1;
                PR2= 31250;
            }
```

```
        }
         }
// Las teclas 7 y 9 tiene 5 combinaciones distintas de
// letras mientras que las demás solo permiten 4
    if ( ((fila == 3) && (columna == 1))||
        ( (fila == 3) && (columna == 3)))
    desplazamiento = desplazamiento%5;
// Si  desplazamiento se pasa de cinco vuelve a
// empezar direccionamiento circular
    else
    desplazamiento = desplazamiento%4;
// Lo mismo para los grupos de letras de 4
    cont = ((columna-1)*5)+ (15 * (fila-1)) + desplazamiento;
// Una vez obtenido el desplazamiento , la columna y fila,
// Se busca en el vector el valor y guardamos su contenido
    if (fila == 0 && columna == 0)
// Si no se ha pulsado ninguna tecla
    cont = desplazamiento;
    if (!(fila == 4 && columna == 3) &&
        !(fila_nueva == 4 && columna_nueva == 1) )
// Si no es *( tengo que borrar)  , ni # lo guardo
    {
        dato = letras[cont];
/***************PROTOCOLO************************/
                enviar_dato(dato);
        }
        else                    // si es * o se pulso # no lo almaceno
                pulsado = 0;
//Inicializamos la variable pulsado para volver a pulsarlo
 } //while(1)
}//main
```

# Solución: Módulo $I^2C$ como esclavo(LCD)

```
// Se debe incluir en el fichero en que se vaya a usar, una sola vez
#include <dspic.h>
#include "binario.h"

// Definimos las conexiones
#define LCD_Data     PORTB        /* Puerto de datos */
#define LCD_RS       RB9          /* Comando o dato */
#define LCD_RW       RB10         /* Lectura/Escritura LCD */
#define LCD_E        RB8          /* Habilita LCD */

// COMANDOS para el LCD HD44780
#define LCDClear     B00000001    /* Borra y cursor al principio */
#define LCDCasa      B00000010    /* Cursor al principio */
#define LCDInc       B00000110    /* Selecciona incrementos */
                     /*     **    Bit 1 I/D (=0 dec posición)    */
                     /*                     (=1 inc posición)*   */
                     /*           Bit 0 S   (=0 no despl. disp.)**/
                     /*                     (=1 desplaza disp.)  */
#define LCDDec       B00000100    /* Sel. decrementos, no despl.  */
#define LCDOn        B00001100    /* Control del display          */
                     /*    ***    Bit 2 D  (=0 display apagado)  */
                     /*                    (=1 disp.encendido)*   */
                     /*           Bit 1 C  (=0 sin cursor)*       */
                     /*                    (=1 con cursor)        */
                     /*           Bit 0 B  (=0 sin parpadeo)*     */
                     /*                    (=1 con parpadeo)      */
#define LCDOff       B00001000    /* Disp. off, sin cursor, no parp.  */
#define CursOn       B00001110    /* Disp. on, con cursor, no parp.   */
#define CursOff      B00001100    /* Disp. on, sin cursor, no parp.   */
#define CursBlink    B00001111    /* Disp. on, con cursor, con parp.  */
#define LCDIzquierda B00011000    /* Desplaz. de cursor o disp.   */
                     /*    **--   Bit 4 S/C (=0) Mueve el cursor */
                     /*                     (=1) Despl. el disp.**/
                     /*           Bit 3 R/L (=0) A la izquierda* */
                     /*                     (=1) A la derecha    */
#define LCDDerecha   B00011100    /* Desplaza a la derecha el disp.   */
#define CursIzquierda B00010000   /* Cursor a la izquierda            */
#define CursDerecha  B00010100    /* Cursor a la derecha              */
#define LCDFuncion   B00111000    /* Bit 4 DL (=0 4 bit bus)          */
                     /*  ***--                (=1 8 bit bus)*         */
                     /*             Bit 3 N   (=0 disp. de 1 linea)   */
                     /*                       (=1 disp. de 2 líneas)* */
                     /*             Bit 2 F   (=0 font 5x8)*          */
                     /*                       (=1 font 5x10)          */
#define LCDCGRAM     B01000000    /* Pone dirección CGRAM */

#define LCDLinea1    B10000000    /* Posición  0 (00h). Línea 1   */
#define LCDLinea2    B11000000    /* Posición 64 (40h). Línea 2   */

// Subrutinas para el manejo del LCD ................
void Pausa_5ms(void)
{
TMR2=0;            // Inicializa temporizador
while(TMR2<=5000);// 5ms=1us*Cuenta --> Cuenta =5000
TMR2=0;
T2TCKPS0=0;        // Preescala 1:1
T2TCKPS1=0;        // este bit se borra con la inicializacion del temp2
}
void LCD_Port(void)
```

```
{
    TRISB = 0x0000;// PORTB todo salidas
    LCD_E = 0;
        TRISF = 0x0FFFF;
}
void LCD_Comando_Inicio(unsigned char c)
{
    LCD_RW   = 0;   // Escribe
    LCD_RS   = 0;   // Comando
    LCD_Data = c;   // Pone comando
                    // Durante el arranque no podemos leer el estado
    LCD_E    = 1;   // Habilita LCD
    LCD_E    = 0;   // Deshabilita LCD
}
void LCD_Comando(unsigned char c)
{
    LCD_RW   = 0;   // Escribe
    LCD_RS   = 0;   // Comando
    LCD_Data = c;   // Pone comando
    Pausa_5ms();
    LCD_E    = 1;   // Habilita LCD
    LCD_E    = 0;   // Deshabilita LCD
}
void LCD_Caracter(unsigned char c)
{
    LCD_RW   = 0;   // Escribe
    LCD_RS   = 0;   // Comando
    Pausa_5ms();
    LCD_Data = c;   // Pone el dato
    LCD_RS   = 1;   // Dato
    asm("NOP");
    LCD_E    = 1;   // Habilita LCD
    asm("NOP"); // Esperamos un pequeño retardo
    asm("NOP");
    LCD_E    = 0;   // Deshabilita LCD
}
void LCD_Reset(void)
{
    LCD_Comando_Inicio(LCDFuncion); // Define modo de operación
    Pausa_5ms();
    LCD_Comando_Inicio(LCDFuncion);
    Pausa_5ms();
    LCD_Comando_Inicio(LCDFuncion);
    Pausa_5ms();
    LCD_Comando_Inicio(LCDFuncion);
    Pausa_5ms();
    LCD_Comando(LCDClear );         // Borra pantalla
    LCD_Comando(LCDInc   );         // Autoincremento posición
    LCD_Comando(CursBlink);         // Cursor parpadeando
}
unsigned char recibir(void)
{
        volatile unsigned char dato;
        I2C_RCEN = 1; // Recepcion habilitada
        while(!I2C_S);// Espera e recibir la condicion de star
                      // Ahora recibe la direccion
        while(!I2C_RBF);
        dato=I2CRCV;  // Lee el dato recibido
        I2C_I2COV = 0;
        I2C_RCEN = 1;
        while(I2C_RBF == 0);// Espera a recibir el dato
        dato=I2CRCV;        // Lee el byte de control recibido
```

```
        I2C_I2COV = 0;
        I2C_RCEN = 1;
        while(I2C_RBF == 0);// Espera a recibir el dato
        dato=I2CRCV;        // Lee el dato recibido
        I2C_I2COV = 0;
        I2C_RCEN = 1;
        while(I2C_P == 0); // Bit de Stop
        // Cuando termine de recibir devuelvo el dato
        return dato;
}
void configura_i2c(void)
{
        volatile unsigned char dato;
        I2C_I2CEN = 1;  // I2C habilitado
        I2C_A10M = 0;   // Transmision de 7 bits
        I2C_ACKDT = 0;  // Envia ACK
        I2C_IPMIEN=1;   // Acepta todas las direcciones
        I2CADD=B00111010;
        I2CBRG=8;       // Configuracion de la velocidad
        dato=I2CRCV;
        I2C_I2COV = 0;
}
//-------------------------------------------------------------

