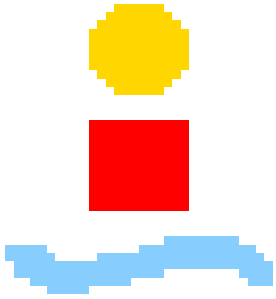
UNIVERSIDAD DE SEVILLA

ESCUELA SUPERIOR DE INGENIEROS



PROYECTO FIN DE CARRERA

DISEÑO DE UNA INTERFAZ PIC-MODEM PARA COMUNICACIONES POR LÍNEA DE POTENCIA

Dirigido por:
Dr. D. Ramón González Carvajal

Realizado por:
Mauricio Béjar Pérez

ENERO 2004

A mi familia y amigos por su apoyo y comprensión
Y a Ramón por su ayuda y orientación

ÍNDICE DE LA MEMORIA

1. OBJETIVOS A REALIZAR

Pretendemos realizar un interfaz hardware para un MODEM de potencia.Mediante este interfaz podremos controlar el MODEM de potencia, configurándolo y utilizándolo para transmitir y recibir a ráfagas.
Debemos por ello tener un control de todas las entradas y salidas del MODEM concernientes a su funcionamiento mediante un microcontrolador PIC que nos permite programar en él todas las veces que hagan falta.
También necesitaremos controlar el propio microcontrolador PIC de alguna manera, para supervisar el control y ajustarlo a los requerimientos propios de los que consta una transmisión o una recepción a ráfagas del MODEM. Ya que si bien este controla el MODEM, debemos decirle al controlador la secuencia de control para hacer una cosa u otra.
Para todo esto necesitaremos desarrollar realmente dos interfaces:

- Un interfaz hardware (el interfaz en si) entre el PIC y el MODEM.
- Un interfaz software de control entre un PC(dispositivo que usaremos para controlar la tarjeta controladora) y el PIC sobre el interfaz RS-232 del puerto serie de ambos(USART adaptada y COM1 o COM2 del PC) mediante un cable serie.

1.1 Interfaz hardware micro-MODEM

Mediante este interfaz controlaremos el MODEM en todos sus aspectos, pudiendo leer y escribir en sus registros para configurarlos y realizar transmisiones y recepciones con él. Este interfaz debe cumplir con ciertos requisitos para realizar accesos de escritura o lectura al MODEM según se puede ver a continuación.
Además deberá controlar las otras señales de control del MODEM para su activación según el modo, resetearlo y gestionar las interrupciones de este para una buena comunicación.

1.1.1 Requisitos y funcionamiento del acceso al MODEM

El MODEM de potencia tiene sus características propias que el interfaz debe cumplir para llevar a cabo un ciclo de escritura o uno de lectura .básicamente se trata de una lectura o escritura de un registro del MODEM para configuración o funcionamiento en medio de un proceso(transmisión de una ráfaga).
Según la hoja de catalogo del MODEM sus especificaciones para los ciclos de lectura y escritura tienen en cuenta los siguientes tiempos:

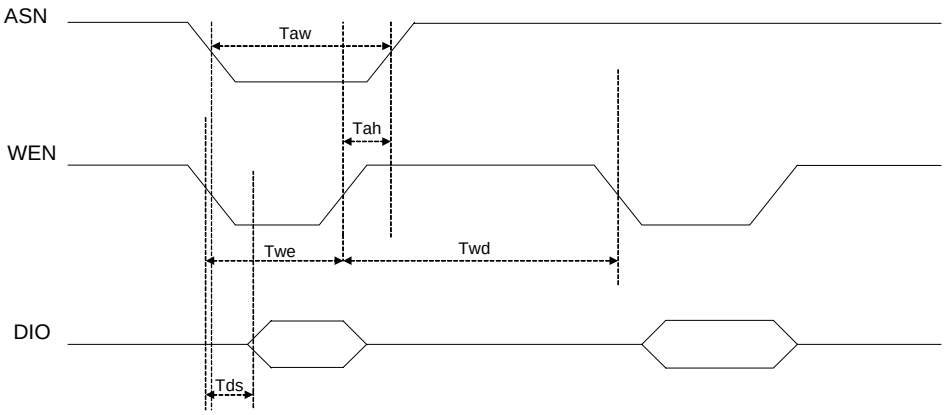
Nombre	Significado	Mínimo	Máximo
Taw	anchura de escritura de dirección	333 ns	∞
Twe	Anchura de habilitación de	333 ns	∞

	escritura		
Tds	Retraso en el establecimiento de datos	- ∞	166 ns
Tdh	Tiempo de mantenimiento de datos	333 ns	∞
Tah	Mantenimiento de dirección de escritura	0 ns	∞
Twd	Retraso de separación de escritura	333 ns	∞

Tabla de tiempos específicos en los ciclos de lectura y escritura

1.1.2 Ciclo de Escritura

Podemos ver el ciclo de escritura del MODEM controlado por las señales del bus de control (ASN,WEN), que serán externas al MODEM, proporcionadas por el dispositivo controlador, en este caso el microcontrolador:

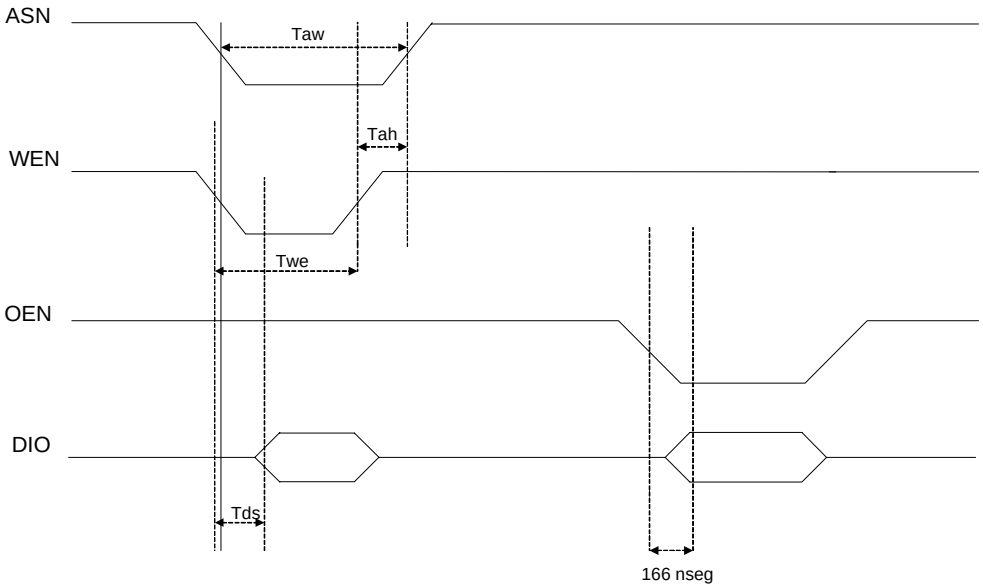


Ciclo de escritura del MODEM

Como se puede apreciar el MODEM utiliza el mismo bus bidireccional para capturar direcciones y datos, primero se le debe pasar la dirección y validarla con ASN a la vez que se indica que se la capture (WEN), y luego se le pasa el dato a escribir en el registro seleccionado (ora vez mediante WEN).

1.1.3 Ciclo de Lectura

Podemos ver el ciclo de escritura del MODEM controlado por las señales del bus de control (ASN,WEN,OEN), que serán externas al MODEM , proporcionadas por el dispositivo controlador , en este caso el microcontrolador:



Ciclo de lectura del MODEM

Para el ciclo de lectura igualmente se le pasa primero la dirección del registro a leer, validándola con ASN y haciendo que se capture con WEN , y luego con OEN se habilitan los datos leídos del MODEM , de manera que el dispositivo que lo controla debe capturarlos.

1.1.4 Funciones a realizar

El interfaz debe cumplir con los siguientes requisitos:

- Control y funcionamiento de los ciclos de lectura y escritura mediante el control de los buses de datos/direcciones y de control.
- Control de la señal de activación del MODEM.
- Control de la señal de reset , para realizar resets cuando se quiera y de la duración que se quiera.
- Control de las señales de activación de los modos test.
- Proporcionar señales exteriores de PLLs para el bloque receptor del MODEM.

1.2 Interfaz de control PC-micro

Este otro interfaz surge de manera natural, al necesitar un control y supervisión el anterior que necesitaría una programación del controlador cada vez que se quisiera hacer algo y seria muy trabajoso de utilizar.Con este interfaz software sobre una simple conexión serie podemos controlar el MODEM mandándole instrucciones al controlador para que realice las funciones pertinentes como son:

- Lecturas de registros.
- Escrituras de registros.
- Habilitación o deshabilitación de señales de control.

Así pues pretendemos controlar el MODEM a partir de una aplicación software en un PC , de manera que se puedan llevar a cabo transmisiones y recepciones y comprobar el funcionamiento de los bloques de que consta el MODEM.

1.2.1 Modos de trabajo

Pretendemos hacer un controlador que pueda ser utilizado desde el puerto serie, de manera que lo desarrollaremos para poder gobernarlo por el puerto serie de un PC u otro dispositivo que pueda llevar a cabo una transmisión serie asíncrona.

Para ello utilizaremos dos modos de trabajo, que implicaran cada uno un programa del microcontrolador:

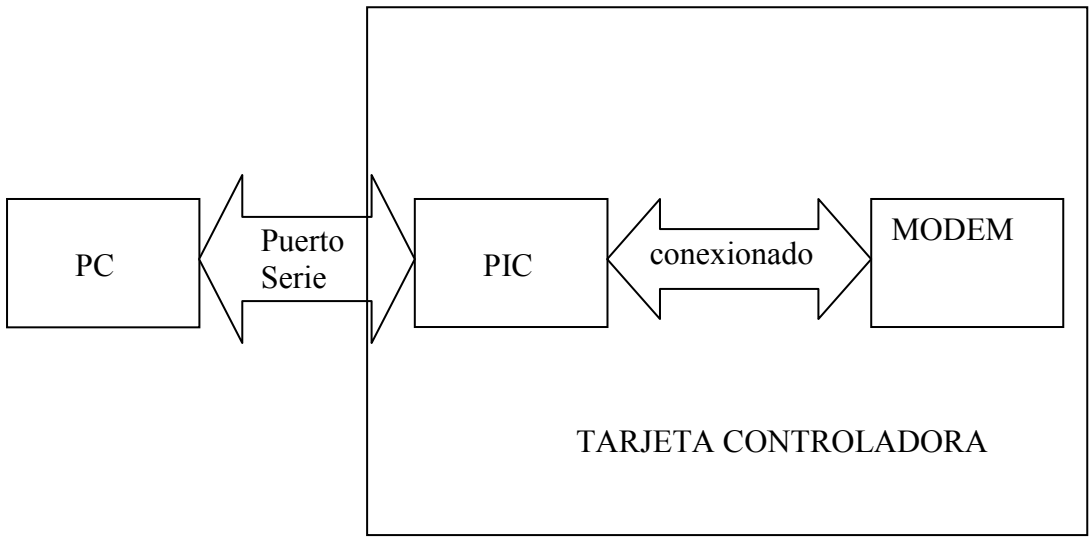
- Externo: en el cual se gobiernan las transmisiones y recepciones a ráfagas por el puerto serie de la tarjeta controladora.
- Interno: en el cual una vez suministrados los datos, de configuración y a transmitir, el proceso es automático no dependiendo de las instrucciones enviadas por el puerto serie (aunque sigue teniéndose control por él).

2. REALIZACION DE LOS OBJETIVOS

Para llevar a cabo todo esto necesitaremos diseñar una placa controladora cuyo núcleo será un microcontrolador y el propio MODEM (a controlar por el anterior), de manera que podamos conectarla al puerto serie de un PC y este gestione la placa controladora mediante una aplicación software.
Procedemos a explicar como se van a implementar los anteriores interfaces, su conexión y como se van a desarrollar.
En posteriores capítulos se incidirá exhaustivamente en cada aspecto de la implementación de los anteriores objetivos, como son el diseño hardware y el software, así como su funcionamiento.

2.1 Diseño de una placa controladora

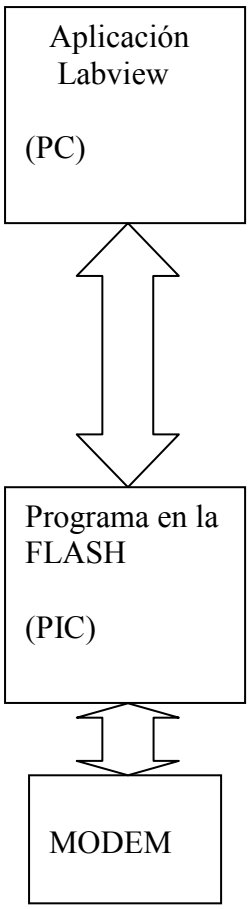
La placa debe contener al MODEM y su controlador junto con todas las necesidades circuitales que estos necesiten (circuitos básicos de funcionamiento de estos), así como un conector para un cable serie que la conectara al PC y una toma de alimentación de la que se obtengan todas las tensiones de la placa.
Como se puede ver en el siguiente esquema, la idea es tener MODEM y controlador (en este caso un PIC, fácilmente reprogramable) en una misma placa para una fácil conexión al PC desde el cual se controlara la placa:



Esquema de la tarjeta controladora y su conexión al PC

2.2 Gobierno del MODEM por el PIC

Se trata de implementar el interfaz hardware de control del MODEM atendiendo a los requisitos de este para realizar lecturas y escrituras de sus registros.
A parte de las conexiones entre el PIC y el MODEM para su correcto control, el PIC deberá tener un programa cargado en su memoria FLASH que permitirá el control de las señales de control del MODEM así como la escritura y lectura de sus registros según los requisitos de los ciclos de acceso de este.
Realizaremos dos versiones de este programa atendiendo al modo en el que queramos realizar el control para las transmisiones, interno o externo como ya se vio antes.
En el siguiente esquema se puede ver la jerarquía de control utilizada para controlar el MODEM:



Esquema jerárquico de control

2.3 Gobierno del PIC por el PC

Este interfaz software de control de la placa es necesario para llevar a cabo un control supervisado sobre el MODEM sin tener que estar cambiando de programa del PIC.
Este control lo llevaremos a través del puerto serie del PC en comunicación con la USART adaptada al RS-232 del PIC, mediante el intercambio de datos de un extremo a otro.
Mediante el programa del PIC se implementan unas rutinas que realizara el PIC para controlar el MODEM en función de datos que le lleguen al PIC por el puerto serie .De manera que las lecturas y escrituras de registro así como el resto de necesidades de control del MODEM sean activadas (llevadas a cabo por el PIC) al recibir el PIC cierta secuencia de bytes por la USART (esta junto con la adaptación al RS-232 la consideraremos a efectos prácticos como un puerto serie).
Esta secuencia la llamaremos instrucción, y tendrá una longitud variable en función a criterios que luego desarrollaremos en el capitulo del programa del PIC, contando con identificador en si de la instrucción, datos y parámetros (para una mejor versatilidad de la instrucción implementada).
El control del paso de estas instrucciones y/o datos hacia el PC lo llevaran a cabo unas aplicaciones software basadas en Labview a ejecutar sobre el PC, permitiéndonos una configuración y ejecución de transmisiones, recepciones o cualquier prueba que queramos realizar sobre el MODEM. Estas aplicaciones también se verán en detalle en el capitulo dedicado a ellas, en el que se vera que además existe un pequeño protocolo para el paso de datos a transmitir en el modo externo de trabajo (control de transmisión desde el programa de labview).
Se puede apreciar en el esquema anterior la jerarquía de los dispositivos implicados en el control del MODEM.

2.4 Uso de programas

Para el diseño y/o manejo de los anteriores interfaces y programas (tanto del PIC con las aplicaciones software para PC) utilizaremos los siguientes programas que describimos según el uso de cada uno de ellos:

2.4.1 Diseño del PCB

Para el diseño de la placa controladora en si, es decir el PCB, utilizamos el PCAD2001, realizando un diseño a dos caras en el menor espacio posible .Utilizamos vías donde es

necesario reduciendo el número de estas a las imprescindibles, para unir las dos caras del circuito, recurriendo a conexiones de resistencias o condensadores en los demás casos .Para cada chip utilizamos su zócalo correspondiente para que así sean fáciles de reemplazar.
Realizamos dos versiones de la placa controladora, la segunda es una versión mejorada en la que en vez de mallado para los filtros externos aun no probados se introducen directamente los filtros en si, y desacoplamos las dos etapas de regulación para conseguir las alimentaciones de la placa (que en la anterior estaban en cascada).

2.4.2 Programación del Microcontrolador PIC

Aquí tenemos que diferenciar entre lo que es la creación y depuración del programa del PIC, y la manera de programar el PIC:

- La depuración y ensamblado del Programa a escribir en el PIC se lleva a cabo en MPLAB.

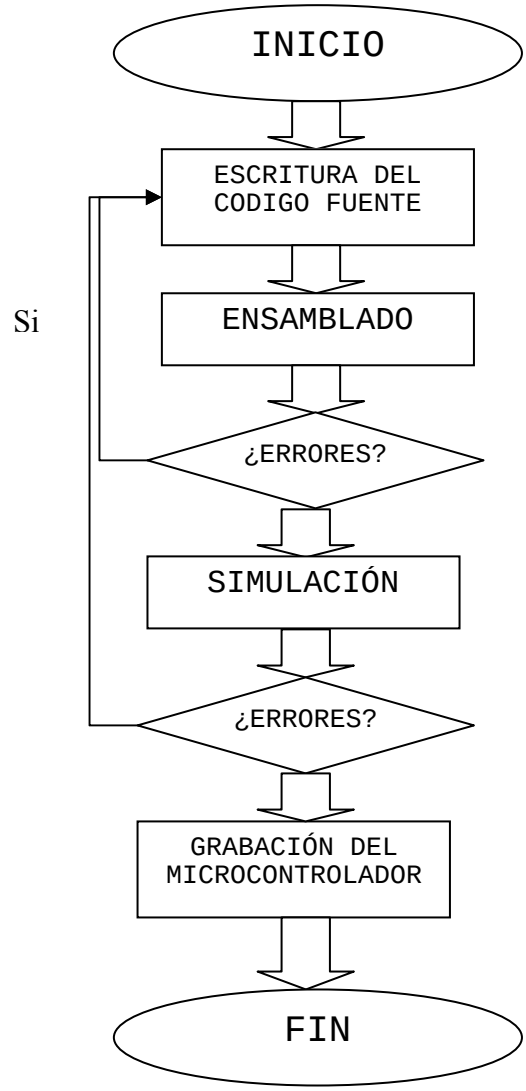
Para la programación del PIC, se siguen las siguientes fases de implementación, en las que de hecho, se supone realizada una fase de estudio del problema, elegido ya el mejor microcontrolador(en este caso el 16F877), así como decidido el sistema de conexión de pines de E/S correcto.

Como se puede ver en el siguiente esquema mediante un editor de texto o directamente utilizando el que nos proporciona MPLAB creamos el archivo .ASM del programa del PIC en ensamblador (el del 16F877) junto con las directrices de MPASM necesarias para la configuración (palabra de configuración) del PIC y la ordenación de los bloques del programa en la memoria FLAS del PIC , esencialmente poniendo las direcciones de salto en la de reset y de interrupción con las correspondientes a el inicio del programa en si y a la rutina de interrupciones.

Posteriormente se procede al ensamblado mediante el MPLAB, queremos que la salida sea un archivo hexadecimal para que podamos programar el PIC con la placa grabadora que veremos después no hay errores seguimos a delante y en caso de necesidad simulamos lo que nos permita el MPSIM (simulador contenido en MPLAB), para comprobar el correcto funcionamiento de lo que hemos programado (así vemos que realmente el PIC responderá como nosotros queremos).

En caso de que haya errores o el comportamiento no sea adecuado, volveremos a rescribir el programa en ensamblador.Esto lo realizaremos hasta alcanzar el comportamiento deseado.

Una vez conseguido el programa deseado y realizado el ensamblado (por lo que tenemos un archivo hexadecimal .HEX) realizamos la grabación en la memoria FLASH del PIC como se detalla en el siguiente punto.



Esquema de creación del programa del PIC

El MPLAB es un entorno de desarrollo integrado que le permite escribir y codificar los microcontroladores PIC de Microchip para ejecutarlos. El MPLAB incluye un editor de texto, funciones para el manejo de proyectos, un simulador interno y una variedad de herramientas que lo ayudarán a mantener y ejecutar su aplicación. También provee una interfase de usuario para todos los productos con lenguaje Microchip, programadores de dispositivos, sistemas emuladores y herramientas de tercer orden.

Para más información sobre el MPLAB ver el pequeño manual en el anexo adjunto al final de la memoria del proyecto.

- Programación del PIC mediante tarjeta programadora , para lo cual utilizaremos el programa Icprog (grabador de programas y datos) que nos permite cargar el archivo hexadecimal creado anteriormente con el MPLAB a partir del programa en ensamblador (junto con las directrices de programación en MPASM , como son la palabra de control y ubicación de las partes del programa en la memoria FLASH).Esta grabación de datos se realiza por el puerto serie del PC(el cual proporciona las tensiones necesarias para el proceso) conectado al conector serie de la placa programadora (grabador JMD).
  El Icprog es un programa de libre distribución que podemos encontrar en

  ic-prog.net , aunque si utilizamos el entorno Windows XP necesitaremos un archivo de sistema adicional para lograr que funcione , icprog.sys .

Para un detallado estudio de la placa programadora así como un pequeño manual de funcionamiento del Icprog consultar los anexos correspondientes al final de la memoria del proyecto.

2.4.3 Diseño de aplicaciones de control para PC

Para controlar desde el PC todo el proceso necesitamos unas aplicaciones software que al ejecutarlas den lugar al control deseado.Estas se desarrollaran con el programa Labview , sobre el que funcionan.
Labview permite crear instrumentos virtuales y aplicaciones de uso en tiempo real , crearemos una para el control paso a paso de la placa al que llamaremos panel de control , que al ejecutarla (una vez dentro presionar el botón de la flecha blanca arriba) permita el mando de una secuencia de instrucción y /o la recepción de un dato o parámetro desde el PIC.
Además crearemos varias aplicaciones para automatizar el proceso de configuración, transmisión y recepción del MODEM, creando un sistema de programas y datos en archivos almacenadles que luego se pueden transferir secuencialmente al PIC (por el puerto serie) para la ejecución de las anteriores tareas, incluyendo el almacenamiento de los datos recibidos por el MODEM en un archivo.
Todo esto se vera con más detalle en el capitulo dedicado a las aplicaciones de Labview.

2.5 Información suplementaria

Como se ha venido comentando antes se adjuntan una serie de anexos correspondientes a las siguientes materias para la mejor comprensión del desarrollo y la funcionalidad de la placa controladora en si.

2.5.1 Placa programadora por puerto serie para PIC

En este anexo contamos con el esquema y descripción de la placa grabadora JMD así como su software de funcionamiento Icprog , para su uso en el caso de tener que reprogramar la FLASH del PIC con un programa que incluya alguna función no alcanzada con el actual , o por si hay malfuncionamiento del PIC y debe ser reemplazado por otro.
Además se incluye un pequeño manual de MPLAB para la obtención del archivo hexadecimal que necesita la placa grabadora a partir del archivo .asm en el que se escriba el nuevo programa (modificación del actual o no) , y que describe la funcionalidad y manejo básico del MPLAB en cado uno de los pasos del esquema de creación de programa del PIC anteriormente visto(edición , simulación) .

2.5.2 Información del controlador y el MODEM

Estos anexos contienen información básica sobre los Principales chips implicados en la placa controladora:

- El controlador en si (PIC16F877) , con sus características básicas(según su datasheet) , el funcionamiento de sus periféricos utilizados y algún otro que podría emplearse en posteriores reprogramaciones.Asi como su juego básico de instrucciones en ensambladora sus principales registros y organización de memoria.
- El MODEM a controlar ; sus características básicas , su funcionamiento ,sus registros y su esquema circuital procedentes de su datasheet para su mejor comprensión y utilización de la placa controladora , pues aunque se han implementado algunas funciones , aun se pueden hacer muchas más en base al interfaz hardware que nos controla el MODEM , especialmente debido a los accesos de escritura y lectura del mismo , ya que permiten configurarlo de cualquiera manera imaginable.

3. ESQUEMA Y REALIZACION DEL HARDWARE

Debemos realizar una placa controladora para poder gobernar el MODEM , para ello en la misma placa dispondremos el controlador y el MODEM interconectados (para un correcto control del MODEM a través del microcontrolador PIC) así como los circuitos necesarios para cubrir las necesidades circuitales de cada uno de los dos.

3.1 Funcionalidad y necesidades de la placa

La placa controladora debe satisfacer las siguientes necesidades del MODEM:

- Alimentación a 3.3V.
- Señal de reloj externa a 6 MHz.
- Gobierno de la señal de Reset de manera externa.
- Señal de interrupción a nivel alto disponible para dispositivo externo.
- Conexión del bus de datos/direcciones Dio para correcto funcionamiento de los ciclos de lectura y escritura.
- Conexión del bus de control del bus Dio (ASN,WEN,OEN) para correcto gobierno de los ciclos de lectura y escritura , ejecutados sobre el MODEM por el controlador externo.
- Señales de frecuencia PLLs externas para uso opcional.
- Tensiones de referencia estables para correcto funcionamiento de los transmisores y el receptor del MODEM.
- Salida accesible de los Transmisores del MODEM .
- Salida accesible de las entradas del MODEM.
- Control o anulación de los modos de test del MODEM.
- Filtros externos para el funcionamiento de los PLLs internos del MODEM.
- Fuente de intensidad constante de 5mA para un correcto funcionamiento.