volatile unsigned char cursor;  // Almacena la posición actual

void main()
{
        volatile unsigned char dato;
        T2CON = 0X8000; // Ton = 1; TCKPS0=0 y TCKPS1 = 1 Escala 1:1
        ADPCFG = 0xffff;
        SWDTEN = 0;     //Desahabilita el perro guardian
        TRISB = 0x0000;
        LCD_Port();
// Rutina que configura los puetos empleados por el LCD
//Esperamos 15ms hasta que Vcc alcance 4,5 V para que funcione bien el LCD
        Pausa_5ms();
        Pausa_5ms();
        Pausa_5ms();
        LCD_Reset();    // Rutina que resetea el LCD
        configura_i2c();
        while(1)
        {
                dato= recibir();
                if(cursor==16)  // Cunado el cursor sea 16 cambio de linea
                {
                        LCD_Comando(LCDLinea2);
                }
                if(cursor==32)  // Cuando el cursor es 32 borro la pantalla
                {
                        LCD_Comando(LCDClear);
                        LCD_Comando(LCDLinea1);
                        cursor=0;
                }
                LCD_Caracter(dato);     // Envio el dato
                cursor ++;              // Actualizo cursor
}// while
}//main
```

A continuación se muestra una foto de la placa final:

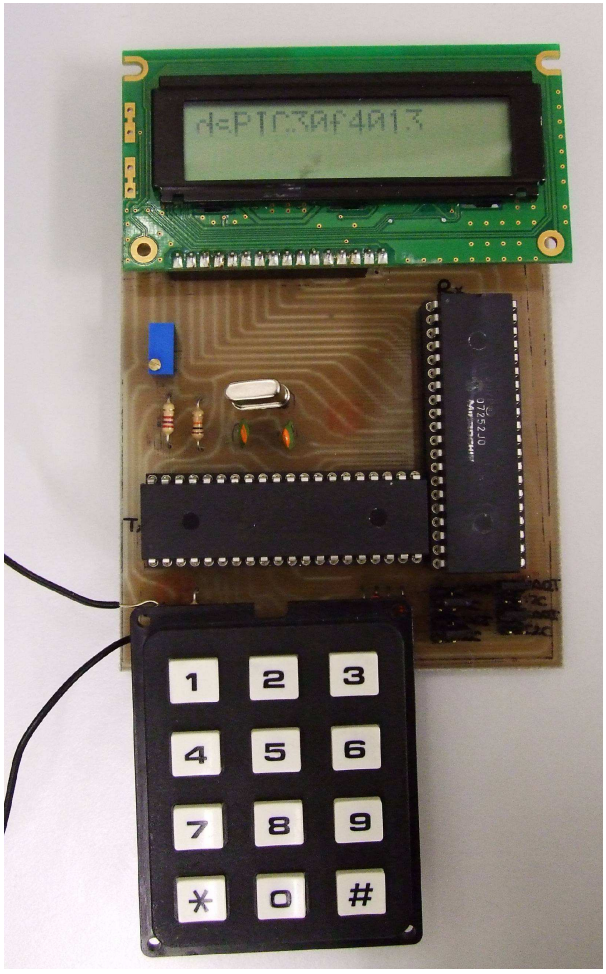



Figura 6.6: Placa de pruebas del módulo $I^2C$

## 6.4. Práctica: Módulo comparador de salida (dsPIC) . Control de motores DC.

Se desea controlar la velocidad y el sentido de giro de un motor de continua cuyas especificaciones se adjuntan posteriormente. Cuando la corriente eléctrica pasa a través de un hilo conductor dentro de un campo magnético que producen un par de fuerzas que hace mover el motor. Si cambia el sentido de la corriente, cambiará el sentido de giro del motor. Existe una relación entre la corriente que pasa por el devanado y la fuerza generada. Además se da el hecho de que a mayor tensión de alimentación mayor será la velocidad de giro del motor. Existirá una tensión crítica a partir de la cual el motor comenzara a girar y una tensión nominal (y máxima) de funcionamiento con par máximo.
El esquema de funcionamiento de los motores de continua se pude ver en la siguiente gráfica:

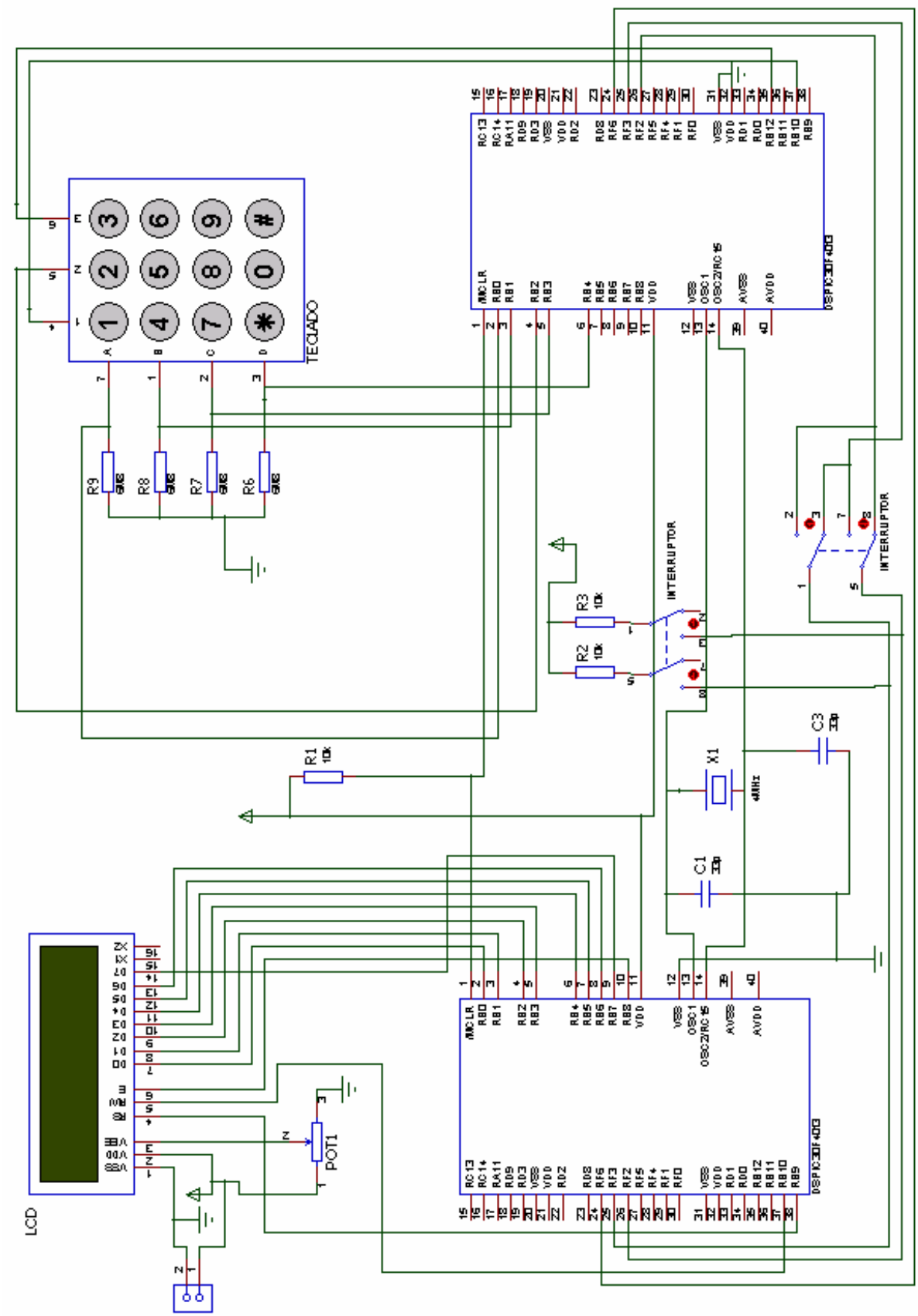



Figura 6.7: Esquema hardware de la placa de pruebas del módulo $I^2C$

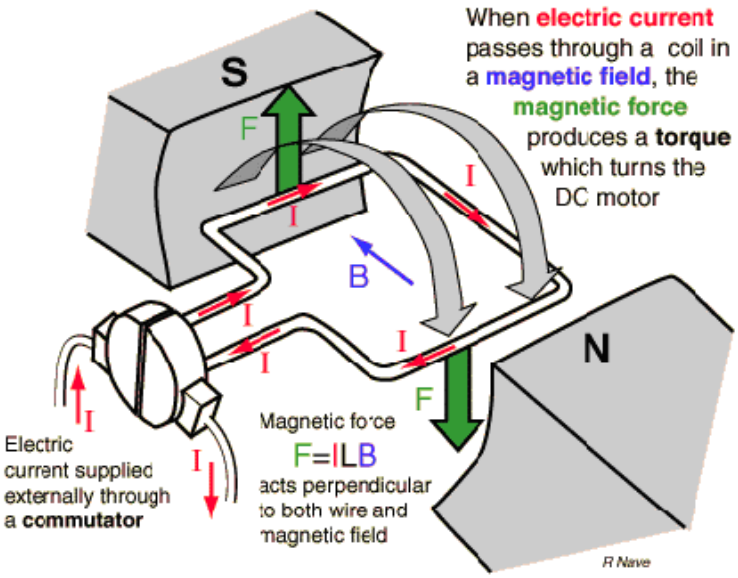



Figura 6.8: Funcionamiento de un motor de continua.

En la siguiente figura se ve un desglose típico de un motor de continua:

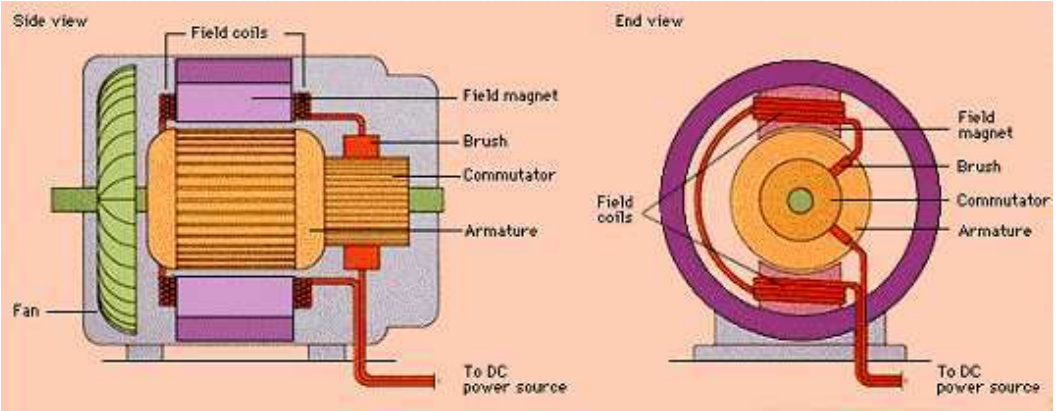



Figura 6.9: Desglose de un motor de continua.

## ETAPA DE POTENCIA.

La etapa de potencia típica para la alimentación de un motor de continua es la que se describe a continuación utilizando el integrado L293 cuya primera hoja de datos se adjunta. Se trata de un puente H que nos permita cambiar el sentido de giro del motor. Los diodos que aparecen son necesarios para evitar los picos de corriente generados por el motor, si se trabaja con el integrado L293B (El L293D lleva integrados estos diodos).
Se puede emplear el diodo 1N4007 disponible en el laboratorio.

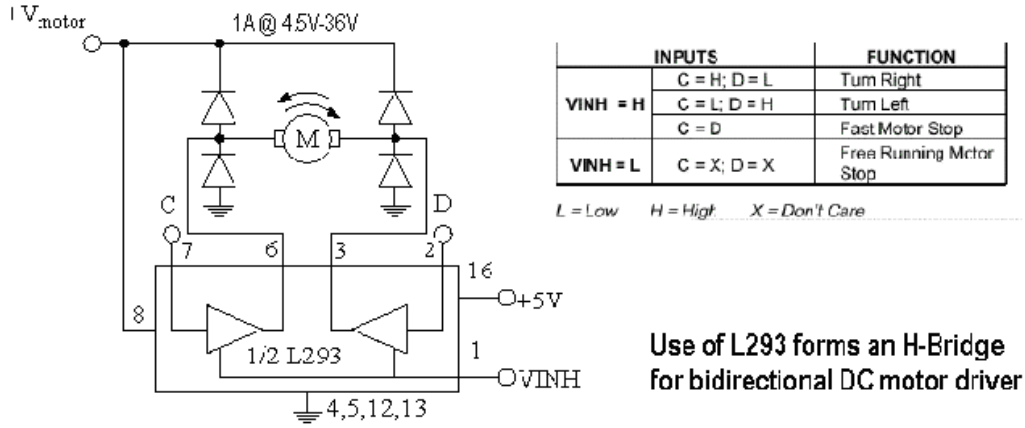



Figura 6.10: Integrado L293B.

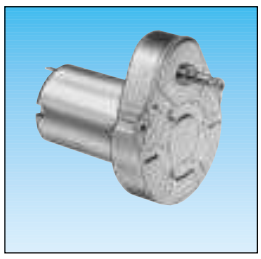
Se pide el diseño del software para controlar no solo el puente H que haga girar el motor en los dos sentidos sino también las rutinas necesarias para controlar la velocidad del motor empleando la modulación de la anchura de los pulsos PWM. Con la modulación PWM conseguimos variar la velocidad del motor como si la tensión efectiva aplicada fuera menor, pero con la ventaja de que la tensión aplicada es la nominal lo que nos da el par máximo aún cuando la velocidad de giro sea menor. Además nos permitirá hacer girar el motor a una velocidad inferior a la lograda con la tensión umbral de arranque del motor. La frecuencia de la señal PWM será de 1KHz. Se empleará una entrada para seleccionar el sentido de giro del motor. Otra patilla será el freno del motor, de forma que cuando se active el motor se frene rápidamente. Otra entrada incrementará en una unidad el ciclo de trabajo, mientras que otro lo decrementará en una unidad. Inicialmente el ciclo de trabajo estará en el 50
Se proporcionara el puente H montado y el motor de continua según las especificaciones.

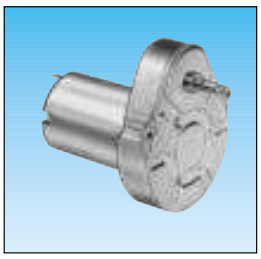
**Para sacar nota:**

Utilizar el teclado alfanumérico para controlar el ciclo de trabajo del motor de continua de tal manera que se pueda introducir por el teclado ciclos de trabajo del 00 al 99 por ciento y de esta manera varíe la velocidad del motor. Por defecto el motor girará con una frecuencia de PWM de 1KHz y ciclo de trabajo 50.
¿cual es el valor más pequeño del ciclo de trabajo para el que sigue girando el motor? Puedes ayudarte de los displays de 7 segmentos (primero direccionas el dígito más significativo, activando el cátodo, sacas el dato más significativo, luego el dígito menos significativo y sacas el dato, y repites indefinidamente para que sea visible).¿Se ha mejorado la tensión umbral de arranque del motor?

¿Que ocurrirá si la frecuencia de la señal PWM fuese mucho mayor?.Compruébalo.

- **D.C. geared motors : 0.3 to 430 rpm**
- **Gearbox torque ratings from : 0.5 to 2 Nm, high-performance plastic gears**
- **Motors : max. usable power 1 to 3.9 W interference suppression on standard products**





**Applications**

- Blood analysers
- Machines for making spectacle lenses
- Microvalves for heating systems
- Change-giving systems
- Packaging machines
- Moving light displays
- Etc.

**Made to order products, available on request**

**Motors** :
- other supply voltages
- motors with 2 ball bearings
- shaft lengths at front and/or rear
- connection by radial tags or leads
- specific interference suppression
- encoder : 5 or 12 pulses per revolution

**Gearboxes :**
- special shaft
- ball bearings
- special lubrication
- friction clutch
- M3 gearbox fixings
- other speeds

**Types**

| | | 82 861 0 | 82 861 0 | 82 841 0 | 82 841 0 |
|---|---|---|---|---|---|
| **Nominal voltage** | | 12 V | 24 V | 12 V | 24 V |
| **Output speeds** (rpm) | Ratios (i) | **Part numbers** | | | |
| **430** | 10 | 82 861 006 | 82 861 015 | ● | ● |
| **215** | 20 | 82 861 007 | 82 861 016 | ● | ● |
| **179** | 24 | ● | ● | ● | ● |
| **143** | 30 | 82 861 008 | 82 861 017 | ● | ● |
| **108** | 40 | 82 861 009 | 82 861 018 | ● | ● |
| **90** | 48 | ● | ● | ● | ● |
| **54** | 80 | 82 861 010 | 82 861 019 | ● | ● |
| **49** | 90 | ● | ● | ● | ● |
| **29** | 150 | ● | ● | ● | ● |
| **27** | 160 | — | — | — | — |
| **22** | 200 | 82 861 011 | 82 861 020 | ● | ● |
| **13** | 320 | — | — | — | — |
| **11** | 375 | 82 861 012 | 82 861 021 | ● | ● |
| **8.6** | 500 | 82 861 013 | 82 861 022 | ● | ● |
| **7.2** | 600 | — | — | — | — |
| **5.8** | 750 | ● | ● | ● | ● |
| **5.4** | 800 | — | — | — | — |
| **3.6** | 1200 | 82 861 014 | 82 861 023 | ● | ● |
| **2.9** | 1500 | — | — | — | — |
| **1.8** | 2400 | ● | ● | ● | ● |
| **0.90** | 4800 | — | — | — | — |
| **0.80** | 5400 | ● | ● | ● | ● |
| **0.36** | 12000 | ● | ● | ● | ● |

**Standard gearbox shaft : see dimensions**

**General characteristics**

| | | | | | |
|---|---|---|---|---|---|
| Motor | | 82 860 0 | 82 860 0 | 82 840 0 | 82 840 0 |
| Gearbox | | 81 021 0 | 81 021 0 | 81 021 0 | 81 021 0 |
| Maximum permitted torque on gearbox for continuous rating (for 1 million revolutions) | Nm | 0.5 | 0.5 | 0.5 | 0.5 |
| Axial load (static) | daN | 1 | 1 | 1 | 1 |
| Radial load (static) | daN | 8 | 8 | 8 | 8 |
| Maximum usable power | W | 3.9 | 3.9 | 1 | 1 |
| Maximum usable power | W | 3 | 3 | 0.9 | 0.9 |
| Gearbox case temperature rise | °C | 50 | 50 | 50 | 50 |
| Weight | g | 160 | 160 | 130 | 130 |

**Options : for standard products non stocked**

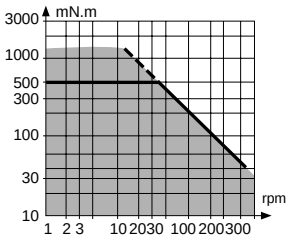
| | | | | | | |
|---|---|---|---|---|---|---|
| Output shaft | Ø 4 mm | 79 200 967 | ● | ● | ● | ● |
| | | 79 200 779 | ● | ● | ● | ● |
| | Ø 6 mm | 70 999 421 | ● | ● | ● | ● |
| | | 79 202 573 | — | — | — | — |
| With magnetic encoder 1 pulse per revolution | | | 82 861 5 | 82 861 5 | — | — |

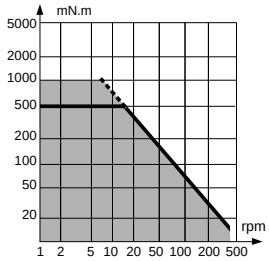
**Nominal speed and torque curves**

The shaded zone represents the operating range of the geared motor.

The horizontal line marks the maximum torque available in continuous duty cycle for a given life.

For higher torque ratings, service life will be reduced.




**Other information**

Basic principles : see page 1/7.
Motor 82 840 0, 82 860 0 : see page 1/9.

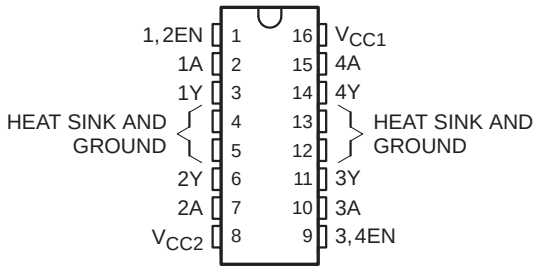
Figura 6.11: Hoja de características del motor modelo 82861010.

# L293
# QUADRUPLE HALF-H DRIVER

SLRS005 – SEPTEMBER 1986 – REVISED MAY 1990

- **1-A Output Current Capability Per Driver**
- **Pulsed Current 2-A Driver**
- **Wide Supply Voltage Range: 4.5 V to 36 V**
- **Separate Input-Logic Supply**
- **NE Package Designed for Heat Sinking**
- **Thermal Shutdown**
- **Internal ESD Protection**
- **High-Noise-Immunity Inputs**
- **Functional Replacement for SGS L293**

**NE PACKAGE**
**(TOP VIEW)**




## description

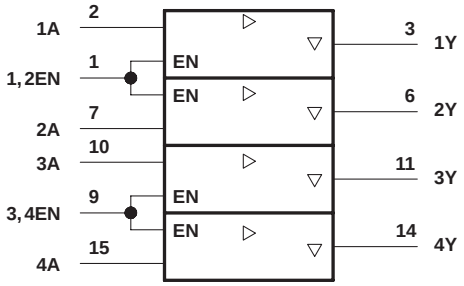
The L293 is a quadruple high-current half-H driver designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. It is designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in a high-impedance state. With the proper data inputs, each pair of drivers form a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

External high-speed output clamp diodes should be used for inductive transient suppression. A $V_{CC1}$ terminal, separate from $V_{CC2}$, is provided for the logic inputs to minimize device power dissipation.

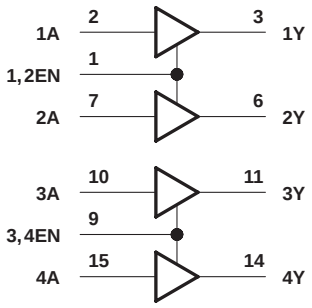
The L293 is designed for operation from 0°C to 70°C.

## logic symbol†




† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC publication 617-12.

## logic diagram




**FUNCTION TABLE**
**(each driver)**

| INPUTS‡ | | OUTPUT |
|---|---|---|
| A | EN | Y |
| H | H | H |
| L | H | L |
| X | L | Z |

H = high-level, L = low-level,
X = irrelevant, Z = high-impedance (off)
‡ In the thermal shutdown mode, the output is in the high-impedance state regardless of the input levels.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 1990, Texas Instruments Incorporated

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265



Figura 6.12: Hoja de características del puente L293.

# Solución: Control de un motor de continua.

```
#include <dspic.h>
#include "binario.h"
//Modulación de anchura de pulsos
//Para un pulso mas ancho, el valor eficaz de la señal es mayor
// que para un pulso mas estrecho, por lo que variando
//la anchura del pulso del tren de pulsos de la señal se puede
//conseguir una señal cuyo valor eficaz(valor medio)
//varie de la forma deseada.
//El movimiento de un motor eléctrico se consigue mediante la
//variación continua de un campo eléctrico o magnético
//para obtener asi un campo rotatorio. Variando la corriente de
// alimentacion del motor se consigue variar la velocidad del motor
//y se va acontrolar mediante PWM, variando el porcentaje de tiempo
//de la señal rectangular en estado en alta y baja , es decir,
// variando la anchura del pulso de la señal, varia la potencia que
// se entrega al motor variando la velocidad de giro de este
//que se puede controlar con mucha precision.

// Declaracion de varibles globales
volatile unsigned char cont,columna, fila,desplazamiento,pulsado;
//ont -->Variable auxiliar para calcular el desplazamiento para obtener
// la letra seleccionada
/*columna --> Variable auxiliar que almacena columna seleccionada
fila --> Variable auxiliar para almacenar la fila seleccionada
desplazamiento --> Variable auxiliar que indica la letra
que es en cada tecla
pulsado --> Variable auxiliar que indica si se ha pulsado otra tecla
 */
volatile unsigned char barrido,dato,columna_nueva,fila_nueva,aux,tiempo;

/* barrido --> almacena el valor de las filas
dato --> Variable en la que se almacena el valor final de la
letra seleccionada
columna --> Variable auxiliar que almacena columna
seleccionada en una nueva
// pulsacion para comparar con la anterior
fila --> Variable auxiliar para almacenar la fila
 seleccionada en una nueva pulsacion
aux --> Variable auxiliar que guarda las filas
para poder comparar
tiempo --> Variable auxiliar para poder contar 1 segundo
*/
unsigned int letras[] = {1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,4,5,
5,5,5,5,6,6,6,6,6,7,7,7,7,7,8,8,8,8,8,9,9,9,9,9,0,0,0,0,0,0,0,0,0,0};
// Adapt0 letras para no modificar el codigo de barrido del teclado
// En el vector letras se guardan todas las letras ynúmeros
// del teclado en grupos de cinco para poder repetir las
// letras si se pulsa una tecla más de cinco veces
// Defino los nombres de las entradas y salidas con
//las que trabajare para una mejor identificación.
#define Fila0 RB0
#define Fila1 RB1
#define Fila2 RB3
#define Fila3 RB4
#define Columna0 RB10
#define Columna1 RB2
#define Columna2 RB12

// FUNCIONES EMPLEADAS
```

```
// Función que espera un tiempo para evitar rebotes de las entradas de 20
void rebotes(void)
{
        TMR2=0;
        PR2 =1250;
        while(!T2IF);           // Tiempo que espera  20 ms
        T2IF  = 0;              // Borro flag
        T2TCKPS0=0;             // Preescala 1:64, este bit se borra con
        T2TCKPS1=1;             // la inicializacion del temp2
}
void modificar_velocidad(unsigned int ciclo)
    //le llega ciclo de trabajo en %
{
        OC1RS=(10*ciclo);
        while(T3IF==0);
//Espero porque sino el valor que pone no es el correcto
        T3IF=0;

}//FIN FUNCION

//PROGRAMA PRINCIPAL BARRIDO TECLADO
unsigned int ciclo;
//variable auxiliar para control de motor
main()
{
double var;
//variable auxiliar     para modificar ciclo de trabajo
int segundo;
//variable auxiliar para leer el ciclo de trabajo
        TRISB = 0x001B;
//1B Columnas son salidas y Filas entradas
        ADPCFG = 0XFFFF;
//Se utiliza las patillas del puertoB como E/S digitales.
        T2CON = 0X8020;
//Configuración del temporizador para el teclado

//CONFIGURACION PWM SIN CONTROL DE FALLO
//Dentro del registro OC1CON tenemos los siguientes bits

//OC1_OCSIDL=0;
// Desactiva el comparador en modo Idle si es 1
OC1_OCTSEL=1;
// Selecciona el temporizador que se va a usar
//entre TMR2 si 0 y TMR3 si 1
OC1_OCM0=0;
// Selecciona el modo en el que trabaja el modulo comparador
// de salida en este caso PWM
OC1_OCM1=1;
// sin control de fallo
OC1_OCM2=1;
OC1RS=0x1F3;
//Este registro de 16 bits almacena el valor de ciclo
//de trabajo actual de la señal
OC1R=0x1F3;
//contiene el ciclo de trabajo actual
// Inicialmente sera 50%, luego la mitad del perido
//de la señal PWM 999/2
// Cuando se finalice el periodo de la señal PWM se
//actualiza OC1R con OCRS
//TEMPORIZADOR 3 PARA PWM
T3CON = 0X8000; /*
    bit 15 TON: Timer On Control bit
```

```
    *   1 = Starts the timer
        0 = Stops the timer
        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 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 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)*/

PR3=0X3E7;
// Cuenta :  PWMper = 1useg * ( PER2+1)* Escala)  Escala = 1,
//PWMper=1/1KHz
//Escala=1
//PER2=999
TMR3=0;
T3IF=0;
var=0.5;
//valor inicial de ciclo de trabajo
dato=0;
segundo=0;
ciclo=0;
while(1)
//Se realiza un barrido en un bucle infinito
{
        columna = 0;
// Inicializa variables
        fila  = 0;
        cont  = 0;
// Mientras no se pulse ninguna tecla
// o se pulse un * sigo haciendo el bucle
        while (( columna == 0) && (fila == 0)||
              (columna ==1) && (fila==4))
        {
                columna = 0;
// Se hace un barrido activando todas las
//columnas de forma secuencial
//y se comprueba que letra está activada
                if(pulsado == 0)
// Si no se ha pulsado otra tecla entro en el if
                {
                Columna0 = 1;
                Columna1 = 1;
                Columna2 = 1;
// Activo todos las columnas
            aux = ((PORTB & B00000011 ) |
                        ((PORTB & B00011000) >> 1));
```