Así mismo el PIC tiene las siguientes necesidades:

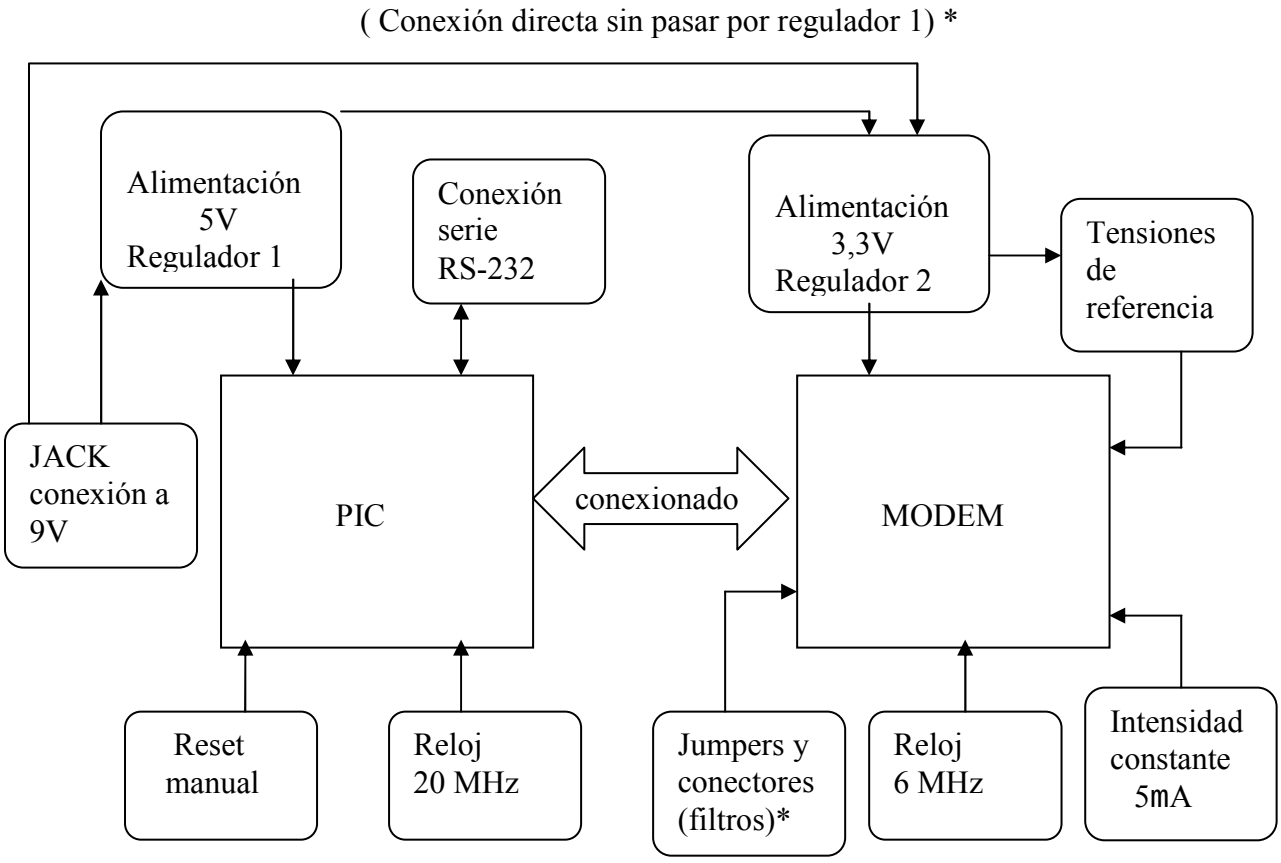
- Alimentación a 5V.
- Señal externa de reloj a 20 MHz.
- Reset manual mediante pulsador.
- Circuito de comunicación serie para la USART adaptándola al interfaz RS-232 ,con lo que la consideraremos como un puerto serie .

Además la placa necesitara de una fuente de alimentación de la cual obtener las alimentaciones de todos los componentes del circuito.Esto se lleva a cabo mediante un conector JACK al cual le siguen dos circuitos de regulación (cada uno para una tensión distinta , la de 5V y la de 3.3V requeridas).La alimentación de la placa se leva a cabo por tanto con la conexión de una pila de 9V o un transformador de la red eléctrica a 9V conectado al JACK de la placa.

La funcionalidad de la placa es satisfacer las anteriores características de los elementos antes expuestos para el correcto gobierno del MODEM por el controlador PIC. Este lo programaremos para poder gobernarlo desde un puerto serie (PC u otro dispositivo que cumpla con el interfaz RS-232).

3.2 Esquema general realizado

Para realizar todo lo dicho anteriormente utilizamos los circuitos necesarios ,que podemos agrupar en bloques , viendo así más fácilmente el circuito total implementado en la placa controladora mediante el siguiente esquema de bloques:



Esquema general de la placa controladora en sus dos versiones

*Se refiere a la segunda placa (modificada)

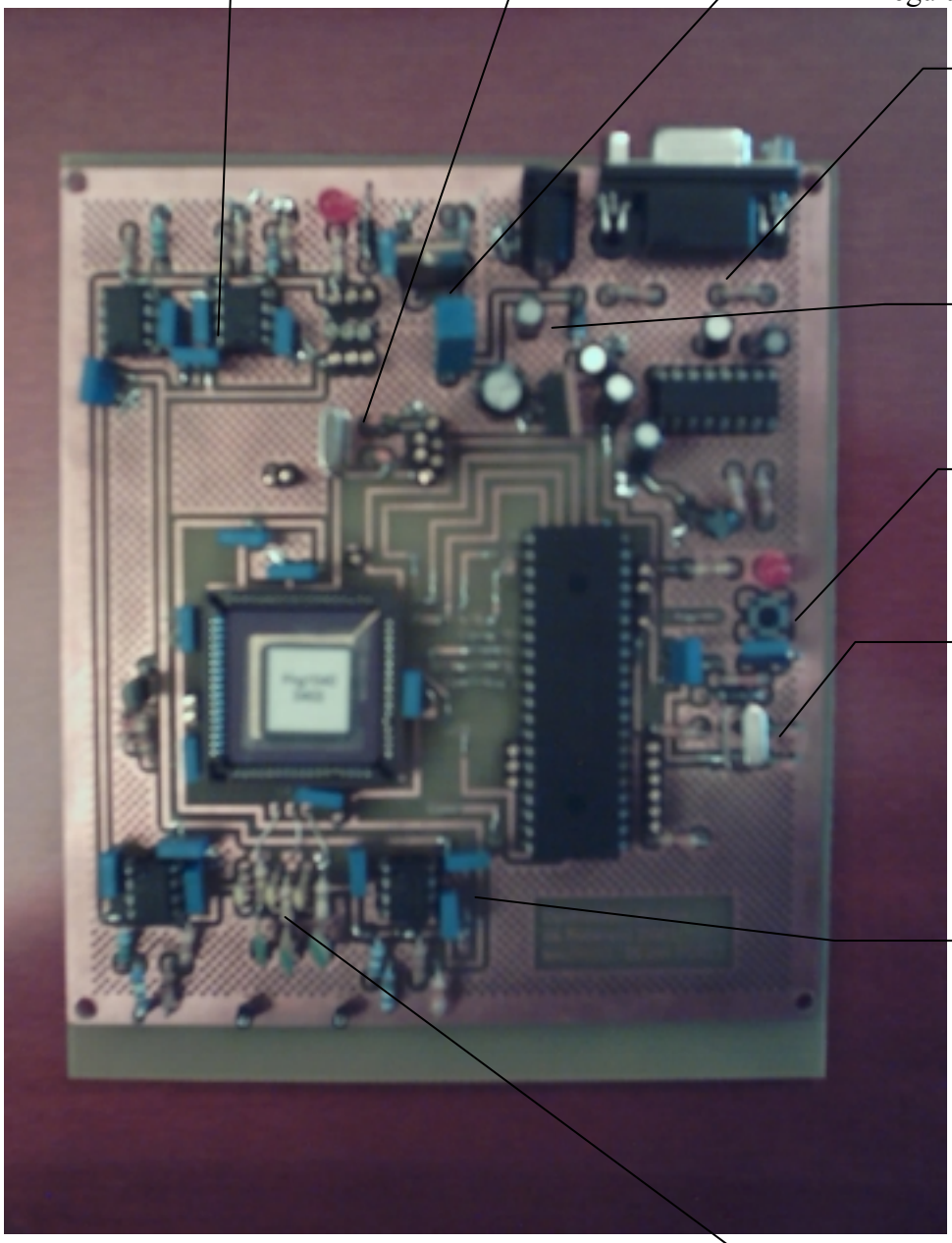
3.2.1 Descripción de los bloques

Procedemos a explicar brevemente los bloques de que consta la placa:

- Alimentación :JACK y reguladores.
  Este bloque proporciona la alimentación a la placa , así como los niveles de tensión necesarios para el funcionamiento del MODEM y del PIC.
  En la primera placa los dos reguladores estarán puestos en serie de manera que el segundo regulador se alimenta del primero , alimentándose el primero directamente del JACK.En cambio en la segunda placa separaremos los reguladores para menor sobrecarga del primer regulador , alimentándose los dos del JACK .
  Gracias al JACK podremos alimentar la placa con un transformador conectado a la red eléctrica o directamente con una pila de 9V.
- Relojes: Uno de 6 MHz para el MODEM y otro de 20 MHz para el PIC.Nos proporcionan las señales de reloj necesarias mediante un cuarzo y un par de condensadores (al ser excitados por una tensión proveniente del dispositivo al que necesita la señal de reloj).
- Jumpers y conectores: Disponemos de conectores a las tensiones de alimentación del PIC y del MODEM así como a tierra así como un par de Jumpers para anular los modos de Test del MODEM o gobernarlos mediante el PIC.
  La primera placa además cuenta con un mallado de conexiones para insertar los filtros externos necesarios para los PLLs internos del MODEM.
- Tensiones de referencia : Proporciona las tensiones de referencia necesarias para el perfecto funcionamiento del MODEM a partir de la salida del segundo regulador , además estas tensiones son estabilizadas mediante seguidores de tensión realizaos con amplificadores operacionales.
- Intensidad constante: Mediante una fuente de intensidad constante alimentada por la salida del primer regulador se obtiene la intensidad necesaria para el funcionamiento del MODEM.
- Conexión serie:Se trata de un circuito adaptador al estándar RS-232 para la comunicación serie con un puerto serie estándar como el de un PC.Es un bloque vital ya que nos proporcionara la comunicación con la tarjeta controladora.
- Reset manual: mediante este bloque proporcionamos la manera de resetear el controlador de manera manual pulsando un simple boton.Al resetear el controlador se resetea automáticamente el MODEM , al estar su reset gobernado por el PIC.
- PIC: Es el controlador en si , en un zócalo de manera que se pueda extraer y programar a voluntad.
- MODEM : Es el MODEM , en un zócalo de manera que se pueda extraer e intercambiar a voluntad.
- Interconexionado: bloque de rutado entre el MODEM y el PIC , con frecuencia se emplearan vías para establecer la conexión debido a la gran cantidad de estas y su disposición en la placa.

3.3 Elección de componentes y división del circuito

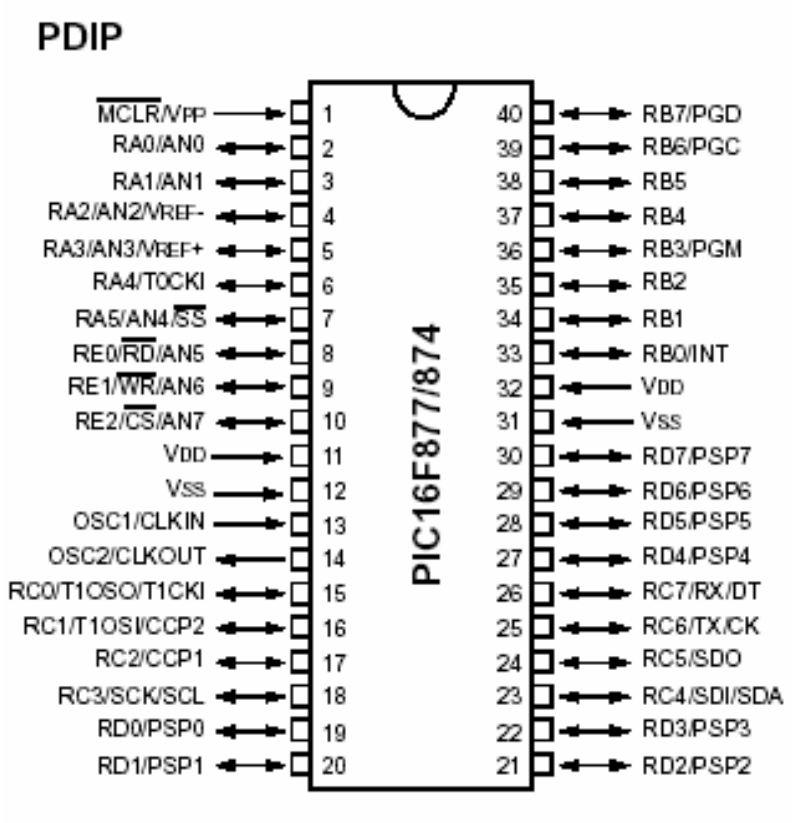
La placa controladora se basa en el PIC 16F877 de microchip dado que nos proporciona un control adecuado del MODEM utilizando casi todas sus características ; sus periféricos USART , Timers y PWM , así como la totalidad de sus puertos : quedando solamente el puerto A y algunos pines de los puertos B y C sin utilizar , pero accesibles para uso posterior(posibles modificaciones del programa).
Además como hemos visto anteriormente tenemos en cuenta las necesidades circuitales del MODEM para su buen funcionamiento .
Veremos a continuación cada uno de los bloques funcionales que necesitan el PIC y el MODEM descritos en detalle en su implementación circuital , así como la interconexión de estos para realizar el control deseado.



2º placa controladora , con filtros externos

3.4 Circuitos básicos del Microcontrolador

Microchip posee diversos encapsulados para la protección del microcontrolador PIC16F877; aquí se representa el encapsulado PDIP que se puede conectar a la placa microcontroladora sin necesidad de soldarlo a ella, mediante zócalos.



El PIC se encajara en un zócalo PDIP-40 para su fácil manejo , ya que será necesario extraerlo si queremos reprogramarlo o cambiarlo por otro.
El PIC necesita de unos circuitos básicos para su funcionamiento , estos se conectan a él mediante los siguientes pines:

Pines del PIC conectados a circuitos básicos para su funcionamiento	Conexión a bloque
1	Reset manual
11,32	Alimentación (5V)
12,31	GND
13,14	CLK 20 MHz
26,27	Comunicación serie (RS232)

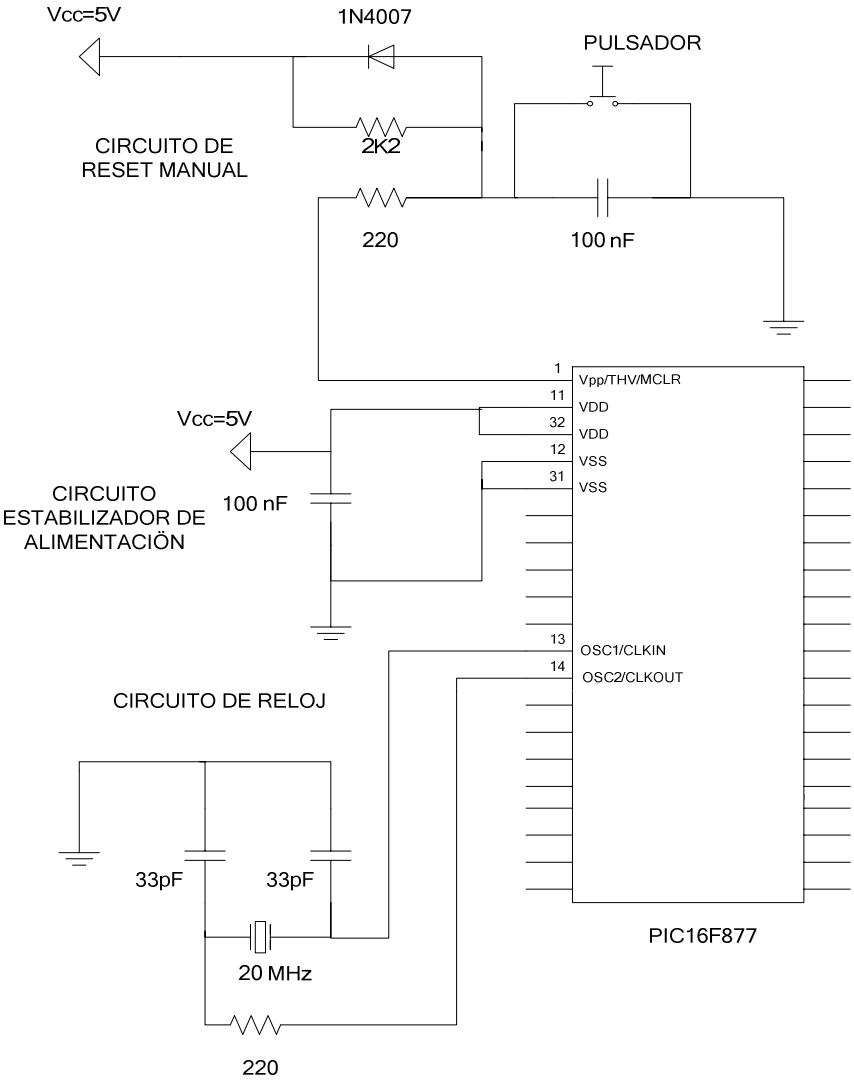
Tabla de pines del PIC conectados a circuitos básicos

Procedemos a ver en detalle cada uno de estos circuitos:

- Circuito de Reset manual:
  Este circuito cuenta con un pulsador para resetear el PIC reinicializando la placa controladora a su estado de espera de ordenes por el puerto serie sin tener que desconectar la alimentación, consta de los siguientes elementos conectados como se muestra en el esquema , alimentándose a 5 V, y conectándose al pin pertinente del PIC.

El interruptor pulsador, conectado al negativo de la alimentación esta en paralelo con un condensador de 100nF que se usa para estabilizar y eliminar los ruidos que introduce el pulsador. Este circuito también posee una resistencia de Pull-Up, y un diodo 1N4007 de protección.

- Circuito de reloj:
  Se basa en un cuarzo de la frecuencia que necesita el PIC(20MHz) junto con un par de condensadores cerámicos de 33pF y una resistencia limitadora para su conexión al PIC , tal y como se muestra en el esquema.
- Circuito estabilizador de tensión:
  Se trata simplemente de un condensador de estabilización entre la alimentación y la tierra del PIC , ver el siguiente esquema.

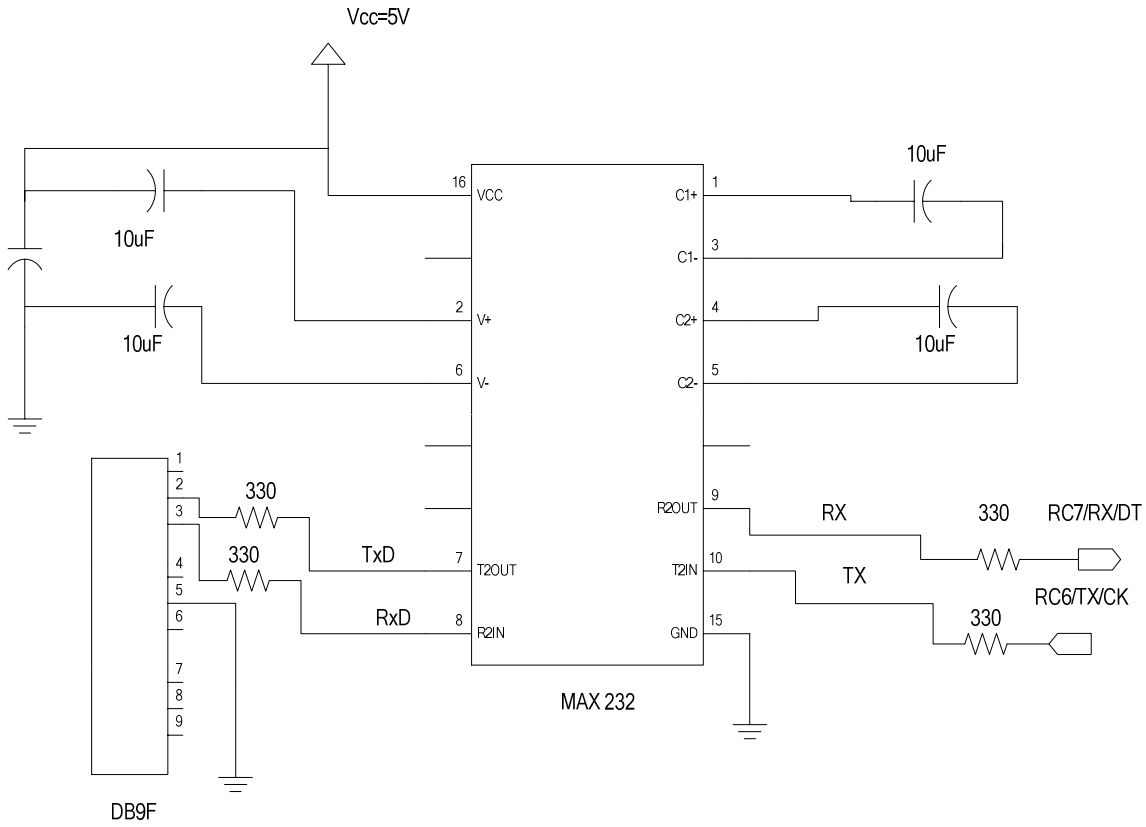


Circuitos básicos del PIC

- Circuito de comunicación serie :

Es un circuito de adaptación al estándar RS-232 para poder operar con la USART del PIC a través de un cable serie normalizado al anterior interfaz mediante un conector DB9.

Se basa en el chip adaptador MAX 232 al cual se le conectan los siguientes elementos para un correcto funcionamiento , como se puede ver en el esquema:



Circuito adaptador al interfaz RS-232

De nuevo este circuito también esta alimentado a 5 V , por lo que utiliza alimentación de la salida del primer regulador.

Este circuito se encarga de la comunicación serie entre el microcontrolador y el PC, esta comunicación es full-duplex o bidireccional asíncrona. Para la realización de este tipo se utiliza el protocolo RS-232-C, donde cada palabra de información o dato se envía independientemente de las demás.

Este circuito está formado básicamente por el integrado MAX232, que se encarga de convertir los niveles lógicos TTL presentes en la patita 10 de transmisión (Tx), a niveles lógicos RS-232, que se obtienen por la patita 7 (TxD) y se aplican al conector serie DB9F.La señal de recepción RxD llega desde el canal serie a la patita 8 con niveles RS-232 que son convertidos a niveles TTL y se obtienen por la patita 9 (Rx).

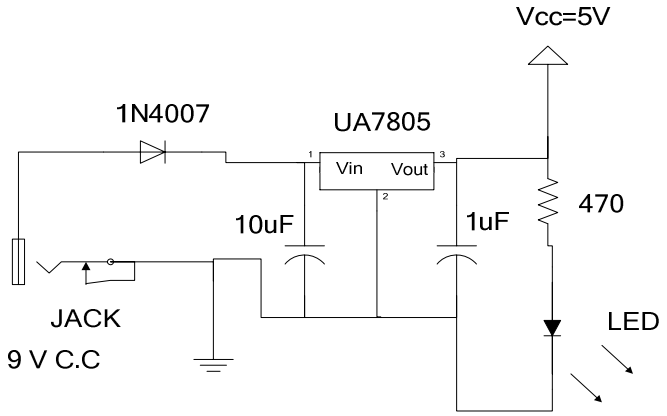
Tanto la señal de recepción Rx como la de transmisión Tx se controlan desde las patitas RC6 y RC7 cuando se está trabajando con un microcontrolador que posea un módulo USART por hardware, capaz de soportar la comunicación serie sincronía y asíncrona.

Para el correcto funcionamiento del circuito MAX232, se le conectan 4 condensadores electrolíticos de 10uF respetando la polaridad indicada en el. También se incluye un condensador de 10uF entre Vcc y GND para mantener la estabilidad de la tensión de alimentación en el funcionamiento de la comunicación serie.

Por ultimo en el circuito se incluyen 4 resistencias limitadoras de 330Ω, para los canales de transmisión y recepción, tanto en los niveles TTL como en los niveles RS232.

- Circuito de alimentación:

Se trata del circuito inicial del que parten todas la alimentaciones de la placa a partir de la tensión continua suministrada en el JACK de entrada.
Mediante el siguiente montaje del primer regulador (ya que tendremos otra etapa de regulación para obtener las tensiones del MODEM acoplada a esta) obtenemos un nivel de continua de 5 V necesario para la alimentación del PIC y sus circuitos básicos de funcionamiento , además de servir para las etapas estabilizadoras de las tensiones de referencia del MODEM(alimentación de los OPAs).El LED nos dirá si esta funcionando la alimentación .
Se basa en un montaje del regulador UA7805 que da salida directa de 5 V , como se puede ver en el siguiente esquema:



Circuito de alimentación del primer regulador

La tensión de entrada es aplicada a la placa mediante un conector Power Jack, que es un conector hembra de 2.1 mm x 5.5 mm coaxial y posee una toma de tierra en el centro del conector que puede ser utilizada.

Esta tensión es proporcionada mediante una fuente de alimentación que está compuesta de un transformador de 9 VAC con una tensión de entrada de 230 V ~ 50/60 Hz y una tensión de salida de 9 V C.C con una intensidad 500 mA. También se puede usarse una pila de 9 voltios.