```
// aux es el valor de las filas
            while ((Fila0==0)&&(Fila1==0)&&
                  (Fila2==0)&&(Fila3==0))
// Espero a que se pulse alguna tecla
            {
                aux = ((PORTB & B00000011 ) |
                            ((PORTB & B00011000) >> 1));
            }
// Si se detecta que se pulsa alguna tecla esperamos los rebotes
// y se vuelve a leer y nos quedamos con el ultimo valor leído

                columna = 0;
// Se inicializa para volver a usarlas
                fila = 0;
                rebotes();
// Se llama a la función que espera por los rebotes
// Se vuelve a leer el teclado para quedarnos con el dato
//correcto  hacer el barrido de nuevo.
                Columna0 = 1;
// colunma0 = 1 es la que está activada
                Columna1 = 0;
// las otras columnas desactivadas
                Columna2 = 0;
                barrido = ((PORTB & B00000011 ) |
                        ((PORTB & B00011000) >> 1));
//Se queda solo con el valor de las filas empleando una máscara
                barrido = 0x0f & (barrido);
// Trabajamos con lógica positiva en las comparaciones por
// lo que negamos la variable obtenida.
// Si la que esta activa es la columna cero en la variable
// columna indico que es la primera columna.
                if(barrido != 0)
                {
                   columna = 1;
                }
                else     // Sino testeo otra columna
                {
                   Columna0 = 0;   //  Columna 1  = 0 y el resto a 1
                   Columna1 = 1;
                   Columna2 = 0;
                   barrido = ((PORTB & B00000011 ) |
                     ((PORTB & B00011000) >> 1));
                   barrido = 0x0f & (barrido);
                   if(barrido != 0)
// Está activa la columna dos lo guardo en la variable columna
                   {
                         columna = 2;
                   }
                   else
                   {
                         Columna0 = 0;
                         Columna1 = 0;  // Columna 2= 0 y el resto a 1
                         Columna2 = 1;
                         barrido = ((PORTB & B00000011 ) |
                       ((PORTB & B00011000) >> 1));
                         barrido = 0x0f & (barrido);
                         if(barrido != 0)  // Está activa la columna tres
                         {
                             columna = 3;
                     }
                    }
//Aquí ya se sabe que columna está activada
```

```
                }

                fila = barrido;
// Barrido tiene  la fila activada y el resto a 0 Comparamos y buscamos
// que fila es la que esta activada. Para saber que fila está activada
// solo tenemos que ver el valor almacenado en Barrido
// Si fila ya es 1 en el puerto tenemos 00000001 si es 2 -->00000010
// pero si la fila es 3 o 4 no coincide con 00000011 y 00000100
// por lo que se testea lo que tiene que valer
// y guardamos nosotros directamente el valor en la variable
// fila para que coincida con los valores en binario
                 if ( fila == B00000100)
                 {
                        fila = 3;
                 }
                 if ( fila == B00001000)
                 {
                        fila = 4;
                 }
          }     // cierro if del  pulsado == 0
          else  // Si se ha pulsado una nueva guardo
                // las anteriores en columna y fila.
          {
                 columna = columna_nueva;
                 fila=fila_nueva;
          }
      }
      Columna0 = 1;
      Columna1 = 1;
      Columna2 = 1;  // Activo todos las columnas
      aux = ((PORTB & B00000011 ) | ((PORTB & B00011000) >> 1));
                     // aux es el valor de las filas
    while ((Fila0==1)||(Fila1==1)||(Fila2==1)||(Fila3==1))
                     // Si sigue activada espero a que se levante
                     // el dedo para no contarla como varias pulsaciones.
          {
             aux = ((PORTB & B00000011 ) | ((PORTB & B00011000) >> 1));
          }
          TMR2= 0;
          T2TCKPS0=0;        // Preescala 1:64
          T2TCKPS1=1;
          PR2= 31250;
          pulsado = 0;   //Inicialización
          desplazamiento  = 0;
          tiempo = 0;
          while((tiempo != 2) && !pulsado)
// Si todavía no hemos contado un segundo
//y se pulsa otra tecla entro en el while
          {
//Se repite hacer el barrido y se guarda en
//columna_nueva y fila_nueva igual que antes.
          Columna0 =1;
              Columna1 = 1;
              Columna2 = 1;        // Activo todos las columnas
          aux = ((PORTB & B00000011 ) | ((PORTB & B00011000) >> 1));
// aux es el valor de las filas
                 while ((Fila0==0)&&(Fila1==0)&&(Fila2==0)&&
                     (Fila3==0) && tiempo !=2)
// Espero a que se pulse alguna tecla
          {
              aux = ((PORTB & B00000011 ) | ((PORTB & B00011000) >> 1));
              if(T2IF == 1) // Se espera otro medio segundo
```

```
            {
                T2IF = 0;
                PR2= 31250;
                tiempo++;
                TMR2=0;
                T2TCKPS0=0;
                T2TCKPS1=1;
            }
        }
        if ( tiempo != 2)
// Si ya paso el segundo no hago el barrido
        {
            rebotes();
// Primero se espera a que pasen los rebotes
//y después leemos la tecla pulsada.
            Columna0 = 1;        // colunma0 = 1
            Columna1 = 0;
            Columna2 = 0;
            barrido = ((PORTB & B00000011 ) |
            ((PORTB & B00011000) >> 1));
//Se queda solo con el valor de las filas
            barrido = 0x0f & (barrido);
            if(barrido != 0)   // Está activa la columna uno
            {
                columna_nueva = 1;
            }
            else
            {
                Columna0 = 0;   //  Columna 1 = 1 y el resto a 0
                Columna1 = 1;
                Columna2 = 0;
                barrido = ((PORTB & B00000011 ) |
                            ((PORTB & B00011000) >> 1));
                //Se queda solo con el valor de las filas
                barrido = 0x0f & (barrido);
                if(barrido != 0)     // Está activa la columna dos
                {
                   columna_nueva = 2;
                }
                else
                {
                    Columna0 = 0;
                    Columna1 = 0;        // Columna 2= 1 y el resto a 0
                    Columna2 = 1;
                    barrido = ((PORTB & B00000011 ) |
                                ((PORTB & B00011000) >> 1));
                   //Se queda solo con el valor de las filas
                    barrido = 0x0f & (barrido);
                    if(barrido != 0)     // Está activa la columna tres
                    {
                        columna_nueva = 3;
                    }
                            }
                     }
              fila_nueva=barrido;
// Barrido tiene a uno la fila activada
//y el resto a 0
              if ( fila_nueva == B00000100)
              {
                     fila_nueva = 3;
              }
              if ( fila_nueva == B00001000)
```

```
                {
                        fila_nueva = 4;
                }

                if ( (fila_nueva != 0) && (columna_nueva != 0))
                {
//  Si pulso una tecla distinta o pulso #
//salgo del bucle de espera, poniendo pulsado = 1
                  if( (( fila_nueva!= fila) || (columna_nueva!=columna))||
                    ( fila_nueva == 4  &&  columna_nueva==3))
                  {
                      pulsado = 1;
                  }
                  else // Sino se cumple ninguna de las anteriores
                  {
                      desplazamiento++;
// Es la misma tecla e incrementamos
// desplazamiento para pasar a la
// siguiente letra o numero de esa tecla
                      tiempo = 0;
// Variable auxiliar para poder
//contar 1 segundo, cuenta 0.5s en 0,5s 1=2*0.5
                      Columna0 = Columna1 = Columna2 = 1;
// Activo todos las columnas
                      aux = ((PORTB & B00000011 ) |
                                ((PORTB & B00011000) >> 1));
// aux es el valor de las filas
                      while (aux != B00000000)
// Si sigue activada espero a que se levante
// el dedo para no contarla como varias pulsaciones
                      {
                           aux = ((PORTB & B00000011 ) |
                                   ((PORTB & B00011000) >> 1));
                      }
// Ya se levantó el dedo ahora espero por
// los rebotes que ha podido causar esta pulsación
                      TMR2=0;
                      T2TCKPS0=0;
                      T2TCKPS1=1;
                      PR2= 31250;
                  }
              }
              if(T2IF == 1)        // Se espera otro medio segundo
              {
                  T2IF = 0;
                  tiempo++;
                  TMR2=0;
                  T2TCKPS0=0;
                  T2TCKPS1=1;
                  PR2= 31250;
              }
          }
           }
// Las teclas 7 y 9 tiene 5 combinaciones distintas de
// letras mientras que las demás solo permiten 4
     if ( ((fila == 3) && (columna == 1))||
         ( (fila == 3) && (columna == 3)))
     desplazamiento = desplazamiento%5;
// Si  desplazamiento se pasa de cinco vuelve a
// empezar direccionamiento circular
     else
     desplazamiento = desplazamiento%4;
```

```
// Lo mismo para los grupos de letras de 4
    cont = ((columna-1)*5)+ (15 * (fila-1)) + desplazamiento;
// Una vez obtenido el desplazamiento , la columna y fila,
// Se busca en el vector el valor y guardamos su contenido
    if (fila == 0 && columna == 0)
// Si no se ha pulsado ninguna tecla
    cont = desplazamiento;
    if (!(fila == 4 && columna == 3) &&
        !(fila_nueva == 4 && columna_nueva == 1) )
// Si no es *( tengo que borrar)  , ni # lo guardo
    {
        dato = letras[cont];
//Almaceno número correspondiente

/****************MODIFICACION DE VELOCIDAD************************/
/*******SEGUN EL CICLO DE TRABAJO INTRODUCIDO********************/

                if(segundo==1)
                {
//si ya se ha introducido ciclo de
// trabajo completo modifico velocidad
                  ciclo=ciclo+dato;
                  modificar_velocidad(ciclo);
                  segundo=0;
                  ciclo=0;
                  dato=0;
                }
                else
                {
                  segundo=segundo+1;
                  ciclo=dato*10;
                  dato=0;
                }
        }
        else
// si es * o se pulso # no lo almaceno
        pulsado = 0;
 //Inicializamos la variable pulsado para volver a pulsarlo
        } //While(1)
}//main
```

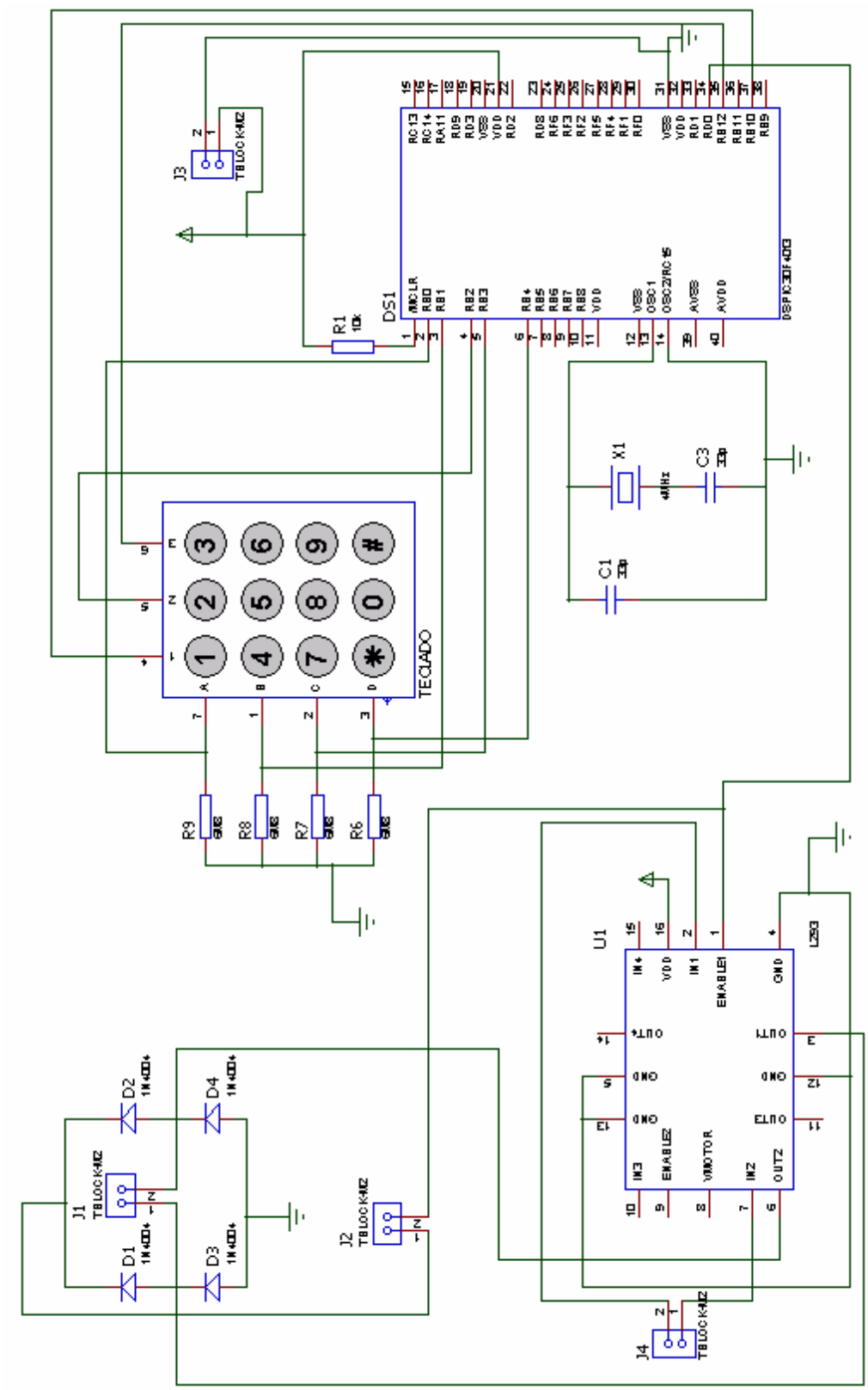



Figura 6.13: Esquema hardware de la placa de pruebas del módulo comparador de salida.

En esta foto se puede comprobar que los elementos básicos son el motor y

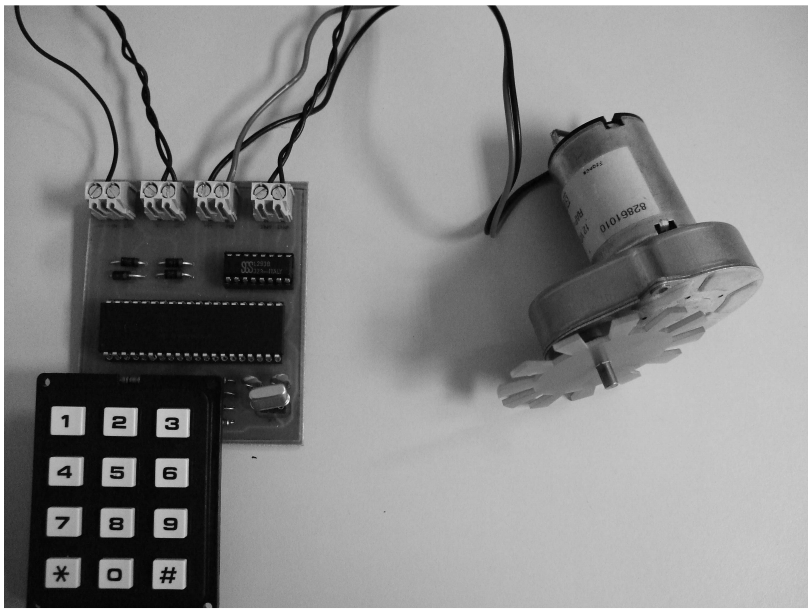



Figura 6.14: Placa de pruebas del módulo Comparador de salida.

el teclado, por este último se introduce el ciclo de trabajo deseado que llega al dsPIC el cual hace un barrido para identificar el valor introducido por el teclado posteriormente modifica el periodo de la señal PWM y paso esta señal por un filtro paso bajo para quedarme con el valor de continua de la señal, de tal manera que a mayor tensión de continua más velocidad y a menor tensión menos velocidad.

## 6.5. Práctica: Diseño de un Filtro FIR.

Se pretende realizar el diseño de un filtro FIR que permita el paso de señales de una determinada frecuencia sin atenuarlas y atenue el resto de señales que no estén en ese rango de frecuencias.
Para la realización de la práctica se requiere la utilización de un generador de señales que permite obtener una señal de una amplitud y frecuencia deseada. Se trabajará con una señal senoidal de 1 voltio y la frecuencia es el parámetro que va a permitir comprobar el funcionamiento del filtro.
**Se debe de ajustar el offset antes de introducir la señal en el dsPIC.**
Con relación al dsPIC se recomiendo visualizar las hojas de características para ver que tensiones soporta el dsPIC para no romperlo al introducir la señal a tratar.
Para comprobar el funcionamiento se utilizará el osciloscopio, se debe visualizar la señal de entrada y de la salida procesada del dsPIC.
Para trabajar en tiempo real grabaremos el programa en WinPic800 con el siguiente configuración XT w/PLL 16X-XT crystal oscillator with 16X PLL.(TCY=4/(4*16M para hacer los cálculos correctos.)
Para poder realizar este filtro se utilizará el módulo CAD del dsPIC, que me permitirá tomar muestras de la señal de estrada y poder procesar cada muestra. Posteriormente se procesa mediante:

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

Donde $y_n$ es un vector que guarda los valores de todas las muestras procesadas, x es un vector que contiene las muestras capturadas de la señal de entrada y h es un vector que guarda los valores del filtro. Para poder obtener los valores del filtro se puede utilizar la herramienta Matlab y el comando fir1(para conocer este comando escribir en la pantalla inicial de Matlab help fir1).
Posteriormente se generará la señal procesada mediante el módulo comparador de salida con PWM y se visualizará en el osciloscopio.
Hay que tener en cuenta que la frecuencia de PWM debe de ser la misma que la frecuencia de muestreo y se desea que tenga un valor de 2Khz por lo que solo se podrán visualizar correctamente señales de 1KHz, debido al solapamieto.
Comprobar como variando la frecuencia de la señal de entrada(sin variar la amplitud de esta),la señal de salida tiene la misma frecuencia que la señal de entrada pero se puede observar que aquellas frecuencias que no estén dentro del ancho de banda del filtro son atenuadas considerablemente.

Se pide realizar un filtro paso bajo que deje pasar frecuencias hasta 500Hz ($B = 100 * fir1(4, 0{,}5)$) y comprobar su funcionamiento.
Realizar lo mismo pero con un filtro pasa banda ($B = 100 * fir1(4, [0{,}10{,}4])$).

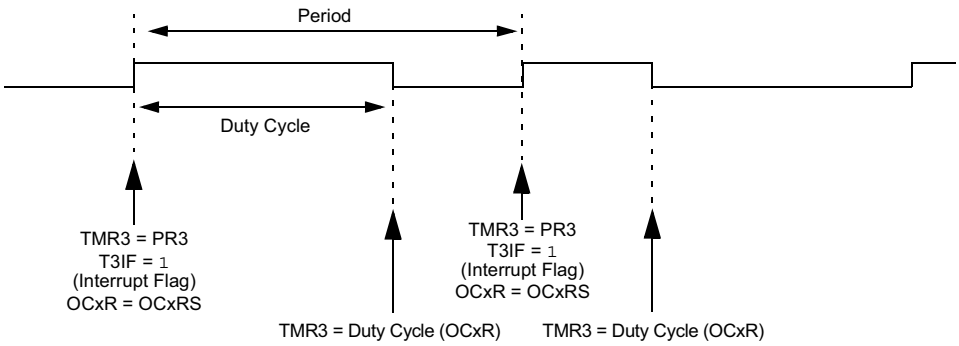



Figura 6.15: Señal PWM.

En la salida del dsPIC hay que colocar un filtro pasa baja para que solo quede el valor medio de la señal que se genera PWM este valor medio me da el valor de la muestra. Se pide calcular el valor de los componentes del filtro teniendo en cuenta los valores que se piden.

**Para sacar nota:**

Realizar un filtro IIR:

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

Donde b y a son los componentes del filtro y se obtienen con el comando $B = butter(4, [0{,}5])$ en Matlab.

Posteriormente se muestra la solución software que resuelve esta práctica y en la que se incluyen diferentes módulos del dsPIC30F4013 como por ejemplo el Convertidor Analógico Digital, el módulo Comparador de salida y además temporizadores para las distintas aplicaciones.

## *Solución: Filtro FIR.*

```
/***REALIZACION DE UN FILTRO FIR PARA UNA ENTRADA SENOIDAL DE FRECUENCIA***/
#include <dspic.h>
#include "binario.h"

//Para trabajar en tiempo real grabaremos nuestro programa  en WinPic800
//con el siguiente oscilador XT w/PLL 16X-XT crystal oscillator
// with 16X PLL por lo que  TCY=4/(4*16MHZ)=1/16us para
//hacer los calculos correctos pruebas realizadas con
//una señal senoidal de amplitud 1 voltio.
//Variables auxiliares
int j=5;
//i y j son variables auxiliares para realizar un barrido
int i=0;
int p=0;
//Vectores que contienen los elementos de la señal,
//del filtro y la salida correspondiente
unsigned int ValorCAD[10];
//las tensiones de entada varian entre un voltio y 0
signed long int salida;
```

```
//vector donde se almacenan los  datos a transmitir
// Obtengo los valores del filtro mediante el programa matlab
// Mediante la formula B=fir1(4,[0.1 0.4])
// Tenemos 5 valores que proporcionan un filtro pasabanda
// del 10%Wmuestreo/2 al 50%Wmuestreo/2 en este caso
// fmuestreo=1KHz
// Los valores obtenidos son valores decimales
//pero quierotrabajar con lógica entera por lo que
// les multiplicamos por 100 posteriormente se divide
// la salida entre 100 para obtener la salida correcta
// signed int fir[]={0,24,66,24,0};
// Primer ejemplo Filtro pasa banda entre 16-64Hz B=100*fir1(4,[0.1 0.4])
signed int fir[]={0,20,59,20,0};
//Segundo ejemplo Filtro paso bajo hasta 80 Hz   B=100*fir1(4,0.5)
/*****************************************************************/
//Interrupción CAD cuando finaliza la conversión
//se guarda valor en ValorCAD
void interrupt conversion(void)@ ADC_VCTR
{
        ADIF=0;
        ValorCAD[j] = ADCBUF0;
}
/*****************************************************************/
void Pausa(void)
{
TMR2=0;
T2TCKPS0=1;     // Preescala 1:8
T2TCKPS1=0;
while(T2IF==0); //Esperamos hasta que el temporizador termine de contar
T2IF=0;
}
 void pwm(signed long int)
{
while(T3IF==0); //Esperamos hasta que el temporizador termine de contar
T3IF=0;
OC1RS=salida;
}
/*************************************************/
signed long int filtro(void)
{
        for (i =0; i < 5 ;  i++) // i=Numero de coeficientes del filtro
                {
                salida = salida + (fir[i]* ValorCAD[j-i]/100);
                }
//Se divide entre 100 poruqe en un principio multiplicamos
//los componentes del filtro por 100
        j++;
//cuando llegamos al fianal del buffer ValorCAD
        if(j==10){
//Se guardan los valores antiguos en la posiciones
                j=5;
// 0-4 para usar las muestras antiguas
                for(i=0;i<5;i++)
                {
                        ValorCAD[i]=ValorCAD[i+5];
                }
        }
        return salida;
}
//Programa principal
void main()
```

```
{
//Configuración del CAD
ADPCFG = 0xEFFF;
ADCON1 = 0x0001;
/*  bit 15 ADON: Mode de operación                                          86
        1 = Convertidor A/D funcionando.
    *0 = Convertidor A/D apagado.
        bit 13 ADSIDL: Parar en modo Idle
        1 = Deja de funcionar en modo Idle
        *0 = Continua funcionando en modo Idle.                             91
        bit 9-8 FORM<1:0>: Formato de los datos de salida
        11 = Fraccional simple (DOUT = sddd dddd dd00 0000)
        10 = Fraccional (DOUT = dddd dddd dd00 0000)
        01 = Entero simple (DOUT = ssss sssd dddd dddd)
    *00 = Entero (DOUT = 0000 00dd dddd dddd)                               96
        bit 7-5 SSRC<2:0>: Selecciona la fuente de disparo
        de la conversión
        111 = Contador interno finaliza el muestreo e
        inicializa la conversión (auto convert)
        110 = Reservado                                                     101
        101 = Reservado
        100 = Reservado
        011 = Intervalo controlado por el motor PWM finaliza
        el muestreo e inicializa la conversión
        010 = GP Timer3 compare finaliza el muestreo e                      106
        inicializa la conversión
        001 = Transición en el pin INT0 finaliza el muestreo
         e inicializa la conversión
        *000 = Borrar el bit SAMP  finaliza el muestreo e
        inicializa la conversión                                            111
        bit 3 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)              116
        *0 = Muestro de multiples canales en secuencia individual.
        bit 2 ASAM: Bit de 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.             121
        bit 1 SAMP: Habilita el muestreo del convertidor
        1 = Al menos un amplificador sample/hold esta habilitado.
        *0 = Todos los amplificadores sample and hold están deshabilitados
        Cuando el bit ASAM es cero, escribiendo 1en
        este bit empieza el muestreo.Cuando los bit                         126
        SSRC = 000 finaliza el muestreo
        y empieza la conversión.
        bit 0 DONE: Bit de estado de la conversión
        1 = Conversión A/D esta hecha
        0 = Conversión A/D no esta hecha                                    131
                Se borra por software o empezando una
        nueva conversión. Si se borra este bit
        no afecta a las operaciones que estén en progreso.*/
ADCHS  = 0x000C;
// Connect RB3/AN3 as CH0 input .                                           136
/*      bit 15-14 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.                                 141
        10= El CH1 negativo es la entrada AN6.
        El CH2 negativo es la entrada AN7.
```