Para la obtención de una tensión continua de 5 voltios estable, se utiliza un regulador de tensión (UA7805), dos condensadores electrolíticos de 10 y 1 uF, para ayudar en la estabilización de dicha tensión; y un diodo 1N4007 que proporciona una protección de polaridad para la fuente de alimentación externa. El dispositivo regulador de tensión UA7805 posee 3 terminales, uno por donde se le proporciona la tensión de entrada (V_I) que puede variar entre 7 y 25 voltios, otro es el terminal donde se conecta

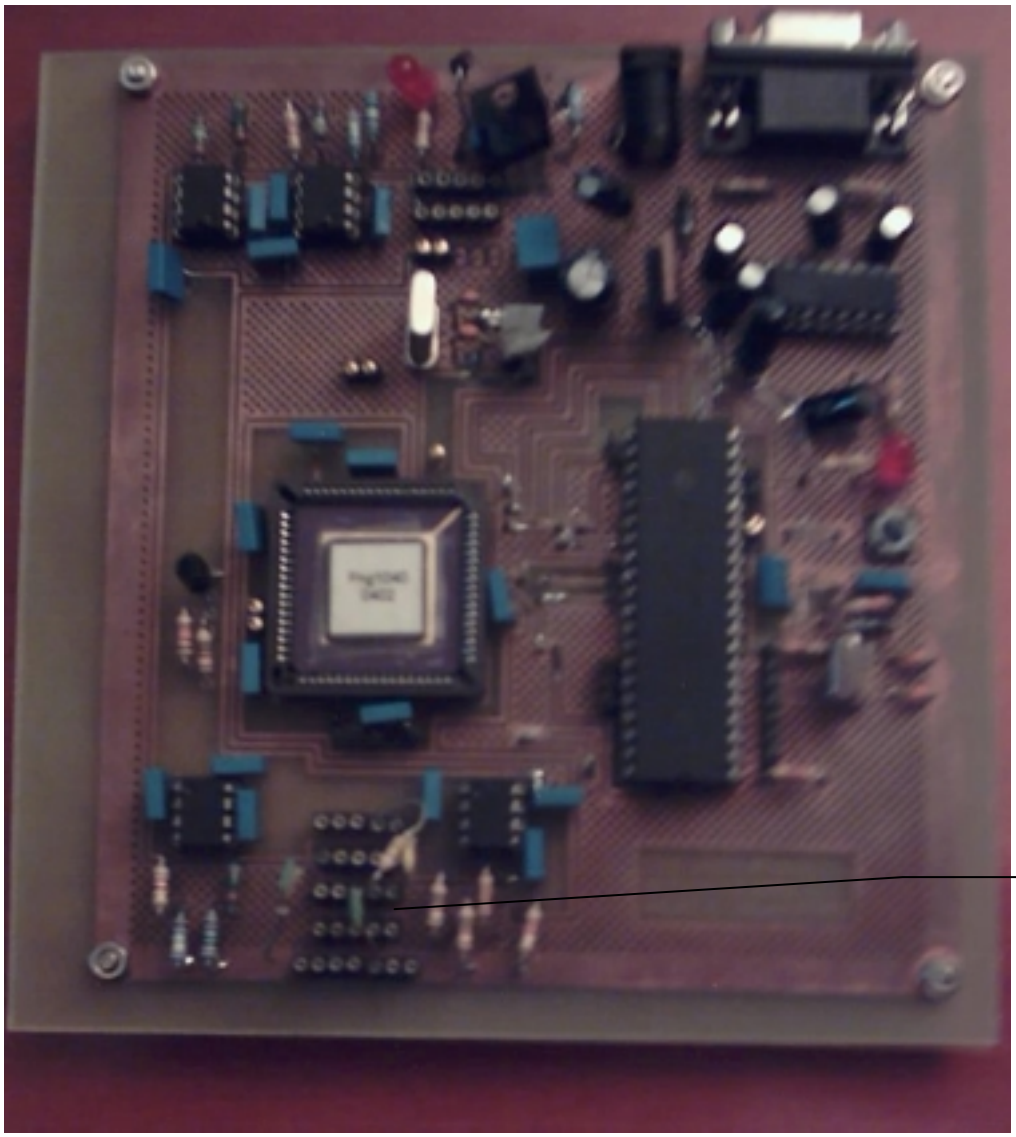
el negativo de la alimentación (COMMON), y por ultimo el terminal de la tensión de salida (V_O), que es la que nos proporciona los 5 voltios.

Por ultimo este circuito de alimentación posee un diodo LED y una resistencia limitadora de 470 Ω, que proporciona una indicación de cuando está conectada la alimentación.

A parte de los pines utilizados anteriormente ,y los utilizados en la interconexión con el MODEM para su control , existen pines del PIC libres accesibles desde pines externos .Estos podrían utilizarse en posteriores reprogramaciones del controlador por si hicieran falta :

Pines libres del PIC a conector externo	Tipo
RBi , i=1..3	E/S programable
RCi , i=0,2,3	E/S programable
RAi , i=0..5	E/S programable

Tabla de pines accesibles del PIC sin propósito definido



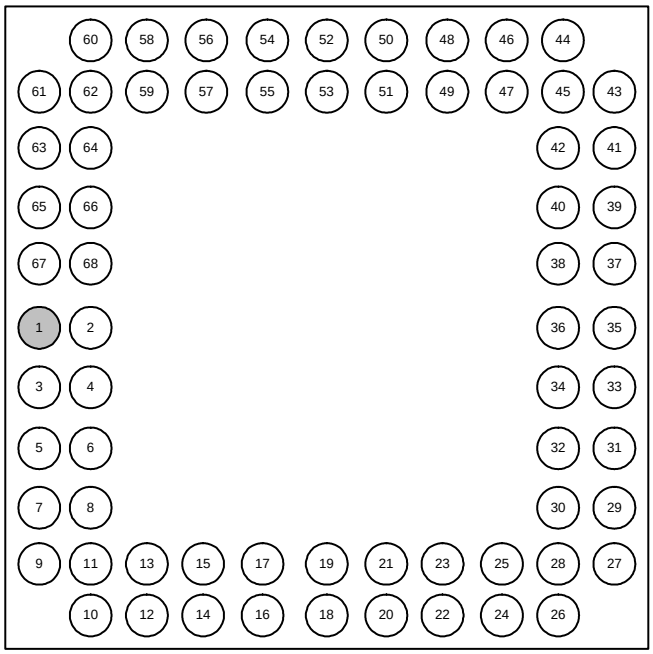
Placa 1 con mallado de conexiones y sin separación de reguladores

3.5 Circuitos básicos del MODEM

El módem bajo prueba se trata de un chip de 68 pines, de los cuales sólo 54 pines se encuentran conectados y los 14 restantes al no estar conectados deben dejarse al aire. Para conocer un poco el esquema debemos conocer la correspondencia entre los pines del chip y las señales que llevan conectadas, porque sólo de esta forma podremos abordar su conexión con el resto del circuito diseñado en la placa. Para la inclusión del

chip en la placa diseñada necesitaremos de la utilización de un zócalo de tipo PGA 68 en el cual se introducirá el chip.

En la gráfica siguiente se puede observar un esquema en el que se muestra la numeración de los pines del zócalo a utilizar, esto es necesario conocerlo para el correcto conexionado del zócalo a la placa diseñada.



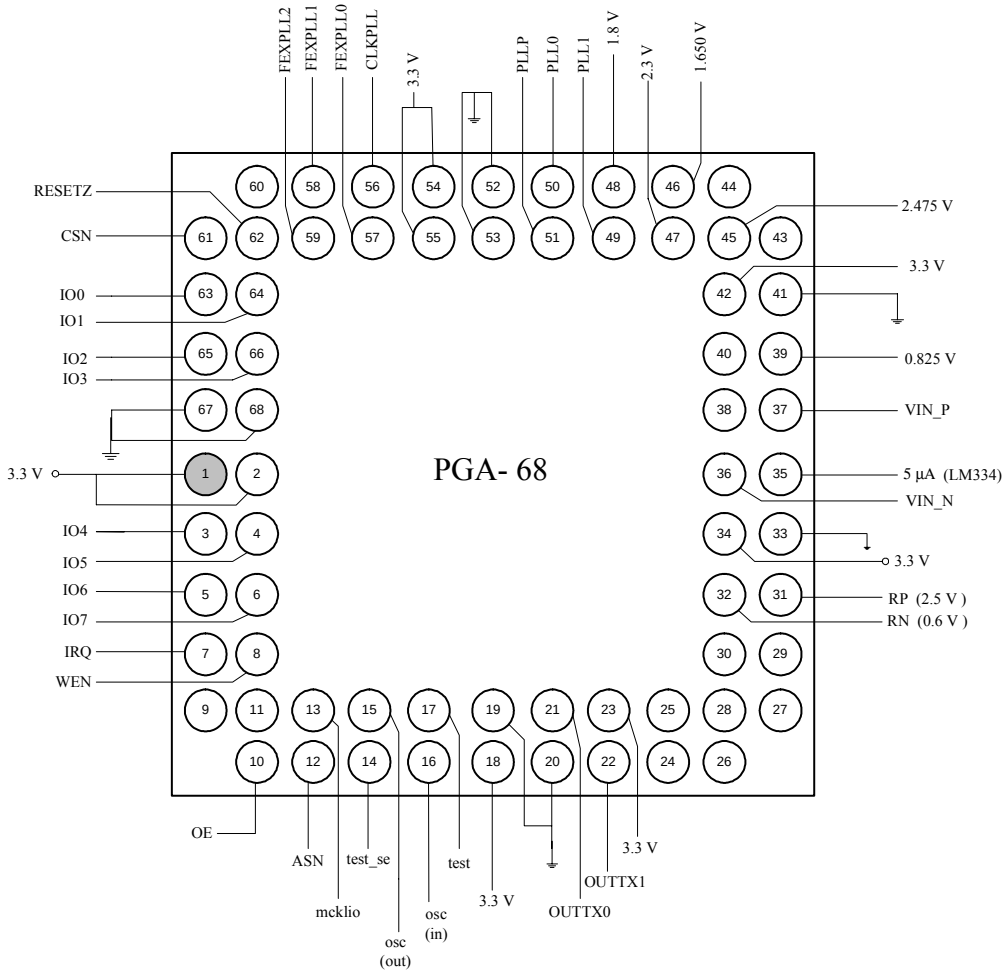
Numeración y ubicación de los pines del zócalo PGA-68

En la tabla siguiente se muestra la correspondencia entre los pines del chip y las señales asociadas a los mismos:

Nº Pin	Señal
1	PWR03P
2	PWR03C
3	io4
4	io5
5	io6
6	io7
7	Irq
8	Wen
9	N.C.
10	Oen
11	N.C
12	Asn
13	Mcklio
14	test_se
15	osc (out)
16	osc (in)
17	test
18	PWR01P, PWR01C
19	GND01P,GND01C
20	TXVSS (1,2,3)
21	OUTTX0 (a,b,c)
22	OUTTX1 (a,b,c)
23	TXVDD (1,2,3)
24	N.C.
25	N.C.
26	N.C.
27	N.C.
28	N.C.
29	N.C.
30	N.C.
31	RN
32	RP
33	ANAVSS2
34	ANAVDD2

Nº Pin	Señal
35	IREF5u
36	VIN_N
37	VIN_P
38	N.C.
39	VREF0825
40	N.C.
41	ANAVSS1
42	ANAVDD1
43	N.C.
44	N.C.
45	VREF_2475
46	VREF_1650
47	VREF_2300
48	VREF_1800
49	LPFPLL1
50	LPFPLL0
51	LPFPLLP
52	GND02P
53	GND02C
54	PWR02P
55	PWR02C
56	CLKPLL
57	FEXPLL0
58	FEXPLL1
59	FEXPLL2
60	N.C.
61	Csn
62	Resetz
63	io0
64	io1
65	io2
66	io3
67	GND03C
68	GND03P

Por último para poder realizar el conexionado de forma correcta vamos a ver un esquema en el que se muestra la señal eléctrica que debe conectarse a cada pin del zócalo PGA-68, esto vamos a verlo así para que pueda entenderse mejor el PCB diseñado y que pueda ser utilizado con posterioridad para realizar otro diseño diferente.



Esquema de conexión del zócalo PGA-68

.

En la siguiente tabla podemos observar la utilización de los pines del MODEM que no se conectan al PIC.Entre ellos podemos distinguir los que se conectan a un bloque circuital básico para el funcionamiento del MODEM y los que simplemente no se conectan a nada al no tener ninguna función especifica:

Pines del MODEM conectados a circuitos básicos para su funcionamiento	Conexión y funcionalidad
9,11,24-30,38,40,43,44,60	Nada
1,2,18,23,34,42,54,55	Alimentación (3,3 V)
19,20,33,41,52,53	GND
15,16	CLK 6 MHz
31	Tensión de referencia Tx (2,5 V)
32	Tensión de referencia Tx (0,6 V)
35	Regulador intensidad 5mA
39	Tensión de referencia Rx (0,825 V)
45	Tensión de referencia Rx (2,475 V)
46	Tensión de referencia Rx (1,650 V)
47	Tensión de referencia Rx (2,3 V)
58	Tensión de referencia Rx (1,8 V)
50	PLL0 * ,Conexión a filtro externo
51	PLL1 *,Conexión a filtro externo
49	PLLP *,Conexión a filtro externo

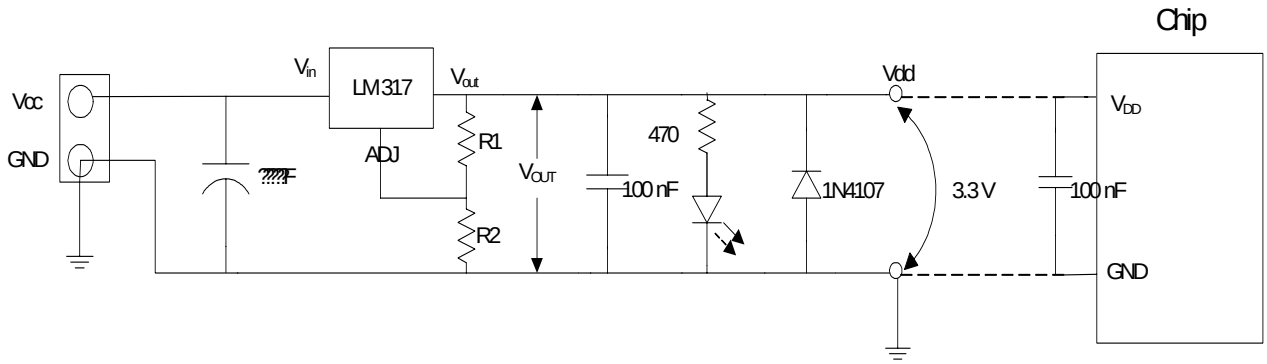
Tabla de pines del MODEM de circuitos básicos

*Conectados a un mallado de conexiones donde pondremos los filtros externos necesarios para el funcionamiento de los PLLs internos.

Pasamos a continuación a describir los bloques circuitales básicos del MODEM :

- Circuito de alimentación:

El circuito de alimentación consta básicamente de una fuente de alimentación de 3.3V , esta fuente se consigue con un regulador de tensión LM317 de *National Semiconductor* al que se le añaden las resistencias correspondientes para que nos proporcione a su salida los 3.3V constantes necesarios para la alimentación del chip**.**La tensión de partida Vcc del circuito es la salida del primer regulador , es decir la etapa de regulación que transforma la tensión de entrada a la placa por el JACK a 5 V. Podemos ver un esquema del circuito de alimentación en la siguiente figura:

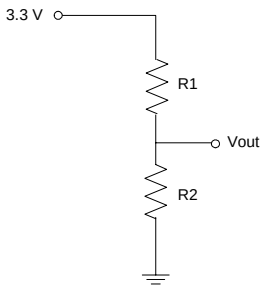


Esquema de alimentación del MODEM , 2º regulador

Los valores de las resistencias que aparecen en la figura anterior y que nos permiten obtener los 3.3 V son R1= 240 Ω y R2= 392 Ω.
Aun así debido a irregularidades en la regulación el valor de salida llega a estabilizarse en torno a los 3.8V en vez del valor requerido , aunque no se ha encontrado la causa(calentamiento de los reguladores o alguna irregularidad no detectada en el LM317).
El LED nos indica si esta conectada la alimentación o no , se utilizan también condensadores a la entrada y salida para estabilización y control de picos , y un diodo de protección.
A las entradas del MODEM de alimentación se estabilizan con un condensador de 100nF entre los 3.3V y tierra(1 condensador para cada entrada de alimentación).

- Tensiones de referencia:

Además de esta tensión de alimentación , el integrado requiere otros 7 niveles más de tensión para la alimentación de los distintos subsistemas que lleva en su interior, como el convertidor sigma-delta, etc. Estos 7 niveles de tensión adicionales se consiguen a partir de los 3.3 V que ya tenemos colocando el mismo número de divisores resistivos a la salida del regulador de tensión con los valores adecuados de las resistencias para obtener los diferentes niveles de tensión. Veamos a continuación un ejemplo del divisor resistivo:



Esquema de divisor de tensión

Seguidamente vemos las tensiones de referencia necesarias para el MODEM ordenadas según su ubicación en la placa de izquierda a derecha , en el sentido inverso a las agujas del reloj ,

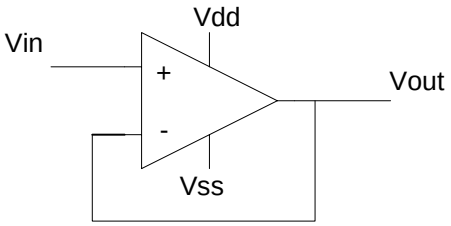
comenzando en la parte izquierda de la placa (teniendo el conector DB9 y el JACK hacia la izquierda)

Vref teórico(V)	R1(Ω)	R2(Ω)	Vref (V)*
0.6	2430	604	0.6569
2.5	820	2490	2.4824
0.825	2490	825	0.8212
2.475	820	2430	2.4673
2.3	825	2K	2.3362
1.65	1K5	1K5	1.65
1.8	1K	1K2	1.8

Tabla de los divisores de tensión

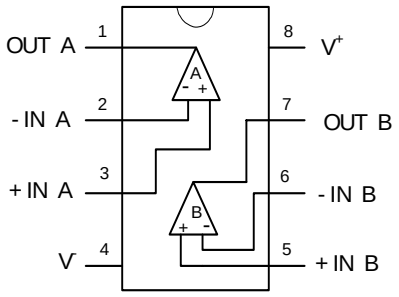
* Debido a las tolerancias de las resistencias de un 5% y a cierta irregularidad en la etapa previa de regulación el valor real varía un poco , aumentando algunas centésimas de voltio.

Cada una de estas tensiones se hacen pasar por un amplificador operacional a través del integrado LM6032 de *National Semiconductor* utilizando para cada uno de ellos una configuración de seguidor de tensión como el de la figura con la intención de que nos regulen la corriente que pasará al chip



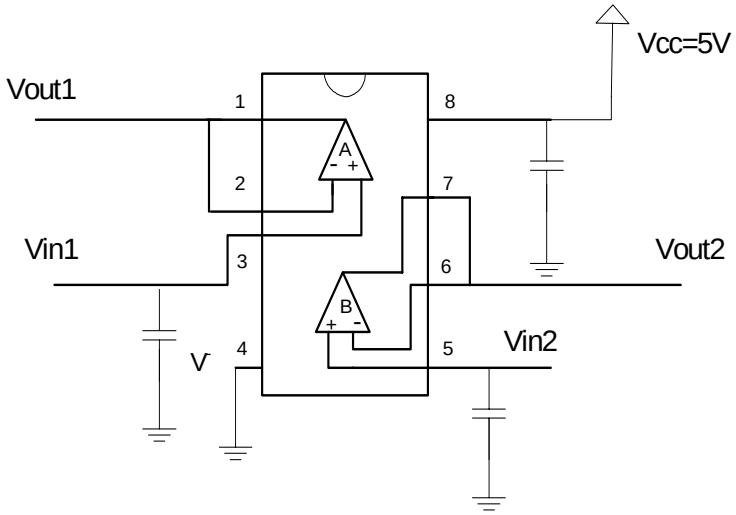
esquema de seguidor de tensión utilizado

Para poder realizar correctamente el montaje de seguidor de tensión para el amplificador operacional es necesario conocer el patillaje y el esquema interno del integrado *LM6032*, a continuación se muestra una figura en la que se detalla dicho esquema:



Esquema de los chips 6032

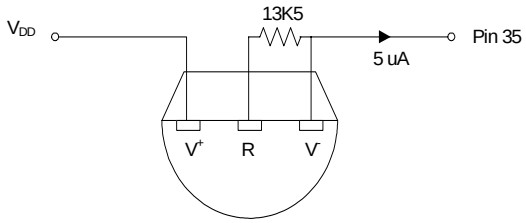
Una vez que se saca la corriente del *LM6032* ya pasa directamente al chip , a cada entrada del integrado ponemos un condensador de 100 nF que nos protege la señal de posibles picos también se opera de igual modo(condensadores de 100nF) para la alimentación del chip , utilizándose alimentación a 5 V de la salida del primer regulador como se puede ver en el esquema:



Circuito de estabilización de las tensiones de referencia

- Circuito de intensidad constante:

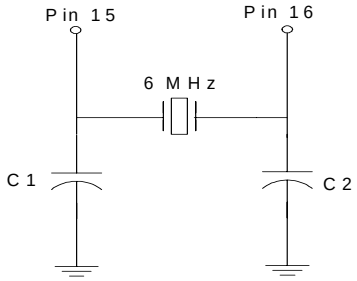
Se basa en el chip LM334 de *National Semiconductor* al que alimentamos con la tensión que proporciona la salida del primer regulador (5 V) y una resistencia en función del valor de intensidad que queramos conseguir , en nuestro caso 5µA .Debido a no tener resistencias de ese valor la componemos con dos en serie que alcanzan el valor deseado, que se puede ver en el siguiente esquema de conexionado , así como la conexión al MODEM.



Circuito de la fuente de intensidad constante

- Circuito de reloj:

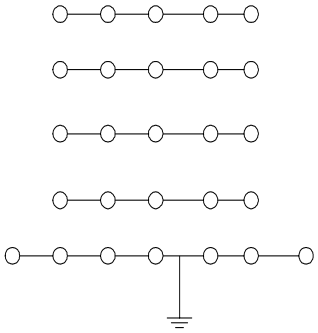
Utilizamos el siguiente montaje para obtener la señal deseada de reloj .Se basa en un cuarzo de 6MHZ y dos condensadores cerámicos de 33pF.



Circuito de la señal de reloj del MODEM

- Circuito de mallado y filtros externos:

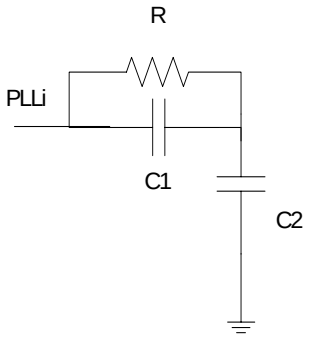
Realizamos un mallado de conexiones en el que podamos insertar los componentes de los filtros externos calculados para el funcionamiento de los PLLs internos del MODEM.
El esquema del mallado es el siguiente:



Mallado de conexiones para filtros externos

De manera que se conectaran los condensadores y resistencias calculados para los filtros externos fijándose el circuito a la tierra de la placa(ultima fila del mallado) , y conectándose mediante un cable el pin externo correspondiente a la salida para PLL del MODEM al filtro contenido en el mallado.

Los filtros que hacen que funcionen correctamente los PLLs internos del MODEM son tres filtros idénticos con esta estructura y componentes:



Esquema de los filtros externos utilizados

R vale 150KΩ , y el valor de los condensadores son : C2=22nF y C1=2n2F.
Se vio en su utilización que los filtros son correctos.

Además de los pines empleados anteriormente , y de los que se emplean para el control del MODEM mediante el PIC , existen otros accesibles mediante pines externos para el funcionamiento y supervisión del MODEM:

Pin del MODEM	Nº de pin	Tipo
Mcklio	13	E/S
OUTTXO0	21	Salida
OUTTXO1	22	Salida
Vin_N	36	Entrada
Vin_P	37	Entrada
CLKPLL	56	Salida

Tabla de pines a conector externo del MODEM

La funcionalidad de estos es la siguiente:

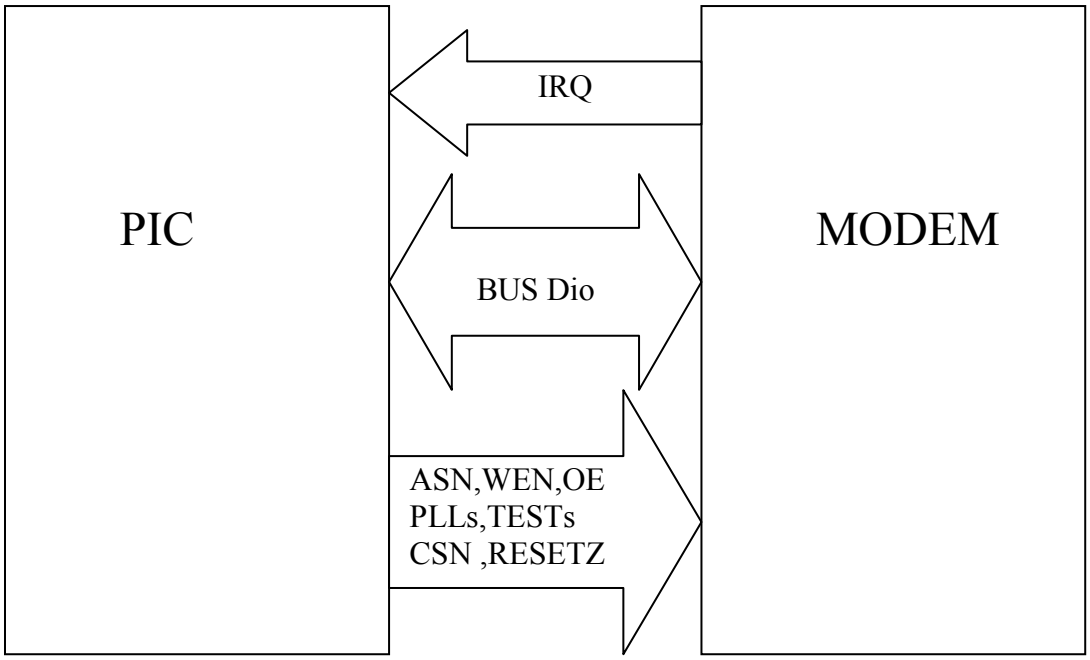
- Mcklio: Señal de sincronismo de canal 0 (para test).
- Salidas del MODEM : OUTTXO0 correspondiente a la salida del transmisor 0 del MODEM diseñado enteramente , y OUTTXO1 correspondiente a la salida del transmisor 1 del MODEM de baja potencia.
- Entradas del MODEM: Señales Vin_N y Vin_P correspondientes a la entrada del receptor del MODEM , las dos deben utilizarse para la misma recepción , ajustándose una de ellas al valor de tierra común a las placas y la otra al transmisor de otro (o del mismo en bucle cerrado para pruebas) MODEM.
- CLKPLL: Señal que nos permite observar el PLL interno del canal 0 del receptor del MODEM.

3.6 Esquema de conexiones PIC-MODEM

Las conexiones entre el PIC y el MODEM permiten el control del uno sobre el otro mediante señales que serán de tres tipos:

- Bidireccionales para el intercambio de datos ; mediante estas se lleva a cabo la comunicación entre los dos dispositivos , es decir las lecturas y escrituras de los registros del MODEM.
- Unidireccionales a MODEM ; señales de control propiamente dichas , activan o desactivan , sirven para funciones básicas como el reset , el habilitador del MODEM … ,y el control de las lecturas y escrituras.
- Unidireccionales a PIC ; en realidad solo es una señal , la de interrupción , que dará aviso al PIC de algún evento que hayamos programado previamente.

Lo anterior se puede ver claramente en el siguiente esquema :



Esquema de interconexionado PIC-MODEM

Los pines involucrados en estas conexiones en el MODEM y el PIC son los siguientes:

Nº Pin del MODEM	Señal del MODEM	Pin del PIC	Nº Pin
62	RESETZ	RB5	38
61	CSN	RB4	37
63-66 y 3-6	DIOi,i=0-7	RDi,i=0-7	19-22 y 27-30
7	IRQ	RB0	33
8	WEN	RE0	8
10	OEN	RE1	9
12	ASN	RE2	10
14	TEST-esc	GND*	
		RC5	24
17	TEST	GND*	
		RC4	23
59	FEXPLL2	RB7	40
58	FEXPLL1	RC1	16
57	FEXPLL0	RB6	39

Tabla de conexiones PIC-MODEM

*Conexión a tierra del jumper.

3.6.1 Control del MODEM

Procedemos a comentar las señales involucradas en el interfaz y control atendiendo al grupo al que pertenecen y su funcionalidad :

- Bidireccionales:
  - Bus de datos/direcciones(DIO): Se trata del bus de intercambio de datos(bytes) entre el MODEM y el PIC de manera que es bidireccional y controlado por el bus de control para los accesos de escritura y lectura , ya que siempre será de entrada para el MODEM salvo cuando se lean datos de este . Al escribir o leer datos del MODEM se debe cumplir con los requisitos de los ciclos de acceso de este , por lo que primero se le debe escribir la dirección del registro(PIC->MODEM) y posteriormente se escribe el dato o se lee el dato , si se lee un dato del MODEM se debe cambiar a entrada para el MODEM.
    Al estar conectado al puerto D del PIC que es bidireccional ,este mismo cambia la direccionalidad del puerto en función de si se debe escribir o leer de el , efectuándose fácilmente los accesos al MODEM.
- Unidireccionales a MODEM:
  - Bus de control(ASN,WEN,OEN): Este conjunto de señales gobiernan los accesos al MODEM , se corresponde con el puerto E del PIC configurado como salida de este ultimo , sigue el protocolo de acceso que utiliza el MODEM , siendo gobernado por el PIC fácilmente para realizar las lecturas y escrituras.

- Señales de Test:señales de activación para los modos de test , opcionalmente al control de estas señales (activas a nivel alto) se puede seleccionar su desactivación permanente mediante un jumper que las conecte a tierra.
- CSN:Señal de activación del MODEM a nivel bajo , nos sirve para seleccionar al MODEM antes de operar con el.
- RESETZ:Señal de reset del MODEM gobernado por el PIC , de manera que podemos resetearlo de manera software(mediante nuestro software sin necesidad de resetear manualmente la placa).
- Frecuencias PLLs externas:Frecuencias PLLs de los canales de recepción del MODEM opcionales al funcionamiento de los PLLs internos del MODEM y que son generadas por el PIC (PIC->MODEM).

➢ Unidireccional a PIC:

- IRQ: Esta señal de interrupciones del MODEM se conecta al pin de interrupciones externas del PIC que se programa como interrupciones a nivel alto , ya que así son las del MODEM, informándonos de algún evento que queramos resaltar en el funcionamiento del MODEM como puede ser la llegada de un dato al receptor o la necesidad de transmitir otro dato del transmisor.

3.7 Modificaciones del PCB(2ª placa)

Las modificaciones con respecto a la primera placa son básicamente dos:

- Desacople de las dos etapas de regulación , partiendo cada una de la alimentación del JACK en vez de estar las dos en cascada debido a las irregularidades encontradas en el 2º regulador y al excesivo calentamiento de ambos ; no obstante no se consiguió arreglar el desvío de 0.5V de más en la salida del segundo regulador , pero si se alivio el calentamiento de los dos a pleno funcionamiento del MODEM.
- Integración de los 3 filtros externos necesarios para el funcionamiento de los PLLs internos del MODEM y conexión a los pines correspondientes del MODEM mediante conexión aérea(cable por fuera), eliminando así el mallado de conexiones de prueba al comprobarse que eran los filtros adecuados.

Además debido a cuestiones de disponibilidad de resistencias se variaron los divisores de tensión quedando así:

Vref teórico(V)	R1(Ω)	R2(Ω)	Vref (V)*
0.6	2K	470	0.6279
2.5	470	1K5	2.5126
0.825	1K5	475	0.7936
2.475	475	604+825	2.4767
2.3	470	1K	2.2448
1.65	909	909	1.65
1.8	1K	1K2	1.8

3.8 Listado de componentes para la fabricación de las placas

A continuación se detalla la lista de componentes utilizados en la fabricación de cada una de las placas por si hiciera falta realizar alguna otra o buscar algún elemento en si que reemplazar:

3.8.1 Placa primera

Vemos el listado global de componentes utilizados:

- Zócalos: 1 PGA-68 , 1 DIP-40 , 1 DIP-16 y 4 DIP-8
- Pines: 37 zócalos y 23 externos.
- Chips:1 PIC16F877, 4 LM6032 , 1 MAX232,1 MODEM , 1 LM317 , 1 LM334 y 1 UA7805
- Diodos: 2 LEDS , 3 1N4007
- 2 Jumpers y 1 Pulsador
- Conectores: 1 DB9 y un JACK
- Relojes: 1 6 MHz y 1 20 MHz
- Condensadores: 20 100nF , 4 cerámicos 33pF , 1 1uF , 7 electrolíticos 10uF, 1 electrolítico 470uF,3 22nF y 3 2,2nF
- Resistencias: 1 2k2,2 220 ,2 470, 1 240, 4 330,1 392,1 12K,3 1K5,2 2430,1 604,2 820,2 2490,2 825 ,1 2K,1 1K ,1 1K2,3 150K.

A continuación lo desglosamos según los bloques circuitales descritos:

- Circuitos del PIC:

- Alimentación y estabilización: 1 LED , 1diodo 1N4007, 1 condensador de 100nF, 1 UA7805,1 JACK,1 condensador electrolítico de 10uF, , 1 condensador de 1uF,1 resistencia de 470
- Comunicación serie: 1 zócalo DIP-16, , 1 MAX232, 1 DB9,5 condensadores electrolíticos de 10uF,4 resistencias de 330
- Reset manual: 1 Pulsador,1 LED , 1diodo 1N4007, 1 condensador de 100nF, 1 resistencia 220,1 resistencia 2K2
- Reloj: 1 cuarzo 20 MHz, 2 condensadores cerámicos de 33pF,1 resistencia de 220
- PIC: 1 zócalo DIP-40, 1 PIC16F877

➢ Circuitos del MODEM:

- Alimentación y estabilización: 1 LED , 1diodo 1N4007,7 condensadores de 100nF, , 1 LM317,1 condensador electrolítico 470uF,1 resistencia de 470,1 resistencia de 240,1 resistencia de 392
- Tensiones de referencia: 4 zócalos DIP-8, 4 LM6032,11 condensadores de 100nF, Resistencias: 2 1K5,2 2430,1 604,2 820,2 2490,2 825 ,1 2K,1 1K ,1 1K2.
- Intensidad constante: 1 LM334,1 resistencia de 12K,1 resistencia de 1K5
- Mallado: 27 pines de zócalo
- Filtros externos:3 resistencias de 150K, 3 condensadores de 22nF,3 de 2n2F
- Reloj: 1 cuarzo 6 MHz,2 condensadores cerámicos de 33pF
- Jumpers:2 jumpers y 6 pines externos
- MODEM: 1 zócalo PGA-68,1 MODEM

3.8.2 Placa segunda , versión corregida

Listado global de componentes utilizados :

- Zócalos: 1 PGA-68 , 1 DIP-40 , 1 DIP-16 y 4 DIP-8
- Pines: 30 externos y 5 zócalos.
- Chips:1 PIC16F877, 4 LM6032 , 1 MAX232,1 MODEM , 1 LM317 , 1 LM334 y 1 UA7805
- Diodos: 2 LEDS , 3 1N4007
- 2 Jumpers y 1 Pulsador
- Conectores: 1 DB9 y un JACK
- Relojes: 1 6 MHz y 1 20 MHz
- Condensadores: 20 100nF , 4 cerámicos 33pF , 1 1uF , 7 electrolíticos 10uF, 1 electrolítico 470uF,3 22nF y 3 2,2nF
- Resistencias: 2 220 ,4 470, 1 240, 4 330,1 392,1 12K,3 1K5,1 604,2 909 ,1 2K,1 825,2 475,2 1K ,1 1K2,3 150K.

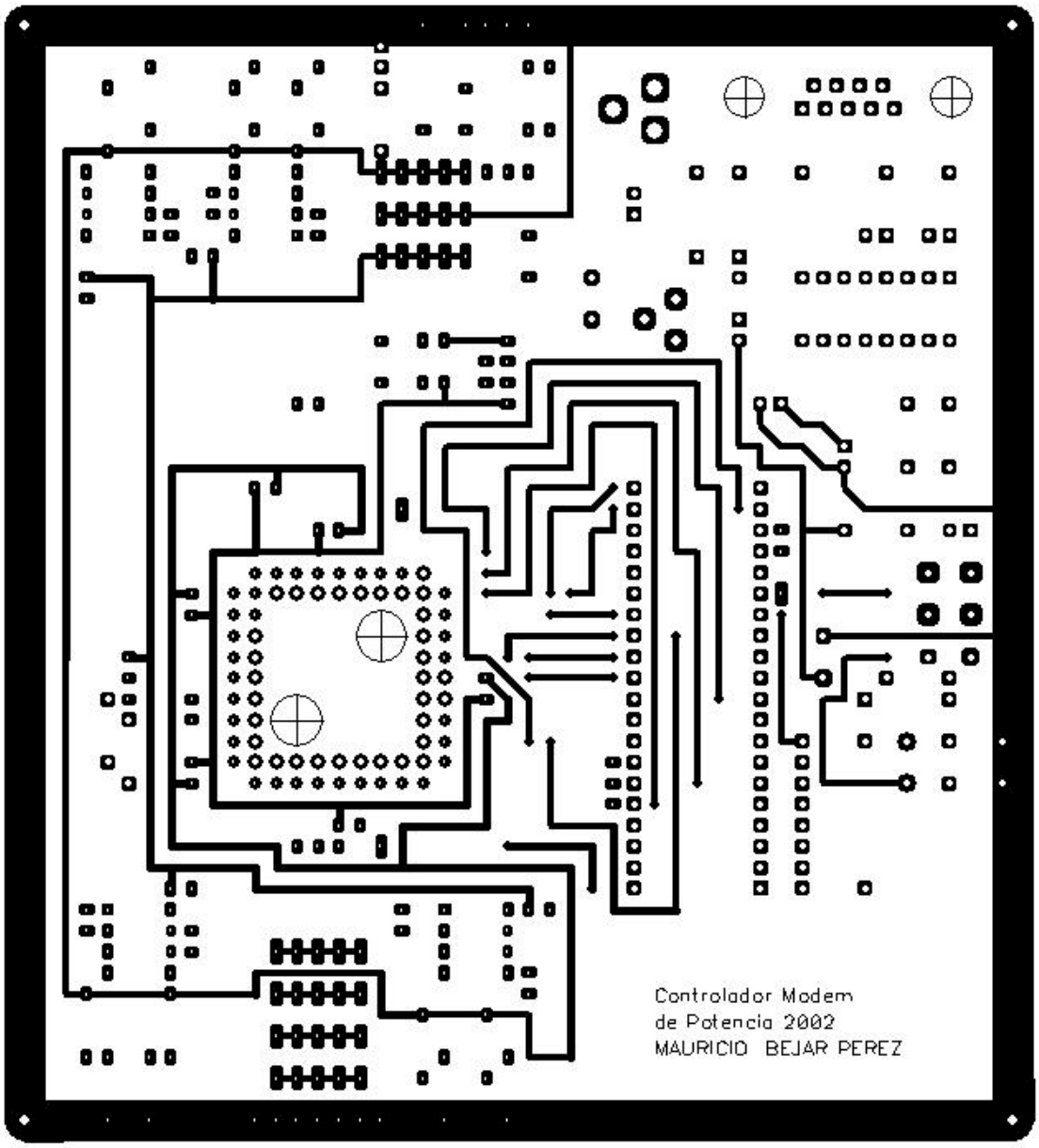
Desglose de los componentes según los bloques circuitales:

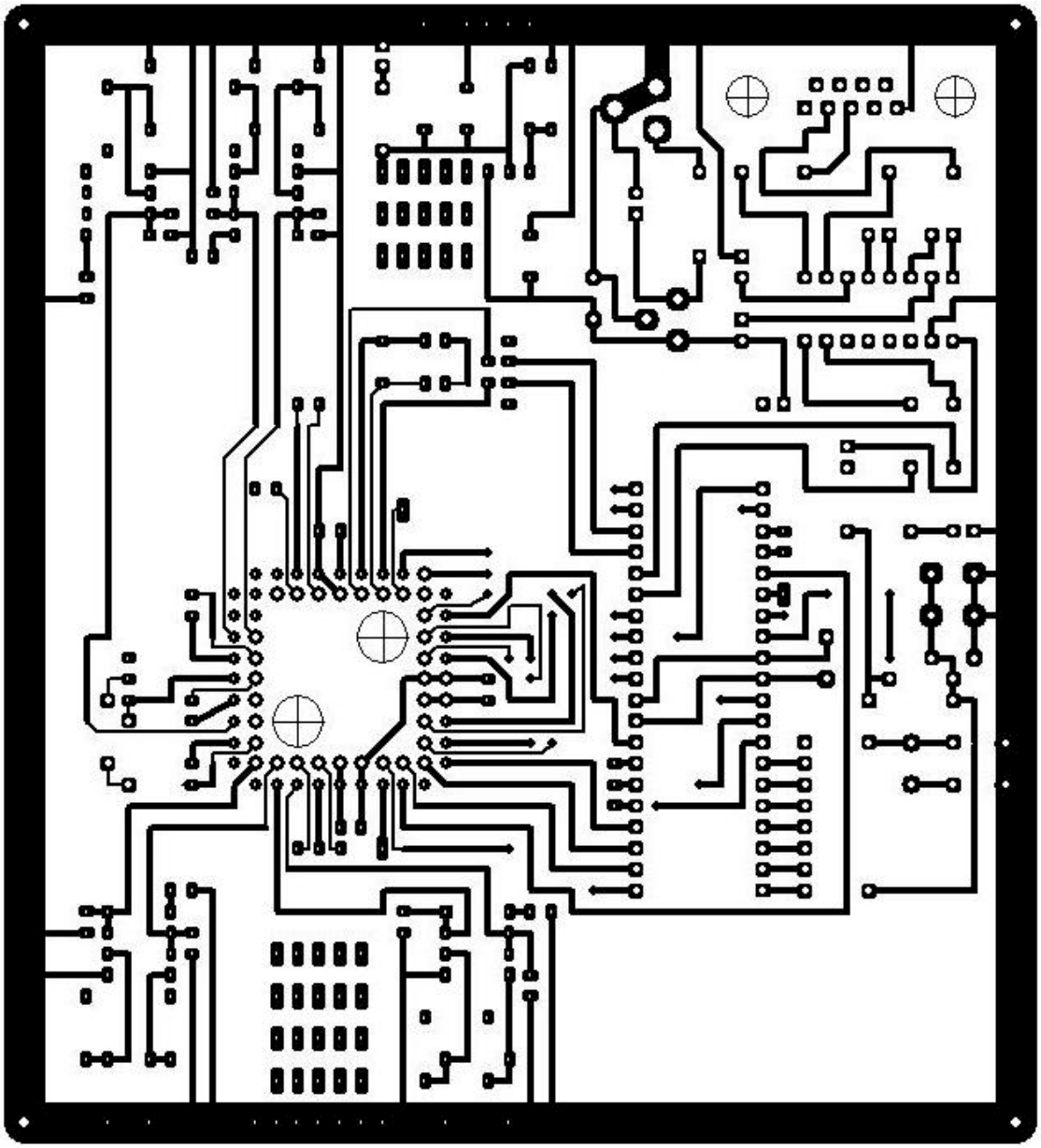
➢ Circuitos del PIC:

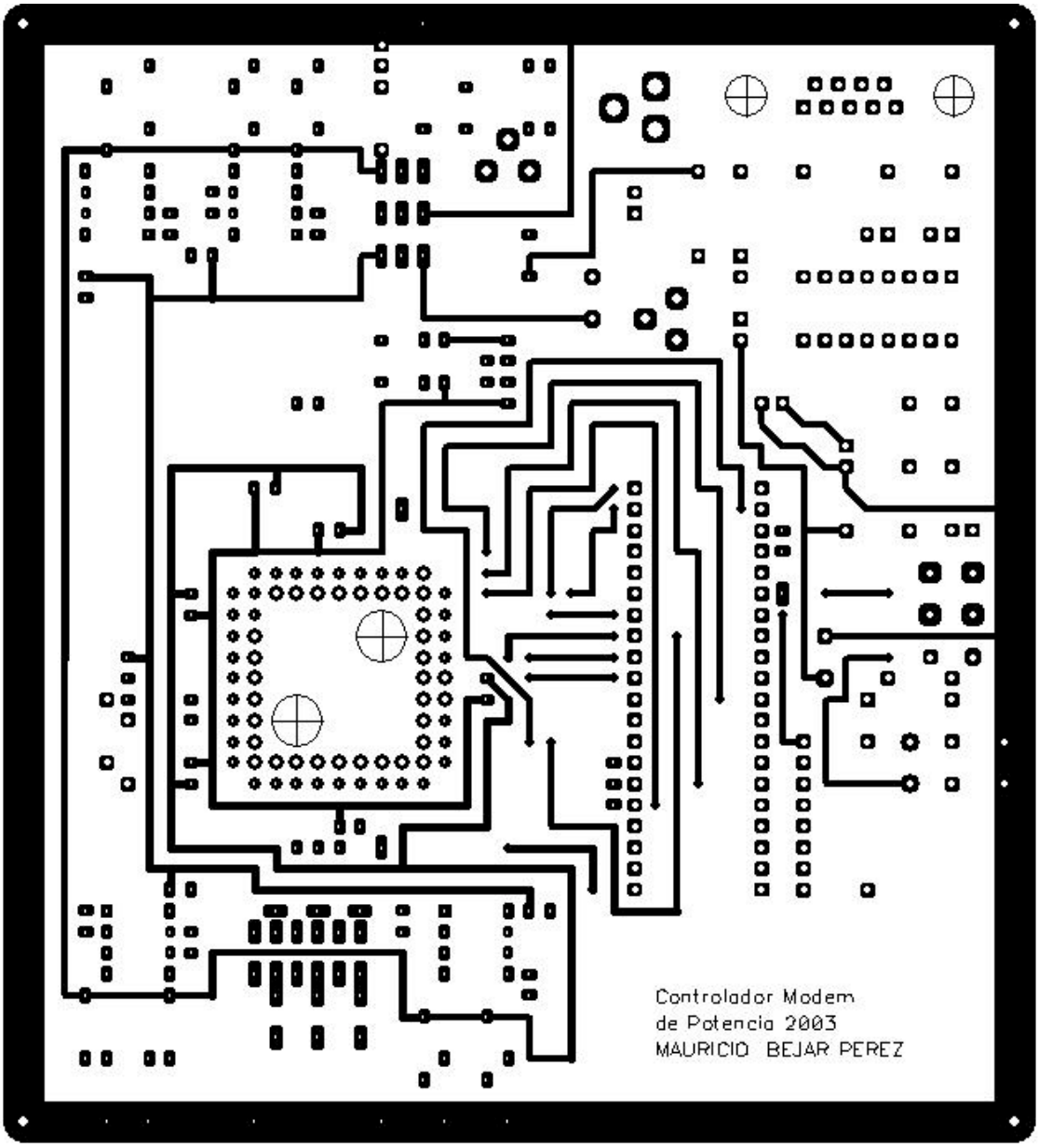
- Alimentación y estabilización: 1 LED , 1diodo 1N4007, 1 condensador de 100nF, 1 UA7805,1 JACK,1 condensador electrolítico de 10uF, , 1 condensador de 1uF,1 resistencia de 470
- Comunicación serie: 1 zócalo DIP-16, , 1 MAX232, 1 DB9,5 condensadores electrolíticos de 10uF,4 resistencias de 330
- Reset manual: 1 Pulsador,1 LED , 1diodo 1N4007, 1 condensador de 100nF, 1 resistencia 220,1 resistencia 2K2
- Reloj: 1 cuarzo 20 MHz, 2 condensadores cerámicos de 33pF,1 resistencia de 220
- PIC: 1 zócalo DIP-40, 1 PIC16F877

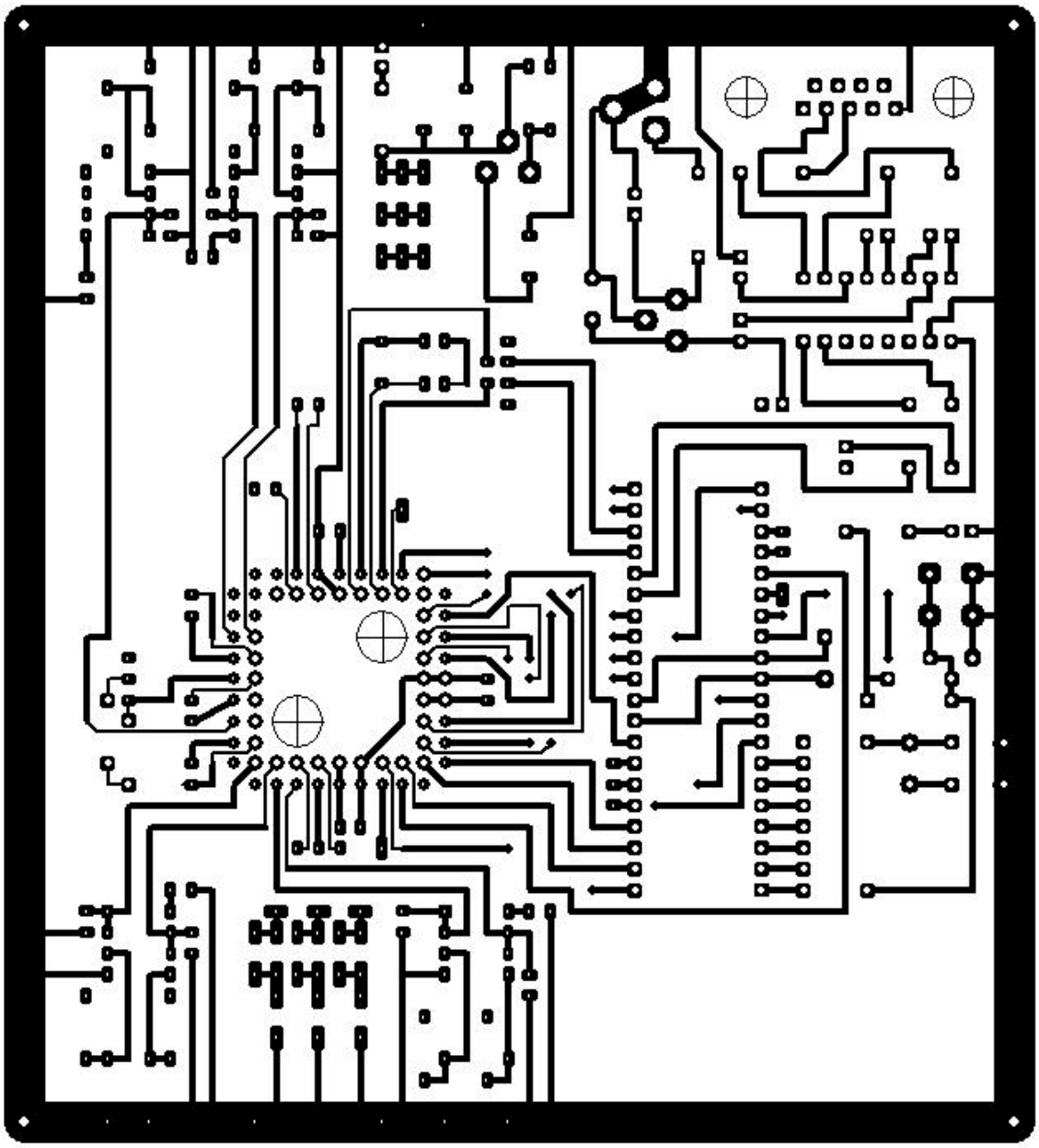
➢ Circuitos del MODEM:

- Alimentación y estabilización: 1 LED , 1diodo 1N4007,7 condensadores de 100nF, , 1 LM317,1 condensador electrolítico 470uF,1 resistencia de 470,1 resistencia de 240,1 resistencia de 392
- Tensiones de referencia: 4 zócalos DIP-8, 4 LM6032,11 condensadores de 100nF, Resistencias: 2 470, 2 1K5,1 604,2 909 ,1 2K,1 825,2 475,2 1K ,1 1K2
- Intensidad constante: 1 LM334,1 resistencia de 12K,1 resistencia de 1K5
- Filtros externos:3 resistencias de 150K, 3 condensadores de 22nF,3 de 2n2F
- Reloj: 1 cuarzo 6 MHz,2 condensadores cerámicos de 33pF
- Jumpers:2 jumpers y 6 pines externos
- MODEM: 1 zocalo PGA-68,1 MODEM


Controlador Modem
de Potencia 2002
MAURICIO BEJAR PEREZ




Controlador Modem
de Potencia 2003
MAURICIO BEJAR PEREZ



4. PROGRAMACIÓN DEL MICRO

En este capitulo veremos con detalle los programas del micro para implementar un controlador programable por el puerto serie, es decir funcionara como un pequeño procesador de instrucciones que le serán suministradas por el puerto serie.
Se crearon dos versiones del programa (en esencia es solo uno); el normal y el modificado, o 2º modo de programación, que nos facilita ciertos aspectos a la hora de llevar a cabo transmisiones a ráfagas y recepciones con el MODEM.