```
        El CH8 negativo es la entrada AN8.
*       0x= El CH1 negativo es la entrada VREF - .
        El CH1 negativo es la entrada VREF -.
        El CH1 negativo es la entrada VREF -.
        bit 13 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 CH8 positivo es la entrada AN2.
        bit 12 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 VREF -.
        bit 11-8 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.
        bit 7-6 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 CH8 negativo es la entrada AN8.
*       0x= El CH1 negativo es la entrada VREF - .
        El CH1 negativo es la entrada VREF -.
        El CH1 negativo es la entrada VREF -.
        bit 5 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 CH8 positivo es la entrada AN2.
        bit 4 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 VREF -.
        3-0 CH0SA<3:0>: Selecciona la entrada negativa del
        canal 0 para la configuración del MUX A.
        1111= El CH0 positivo es la entrada AN15.
        1110= El CH0 positivo es la entrada AN14.
        1101= El CH0 positivo es la entrada AN13.
*       1100= El CH0 positivo es la entrada AN12
        0011= El CH0 positivo es la entrada AN3.
        0010= El CH0 positivo es la entrada AN2.
        0001= El CH0 positivo es la entrada AN1.
        0000= El CH0 positivo es la entrada AN0.        */
ADCSSL = 0x1000;        // Canal AN12
ADCON3 = 0x0202;        // Manual Sample, Tad = internal 2 Tcy
/*      bit 12-8 SAMC<4:0>: Indica el tiempo de muestreo automático
        11111 = 31 TAD
        00001 = 1 TAD
```

```
        00000 = 0 TAD (si sólo se permite realizar conversiones
        secuencial usando más de un amplificador S / H)
        bit 7 ADRC: Fuente del reloj del conversor A/D
        1 = Reloj RC interno para el convertidor A/D
*       0 = Reloj obtenido del reloj del sistema.
        bit 5-0 ADCS<5:0>: Bits que realiza la
    selección del reloj de conversión.
        111111 = TCY/2 * (ADCS<5:0> + 1) = 32 * TCY
*       000010 = TCY /2 * (ADCS<5:0> + 1) = 3/2*TCY
        000001 = TCY /2 * (ADCS<5:0> + 1) = TCY
        000000 = TCY /2 * (ADCS<5:0> + 1) = TCY /2      */
ADCON2 = 0x0000;/*
        bit 15-13 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
        --------|-----------------------|---------------------------

        bit 10 CSCNA: Selecciona la entrada de muestreo
    para el canal CH0+ S/H, para el MUX A pone a uno
    la entrada del multiplexor
        1 = Escanea las entradas
*       0 = No escanea las entadas
        bit 9-8 CHPS<1:0>: Bits de 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 SIMSAM (ADCON1<3>) = 0 múltiples canales
    son muestreados secuencialmente.
        Cuando el bit SMSAM (ADCON1<3>) = 1 múltiples canales
    son muestreados según CHPS<1:0>
        bit 7 BUFS: Bit de estado del buffer de salida. Solo valido cuando
    BUFM = 1 (Dirección dividida en 2x8).
        1 = Posiciones 0x8-0xF del buffer de salida del A/D están llenas,
    el usuario debe acceder a los datos de las posiciones 0x0-0x7.
*   0 = Posiciones 0x0-0x7 del buffer de salida del A/D están llenas,
    el usuario debe acceder a los datos de las posiciones 0x8-0xF.
        bit 5-2 SMPI<3:0>: Bits de selección de interrupciones
    de la secuencia de muestreo conversión.
        1111 = Se genera una interrupción cada 16ª muestreo/conversión.
        1110 = Se genera una interrupción cada 15° muestreo/conversión.
        .....
        0001 = Se genera una interrupción cada 2° muestreo/conversión.
*       0000 = Se genera una interrupción cada vez que se termina una
              secuencia muestreo/conversión.
        bit 1 BUFM: Bit de selección del modo del Buffer.
        1 = Buffer configurado como 2 buffer de 8 palabras cada una,
    ADCBUF (15...8), ADCBUF (7...0)
*       0 = Buffer configurado como un único buffer de
    16 palabras ADCBUF (15...0.)
        bit 0 ALTS: Bit de selección de una entrada alternativa
    del modo  de muestreo
        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 Uses MUX A.
        *       0 = Utiliza siempre el multiplexor MUX A.       */
```

```
ADCON1bits.ADON =1;
// enciendo A/D
TRISB=0x100F;
//Inicialización el puerto B donde RB12
//es entrada de la señal a filtrar

//CONFIGURACIÓN PWM SIN CONTROL DE FALLO
//Dentro del registro OC1CON tenemos los siguientes bits
OC1_OCSIDL=0;
// Desactiva el comparador en modo Idle si es 1
OC1_OCTSEL=1;
// Selecciona el temporizador que vamos a usar entre TMR2 si 0 y TMR3 si 1
OC1_OCM0=0;
// Selecciona el modo en el que trabaja
//el modulo comparador de salida en este caso PWM
OC1_OCM1=1;
// sin control de fallo
OC1_OCM2=1;
// Cuando se finalice el periodo de la señal PWM
// se actualiza OC1R con OCRS
//OC1RS=0x0000;
//Este registro de 16 bits almacena el valor de
// ciclo de trabajo actual de la señal
//OC1R=0x0000;
//contiene el ciclo de trabajo actual
T3CON = 0X8000;
//Misma configuracion que el T2CON pero con una preescala 1:1
PR3=1023;
//Frecuencia de PWM de 1KHz y una resolucion de 10 bits PR3=1023
// como el convertidor para una frecuencia de oscilacion de  4MHZ

//TEMPORIZADOR
T2CON = 0X8010;
/*      bit 15 TON: Timer On Control bit
*       1 = Starts the timer
        0 = Stops the timer
        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)*/
T2TCKPS0=1;     // Preescala 1:8
T2TCKPS1=0;
```

```
PR2=1024;
// La frecuencia de muestreo sea igual que la frecuencia de PWM
// Fmuestreo 2KHz -->frecuencias hasta 1KHZ
ADIF=0;
// Flag indicador del CAd lo inicializamos
ADIE=1;
// Se habilita interrupcion del CAD
NSTDIS=1;
// Deshabilita interrpciones anidadas
T2IF=0;
// Habilita interrupciones de los temporizadores
T3IF=0;
// Iniciaizacion de flags
TMR3=0;
TMR2=0;
  while(1){
//Obtengo muestras de la señal de entrada con el módulo CAD
        ADCON1bits.SAMP = 1;
//Empieza muestreo
        Pausa();
//Retardo S/H
        ADCON1bits.SAMP = 0;
//Empieza conversioón
//Espera a que termine la conversión
//Se guardan las muestras en un buffer 10
//Las primeras cinco posiciones se utilizan para almecenar datos
//antiguos y se empieza almecenar en la posicion 5
//Esperamos a que terine de convertir
//Filtrado de la señal
        salida=0;
//Vector que guarda la señal procesada
        salida=filtro();
//Llamamos a la funcion que filtra la señal de entrada
        pwm(salida);
//Espero hasta que termine el perido de
//la señal PWM para cargar el nuevo valor
//del ciclo trabjo Temporizador 3 genera una interrpcion
//sacamos la señal filtrada con PWM
 }//while
}//main
//Se debe colocar un filtro paso bajo para obtener la señal senoidal de
//la señal PWM para ello colocamos un filtro RC (R=68K, C=10nF) que
//me permita quedarme con el valor medio de la señal PWM que corresponde al
//valor de la muestra de la senoide.
//Hay que tener en cuenta que el dsPIC genera una señal PWM de amplitud
// 5V que habra que tener en cuenta a la hora de generar la señal PWM
```

En la siguiente fotografía se puede apreciar el resultado de la práctica en un osciloscopio disponible en el laboratorio, se aprecia la señal de entrada y la señal de salida después de filtro, en este caso la señal original pasa por completo debido a que está dentro de los rangos del filtro, pero si se aumenta la frecuencia de la señal de salida la señal se verá atenuada de forma considerable.

A continuación se muestra la solución software de un filtro IIR cuyo objetivo es el mismo que en el caso anterior filtrar una señal de una determinada frecuencia.

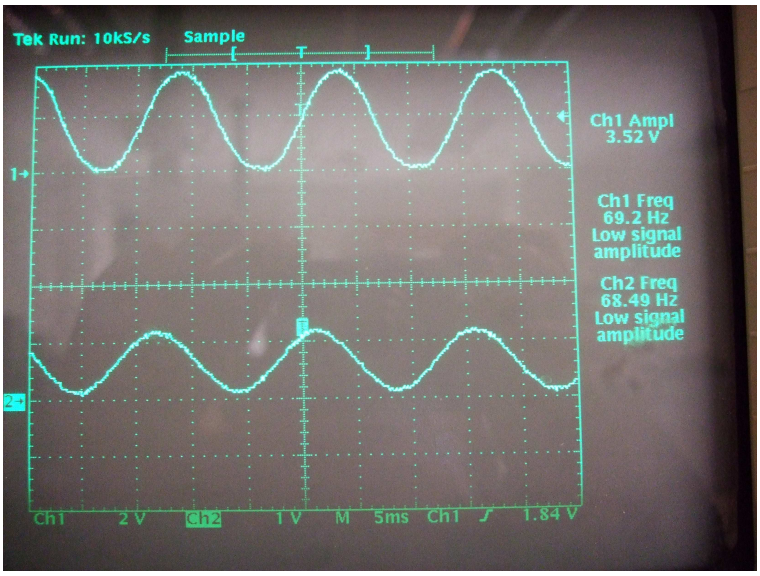



Figura 6.16: Representación en el osciloscopio de la señal de entrada y de la salida procesada.

La única diferencia con respecto al anterior es la forma de procesar las muestras obtenidas con módulo CAD y que en este caso no utilizo interrupción en el CAD aunque podría utilizarse también como en el caso anterior.

## *Solución: Filtro IIR.*

```
/***REALIZACION DE UN FILTRO IIR PARA UNA ENTRADA SENOIDAL DE FRECUENCIA***/

#include <dspic.h>
#include "binario.h"

//Para trabajar en tiempo real grabaremos nuestro programa  en WinPic800
//con el siguiente oscilador XT w/PLL 16X-XT crystal oscillator with 16X PLL
//Por lo que  TCY=4/(4*16MHZ)=1/16us para hacer los calculos correctos

//FunciÓn empleada
void Pausa(void)
{

TMR2=0;
// Inicializa temporizador

T2TCKPS0=1;
// Preescala 1:8
T2TCKPS1=0;
while(TMR2<=1024);
// Tmuestreo=1us*Cuenta*8/16
// Hacemos que la frecuencia de muestreo sea igual que la frecuencia de PWM
}
// Fmuestreo 2KHz -->frecuencias hasta 1KHZ
//Variables auxiliares
int j=5;
//i y j son variables auxiliares para realizar un barrido
int i=0;
```

```
int p=0;
//Vectores que contienen los elementos de la señal,
// del filtro y la salida correspondiente

unsigned int ValorCAD[10];
//las tensiones de entada varian entre un voltio y 0
signed long int salida[10];
//vector donde se almacenan los  datos a transmitir

// Obtenemos los valores del filtro mediante el programa matlab
// Mediante la formula B=fir1(4,[0.1 0.4])
// Tenemos 5 valores que proporcionan un filtro pasabanda
// del 10%fmuestreo/2 al 50%fmuestreo/2 en este caso
// fmuestreo=1KHz  pasa banda entre 50Hz y 200Hz las demas las atenua
// Los valores obtenidos son valores decimales pero queremos trabajar con
// logica entera por lo que les multiplicamos por 100
// Posteriormente se divide la salida entre 100 para
//obtener la salida correcta

signed int B[]={9,38,56,38,9};
//Valores multiplicados por 100
 signed int A[]={100,0,49,0,2};

//Programa principal
void main()
{
//Configuramos el CAD
//Configuración del CAD
ADPCFG = 0xEFFF;
ADCON1 = 0x0001;
/*  bit 15 ADON: Mode de operación
        1 = Convertidor A/D funcionando.
    *0 = Convertidor A/D apagado.
        bit 13 ADSIDL: Parar en modo Idle
        1 = Deja de funcionar en modo Idle
        *0 = Continua funcionando en modo Idle.
        bit 9-8 FORM<1:0>: Formato de los datos de salida
        11 = Fraccional simple (DOUT = sddd dddd dd00 0000)
        10 = Fraccional (DOUT = dddd dddd dd00 0000)
        01 = Entero simple (DOUT = ssss sssd dddd dddd)
    *00 = Entero (DOUT = 0000 00dd dddd dddd)
        bit 7-5 SSRC<2:0>: Selecciona la fuente de disparo
        de la conversión
        111 = Contador interno finaliza el muestreo e
        inicializa la conversión (auto convert)
        110 = Reservado
        101 = Reservado
        100 = Reservado
        011 = Intervalo controlado por el motor PWM finaliza
        el muestreo e inicializa la conversión
        010 = GP Timer3 compare finaliza el muestreo e
        inicializa la conversión
        001 = Transición en el pin INT0 finaliza el muestreo
         e inicializa la conversión
        *000 = Borrar el bit SAMP  finaliza el muestreo e
        inicializa la conversión
        bit 3 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 multiples canales en secuencia individual.
```

```
        bit 2 ASAM: Bit de 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.
        bit 1 SAMP: Habilita el muestreo del convertidor
        1 = Al menos un amplificador sample/hold esta habilitado.
        *0 = Todos los amplificadores sample and hold están deshabilitados
        Cuando el bit ASAM es cero, escribiendo 1en
        este bit empieza el muestreo.Cuando los bit
        SSRC = 000 finaliza el muestreo
        y empieza la conversión.
        bit 0 DONE: Bit de estado de la conversión
        1 = Conversión A/D esta hecha
        0 = Conversión A/D no esta hecha
                Se borra por software o empezando una
        nueva conversión. Si se borra este bit
        no afecta a las operaciones que estén en progreso.*/
ADCHS   = 0x000C;
// Connect RB3/AN3 as CH0 input .
/*      bit 15-14 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 CH8 negativo es la entrada AN8.
*       0x= El CH1 negativo es la entrada VREF - .
        El CH1 negativo es la entrada VREF -.
        El CH1 negativo es la entrada VREF -.
        bit 13 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 CH8 positivo es la entrada AN2.
        bit 12 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 VREF -.
        bit 11-8 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.
        bit 7-6 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 CH8 negativo es la entrada AN8.
*       0x= El CH1 negativo es la entrada VREF - .
        El CH1 negativo es la entrada VREF -.
        El CH1 negativo es la entrada VREF -.
        bit 5 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 CH8 positivo es la entrada AN2.
        bit 4 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 VREF -.
        3-0 CH0SA<3:0>: Selecciona la entrada negativa del
        canal 0 para la configuración del MUX A.
        1111= El CH0 positivo es la entrada AN15.
        1110= El CH0 positivo es la entrada AN14.
        1101= El CH0 positivo es la entrada AN13.
*       1100= El CH0 positivo es la entrada AN12
        0011= El CH0 positivo es la entrada AN3.
        0010= El CH0 positivo es la entrada AN2.
        0001= El CH0 positivo es la entrada AN1.
        0000= El CH0 positivo es la entrada AN0.        */
ADCSSL = 0x1000;        // Canal AN12
ADCON3 = 0x0202;        // Manual Sample, Tad = internal 2 Tcy
/*      bit 12-8 SAMC<4:0>: Indica el tiempo de muestreo automático
        11111 = 31 TAD
        00001 = 1 TAD
        00000 = 0 TAD (si sólo se permite realizar conversiones
        secuencial usando más de un amplificador S / H)
        bit 7 ADRC: Fuente del reloj del conversor A/D
        1 = Reloj RC interno para el convertidor A/D
*       0 = Reloj obtenido del reloj del sistema.
        bit 5-0 ADCS<5:0>: Bits que realiza la
    selección del reloj de conversión.
        111111 = TCY/2 * (ADCS<5:0> + 1) = 32 * TCY
*       000010 = TCY /2 * (ADCS<5:0> + 1) = 3/2*TCY
        000001 = TCY /2 * (ADCS<5:0> + 1) = TCY
        000000 = TCY /2 * (ADCS<5:0> + 1) = TCY /2      */
ADCON2 = 0x0000;/*
        bit 15-13 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
        --------|-----------------------|---------------------------

        bit 10 CSCNA: Selecciona la entrada de muestreo
    para el canal CH0+ S/H, para el MUX A pone a uno
    la entrada del multiplexor
        1 = Escanea las entradas
*       0 = No escanea las entadas
        bit 9-8 CHPS<1:0>: Bits de 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 SIMSAM (ADCON1<3>) = 0 múltiples canales
    son muestreados secuencialmente.
        Cuando el bit SMSAM (ADCON1<3>) = 1 múltiples canales
```

```
    son muestreados según CHPS<1:0>
        bit 7 BUFS: Bit de estado del buffer de salida. Solo valido cuando
    BUFM = 1 (Dirección dividida en 2x8).
        1 = Posiciones 0x8-0xF del buffer de salida del A/D están llenas,
    el usuario debe acceder a los datos de las posiciones 0x0-0x7.
*   0 = Posiciones 0x0-0x7 del buffer de salida del A/D están llenas,
    el usuario debe acceder a los datos de las posiciones 0x8-0xF.
        bit 5-2 SMPI<3:0>: Bits de selección de interrupciones
    de la secuencia de muestreo conversión.
        1111 = Se genera una interrupción cada 16ª muestreo/conversión.
        1110 = Se genera una interrupción cada 15° muestreo/conversión.
        .....
        0001 = Se genera una interrupción cada 2° muestreo/conversión.
*       0000 = Se genera una interrupción cada vez que se termina una
              secuencia muestreo/conversión.
        bit 1 BUFM: Bit de selección del modo del Buffer.
        1 = Buffer configurado como 2 buffer de 8 palabras cada una,
    ADCBUF (15...8), ADCBUF (7...0)
*       0 = Buffer configurado como un único buffer de
    16 palabras ADCBUF (15...0.)
        bit 0 ALTS: Bit de selección de una entrada alternativa
    del modo  de muestreo
        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 Uses MUX A.
        *       0 = Utiliza siempre el multiplexor MUX A.       */

ADCON1bits.ADON =1;
// enciendo A/D
TRISB=0x100F;

//CONFIGURACION PWM SIN CONTROL DE FALLO
//Dentro del registro OC1CON tenemos los siguientes bits

OC1_OCSIDL=0;
// Desactiva el comparador en modo Idle si es 1
OC1_OCTSEL=1;
// Selecciona el temporizador que vamos a usar entre TMR2 si 0 y TMR3 si 1
OC1_OCM0=0;
// Selecciona el modo en el que trabaja
//el modulo comparador de salida en este caso PWM
OC1_OCM1=1;
// sin control de fallo
OC1_OCM2=1;
// Cuando se finalice el periodo de la señal
//PWM se actualiza OC1R con OCRS
//OC1RS=0x0000;
//Este registro de 16 bits almacena el valor de
// ciclo de trabajo actual de la señal
//OC1R=0x0000;
//contiene el ciclo de trabajo actual
T3CON = 0X8000;
//Misma configuracion que el T2CON pero con una preescala 1:1
PR3=1023;
//Frecuencia de PWM de 1KHz y una resolucion de 10 bits PR3=1023
// como el convertidor para una frecuencia de oscilacion de  4MHZ
//TEMPORIZADOR
//TEMPORIZADOR
T2CON = 0X8010;
/*      bit 15 TON: Timer On Control bit
*       1 = Starts the timer
        0 = Stops the timer
```

```
        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)*/
T2TCKPS0=1;
// Preescala 1:8
T2TCKPS1=0;
PR2=1024;
// La frecuencia de muestreo sea igual que la frecuencia de PWM
// Fmuestreo 2KHz -->frecuencias hasta 1KHZ
while(1){
//Obtengo muestras de la señal de entrada con el módulo CAD
        ADCON1bits.SAMP = 1;
//Empieza muestreo
        Pausa();
//Retardo S/H
        ADCON1bits.SAMP = 0;
//Empieza conversión
        while (!ADCON1bits.DONE);
//Espera a que termine la conversión
        ValorCAD[j] = ADCBUF0;
//Se guardan las muestras en un buffer 10
//Las primeras cinco posiciones se utilizan para almecenar datos
//antiguos y se empieza almecenar en la posicion 5
//Filtrado de la señal
        salida[j]=0;
//Vector que guarda la señal procesada
        for (i =0; i < 5 ;  i++)
// i=Numero de coeficientes del filtro
        {
        salida[j]= salida[j] + (B[i]* ValorCAD[j-i]-A[i]* salida[j-i])/100;
        }
        j++;
   //cuando llegamos al final del buffer ValorCAD
        if(j==10){
//Se guardan los valores antiguos en la posiciones
                j=5;
// 0-4 para usar las muestras antiguas
                for(i=0;i<5;i++)
                {
                        ValorCAD[i]=ValorCAD[i+5];
                        salida[i]=salida[i+5];
                }
```

```
        }
//Se divide entre 100 poruqe en un principio multiplicamos
//los componentes del filtro por 100
//sacamos la señal filtrada con PWM
                        while(T3IF==0);
//Espero porque sino el valor que pone no es el correcto
                        T3IF=0;
                        OC1RS=salida[j];
// La anchura del pulso es proporcional a la seniodal
  }//while
}//main