4.1 Funciones implementadas

El microcontrolador dispondrá de un programa (2 versiones del mismo) que implementara ciertas funciones que queremos realizar con él.
Las funciones están clasificadas según el grupo al que pertenecen, es decir tienen distintas características, por lo que las dividimos en varios grupos:

Grupo	básico	avanzado	configurable	Automático
descripción	Gobierno básico del MODEM	Ayuda en la pruebas de el MODEM	Necesita parámetros	No necesita parámetros

Cada función pertenece a dos de esos grupos no pudiendo darse las combinaciones de grupos que son mutuamente excluyentes, como son los grupos básico y avanzado entre si, y los configurable y automático entre si.

Estas son las funciones:

Función implementada	Instrucciones utilizadas	Nº instrucciones	Grupo al que pertenecen Las instrucciones
Lectura MODEM	1	1	Básica-configurable
Escritura MODEM	2	1	Básica-configurable
Lectura continua de registro	1	1	Avanzada-configurable
Captura de canal 0	3,5	2	Básica-automática
Captura de canal 1	4,5	2	Básica-automática
Habilitar MODEM	7	1	Básica-automática
Modo Test.	6,8	2	Avanzada-automática
Modo test-esc	15,8	2	Avanzada-automática
Frecuencias * Transmisión	12	1	Básica-configurable
Frecuencia 0 externa	9	1	Avanzada-configurable
Frecuencia 0 * interna	9	1	Básica-configurable

Frecuencia 1 externa	10	1	Avanzada-configurable
Frecuencia 1 * interna	10	1	Básica-configurable
Frecuencia P externa	11	1	Avanzada-configurable
Frecuencia P * Interna	11	1	Básica-configurable
Reset MODEM	13,16	2	Básica-automática
Fin Tx MODEM	14	1	Básica-automática
Escribe cola Tx **	2	1	Básica-configurable

*Funciones surgidas de la implementación de las externas mediante instrucciones.
** Solo en el 2º modo de programación.

Procedemos a explicar en detalle cada función:

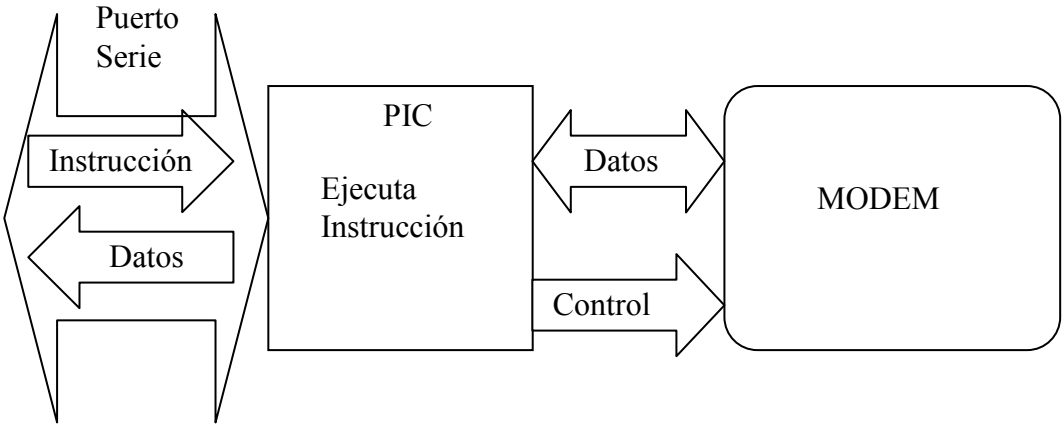
- Lectura MODEM: Lectura de un registro del MODEM cuya dirección conocemos siguiendo el protocolo de lectura del MODEM.
- Escritura MODEM: Escritura en un registro del MODEM, cuya dirección conocemos, del valor que queramos introducir en el, cumpliendo con el protocolo de escritura del MODEM.
- Lectura continúa de registro: Volcado de un registro cuya dirección conocemos sobre el puerto D del PIC para su seguimiento mediante un osciloscopio, debe poder activarse y desactivarse a voluntad.
- Captura de canal 0: Lectura continuada del registro del MODEM en el que se almacenan los datos recibidos por este correspondientes a su canal 0 de recepción, el registro será leído cada vez que se reciba un dato de forma automática .Debe poder desactivarse a voluntad.
- Captura de canal 1: Lectura continuada del registro del MODEM en el que se almacenan los datos recibidos por este correspondientes a su canal 1 de recepción , el registro será leído cada vez que se reciba un dato de forma automática .Debe poder desactivarse a voluntad.
- Habilitar MODEM: Activación de la señal de habilitación del dispositivo, para su funcionamiento.
- Modo Test: Activación del modo Test , debe poder desactivarse a voluntad.
- Modo Test-esc: Activación del modo Test-esc , debe poder desactivarse a voluntad.
- Frecuencias Transmisión : Configura las frecuencias de transmisión (las 3) del MODEM , surge de la implementación con instrucciones de las frecuencias externas , se puede hacer a base de otras funciones (3 escrituras o las 3 frecuencias internas) , pero ahorra tiempo.
- Frecuencia 0 externa: Configura la frecuencia del canal 0 de transmisión y hace que el PIC genere una frecuencia externa de PLL0 acorde a esta .Sirve para la configuración cuando usamos el MODEM en “bucle cerrado”(pruebas con si mismo), o en “bucle abierto”(dos MODEM configurados a las mismas frecuencias de Transmisión y Recepción).
- Frecuencia 0 interna: Configura la frecuencia del canal 0 de transmisión .Sale como consecuencia de la implementación de la externa.
- Frecuencia 1 externa: Configura la frecuencia del canal 1 de transmisión y hace que el PIC genere una frecuencia externa de PLL1 acorde a esta .Sirve para la configuración cuando

usamos el MODEM en “bucle cerrado”(pruebas con si mismo), o en “bucle abierto”(dos MODEM configurados a las mismas frecuencias de Transmisión y Recepción).

- Frecuencia 1 interna: Configura la frecuencia del canal 1 de transmisión .Sale como consecuencia de la implementación de la externa.
- Frecuencia P externa: Configura la frecuencia piloto de transmisión y hace que el PIC genere una frecuencia externa de PLLp acorde a esta .Sirve para la configuración cuando usamos el MODEM en “bucle cerrado”(pruebas con si mismo), o en “bucle abierto”(dos MODEM configurados a las mismas frecuencias de Transmisión y Recepción).
- Frecuencia P interna: Configura la frecuencia piloto de transmisión .Sale como consecuencia de la implementación de la externa.
- Reset MODEM: Resetea el MODEM durante el tiempo que queramos.
- Fin Tx MODEM : Desactiva el transmisor del MODEM .La usamos cuando estamos transmitiendo a ráfagas.
- Escribe cola Tx : Escribe el dato en la cola de transmisión para almacenar los datos a transmitir en la ráfaga. Esta función solo aparece en el 2º modo de programación , ya que en este implementamos la cola de Tx , mientras que el otro no.

4.2 Implementación mediante programación

Pretendemos crear un pequeño procesador de instrucciones en el microcontrolador de manera que ,al leer del puerto serie un byte ,ejecute cierta instrucción permitiendo el control y comunicación con el MODEM , de esta manera el controlador en si(PIC) será transparente al usuario que mande instrucciones por el puerto serie , viendo el MODEM directamente .
Como se puede ver en el esquema :



4.2.1 Juego de instrucciones

Utilizaremos una serie de instrucciones implementadas en el programa de micro de manera que con ellas podamos llevar a cabo las funciones anteriormente descritas , al saber el micro lo que debe hacer al recibir una instrucción podemos gobernar las acciones del controlador suministrándole instrucciones por el puerto serie como antes se ha visto.
.

Las instrucciones se dividen en dos sets debido a que tienen distintas longitudes de identificador de instrucción , el Set 1 tiene un identificador de 1 byte mientras que el 2 tiene uno de 2 bytes . La estructura de una instrucción es la siguiente:

Identificador de instrucción(1-2 bytes)	Parámetros (0-3 bytes)

Por lo que su longitud varia entre 1 y 5 bytes .no obstante no es posible confundirlas entre si , comenzada a recibirse una por el puerto serie se identifica al primer , o como mucho al 2º byte y se sabe con certeza(esta dentro del programa)el número de bytes que se deben recibir para acabar el ciclo de instrucción y comenzar con otra.

A la hora de codificar las instrucciones , es decir asignar un elemento del código a cada una , utilizamos el siguiente método:

- No utilizamos los bytes completos , la longitud es variable permitiendo una identificación más rápida ya que el código no es anidado .
- Las instrucciones se identifican unívocamente con un elemento del código.
- Cada elemento del código se distingue de otro por el lugar que ocupa en el byte el ultimo 1 empezando por la derecha(de ahí hacia la derecha todo serán unos) , en el caso de 2 bytes nos referimos al segundo de ellos al ser el primero común a ellos(aunque el método de identificación es el mismo para el primero).
- Los lugares van del 0(no hay unos) al 8 (8 unos).

Así pues podemos ver la estructura de un elemento del código dentro de un byte:

X	x	0	1	1	1	1	1

Lugar 8º Lugar 1º

Solo utilizaríamos 6 bits para identificar la instrucción (las posiciones marcadas con x no importa el valor que tengan), que se leería desde la derecha hacia la izquierda quedando el ejemplo : 111110 es el elemento del código, al tener que llegar al byte se llenarían de ceros los bits marcados con x.

Debido a la depuración del código y de las funciones a implementar con las instrucciones nos queda libre un elemento del código disponible para ser utilizado en posteriores reprogramaciones :

X	0	1	1	1	1	1	1

Primer byte

X	0	1	1	1	1	1	1

2º byte

Los elementos del código y su asociación a las instrucciones quedan entonces así:

Nº de instrucción	Código byte 1	Código byte 2	Valor real al

			completar los bytes (Hex)
1	0		00
2	10		01
3	110		03
4	1110		07
5	11110		0F
6	111110		1F
7	11111110		7F
8	11111111		FF
9	1111110	0	3F00
10	1111110	10	3F01
11	1111110	110	3F03
12	1111110	1110	3F07
13	1111110	11110	3F0F
14	1111110	111110	3F1F
15	1111110	11111110	3F7F
16	1111110	11111111	3FFF

El juego de instrucciones es el siguiente:

instrucción	Nº	Nº bytes	Byte 1 (Hex)	Byte 2 (Hex)	Parámetros(bytes según orden)	Set instrucciones	Modificadas en 2º modo programación
Lectura	1	3	00		Dir,modo	Set 1	
Escritura	2	3(4*)	01		Dir,dato,(modo)*	Set 1	*para escribir en cola Tx
Lect. Reg 0	3	1	03			Set 1	
Lect. Reg 1	4	1	07			Set 1	
Fin Lect. Reg	5	1	0F			Set 1	
Habilitar Test	6	1	1F			Set 1	
Habilitar Modem	7	1	7F			Set 1	
Fin Test	8	1	FF			Set 1	
Frecuencia 0	9	5	3F	00	Frecmod,frecpic,modo	Set 2	
Frecuencia 1	10	5	3F	01	Frecmod,frecpic,modo	Set 2	
Frecuencia P	11	5	3F	03	Frecmod,frecpic,modo	Set 2	
Frecuencia total	12	5	3F	07	Frecmod0,Frecmod1, FrecmodP	Set 2	
Reset MODEM	13	2	3F	0F		Set 2	
Fin Tx MODEM	14	2	3F	1F		Set 2	
Habilitar Test esc	15	2	3F	7F		Set 2	
Fin reset	16	2	3F	FF		Set 2	

* Solo en el 2º modo de programación

Descripción de las instrucciones :

- Instrucciones de habilitación o desabilitación(No tienen parámetros):
  - Habilitar Test : Activa la señal de habilitación del Modo test para el MODEM(a nivel alto).
  - Habilitar MODEM: Activa el habilitador del MODEM (la señal a nivel bajo)para que este funcione.
  - Fin test : Desactiva las señales de modo Test (las dos).
  - Reset MODEM : Activa la señal de reset (la pone a nivel bajo) , dejando al MODEM en estado de reset.
  - Fin Tx MODEM : Escribe en el registro de estado del MODEM deshabilitando la transmisión.
  - Habilitar Test-esc : Activa la señal de habilitación del Modo test-esc para el MODEM(a nivel alto).
  - Fin reset: desactiva la señal de reset(la pone a nivel alto) sacando al MODEM del estado de reset.
  - Lect. Reg 0 : Activa las interrupciones de recepción del canal 0 , de manera que si se están recibiendo datos por ese canal , estos son mandados por el puerto serie cada vez que se produzca una interrupción de dato nuevo.Es es excluyente con respecto a Lect. Reg 1.
  - Lect. Reg 1 : Activa las interrupciones de recepción del canal 1 , de manera que si se están recibiendo datos por ese canal , estos son mandados por el puerto serie cada vez que se produzca una interrupción de dato nuevo. Es excluyente con respecto a Lect. Reg 0.
  - Fin Lect. Reg : Desactiva las interrupciones para los dos canales , de manera que se corta el flujo de datos(si lo hubiera) hacia el Puerto serie.

- Lectura : Aunque es solo una instrucción , lleva a cabo diversas funciones dada su implementación en el código , al disponer de parámetros que identifiquen una función de otra , y al estar estas relacionadas.Necesita dos parámetros ,Dirección y Modo , si bien a veces solo utiliza uno(Modo).ver en la siguiente tabla los valores de parámetros permitidos.

  Implementa las siguientes funciones o tipos de lectura según el parámetro Modo:

Tipo Lectura	Dirección(Hex) (1 byte)	Modo (1 byte)	Comentarios
Simple	00-82	00	Lectura normal de un registro
Continua habilitada	00-82	01	Salida del registro en puerto D , bloqueada la lectura y escritura
Continua deshabilitada	No importa	03	Desbloquea lectura y escritura

- Lectura simple : lectura del registro cuya dirección esta en el parámetro del mismo nombre , el dato se almacena en un registro del PIC y posteriormente se manda por el puerto serie.
- Lectura continua habilitada : volcado del registro indicado por el parámetro dirección sobre el puerto D del PIC , esto se debe a realizar un ciclo de lectura del MODEM a medias , dejando a este pendiente del fin del protocolo.Mientras se esta en este modo no se puede ni leer ni escribir en el MODEM hasta que no se termine el ciclo de lectura.
- Lectura continua deshabilitada : Se trata de salir del modo volcado en puerto D mediante la finalización del ciclo de lectura que dejo el puerto Dio del MODEM bloqueado , terminando así con el protocolo se vuelve al estado normal.

- Escritura: En el modo básico de programación(o primer modo) no existe el parámetro Modo , y solo se trata de la escritura en un registro(cuya dirección se pasa como parámetro) del dato pasado (parámetro).
  En el 2º modo de programación se añade el parámetro modo , ya que se introduce una modificación para que se pueda escribir de manera normal o en la cola de transmisión , quedando así(ver valores permitidos de los parámetros en la tabla):

Tipo Escritura	Dirección(Hex) (1 byte)	Datos (1 byte)	Modo* (1 byte)	Comentarios
Simple	00-82	00-FF	00	Escritura normal de un registro
A cola *	No importa	00-FF	01	Mete el dato en la cola Tx

*Solo en el 2º modo de programación

Los modos de escritura por tanto son:

- Escritura simple : es la única que existe en el primer modo de programación, en ella se escribe un dato(parámetro) en un registro(cuya dirección es un parámetro dado) del MODEM usando para ello el protocolo de escritura del MODEM.
- Escritura a cola : escritura en una cola , por lo que simplemente se guarda el valor proporcionado(parámetro) en la primera posición libre de la cola(después de todos los anteriores datos) , sin usarse el parámetro de dirección , que carece en este caso de valor.

- Frecuencias: Hay varias instrucciones referentes a la frecuencia , algunas solo introducen en un registro especifico un valor(escriben el valor apropiado en el registro de frecuencia del transmisor) y otras además configuran y activan algún periférico o rutina para generar una señal a cierta frecuencia dada.

Así pues las instrucciones y sus parámetros(con sus valores permitidos) son:

Tipo frecuencia	Frecuencia MODEM (1 byte) (Hex)			Frecuencia PIC(1byte)(Hex)	Modo (1 byte)	Comentarios
Canal 0	01-FB			No importa	00	Solo escribe en el MODEM
Canal 0	01-1E			EA-F7	01	PIC genera señal pll0
Canal 1	01-FB			No importa	00	Solo escribe en el MODEM
Canal 1	01-FB			07-3D	01	PIC genera señal pll1
P	01-FB			No importa	00	Solo escribe en el MODEM
P	01-17			E9-F4	01	PIC genera señal pllP
Todas	01-FB	01-FB	01-FB			Solo escribe en el MODEM

- Frecuencia total : Escribe en los registros de frecuencia del transmisor los valores(pasados como parámetros) para los canales 0,1 y la frecuencia piloto.

Las demás instrucciones de frecuencia dependen del parámetro Modo(ver tabla de arriba) , así estarán las externas y las internas:

- Frecuencia canal 0 interna : Escribe en el registro de frecuencia del transmisor del canal 0 el valor dado por el primer parámetro , no importando el valor de su segundo parámetro(que estaría destinado para el PIC).
- Frecuencia canal 0 externa: Escribe en el registro de frecuencia del transmisor del canal 0 el valor dado por el primer parámetro , y además con el valor del segundo parámetro configura y activa(genera por interrupciones) la señal PLL0 del PIC al MODEM.
- Frecuencia canal 1 interna : Escribe en el registro de frecuencia del transmisor del canal 1 el valor dado por el primer parámetro , no importando el valor de su segundo parámetro(que estaría destinado para el PIC).
- Frecuencia canal 1 externa: Escribe en el registro de frecuencia del transmisor del canal 1 el valor dado por el primer parámetro , y además con el valor del segundo parámetro configura y activa(mediante el PWM) la señal PLL1 del PIC al MODEM.

- Frecuencia P interna : Escribe en el registro de frecuencia piloto del transmisor el valor dado por el primer parámetro , no importando el valor de su segundo parámetro(que estaría destinado para el PIC).
- Frecuencia P externa: Escribe en el registro de frecuencia piloto del transmisor el valor dado por el primer parámetro , y además con el valor del segundo parámetro configura y activa(genera por interrupciones) la señal PLLp del PIC al MODEM.

4.2.2 Mapa de memoria

En el programa del PIC utilizaremos variables ubicadas en registros de su memoria , debido a la disposición de sus registros internos(mapa de memoria) utilizaremos la zona de memoria formada por registros que no tienen asignada ninguna función para el PIC de su primer banco de memoria que , al ser el más utilizado por el programa ,es el que usaremos para ubicar las variables del programa.

Este es el mapa de memoria del programa normal (o 1ª programación) :

Dirección (Hex)	Registro
20	
21	DATMOD
22	ADDRESS
23	BANCO
24	FLAG
25-26	
27	FRECUEN
28	
29	COPIA
2A	PORTM
2B	DATMOD2
2C	DATMOD3
2D	PORTN
2E	
2F	LER
30	TEMPO
31-7F	

Las posiciones en blanco indican que esos registros están disponibles para posteriores variables si hiciera falta modificar el programa.

Y este el del 2º modo de programación(programa ampliado) :

Dirección (Hex)	Registro
20-3F	Cola Tx
40-5F	Cola Rx
60	TEMPO
61	DATMOD
62	ADDRESS
63	TTOP
64	TBOT
65	RTOP
66	RBOT
67	DATPIL
68-69	
6A	CONTRX
6B	CONTTX
6C	COMPW
6D-6E	
6F	LER
70-72	
73	BANCO
74	FLAG
75-78	
79	COPIA
7A	PORTM
7B	DATMOD2
7C	DATMOD3
7D	PORTN
7E-7F	

4.2.3 Descripción de los registros del programa

Registros comunes a los dos programas:

- FLAG ; Registro de banderas de control:

C				V	L	R	

MSB LSB

Banderas de control(todas a 0 después de un reset manual):

- C :
  - Si es igual a 1 la ultima comparación del registro COMPW con 0 es igual a 0.Solo se usa en el 2º modo de programación.
- V :
  - Si es igual a 1 se ha acaba de desconectar el Rx y hay datos para vaciar de la cola de Rx , que deben mandarse al PC por el PS. Solo se usa en el 2º modo de programación.
- L :
  - Si es igual a 1 están activadas las interrupciones por Rx en el MODEM para el canal 0 o el 1, esta activo el modo de adquisición continuo de datos Rx por el MODEM.
- R :
  - Si es igual a 1 hay dato leído del MODEM en DATMOD.

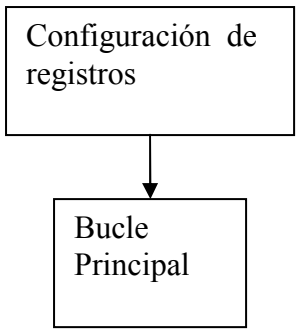
- DATMOD: Registro de almacenamiento de datos procedentes(leídos) del MODEM o para escribir en el MODEM, al escribir en el MODEM y al mandar el dato por el puerto serie el registro queda libre , al capturar los valores leídos o a escribir queda ocupado .De manera que al completarse de manera independiente los ciclos de lectura y escritura no se solapan nunca los valores.
- ADDRESS : Registro donde se almacena la dirección del registro a acceder , ya sea en lectura o en escritura.
- LER: Registro de "instrucción" del microcomputador implementado , es decir ; es donde se almacena el byte de identificación de instrucciones según el cual se ejecuta una instrucción u otra(en el caso de instrucciones del Set 2 se almacena 2 veces consecutivas realizando el mismo proceso).
- BANCO: Registro donde se guarda el valor del registro STATUS cuando se accede a la rutina de interrupción , ya que al operar esta en un banco determinado(el 0) y al cambiarse de banco durante el ciclo de lectura(a banco 1) , puede que se acceda a registros equivocados al tener otro banco seleccionado.
- COPIA: Registro donde se almacena el valor del registro W cuando se ejecuta la rutina de interrupción.
- FRECUEN: Registro que almacena el valor de configuración para el PWM del PIC ,es el valor del periodo de este para generar la señal PLL1.
- PORTN: Registro que almacena el valor de configuración para el Timer 0 , siendo el número que debe tener el timer para que a partir de él y al llegar a desbordarse(interrupción) tenga una duración de la mitad del periodo de la señal a generar (menos el ajuste que haga falta por la rutina de interrupción).
- PORTM: Registro que almacena el valor de configuración para el Timer 1 , siendo el número que debe tener el segundo byte del timer para que a partir de él y al llegar a desbordarse(interrupción) tenga una duración de la mitad del periodo de la señal a generar(menos el ajuste que haga falta por la rutina de interrupción) .
- TEMPO:Registro que almacena el valor de al dirección del canal a leer de manera continua (al llegar interrupciones de recepción).
- DATMOD2:Registro que almacena el valor de configuración de frecuencia para el transmisor del MODEM por el canal1.
- DATMOD3: Registro que almacena el valor de configuración de la frecuencia piloto para el transmisor del MODEM .

Registros propios del 2º modo de programación:

- Cola Tx: 32 registros seguidos donde almacenamos los datos a transmitir en una ráfaga por el MODEM.
- Cola Rx:32 registros seguidos donde almacenamos datos recibidos por el MODEM.
- COMPW:Registro donde almacenamos el valor del registro a comparar con 0.
- DATPIL: Registro temporal de almacenaje del dato a introducir o a sacar de una de las colas.
- CONTTX: Registro contador de datos almacenados en la cola de transmisión.
- CONTRX: Registro contador de datos almacenados en la cola de recepción.
- TTOP: Registro de la dirección de la primera posición libre de la cola de transmisión donde meter el dato.
- TBOT: Registro de la dirección de la posición del primer dato a sacar de la cola de transmisión.
- RTOP: Registro de la dirección de la primera posición libre de la cola de recepción donde meter el dato.
- RBOT: Registro de la dirección de la posición del primer dato a sacar de la cola de recepción.

4.2.4 Programa

El programa se compone(en los dos modos de programación) de una configuración previa de registros del PIC y un bucle principal que se ejecuta continuamente como se puede ver en el esquema siguiente.
La configuración inicial hace que el programa funcione correctamente desde la primera ejecución del bucle principal, configurando los periféricos , puertos e interrupciones pertinentes así como los registros que hemos descrito anteriormente que resultan básicos para el buen funcionamiento del programa.
El bucle inicial lleva a cabo las instrucciones que se le suministren por el puerto serie al PIC.



Además existen ciertas funciones (básicas y avanzadas , según el modo de programación) y una rutina de interrupción que trata las interrupciones según las utilizamos en el programa.

4.2.5 Rutina de interrupción

El uso de interrupciones en el programa generadas por el PIC se reduce a gestionar la generación de las señales externas de PLL que queremos proporcionar al MODEM mediante el Timer 0 y el Timer 1 .Las interrupciones se generan al desbordarse el Timer 0 o el 1 , teniendo que diferenciarse en la rutina de interrupción cual de las dos interrupciones ha sido la causante de la ejecución de la rutina , y proceder en cada caso a el procedimiento adecuado.
El procedimiento que utilizamos para generar señales periódicas por pines del puerto B
Es muy simple :

- Asignamos un valor (alto o bajo) al pin (paso previo de la configuración).
- Configuramos el temporizador correspondiente para la siguiente interrupción.
- En la rutina de interrupción identificamos el estado en el que esta el pin(alto o bajo).
- Invertimos el valor del pin.
- Volvemos de la interrupción(paso posterior).

Como se puede ver en el siguiente diagrama es de vital importancia salvar los registros pertinentes antes de realizar ninguna operación , en este caso salvamos el valor de W (registro propio del PIC para operaciones) puesto que no sabemos si el valor que tiene debe ser utilizado después(esta en medio de un proceso) o no, y el valor del registro de estado STATUS , puesto que nos indica en el banco de memoria del PIC en el que nos encontrábamos realizando operaciones con los registros(el que estaba seleccionado) y si utilizamos un banco inapropiado no accedemos a los registros que queremos.

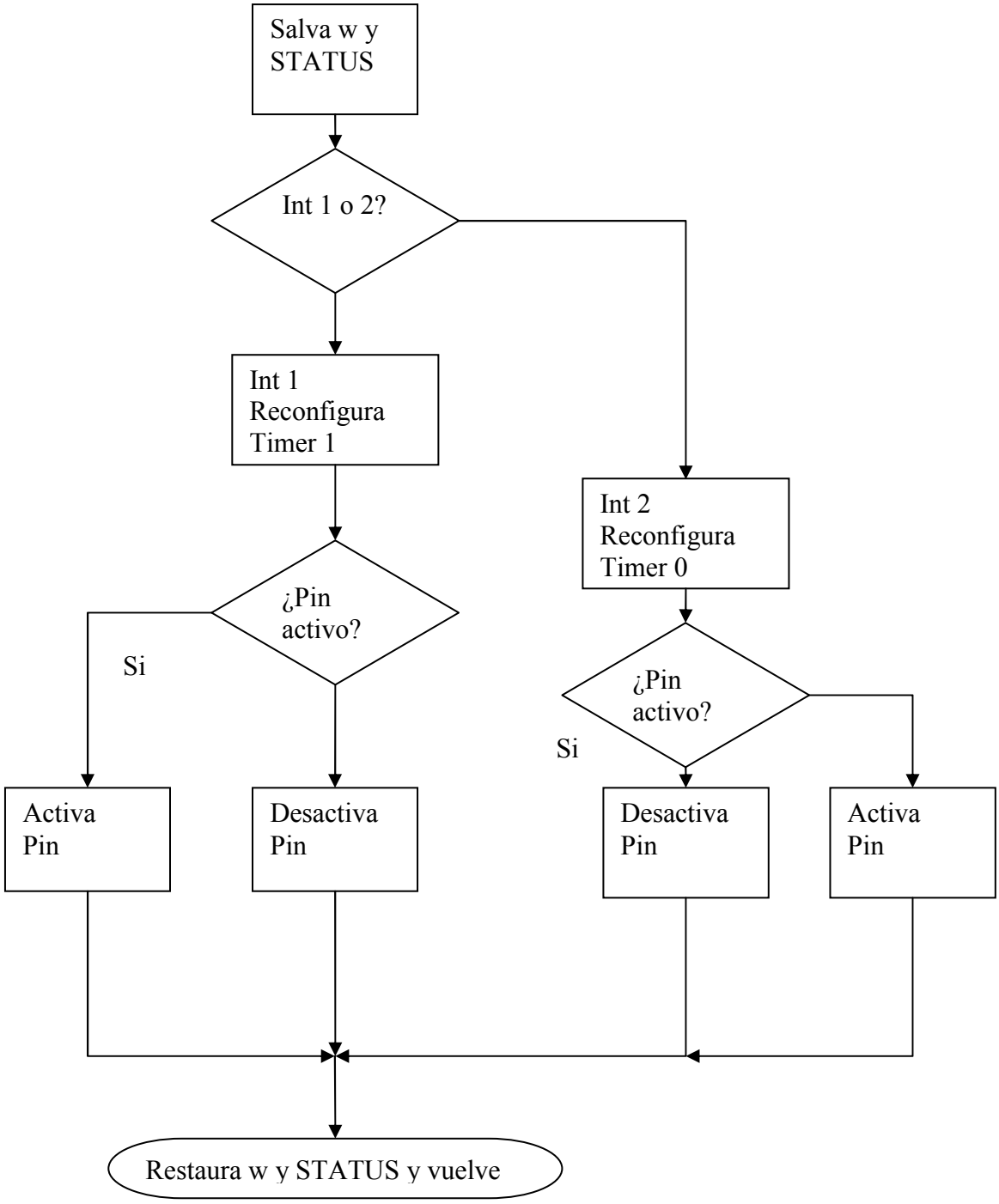


Diagrama funcional

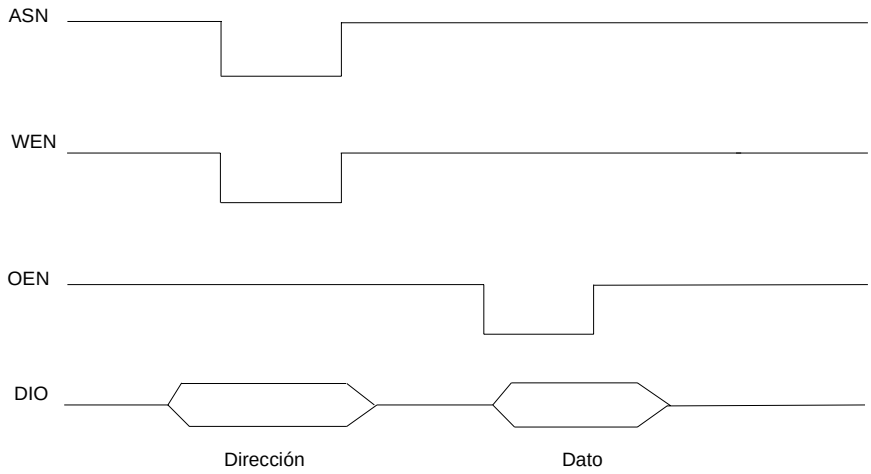
Posteriormente identificamos la causa de la interrupción (salto a la rutina) , y una vez definido el Timer sobre el que tenemos que operar llevamos a cabo el anteriormente mencionado proceso de generación de señal periódica(solo los 3 pasos de en medio).
Finalmente restauramos los valores de los registros W y STATUS y regresamos de la rutina de interrupción.

4.2.6 Funciones básicas

En los dos modos de programación utilizamos unas funciones comunes , estas son las funciones básicas del programa(ya que las dos versiones o modos de programación lo son del mismo programa).

4.2.6.1 Lect

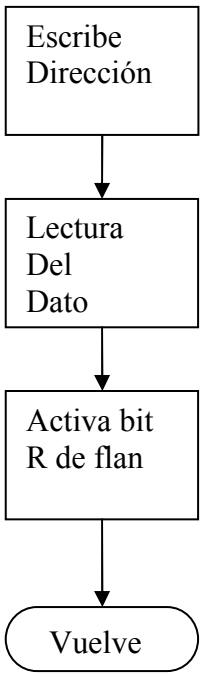
Es la función básica que implementa el ciclo de lectura de MODEM , dándonos el acceso a los registros del MODEM de una manera sencilla y eficaz .
Utiliza el registro ADDRESS como parámetro de la dirección del registro a leer(que emplearemos en el protocolo de lectura) y el DATMOD para almacenar el dato leído del MODEM , posteriormente este se enviara por el puerto serie.



Cronograma de la función Lect

Implementa el protocolo de lectura del MODEM ajustándolo al cronograma anteriormente mostrado , cumpliendo así con las especificaciones del MODEM.

Como se puede ver en el esquema siguiente el proceso que realiza esta función es lineal y bastante simple , adaptándose a las exigencias de protocolo del MODEM:

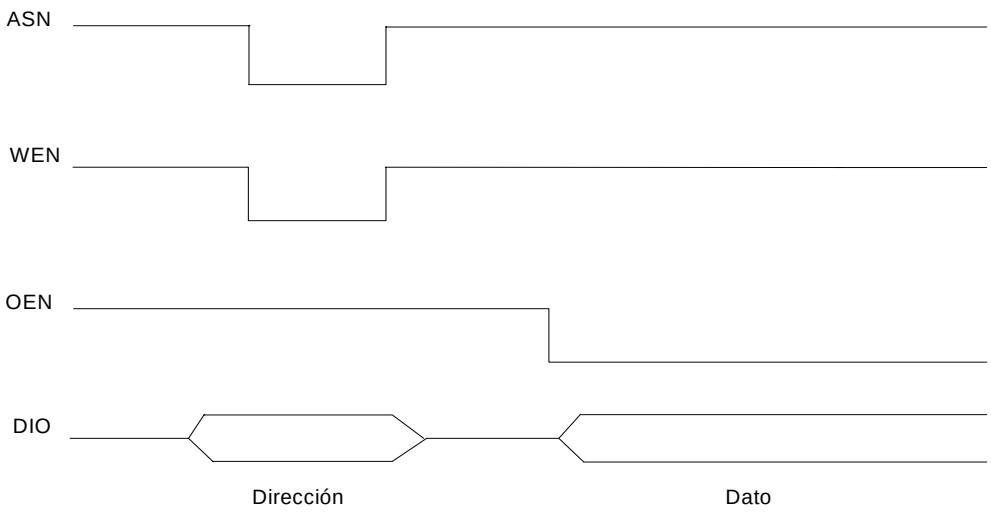


Esquema de la función Lect

Primero escribe por el puerto D la dirección contenida en ADDRESS activando las señales del puerto E (ASN y WEN a nivel bajo) y subiéndolas posteriormente para seleccionar la dirección en el Modem.Luego cambia el estado del puerto D para que sea entrada al PIC en vez de salida , pone a nivel bajo OEN (por el puerto E) durante varios ciclos de instrucción(del PIC de 200ns) capturando el valor en DATMOD y vuelve a poner el puerto D como salida.
Por ultimo activa el bit R del registro FLAG indicando que se ha realizado una lectura (debe mandarse el dato por el puerto serie) , y vuelve al bucle principal.

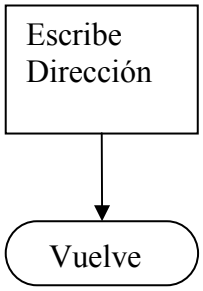
4.2.6.2 Lect cont

Esta función lleva a cabo un ciclo de lectura pero lo interrumpe a la mitad , no llegándose a terminar el protocolo del MODEM , con lo que quedan los datos de Dio (registro a leer) volcados sobre el puerto D del PIC.
Utiliza solamente el registro ADDRESS , en el que esta contenido la dirección del registro a leer , no capturándose el dato en ningún registro .
Implementa parte de el protocolo de lectura del MODEM (dejándolo a medias) ajustándolo al cronograma siguiente , cumpliendo así con las especificaciones del MODEM de lectura , pero dejando el proceso bloqueado antes de capturar el dato .



Cronograma de la función Lect cont

Como se puede observar en el siguiente esquema , la estructura de la función es lineal :



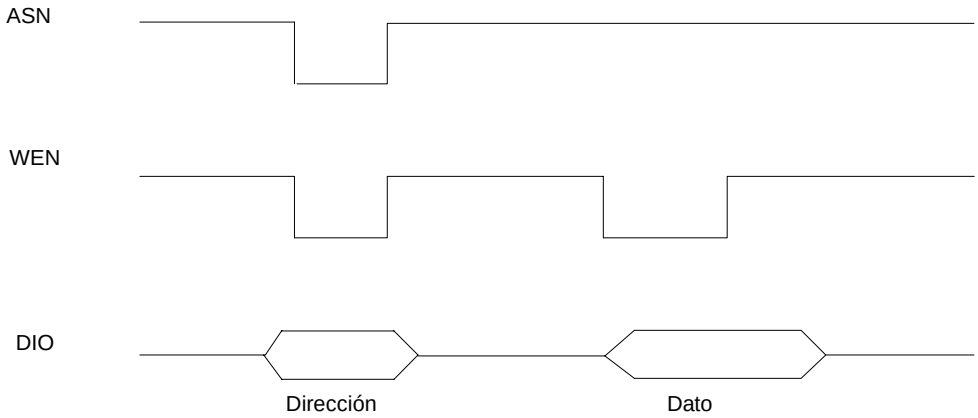
Esquema de la función Lect cont

Primero escribe por el puerto D la dirección contenida en ADDRESS activando las señales del puerto E (ASN y WEN a nivel bajo) y subiéndolas posteriormente para seleccionar la dirección en el Modem.Luego cambia el estado del puerto D para que sea entrada al PIC en vez de salida y pone a nivel bajo OEN (por el puerto E) , dejando de esta manera el dato volcado hasta que queramos desbloquear el puerto(acabando con el protocolo de lectura).Luego vuelve al bucle principal.

4.2.6.3 Escrt

Esta función básica lleva a cabo la escritura de un dato en un registro mediante el protocolo de escritura del MODEM , de esta manera podremos escribir fácilmente valores en los registros indicados del MODEM.

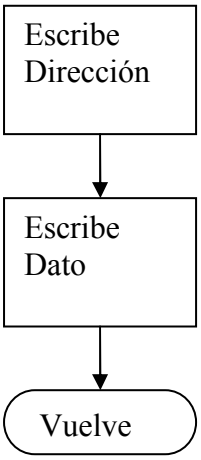
Utiliza el registro ADDRESS donde se le debe haber pasado la dirección del registro a acceder , y escribe el contenido del registro DATMOD en el registro del MODEM(por lo que se le debe haber pasado ya el valor al invocar la función



Cronograma de la función Escrt

El comportamiento de la función se describe por el cronograma anterior , que cumple con el protocolo de escritura del MODEM.

Su estructura es lineal y simple :



Esquema de la función Escrt

Primero escribe por el puerto D la dirección contenida en ADDRESS activando las señales del puerto E (ASN y WEN a nivel bajo) y subiéndolas posteriormente para seleccionar la dirección en el Modem.Luego pone el valor de DATMOD en el puerto D y activa (a nivel bajo) WEN , dejándola así durante varios ciclos de instrucción(del PIC). Posteriormente se vuelve a poner WEN a nivel alto , con lo que se produce la escritura por parte del MODEM en el registro seleccionado , y se vuelve al bucle principal.

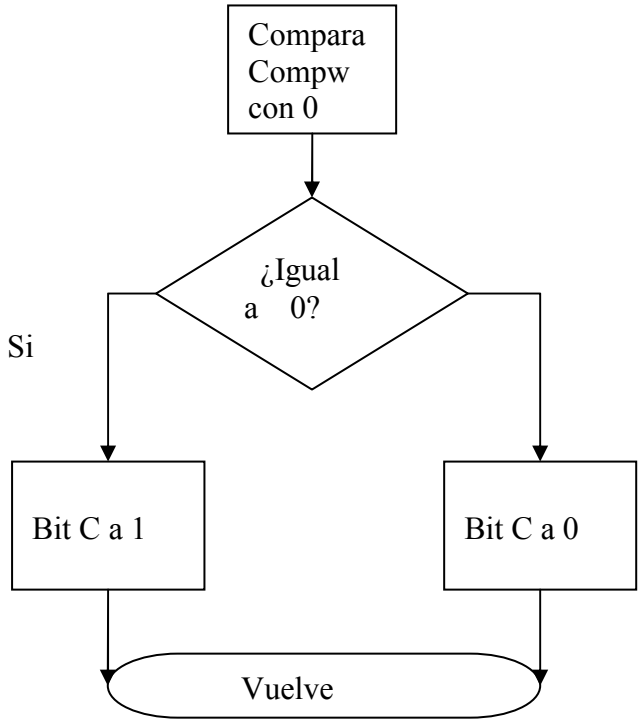
4.2.7 Funciones avanzadas

Estas son funciones que solo existen en el 2º modo de programación , se encargan de gestionar las colas (meter y sacar datos) y de alguna otra función (como es calcular la comparación de un registro con 0).

4.2.7.1 Comp0

Esta función nos calcula la comparación de un registro con cero , almacenando el resultado en un bit del registro de banderas de control (FLAG).
Utiliza el registro COMPW como almacenaje del valor del registro a comparar (debe pasársele el valor antes de llamar a la función).

Sigue el siguiente esquema de funcionamiento:



Esquema de la función Comp0

Primero compara bit a bit el registro COMPW con 0 , y dependiendo de si sean iguales o no mete un valor u otro en el bit C (bit 7) del registro FLAG:

- Si son iguales C=1.
- Si no son iguales C=0.

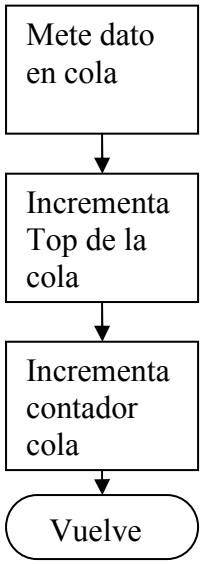
Tras lo cual vuelve al bucle principal.

4.2.7.2 MTX y MRX

Estas dos funciones tienen la misma estructura , aunque pondremos de manifiesto sus diferencias. Se trata de funciones que introducen un dato en una de las colas(la de transmisión ; MTX , y la de recepción ; MRX).Tratan las colas con direccionamiento circular mediante dos índices de posición (xTOP y xBOT) y un contador de los datos que hay dentro de la cola(CONTx).Tambien utilizan el registro DATPIL para almacenar el dato a meter en la cola.
El proceso utilizado tiene en cuenta lo siguiente :

- Usan dirección indirecta del PIC , registros FSR(valor del "puntero") y INDF("puntero" al que se hace referencia en las operaciones).
- Operamos sobre FSR para variar la posición del "puntero".
- xTOP indica la dirección del primer puesto vació en la cola(a introducir el dato).
- xBOT indica la dirección del primer dato a sacar de la cola.
- El puntero se desborda (sale de la zona de memoria donde están los registros de la cola)tras varias operaciones , y al tratarse de direccionamiento circular hay que hacer ciertas operaciones para que después de la ultima posición de memoria (la más alta) se vuelva a la inferior(direccionamiento circular).

El esquema de las funciones es bastante simple y lineal:



Esquema de las funciones MTX y MRX

Primero realizando direccionamiento indirecto sobre la posición xTOP se mete el dato en la cola.Luego se realiza el incremento de manera circular de la posición xTOP , para ello operamos sobre FSR sobre el que realizamos ciertas operaciones según sea la cola para que si se ha desbordado vuelva al espacio de memoria de la cola en cuestión , y guardando el valor en xTOP .

Esto es , según la cola , después de incrementar el registro :

- Cola de transmisión:

    - Comprobamos si se ha desbordado la cola (si se ha salido de rango el bit 6 de FSR estará activado), de estar activado el bit 6 se efectúan las operaciones siguientes , sino no.
    - Le sumamos 32 , con lo que los bits menos significativos quedan al principio de la cola .
    - Le aplicamos el and lógico con 63 para eliminar los bits más significativos que han quedado desbordados por la operación anterior.

- Cola de recepción :

    - Comprobamos si se ha desbordado la cola (si se ha salido de rango el bit 6 de FSR estará activado), de estar activado el bit 5 se efectúan las operaciones siguientes , sino no.
    - Le aplicamos el and lógico con 64 para eliminar los bits no necesarios , dejando el puntero al inicio de la cola.

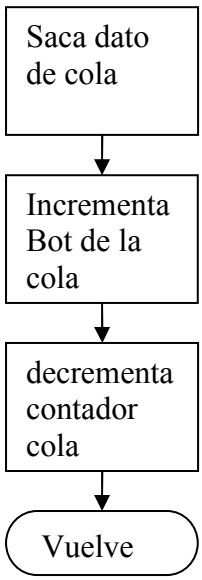
Luego se incrementa el contador de los datos de la cola (CONTx) y se vuelve al bucle principal.

4.2.7.3 STX

Esta función gestiona la extracción de un dato de la cola de transmisión. Trata la cola con direccionamiento circular mediante dos índices de posición (TTOP y TBOT) y un contador de los datos que hay dentro de la cola(CONTTX).También utilizan el registro DATPIL para almacenar el dato a sacar de la cola.

El proceso de direccionamiento circular tiene en cuenta lo mismo que en las anteriores funciones(MTX,MRX).

El esquema de funcionamiento es lineal y simple :



Esquema de la función STX

Primero saca el primer dato de la cola utilizando direccionamiento indirecto , esto es ; se utiliza como valor del puntero (FSR) el registro TBOT .Luego se incrementa de manera circular el valor de la posición TBOT operando sobre FSR, se incrementa FSR y luego :

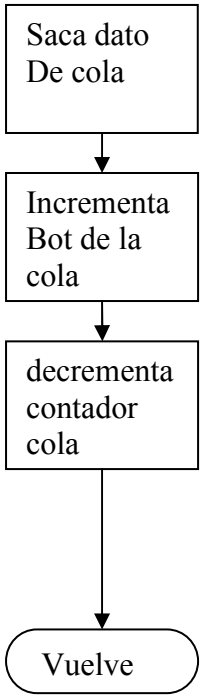
- Comprobamos si se ha desbordado la cola (si se ha salido de rango el bit 6 de FSR estará activado), de estar activado el bit 6 se efectúan las operaciones siguientes , sino no.
- Le sumamos 32 , con lo que los bits menos significativos quedan al principio de la cola y el siguiente queda desbordado.
- Le aplicamos el and lógico con 63 para eliminar los bits más significativos que han quedado desbordados por la operación anterior.

Guardamos el valor de posición así obtenido en TBOT , decrementazos el contador de datos de la cola (CONTTX) y volvemos al bucle principal.

4.2.7.4 SRX

Esta función gestiona la extracción de un dato de la cola de recepción. Trata la cola con direccionamiento circular mediante dos índices de posición (RTOP y RBOT) y un contador de los datos que hay dentro de la cola(CONTRX).También utilizan el registro DATPIL para almacenar el dato a sacar de la cola.
El proceso de direccionamiento circular tiene en cuenta lo mismo que en las anteriores funciones(MTX,MRX).

El esquema de funcionamiento es lineal y simple :



Esquema de la función SRX

Primero saca el primer dato de la cola utilizando direccionamiento indirecto , esto es ; se utiliza como valor del puntero (FSR) el registro RBOT .Luego se incrementa de manera circular el valor de la posición RBOT operando sobre FSR, se incrementa y luego :

- Comprobamos si se ha desbordado la cola (si se ha salido de rango el bit 6 de FSR estará activado), de estar activado el bit 5 se efectúan las operaciones siguientes , sino no.
- Le aplicamos el and lógico con 64 para eliminar los bits no necesarios , dejando el puntero al inicio de la cola.

Guardamos el valor de posición así obtenido en RBOT , decrementamos el contador de datos de la cola (CONTRX) , almacenamos el dato en DATPIL para una posterior transmisión por el puerto serie , y volvemos al bucle principal.

4.2.8 Bucle principal modo normal

Este bucle es la base del programa , se ejecuta de manera continua , permitiendo así la ejecución de las instrucciones que son recibidas por el puerto serie , y controlando el envío de datos por el puerto serie .

Se basa en procesos excluyentes , es decir , que se realiza en cada ejecución del bucle una sola (o las menos posibles) tareas , ya sea identificar y ejecutar una instrucción recibida , leer del MODEM , mandar un dato solicitado por el puerto serie o solicitar un nuevo dato para transmitir.Para estas ultimas tareas se usa el sistema de banderas almacenadas en el registro FLAG , que resulta vital para un buen y coordinado funcionamiento del programa.

El uso que se hace de las interrupciones que nos llegan por el MODEM(interrupciones por el pin RB0 , interrupciones externas) es excluyente en cuanto a los modos de transmisión o recepción. Esto quiere decir que no se activan a la vez las dos interrupciones en el MODEM ya que funciona en modo semi-duplex , o se esta transmitiendo o se esta recibiendo , de manera que las interrupciones se deben en un caso a la recepción y en otro a la transmisión.Entre las interrupciones creadas por recepción también son excluyentes los canales según como se configura , si se activa un modo de lectura continua de un registro no se puede activar el otro hasta haber desactivado el anterior , puesto que así no capturaríamos realmente los datos recibidos al no saber quien ha generado la interrupción.
Su funcionamiento es el siguiente (ver esquema adjunto):

Se mira el puerto serie para ver si ha sido recibida un nuevo byte (identificador de instrucción) para ejecutar una nueva instrucción , si no hay nada se pasa a realizar tareas pendientes que se dejaron en anteriores ejecuciones del bucle principal :

- Se mira el bit R del registro FLAG para ver si se hizo una lectura del MODEM anteriormente y hay dato a transmitir al MODEM :

  - Si hay dato (R=1) se envía por el puerto serie (solo si este esta disponible para hacer la transmisión)y se pone a cero el bit R.En caso de que este ocupado(el puerto serie) se termina el bucle y se vuelve a mirar si hay instrucción nueva (byte recibido en el puerto serie).