//Se debe colocar un filtro paso bajo para obtener la señal senoidal de
//la señal PWM para ello colocamos un filtro RC (R=4.3M, C=1nF) que
//me permita quedarme con el valor medio de la señal PWM que corresponde al
//valor de la muestra de la senoide.
//Hay que tener en cuenta que el dsPIC genera una señal PWM de amplitud
// 5V que habra que tener en cuenta a la hora de generar la señal PWM
```

# Capítulo 7

# Presupuesto

Dentro del presupuesto voy a tener varias partes, una es el presupuesto de diseño que es el presupuesto que cobra el ingeniero por las horas de trabajo invertidas y la otra parte es el presupuesto de laboratorio en el que se incluyen presupuesto de los materiales y el de montaje. Otra parte del presupuesto es el beneficio pero como este proyecto esta orientado al desarrollo de prácticas para la universidad no va a tener ningún beneficio.

## 7.1. Presupuesto Diseño

Este presupuesto es el que cobro por las horas invertidas para la realización del proyecto en el que se incluye material didáctico que facilita el estudio de los dsPIC, enunciados de prácticas orientadas a los alumnos con solución software y para alguna de ellas elaboración de un placa para mostrar su funcionamiento.Han sido 500 horas de trabajo. En la siguiente tabla se muestra el presupuesto de diseño total:

| **Horas** | **Euros/horas** | **Total** |
|---|---|---|
| 500 | 15 | 7500 |

## 7.2. Presupuesto Laboratorio

Para poder entender el presupuesto de laboratorio se muestra a continuación el precio de los distintos placas fabricadas:

### 7.2.1. Presupuesto Grabador

En la siguiente tabla se muestra el precio de cada uno de los componentes necesarios para fabricar cada grabador y al final se muestra el precio unitario por grabador:

| UNI | MATERIAL | PRECIO(UNI) | TOTAL(EUROS) |
|---|---|---|---|
| 1 | Conector DB25 Macho para PCB | 1.3 | 1.3 |
| 1 | J5 Coax power | 0.58 | 0.58 |
| 2 | Resistencias 10KΩ | 0.05 | 0.10 |
| 4 | Resistencias 4.7KΩ | 0.05 | 0.20 |
| 1 | Resistencia 2.2KΩ | 0.05 | 0.05 |
| 1 | Resistencia 680Ω | 0.05 | 0.05 |
| 1 | Resistencia 330Ω | 0.05 | 0.05 |
| 3 | Resistencia 1KΩ | 0.05 | 0.15 |
| 1 | Resistencia 12KΩ | 0.05 | 0.05 |
| 1 | Resistencia 100Ω | 0.05 | 0.05 |
| 1 | Condensador 100pF | 0.1 | 0.1 |
| 2 | Condensadores 100nF | 0.14 | 0.28 |
| 1 | Condensador 330nF | 0.14 | 0.14 |
| 1 | 7805 | 0.60 | 0.60 |
| 1 | Circuito integrado 74HC14 | 0.34 | 0.34 |
| 4 | Transistor BC547 | 0.16 | 0.64 |
| 1 | Transistor BD136 | 0.37 | 0.37 |
| 1 | Diodo 1N4007 | 0.09 | 0.09 |
| 1 | Diodo OA91 | 0.06 | 0.06 |
| 1 | Led Rojo | 0.13 | 0.13 |
| 1 | Borna 2 vias atornillable para PCB | 0.28 | 0.28 |

| UNI | MATERIAL | PRECIO(UNI) | TOTAL(EUROS) |
|---|---|---|---|
| 2 | Conector RJ45 hembra para PCB acodado | 1.13 | 2.26 |
| 1 | Latiguillo RJ45-RJ45(1 metro) | 17.31 | 17.31 |
| 1 | Zócalo 14 pines | 0.54 | 0.54 |
| 1 | Zócalo 40 pines | 1.84 | 1.84 |
| 1 | Tira de pines macho | 0.53 | 0.53 |
| | | | **28.09** |

### 7.2.2. Presupuesto Placa PIC RJ

Como se comentó en los objetivos lo que se busca es que el grabador sirva tanto para los dsPIC30f como para los PIC por lo que es necesario fabricar una placa que adapte este grabador al PIC, el precio de los elementos necesarios para la fabricación de esta placa son:

| UNIDADES | MATERIAL | PRECIO(UNIDAD) | TOTAL(EUROS) |
|---|---|---|---|
| 1 | Zócalo 16 pines | 0.37 | 0.37 |
| 1 | Conector RJ45 | 1.13 | 1.13 |
| | | | **1.5** |

### 7.2.3. Presupuesto Elementos Comunes

Hay una serie de elementos que son comunes a todas las prácticas por lo que no es necesario comprar uno por práctica a desarrollar, el precio de estos elementos se muestra a continuación:

| UNIDADES | MATERIAL | PRECIO(UNIDAD) | TOTAL(EUROS) |
|---|---|---|---|
| 1 | Placa de pruebas dsPICDEM2 | 83 | 83 |
| 1 | Carrete de estaño | 5.89 | 5.89 |
| 1 | Teclado | 7.56 | 7.56 |
| 1 | LCD(HD-44780) | 14.16 | 14.16 |
| 4 | dsPIC30F4013 | 11.55 | 46.2 |
| | | | **156.81** |

### 7.2.4. Presupuesto Placa de alimentación

Para facilitar la conexión del dsPIC30f se fabricó una placa de alimentación que hace llegar las tensiones necesarias a los pines de alimentación y va a garantizar una conexión segura, ya que los dsPIC son muy sensibles a conexiones erróneas.
En la siguiente tabla se muestra en precio de dicha placa, que como es una placa que se puede emplear para distintas prácticas la incluyo en elementos comunes:

### 7.2.5. Presupuesto unitario práctica Cruce de semáforos

El presupuesto del material necesario para fabricar la placa empleada para mostrar el funcionamiento de la práctica de los temporizadores se muestra en la siguiente tabla, donde al final se muestra el total de la inversión necesaria

| UNIDADES | MATERIAL | PRECIO(UNIDAD) | TOTAL(EUROS) |
|---|---|---|---|
| 2 | Condensadores 33pF | 0.20 | 0.40 |
| 1 | Resistencia 10kΩ | 0.05 | 0.20 |
| 1 | Zócalo 40 pines | 1.84 | 1.84 |
| 1 | Cristal de cuarzo 4MHz | 0.97 | 0.97 |
| | | | **3.41** |

| UNIDADES | MATERIAL | PRECIO(UNIDAD) | TOTAL(EUROS) |
|---|---|---|---|
| 1 | Zócalo 40 pines | 1.84 | 1.84 |
| 1 | Conector de dos vias | 0.28 | 0.28 |
| 1 | Resistencia 10kΩ | 0.05 | 0.20 |
| 1 | Cristal de cuarzo 4MHz | 0.97 | 0.97 |
| 8 | Led Rojo | 0.13 | 1.04 |
| 4 | Led Ámbar | 0.13 | 0.52 |
| 1 | Led Verde | 0.13 | 1.04 |
| 10 | Resistencias 150Ω | 0.05 | 0.50 |
| 2 | Pulsadores(OMRON) | 0.6 | 1.2 |
| | | | **7.59** |

para la fabricación de dicha placa. Además para el desarrollo de esta práctica se necesita materiales comunes a varias prácticas que no se soldarán a la placa como es un dsPIC30f4013: El presupuesto empleado para el desarrollo de esta

| UNIDADES | MATERIAL | PRECIO(UNIDAD) | TOTAL(EUROS) |
|---|---|---|---|
| 1 | dsPIC30F4013 | 11.55 | 11.55 |
| | | | **11.55** |

práctica es la suma del necesario para el desarrollo de la placa y en de los elementos comunes.

### 7.2.6. Presupuesto unitario práctica $I^2C$

El presupuesto del material necesario para fabricar la placa empleada para mostrar el funcionamiento de la práctica del $I^2C$ se muestra en la siguiente tabla, donde al final se muestra el total de la inversión necesaria para la fabricación de dicha placa. Además para el desarrollo de esta práctica se necesitan materiales comunes a varias prácticas que no se soldaran a la placa como son un teclado, un LCD y dos dsPIC30f4013: El presupuesto empleado para el desarrollo de esta práctica es la suma del necesario para el desarrollo de la placa y en de los elementos comunes.

| PLACA | MATERIAL COMÚN | TOTAL(EUROS) |
|---|---|---|
| 11.55 | 7.59 | 19.14 |

| UNIDADES | MATERIAL | PRECIO(UNIDAD) | TOTAL(EUROS) |
|---|---|---|---|
| 2 | Zócalos(40 pines) | 1.84 | 3.68 |
| 2 | Condensadores 33pF | 0.20 | 0.40 |
| 1 | Cristal de cuarzo 4MHz | 0.97 | 0.97 |
| 1 | Resistencia 10KΩ | 0.05 | 0.05 |
| 4 | Resistencia 330Ω | 0.05 | 0.20 |
| 2 | Resistencia 2.7KΩ | 0.05 | 0.20 |
| 1 | Conector de 2 vias | 0.028 | 0.56 |
| 1 | Potenciómetro | 1 | 1 |
| 8 | Jumpers | 0.07 | 0.28 |
| | | | **7.34** |

### 7.2.7. Presupuesto Placa Control de Motor

Los componentes necesarios para la realización de esta placa se muestran a continuación acompañados por los precios correspondientes. Además para el desarrollo de esta práctica necesitamos materiales comunes a varias prácticas que no se soldaran a la placa como son un teclado, un motor y un dsPIC30f4013: El presupuesto empleado para el desarrollo de esta práctica es la suma del necesario para el desarrollo de la placa y en de los elementos comunes.

### 7.2.8. Presupuesto Total de Laboratorio

En el caso hipotético de que en el laboratorio haya 20 alumnos repartidos en 10 puestos de trabajo el presupuesto total del laboratorio se divide en dos partes, que es el presupuesto de material mostrado en la siguiente tabla y el presupuesto de montaje que en nuestro caso es cero debido a que serán los alumnos los que suelden las distintas placas.

## 7.3. Amortización coste de diseño

Este presupuesto engloba el presupuesto de diseño y el de laboratorio

| UNIDADES | MATERIAL | PRECIO(UNIDAD) | TOTAL(EUROS) |
|---|---|---|---|
| 1 | Teclado | 7.56 | 7.56 |
| 1 | LCD(HD-44780) | 14.16 | 14.16 |
| 2 | dsPIC30F4013 | 11.55 | 23.10 |
| | | | **44.82** |

| PLACA | MATERIAL COMÚN | TOTAL(EUROS) |
|---|---|---|
| 7.34 | 44.82 | 52.16 |

| UNIDADES | MATERIAL | PRECIO(UNIDAD) | TOTAL(EUROS) |
|---|---|---|---|
| 1 | Zócalo(40 patas) | 1.84 | 1.84 |
| 1 | Resistencia 10kΩ | 0.05 | 0.20 |
| 1 | Cristal de cuarzo 4MHz | 0.97 | 0.97 |
| 2 | Condensadores 33pF | 0.20 | 0.40 |
| 4 | Resistencia 330Ω | 0.05 | 0.20 |
| 4 | Conectores de 2 vias | 0.28 | 1.12 |
| 4 | Diodos 1N4004 | 0.09 | 0.36 |
| 1 | Integrado L293B | 6.1 | 6.1 |
| 1 | Motor Dc | 52.28 | 52.28 |
| 1 | Zócalo torneado 16 patas | 0.37 | 0.37 |
| | | | **63.84** |

| UNIDADES | MATERIAL | PRECIO(UNIDAD) | TOTAL(EUROS) |
|---|---|---|---|
| 1 | Teclado | 7.56 | 7.56 |
| 1 | Motor DC | 52.28 | 52.28 |
| 1 | dsPIC30F4013 | 11.55 | 11.55 |
| | | | **71.39** |

| PLACA | MATERIAL COMÚN | TOTAL(EUROS) |
|---|---|---|
| 763.84 | 71.39 | 135.23 |

| UNI | ELEMENTO | COSTE UNI.) | TOTAL(EUROS) |
|---|---|---|---|
| 10 | Grabador | 28.09 | 280.90 |
| 10 | PIC-RJ | 1.5 | 15 |
| 10 | Placa de pruebas dsPICDEM2 | 83 | 830 |
| 1 | Carrete de estaño | 5.89 | 5.89 |
| 10 | Teclado | 7.56 | 75.60 |
| 10 | LCD(HD-44780) | 14.16 | 141.60 |
| 20 | dsPIC30F4013 | 11.55 | 231 |
| 10 | Placas de alimentación | 3.41 | 34.10 |
| 10 | Placas de $I^2C$ | 7.34 | 73.4 |
| 10 | Placa control de un cruce de semáforos | 7.59 | 75.9 |
| 10 | Placa control motor | 63.84 | 638.4 |
| | | | 2401.79 |

| **Diseño** | **Laboratorio** | **Total(EUROS)** |
|---|---|---|
| 7500 | 2401.79 | 9901.79 |

# Capítulo 8

# Conclusiones

Para finalizar este proyecto se expondrán las conclusiones obtenidas en su realización.
Se pretendía realizar un estudio de un dsPIC de la familia 30F, en concreto se ha utilizado el dsPIC30F4013, y sus diferentes aplicaciones. Se han analizado parte de los módulos que contiene este dispositivo estudiando su funcionamiento y poniéndolo en práctica mediante la realización de una práctica con su enunciado, solución software y hardware que permitió comprobar su correcto funcionamiento.
Se comprobó la sensibilidad de este dispositivo ante una mala alimentación, lo que dio lugar a la creación de un módulo de alimentación que permitiera una correcta alimentación del dispositivo durante la realización de las diferentes pruebas.
Es conveniente recalcar la importancia de revisar todas las soldaduras en la fabricación del grabador, pues es el elemento clave en este proyecto, porque es el que permite grabar en el dsPIC de forma correcta los programas deseados y un cortocircuito en dicha placa podría dañar el puerto paralelo del ordenador al que este conectado.
Ha sido necesario el aprendizaje de diferentes herramientas software para poder realizar este proyecto como son MPLAB y PROTEUS/ARES ,junto a otros, pero quizás estos sean los mas significativos en el primero se desarrollaba la parte software de cada práctica y en Proteus la parte hardware.
También ha sido necesario el aprendizaje del uso de la fresadora del laboratorio que permitía la fabricación de las placas sobre las que se iban a soldar todos los elementos y que permitía comprobar el funcionamiento de los diferentes módulos. Así como el analizador lógico que permitía hacer unas primeras pruebas sin necesidad de tener la placa final.
Cabe destacar que en el programa ARES se tiene la opción de hacer un autorrutado, pero hay que tener en cuenta que aunque cumpla las reglas de diseño puede que posteriormente cuando se va a fresar la placa, varias pistas se unan debido a la anchura de la fresa por lo que conviene separar las pistas todo lo

que sea posible de forma manual en el programa para evitar complicaciones.
En ocasiones a la hora de grabar con el programa WINPIC800 en el dsPIC graba un ejecutable anterior al actual conviene borrar ese ejecutable cada cierto tiempo para evitar complicaciones. Hay que tener cuidado al elegir el oscilador en este programa y escoger el adecuado en cada caso.
Las partes que se han trabajado del dsPIC30F son:

- Temporizadores
- Interrupciones
- UART
- $I^2C$
- SPI
- Captura de entrada
- Comparador de salida(PWM)
- Convertidor Analógico Digital(CAD)

Las prácticas realizadas abarcan estas partes del dsPIC30F por separados y en algunas de las prácticas se mezclan.
Una posible ampliación de este proyecto podría ser el estudio detallado del módulo CAN que contiene este dispositivo y la realización de alguna aplicación.

# Bibliografía


[1] José Mª Angulo Usategui,"Microcontroladores avanzados dsPIC". *Thomson*,2006.

[2] Datasheet dsPIC30F.

[3] www.microchip.com

[4] www.farnell.com

[5] www.micropik.com