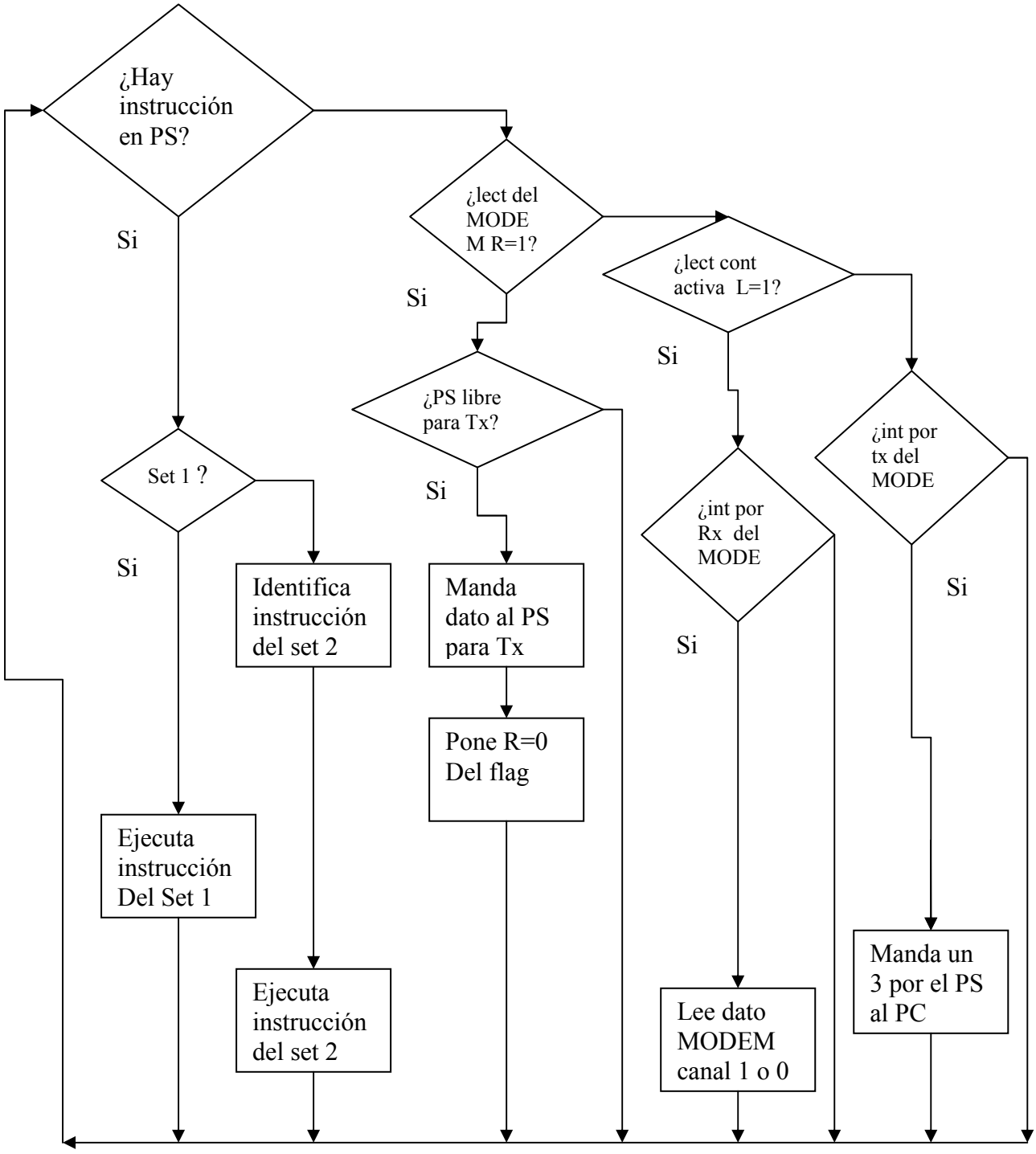
  - Si no hay dato (R=0) se procede a examinar el bit L del registro FLAG , que nos dice si hemos activado en una vuelta anterior del bucle el modo de lectura continua de registro(canal 0 o 1 de recepción del MODEM , registros 00 o 01 en hexadecimal) al ejecutar la instrucción pertinente :

    - Si L=1 , es decir esta activa la lectura continua de registro , comprobamos si el MODEM ha generado un pulso de interrupción

(lo recibimos por RB0 del PIC) mirando el registro de banderas de interrupciones(la correspondiente a la interrupción externa es la que nos interesa).En caso afirmativo procedemos a leer el registro pertinente(canal 0 o 1) del MODEM(se llama a la función Lect después de haber metido en ADDRESS la correspondiente dirección almacenada en TEMPO).
En caso de que no haya interrupción se vuelve de nuevo al principio del bucle.

- Si L=0 entonces se mira si hay interrupción (procedente del MODEM) , si la hay esta se debe a que se ha transmitido un dato(por el MODEM) requiriendo el transmisor un nuevo dato a transmitir para que la transmisión sea continua (dentro de la ráfaga) , por lo que mandamos un 3 por el puerto serie(esto forma parte de un protocolo de requerimiento de datos por el puerto serie que se vera más tarde en las aplicaciones de Labview).
  Si no hay interrupción se vuelve al inicio del bucle.

- Si hay un byte en el puerto serie se identifica el Set al que pertenece la instrucción (primero si es al 1 y luego si es al 2), se identifica la instrucción dentro del Set y se ejecuta la instrucción(se realiza lo mínimo posible en esta vuelta del bucle , dejando el termino de la instrucción a posteriores vueltas) .Luego se vuelve al principio del bucle.



Esquema de funcionamiento del bucle principal

4.2.8.1 Ejecución instrucciones Set 1 y Set 2

Las instrucciones están agrupadas en dos sets (1 y 2) dependiendo de la longitud del identificador de instrucción(1 o 2 bytes al principio de la instrucción).
Así pues esta es la agrupación de instrucciones:

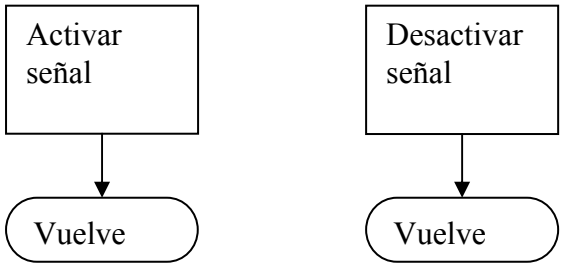
- Set 1:
    - Lectura.
    - Escritura.
    - Lect. Reg. 0.
    - Lect. Reg 1.
    - Fin lect. Reg.
    - Habilitar Test.
    - Habilitar MODEM.
    - Fin Test.
- Set 2:
    - Habilitar Test-esc.
    - Frecuencia 0.
    - Frecuencia 1.
    - Frecuencia P.
    - Frecuencia total.
    - Fin reset.
    - Habilitar reset MODEM.
    - Fin Tx. MODEM.

No obstante algunas de ellas tienen características comunes de ejecución(las tareas que lleva a cabo el programa al ejecutarlas) , por lo que las veremos juntas indistintamente del set al que pertenezcan.
Así pues las agrupamos según las tareas que se ejecutan:

- Habitación y deshabilitación de señales:
    - Aquí se engloban las instrucciones de habilitación siguientes :
        - Habilitar Test.
        - Habilitar Test-esc.
        - Habilitar MODEM.
        - Habilitar reset MODEM.

Estas instrucciones se refieren a la habilitación ya sea por nivel alto(Tests) o por nivel bajo(habilitador MODEM y reset) de una señal generada por el PIC , fijándola a ese valor lógico.

El esquema de funcionamiento es el mismo para todas estas instrucciones :



Esquema de instrucciones de habilitación y deshabilitación.

Como se puede ver en el esquema , al ejecutarse una de estas instrucciones lo que hace es fijar el valor del pin correspondiente al valor requerido escribiendo en el bit correspondiente del puerto un 0 o un 1 lógico :

- Nivel alto (1) para Habilitador de Test y Test-esc.
- Nivel bajo (0) para Habilitador de MODEM y Reset.

Luego volverán al bucle principal.

- Las instrucciones de deshabilitación son las siguientes:

  - Fin Test.
  - Fin reset.

  Estas instrucciones devuelven el pin del puerto correspondiente al valor de inicio anulando el modo que se había activado con una instrucción de habilitación anterior.

Su esquema de funcionamiento se puede ver en el grafico de arriba:

Simplemente se pone el valor lógico adecuado en el bit correspondiente al pin del puerto (se escribe en el registro del puerto) al que se refiere la señal que queremos desactivar :

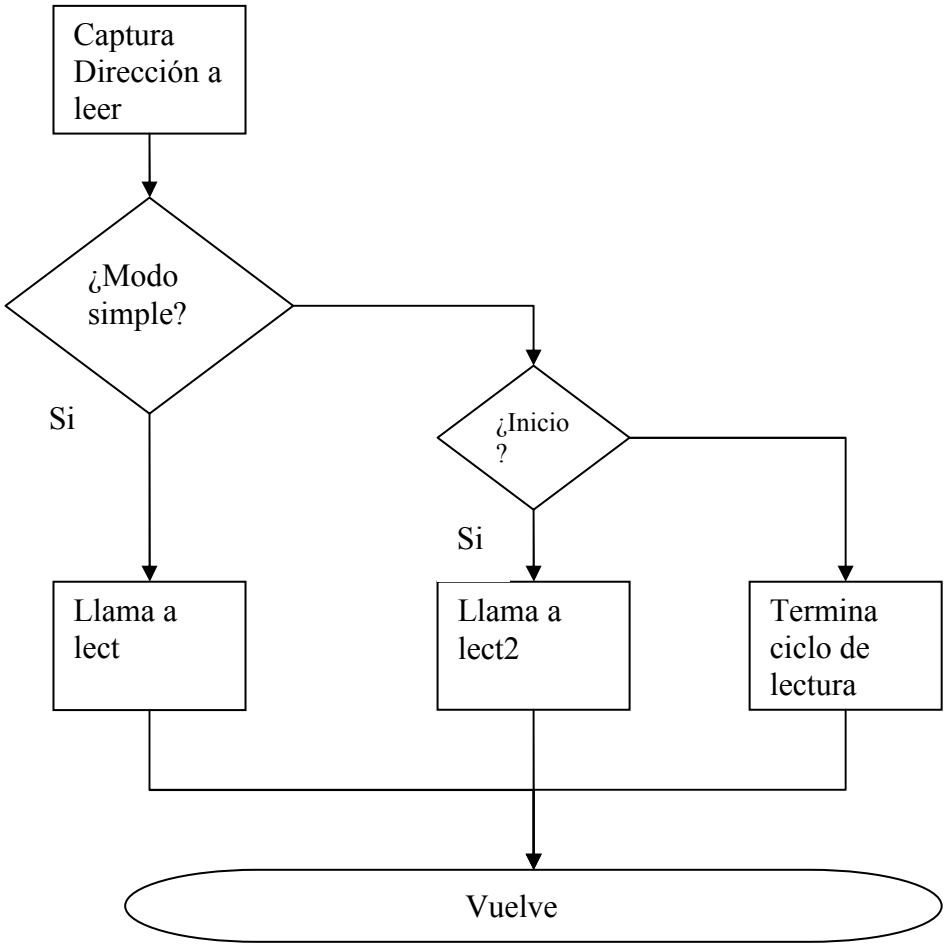
- Para fin test ponemos a nivel bajo la señal (escribimos un 0 en el bit) , desactivando las dos señales de test a la vez (ya que o se da uno o se da otro pero no los dos a la vez, así que no hace falta un desactivador para cada uno).
- Para fin reset ponemos a nivel alto la señal(escribimos un 1 en el bit correspondiente).

Luego volverán al bucle principal.

- Lectura:

Esta instrucción es bastante compleja ya que da lugar a tres posibles usos de la misma(cumpliendo así con dos de las funciones que queremos que realice el controlador). Tiene una parte común respecto a estos usos ya que estos se configuran mediante un parámetro (modo) que se le envía por el puerto serie.Luego en función de esa selección se realizara una función u otra.

Su funcionamiento es ,como se puede ver en el esquema , el siguiente:



Esquema de funcionamiento de la instrucción Lectura

Primeramente se espera en bucle de espera activa(mirando la pertinente bandera de evento , en este caso llegada de datos)del puerto serie a que llegue el primer parámetro , que se guardara en el registro ADDRESS y luego se repite l proceso para el parámetro Modo , pero entonces dependiendo del valor de este se llevan a cabo diferentes tareas:

- Si el valor es 0 entonces se procede a una lectura normal del registro del MODEM indicado por ADDRESS , ello se realiza llamando a la función Lect anteriormente descrita , que deja el dato almacenado en DATMOD y pendiente de ser enviado(R=1) por el puerto serie , ya que a la próxima interacción del bucle principal se realizara esto.
- Si el valor es 1 entonces se procede a un volcado del registro seleccionado por ADDRESS en el puerto D del PIC , para ello se llama a la función Lect2

anteriormente explicada, quedando el puerto D bloqueado en lectura perpetua del registro del MODEM , por lo que podemos ver que esta sucediendo en ese registro(con un osciloscopio).

- Si el valor es 3 entonces se procede a terminar el volcado del registro(que debe anteriormente haberse producido , pues sino no sirve esto de nada).
  Para ello termina el protocolo de lectura del MODEM , es decir subiendo de nuevo la señal OEN (la pone a 1) y devolviendo el puerto D a su estado de salida (desde el PIC).

Después de llevarse la tarea que proceda a cabo se vuelve al inicio del bucle principal.

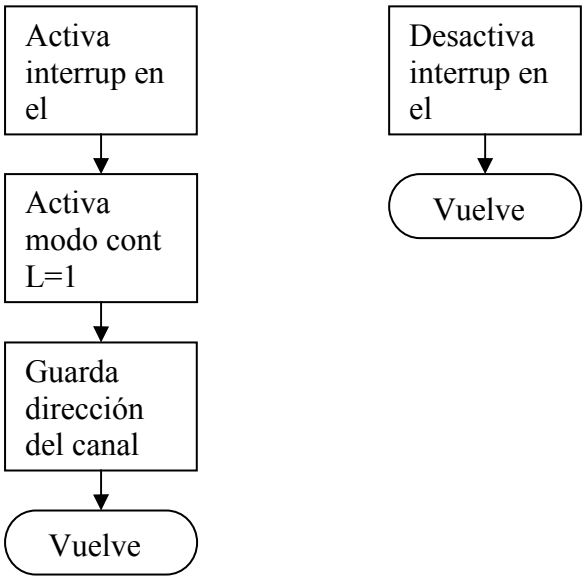
- Lect. Reg 0 y 1 , Fin lect Reg:

Al tratarse de instrucciones relacionadas las veremos en el mismo bloque , ya que se refieren todas a la activación y desactivación del modo continuo de adquisición de datos de uno de los canales de recepción el MODEM(0 o 1).

- Lect. Reg 0 y 1 :

  Aquí lo que se hace es activar la interrupción pertinente en el MODEM (canal 0 o 1) y activar el bit (poner a 1)del modo continuo de adquisición de datos (L del registro FLAG), guardando en TEMPO la dirección del registro(del MODEM) a leer cada vez que le llegue una interrupción al PIC proveniente del MODEM(este a recibido un nuevo dato por el canal seleccionado).

Así pues este es su esquema de funcionamiento:



Esquema de funcionamiento de Lect. Reg 0 y 1 , y de Fin Lect. Reg

Como podemos observar en el esquema su ejecución es secuencial , escribiéndose primero en el registro de interrupciones(habilitador) del MODEM la habilitación de interrupciones por recepción del canal correspondiente según la instrucción (0 o 1). Luego se activa el bit L de FLAG(L=1) para que en posteriores ejecuciones del bucle principal el programa sepa que si hay

interrupciones externas(lo comprueba mirando el registro)se lee del canal de datos del MODEM , para posteriormente mandar el dato por el puerto serie.
Finalmente se guarda en TEMPO el valor de la dirección del registro que debemos leer cada vez que se necesite(llegue nuevo dato al MODEM) y se vuelve al bucle principal.

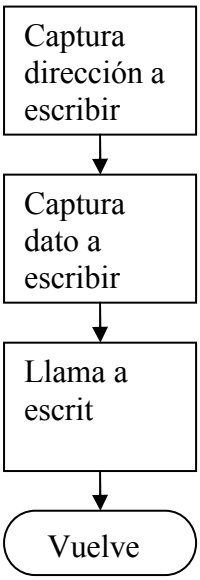
- Fin Lect. Reg:

  Se trata de poner fin a la captura de datos (se supone que luego desactivaremos el Receptor del MODEM , o no nos importan los siguientes datos que se reciban).

Para realizar esto se llevan a cabo los siguientes procesos como se puede ver en el anterior esquema :

Se escribe en el registro habilitador de interrupciones del MODEM deshabilitando la recepción por los canales(no pueden activarse los dos a la vez por lo que solo hace falta un deshabilitador para los dos , deshabilitándolos a los dos a la vez) y se vuelve al principio del bucle principal de nuevo.

- Escritura :
  Esta instrucción en el modo simple de programación(primer programa)se dedica solo a escribir en el registro del MODEM seleccionado por ADDRESS el dato introducido previamente en DATMOD , en el 2º modo se complica , pero ya se vera mas adelante.
  Su esquema y funcionamiento son los siguientes:



Esquema de funcionamiento de la instrucción Escritura
Primero se espera en bucle de captura(espera hasta que en el puerto serie se haya recibido un nuevo byte)el valor que se introduce en ADDRESS . Esto se hace también para el siguiente parámetro que se mete en DATMOD , con lo que ya disponemos de los parámetros adecuados para escribir en el MODEM mediante la función escrit que ya se vio antes .Se realiza la escritura llamando a escrit y después se vuelve al principio del bucle principal.

- Frecuencia 0 y 1 :
  Las agrupamos juntas pues realizan un proceso muy similar , si bien a la hora de configurar y generar la señal requerida(PLL1 o PLL0) se diferencian pero se verán los dos procesos en si.
  Tienen dos posibilidades de funcionar según el parámetro Modo , generando o no una señal PLL para el MODEM , en cualquiera de los dos casos se capturan dos parámetros de datos de configuración de frecuencia , uno para el MODEM(que siempre se utiliza) y otro para el PIC(solos e usa en el modo externo) para configurar la señal a generar.

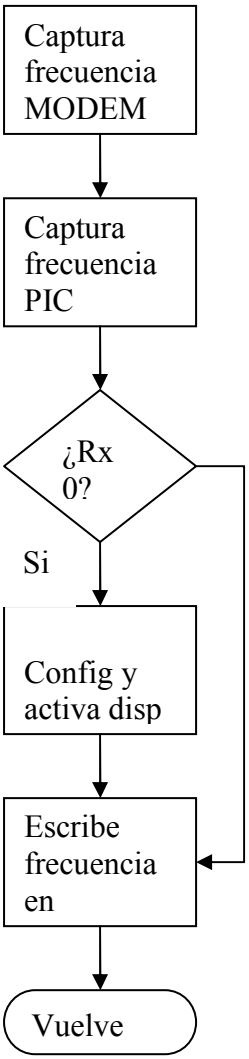
El funcionamiento es como se puede ver en el siguiente esquema:

Primero se entra en un ciclo de captura (como en los otros casos antes mencionados) hasta que se recibe el primer dato por el puerto serie , que se almacena en el registro DATMOD , luego se vuelve a repetir este proceso pero para capturar el valor que nos servirá para configurar la frecuencia generar por el PIC (dependiendo de la instrucción) , que se almacena en el registro PORTN (Frecuencia 0) o en FRECUEN.
De nuevo se captura un nuevo byte por el puerto serie(como antes) , es el parámetro Modo que nos dice si va a ser necesario configurar y generar la frecuencia PLL o no ;

- Si recibimos un 0 no va a ser necesario(interno)
- En otro caso si(externo).

Ahora se procede a configurar y generar la señal , dependiendo de la instrucción se realiza un proceso u otro :

- Frecuencia 0:
  Se introduce el valor de configuración en el registro TEMP0 del timer 0 y se activan las interrupciones por desbordamiento del timer 0, de manera que creara interrupciones cuando se desborde(en la rutina de interrupción esta el medio de generación de la señal PLL0 como ya se vio antes).
- Frecuencia 1:
  Configura el periférico PWM para que genere una señal cuadrada como un reloj con un periodo definido por el parámetro anteriormente dado(FRECUEN da el nº necesario para el contador Timer 2 según el manual del PIC) .



Esquema de funcionamiento de las instrucciones Frecuencia 0 y Frecuencia 1

Posteriormente se realiza la escritura en el registro correspondiente de frecuencia del MODEM(tanto en el caso externo como en el interno) y se vuelve al principio del bucle principal.

- Frecuencia P:

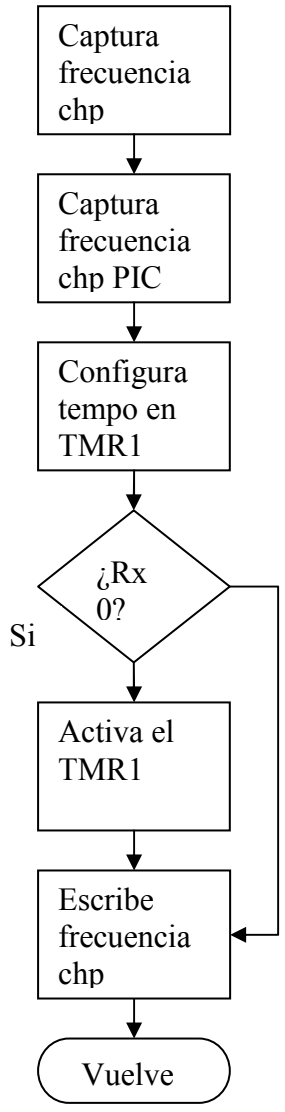
  Al igual que en el caso de las otras frecuencias esta instrucción depende del parámetro Modo para elegir entre dos funcionamientos ; uno el básico (interno) en el que solo se escribe en el registro de la frecuencia piloto del MODEM el valor que nos es dado , y el otro en el que además se configura y genera una señal PLLp e frecuencia dada por el parámetro guardado en el registro PORTM , en cualquiera de los dos caso deben capturarse los dos datos.

Su esquema de funcionamiento es el siguiente:

Primero se entra en un ciclo de captura (como en los otros casos antes mencionados) hasta que se recibe el primer dato por el puerto serie , que se almacena en el registro DATMOD , luego se vuelve a repetir este proceso pero para capturar el valor que nos servirá para configurar la frecuencia a generar por el PIC (dependiendo de la instrucción) , que se almacena en el registro PORTM.
Se configura el Timer 1 con el valor de PORTM para el 2 byte del contador , el primero se pone a FF(Hex).
De nuevo se captura un nuevo byte por el puerto serie(como antes) , es el parámetro Modo que nos dice si va a ser necesario generar la frecuencia PLL o no ;

- Si recibimos un 0 no va a ser necesario(interno)
- En otro caso si(externo).

En el caso externo se termina de configurar el Timer 1 y se activan las interrupciones debidas a este.

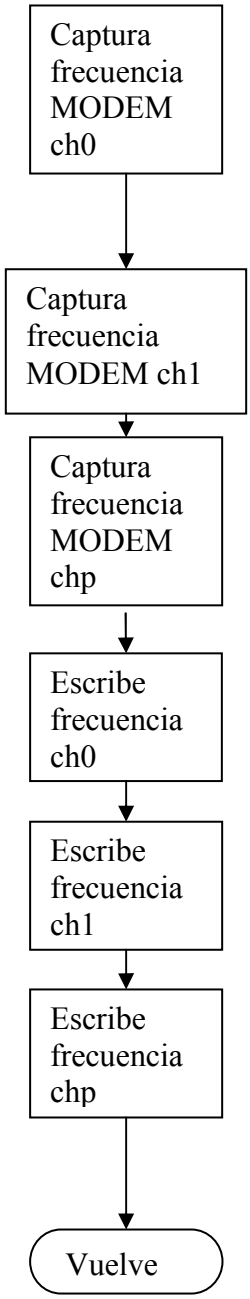


Esquema de la instrucción Frecuencia P

Posteriormente se realiza la escritura en el registro correspondiente de frecuencia del MODEM(tanto en el caso externo como en el interno) y se vuelve al principio del bucle principal.

- Frecuencia total:
  Se trata de configurar los registros de frecuencia del transmisor del MODEM todos de una vez para ello se usan varios registros mas de datos capturados(DATMOD2 y DATMOD3).

Su esquema y funcionamiento es el siguiente:



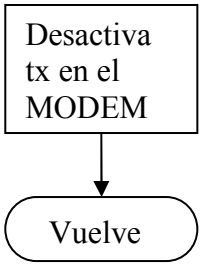
Esquema de la instrucción Frecuencia total

Mediante bucle de captura(como antes) se capturan por el puerto serie los datos a introducir en los registros de configuración del MODEM; frecuencias del canal 0 y 1 , y frecuencia

piloto.Posteriormente y asignando internamente las direcciones de los registros del MODEM a escribir realizamos en el mismo orden las escrituras en el MODEM , volviendo luego al inicio del bucle principal.

- Fin Tx MODEM:
  Esta instrucción permite la desactivación del transmisor del MODEM , de manera que se puede controlar el proceso por el puerto serie.

  Su esquema y funcionamiento es el siguiente:



Esquema de Fin Tx MODEM

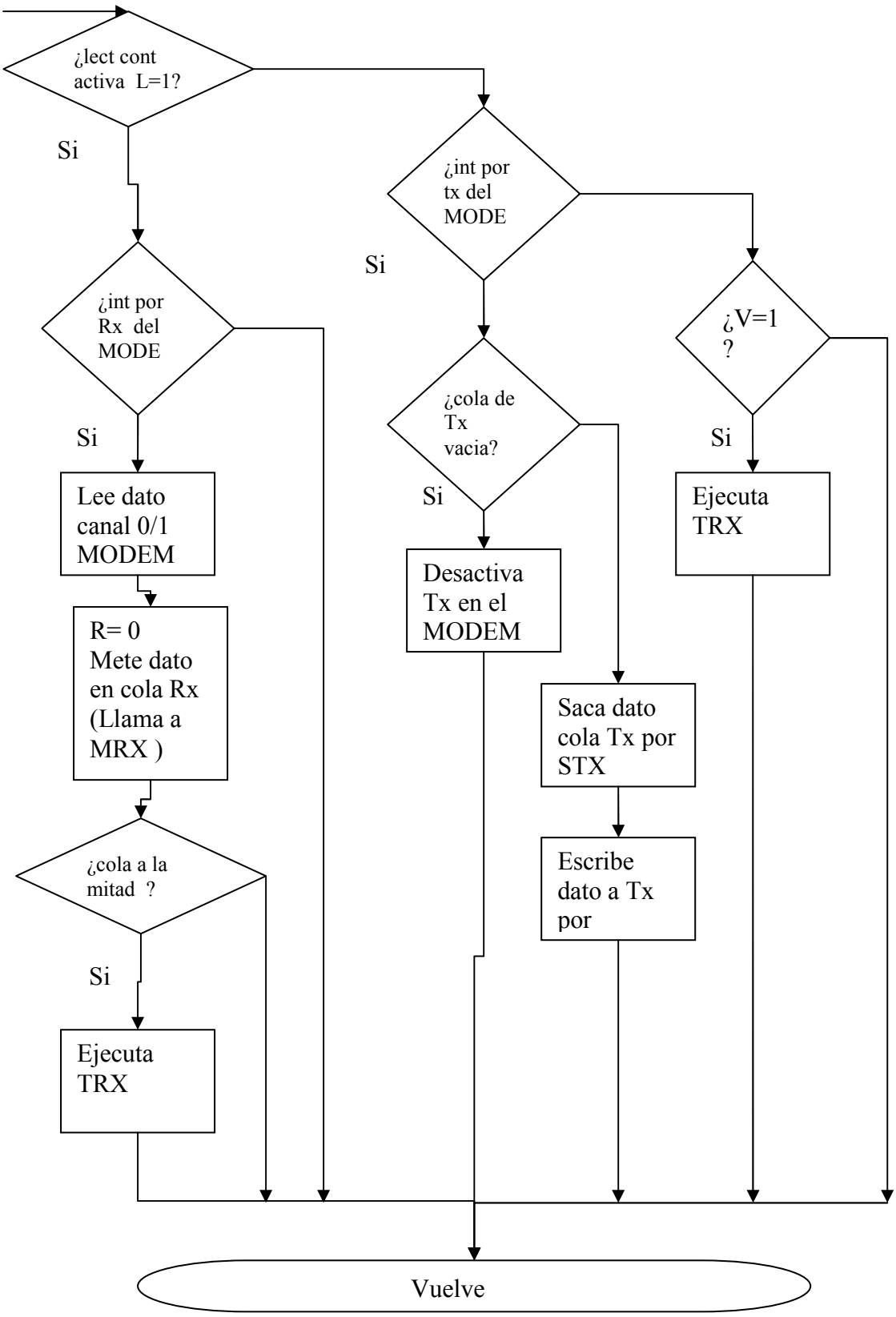
Se lee el registro de estado del transmisor del MODEM de manera que solo modifiquemos el bit que activa la transmisión (poniéndolo a 0) al realizar un and lógico con todo unos menos la posición del bit de habilitación de transmisión .Posteriormente escribimos este valor en el registro del MODEM , desactivando así la transmisión pero sin variar la configuración de este o del receptor(ya que están en el mismo registro).
Finalmente se vuelve al inicio del bucle principal.

4.2.9 2º modo de programación (programa modificado)

Estas son las modificaciones realizadas sobre el programa inicial (básico) que nos permiten un control de la transmisión por ráfagas y la recepción más automático , implementándose una cola para cada una de esas funciones.
Veamos las modificaciones sufridas:

- El bucle principal sufre ciertas modificaciones a partir de la comprobación de si hay adquisición continua de datos recibidos por un canal del MODEM (L=1) o no.

El esquema de la modificación así como su funcionamiento es el siguiente :



Esquema de la modificación en el bucle principal

Se comprueba el bit L del registro FLAG (lectura continua de registro activa o no) , en función de su valor se hará una cosa u otra:

- Si L=1 hay lectura continua de registro por lo que se comprueba si hay interrupciones por causas externas(interrupción generada por el MODEM al recibir un nuevo dato en el canal de recepción 0 o 1) , según el caso se hace:

    - Si hay interrupción se lee el registro del canal de recepción apropiado(0 o 1) del MODEM y posteriormente se llama a la función MRX para meter el dato en cola de recepción (poniéndose a 0 de nuevo el bit R).
      En función de cómo este la cola(vacía o con al menos la mitad de posiciones ocupadas) se ejecuta un bloque de tareas común a la instrucción Fin lect. Reg modificada(que luego veremos) , ejecutándose en el caso de que al menos este medio llena la cola.

Finalmente se vuelve al principio del bucle.

    - Si no hay interrupción se vuelve al principio del bucle.

- Si L=0 entonces puede que haya interrupciones externas generadas por el transmisor del MODEM(necesita un nuevo dato a transmitir) , es decir , puede que estemos en medio de la transmisión de una ráfaga de datos por el MODEM:

    - Si no hay interrupción se comprueba el valor del bit V del registro FLAG, este nos dice si hay datos en la cola de recepción pendientes de vaciar y enviar por el puerto serie de una anterior recepción (se habría desactivado el receptor pero la cola aun tendría datos) de una ráfaga:

        - Si V=0 se vuelve al principio del bucle.
        - Si V=1 se ejecuta el bloque de tareas TRX.

    - Si hay interrupción entonces se comprueba si hay datos aun por transmitir (cola de transmisión vacía o no) , esto se realiza comparando el valor del registro CONTTX con 0 (mediante la función Comp0):

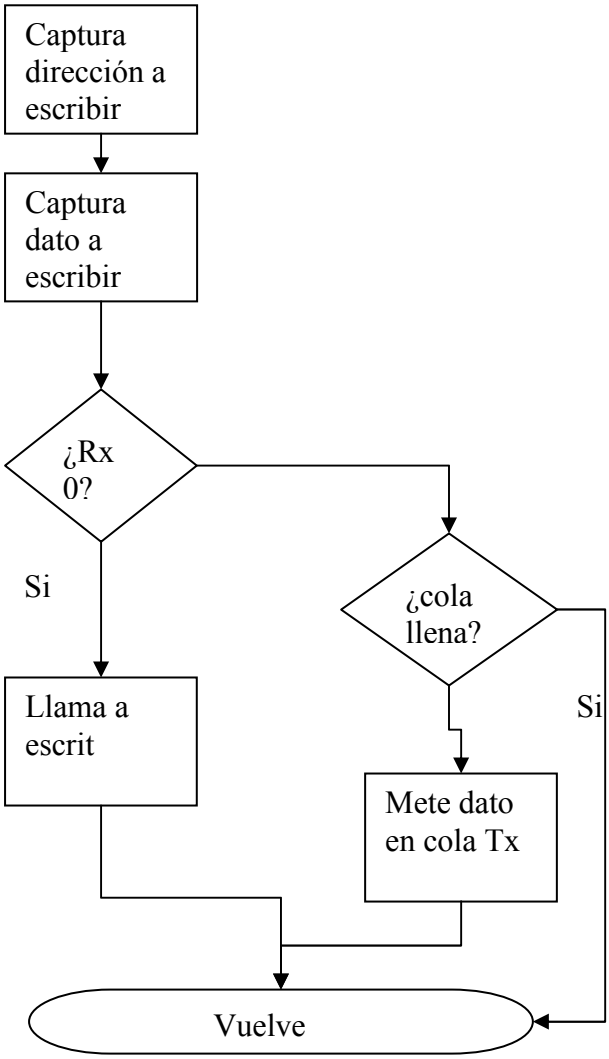
        - Si la cola de transmisión esta vacía (C=1) se desactiva el transmisor del MODEM mediante la ejecución de la instrucción Fin Tx MODEM vista en el modo principal de programación.
        - Si hay datos aun por transmitir (C=0)se saca el primero(mediante la función STX) y se escribe en el registro de dato a transmitir del MODEM (continuando la trasmisión de la ráfaga).

      Luego se vuelve al principio del bucle.

➢ Escritura(modificada):

Esta instrucción sufre modificaciones en este modo de programación ya que tiene que tener en cuenta si realizamos una escritura normal(a registro del MODEM) , o una escritura de dato a transmitir (en la cola de transmisión).

El esquema modificado resultante que nos muestra su comportamiento es el siguiente:



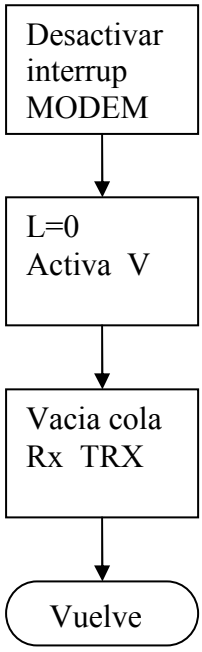
Esquema de la instrucción Escritura modificada

Como se puede observar , la primera parte de captura de dirección(ADDRESS) y dato(DATMOD) es la misma que en el anterior modo de programación(espera en bucle de captura) , y se realizan del mismo orden .Luego se procede a capturar un tercer parámetro característico de la modificación , el Modo , que nos indicara la acción a seguir por la instrucción:

- Si es 0 entonces se ejecuta la escritura de manera normal llamando a la función escrit (que realiza todo el proceso) y luego se vuelve al inicio del bucle principal.
- Si es distinto de 0 (normalmente será un 1) comprobamos si la cola de transmisión esta llena (al tener 32 posiciones se limita a comparar un bit):
    - Si esta llena volvemos al principio del bucle principal.
    - Si aun hay posiciones libres metemos el dato almacenado en DAMOD en DATPIL y llamamos a MTX , con lo que almacenamos el dato en la cola de transmisión .Posteriormente volvemos al inicio del bucle principal.

- Fin lect. Reg modificada:
  Debido a la implementación de una cola de recepción cambia la forma de transmitir los datos que se adquieren del canal de recepción del MODEM , ya que se almacenan y se transmiten cuando la cola esta medio llena(solucionando cualquier posible desajuste de sobre escritura del registro DATMOD si se recibieran datos demasiado rápido).
  El esquema de funcionamiento modificado es el siguiente:

Es escribe en el registro de habilitación de interrupciones del MODEM desactivando las interrupciones por recepción (canales 0 y 1) igual que antes y se desactiva el bit L del registro FLAG(L=0) .Ahora se activa el bit V del registro FLAG(V=1) y se saca el primer dato de la cola de recepción utilizando el bloque TRX que controla si hay datos o no en la cola y si los hay saca el primero y lo transmite por el puerto serie(se queda en espera si el puerto serie no esta disponible , hasta que lo este).

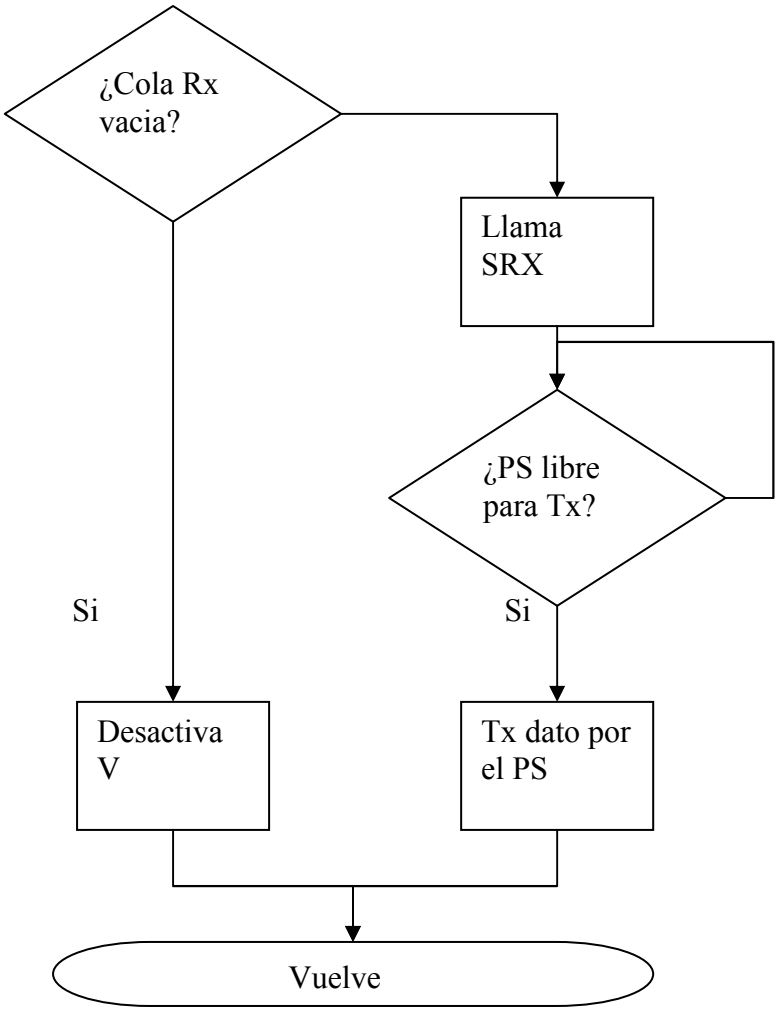


Esquema de la función Fin lect. Reg modificada

Finalmente se vuelve al principio del bucle principal.

- Bloque TRX:
  A pesar de no tratarse de una instrucción(no se ejecuta al recibir un byte por el puerto serie) es un bloque bastante utilizado en esta versión del programa , ya que gestiona la cola de recepción en cuanto a sacar datos de ella y mandarlos por el puerto serie.

Su es esquema de funcionamiento es el siguiente:



Esquema de TRX , bloque de Fin lect. Reg utilizado en el bucle principal modificado

Comprueba si la cola de recepción esta vacía o no , para ello mete el valor del registro CONTRX en COMPW y llama a la función Comp0(descrita anteriormente) , en función del valor del bit C(del registro FLAG) hace una cosa u otra:

- Si C=1 , la cola esta vacía por lo que desactiva V(V=0) y vuelve .
- Si C=0 , aun hay datos en la cola por lo que sacamos el primero llamando a la función SRX(anteriormente explicada).
  Luego se transmite el dato por el puerto serie cuando este no esté ocupado(se queda en espera si lo estuviese) y vuelve.

4.3 Modos de Transmisión y Recepción externo e interno

Disponemos de dos modos de ejecutar las transmisiones a ráfagas y recepciones del MODEM según el modo de programación que usemos ; con el primer programa usamos el modo externo

que esta completamente supervisado por el controlador(PC u otro dispositivo , en nuestro caso una aplicación del Labview sobre PC) que gobierna la placa controladora por el puerto serie , disponiendo incluso de un protocolo para cargar datos en el transmisor del MODEM conforme los vaya necesitando este(transmisión de una ráfaga) , con el segundo programa usamos el modo interno que nos implementa una cola para recepción (donde almacenamos los datos recibidos por el MODEM y luego la descargamos por el puerto serie) y otra de transmisión (donde cargamos los datos de la ráfaga que queremos transmitir , así como la cabecera necesaria en las primeras posiciones) ,automatizándose la transmisión una vez cargados los datos y las instrucciones pertinentes de configuración .

Así los modos quedan de la siguiente manera:

- Modo externo, lo utilizamos cuando la velocidad de transmisión del MODEM es inferior a la de comunicación por el puerto serie (en nuestro caso 9600bps), ya que así el paso de datos desde el puerto serie al MODEM para transmitir será fluido(la ráfaga se realiza adecuadamente) y no necesitaremos una cola de recepción ya que podremos enviar los datos por el puerto serie en menos tiempo que el que nos llegan provenientes del MODEM :

    - Transmisión :
      En este modo una vez configurado el MODEM necesitamos un protocolo para proporcionar datos al transmisor de este con objeto de no necesitar una cola de transmisión para asegurar la continuidad de la ráfaga transmitida , el programa de Labview(PC) controla este paso de datos , y cuando se han terminado de enviar los datos de la ráfaga manda la instrucción de parar transmisor , controlando de esa manera todo el proceso de manera automática pero interviniendo el PC en ella.

    - Recepción:
      En este modo no hace falta una cola de recepción al no llegar datos del MODEM más rápido de lo que podemos enviarlos por el puerto serie , por lo tanto una vez configurado el receptor del MODEM y activado el modo de adquisición de datos recibidos por uno de los canales del MODEM es el programa de Labview el encargado de ordenar el cese del modo y desactivar el receptor cuando se hayan recibido el número de datos necesarios(configurable por el usuario).

- Modo interno , lo utilizamos cuando la velocidad de transmisión del MODEM es superior a la de comunicación por el puerto serie (en nuestro caso 9600bps) o simplemente si queremos tener los procesos más automatizados y disponer de colas de datos en transmisión y en recepción , ya que así el paso de datos desde el puerto serie al MODEM para transmitir no será fluido(la ráfaga no se realiza adecuadamente) y necesitaremos una cola de transmisión un protocolo interno(dentro del mismo PIC) de paso de datos al transmisor del MODEM , también necesitaremos una cola para recepción ya que enviamos los datos por el puerto serie en mas tiempo que el que nos llegan provenientes del MODEM :

    - Transmisión :
      En este modo las ráfagas se realizan bajo el control del PIC una vez configurado el MODEM convenientemente , es decir el PIC dispone de una cola de datos de transmisión(la ráfaga a transmitir con las primeras posiciones ocupadas por la cabecera adecuada) , de manera que cuando se inicia la transmisión es el mismo el

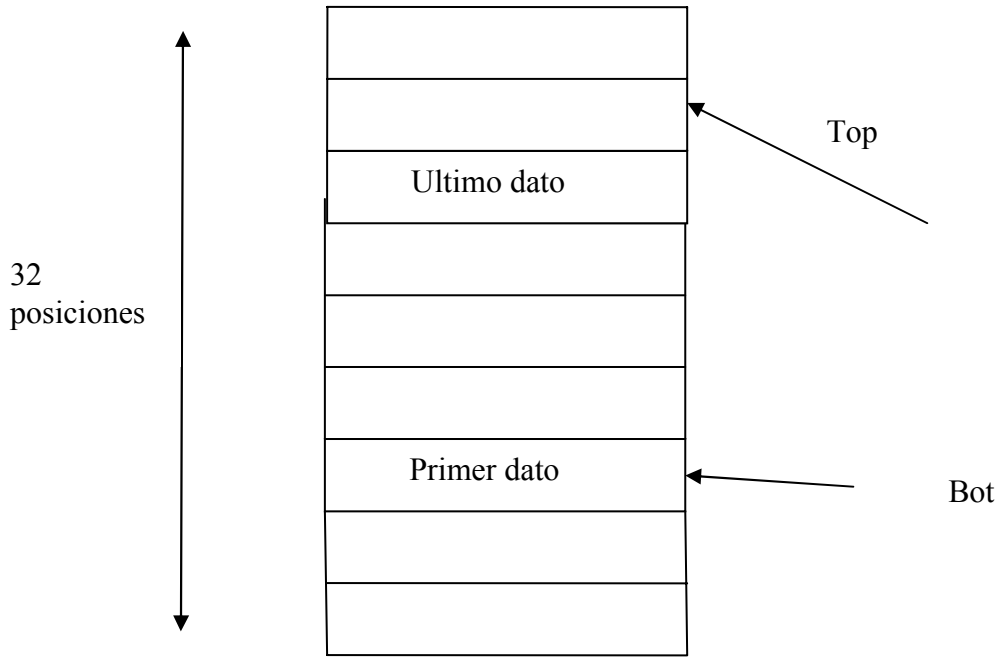
que pasa los datos al MODEM cuando este los pide(ráfaga continua) , y además pone fin a la transmisión cuando se acaban los datos (ejecuta la instrucción Fin Tx MODEM), si bien mientras se esta transmitiendo se pueden ir metiendo nuevos datos por el puerto serie en la cola (esto debe hacerse modificando el programa de Labview).

- Recepción :
  En este modo se almacenan los datos que son recibidos por el canal de recepción del MODEM(después de configurar y activar modo captura de registro continua) en una cola , de manera que se ajuste el flujo de datos si la velocidad de transmisión por el puerto serie fuera más lenta que la de transmisión por el MODEM .El fin de la recepción sigue gobernado por el programa de labview (ya que así como la cola se va vaciando con cada nueva entrada cuando llega a la mitad de su capacidad y se pueden recibir ráfagas de más de 32 datos), aunque cambiando un poco la programación podríamos hacer que cuando se llene la cola el mismo(el PIC) pare la recepción y luego vuelque los datos por el puerto serie(solo recibiríamos ráfagas de 32 datos).

4.4 Colas

Utilizamos dos colas en este 2º modo de programación para actuar de buffer para transmisión y recepción del MODEM.Su estructura y gobierno son los mismos.
Sus características básicas son:

- 32 posiciones , registros de memoria del PIC del primer banco , que se pueden ocupar en su totalidad.
- Disciplina FIFO en el manejo de datos.
- Direccionamiento circular de la cola para un uso practico y transparente al usuario que almacena o saca datos de ella.
- Dos índices que señalan las posiciones del primer dato almacenado y de la primera posición libre de la cola respectivamente (xBOT y xTOP).
- Un índice contador (CONTx) que nos informa de los datos almacenados en la cola , utilizado en la gestión de entrada y salida de datos.
- Inicialmente xBOT=xTOP igual a la posición más baja de la región de memoria ocupada por la cola , pues viene del reposo y no hay datos en la cola.



Esquema de las colas de transmisión y recepción.

4.5 Programas comentados

```
      LIST P=16F877                ; configuración del programa
errorlevel -302              ; para el compilador
INCLUDE "P16F877.INC"   ; incluye archive de direcciones del PIC

__CONFIG _BODEN_OFF & _CP_OFF & _PWRTE_OFF & _WDT_ON & _WRT_ENABLE_ON &
_HS_OSC & _DEBUG_OFF & _CPD_OFF & _LVP_OFF


LER   EQU 0x2F          ; direcciones reservadas para
DATMOD      EQU 0x21    ; los registros del programa
ADDRESS EQU 0x22
TEMPO EQU 0x30
FLAG  EQU 0x24
FRECUEN     EQU 0x27
SEGUR EQU 0x32
COPIA EQU 0x29
PORTM EQU 0x2A
PORTN EQU 0x2D
DATMOD2     EQU 0x2B
DATMOD3     EQU 0x2C
BANCO EQU 0x23
ORG 0x00                ; dirección del reset
goto INICIO   ;cuando hay reset se salta al inicio del programa
```

```
ORG 0x04      ;dirección de interrupción
      goto INTER     ;cuando hay  interrupción se salta a la rutina de
                  ;interrupción
org 0x05

INTER movwf COPIA                   ;rutina de interrupción
movf  STATUS,W
movwf BANCO                   ;salva los registros
bcf  STATUS,RP0
btfss PIR1,TMR1IF             ;identifica la causa de la
; interrupción
goto INTER2
movf  PORTM,W
movwf TMR1L                   ;configura timer 1
movlw b'11111111'
movwf TMR1H
bcf PIR1,TMR1IF
btfss PORTB,6
goto CLAR
bcf PORTB,6       ;  cambio en la señal para generación de
;  frecuencia
FINAL movf  BANCO,W
movwf STATUS      ;devuelve los registros

movf  COPIA,W

retfie                        ;vuelve de la rutina
CLAR  bsf PORTB,6       ;  cambio en la señal para generación de
;  frecuencia
movf  BANCO,W
movwf STATUS      ;devuelve los registros
movf  COPIA,W
retfie            ;vuelve de la rutina
INTER2 bcf  INTCON,2
movf  PORTN,W
movwf TMR0              ;configura timer 0
btfss PORTB,7     ; cambio en la señal para generación de
;  frecuencia
goto CLAR2
bcf PORTB,7
movf  BANCO,W     ;devuelve los registros
movwf STATUS
movf  COPIA,W
retfie            ;vuelve de la rutina
CLAR2 bsf PORTB,7       ; cambio en la señal para generación de
;  frecuencia
movf  BANCO,W
movwf STATUS      ;devuelve los registros
movf  COPIA,W
retfie            ;vuelve de la rutina

LECT  movf ADDRESS,W          ;Función de lectura del MODEM
movlw .2          ;se pasa primero la dirección del registro
movwf PORTE
nop               ;al final en DATMOD queda el dato leído del
; MODEM
movlw .7          ;se sigue el protocolo de lectura
movwf PORTE
bsf STATUS,RP0          ;se cambia el Puerto D a lectura
movlw b'11111111'
movwf TRISD
```

```
bcf STATUS,RP0
movlw .5
movwf PORTE       ;se lee el dato
movf PORTD,W
movwf DATMOD
bsf  PORTE,1
bsf  FLAG,1       ;se activa la bandera de dato para tx
bsf STATUS,RP0
clrf TRISD        ;se configura  el puerto D como salida
bcf STATUS,RP0
bcf STATUS,RP1
return            ;se vuelve
LECT2 movf ADDRESS,W
movwf PORTD
movlw .2          ;se pasa primero la dirección del registro
movwf PORTE       ;siguiendo el protocolo de lectura
nop               ;al final en DATMOD queda el dato leído del
; MODEM
movlw .7
movwf PORTE
bsf STATUS,RP0
movlw b'11111111' ; se configura el puerto D como entrada
movwf TRISD
bcf STATUS,RP0
movlw .5          ;se interrumpe el protocolo de lectura
movwf PORTE       ;quedando la lectura del registro continua
return            ;se vuelve

ESCRT movf ADDRESS,W          ;función de escritura del MODEM
movwf PORTD
movlw .2
movwf PORTE             ;se pasa la dirección al MODEM
nop
movlw .7                ;cumpliendo con el protocolo de escritura
movwf PORTE
movf DATMOD,W           ;se pasa el valor a escribir  en el MODEM
movwf PORTD
bcf  PORTE,0
nop
nop
bsf  PORTE,0            ;se hace que el MODEM capture el dato
return                  ; se vuelve


INICIO clrf FLAG
clrf PORTB        ;csn a nivel alto , MODEM no seleccionado aún
clrf PORTC        ;configuración  de los registros
clrf PORTD
clrf PORTE
bsf  STATUS,RP0
movlw .6
movwf ADCON1      ;configuración
clrf  TRISD
clrf  TRISE
clrf  TRISB
bsf   TRISB,0
bcf   TRISC,6
bcf   TRISC,1
bcf   TRISC,2
bsf   TXSTA,5
bsf   TXSTA,2           ;USART sin interrupciones , se opera por
```

```
; banderas
bsf   SPBRG,0
bsf   SPBRG,7          ; funciona asíncrona a 9,6 kb
bcf   OPTION_REG,5
bcf  STATUS,RP0
bcf  PORTC,5           ;configuración modos test desactivados
bcf   PORTC,4
bsf   INTCON,7         ;configuración interrupciones
bsf   INTCON,6         ;solo periféricos
      bsf   RCSTA,7
bsf   RCSTA,4
bsf   PORTB,5          ;reset y enable del MODEM desactivados
bsf   PORTB,4
movlw .7
movwf PORTE            ;bus de control E a 111

BUCLE clrwdt                ;bucle de funcionamiento
btfss   PIR1,RCIF
goto TRANS        ; lo primero que se recibe es una palabra de
bcf   PIR1,RCIF   ;control , según esta se hacen varias cosas
movf  RCREG,W
movwf LER
btfss   LER,0          ; según los bits, si es 0 el bit 0 se
; procede a leer
goto  LECTU            ; bit5 bit4 bit3 bit2 bit1 bit0
btfss   LER,1          ;  x     x    x    x    0   1
goto  ESCRITU          ;                        se escribe
btfss   LER,2
goto  LETCON0          ;  x     x    x    0    1   1
      ;se leen los datos del canal 0 de manera continua
btfss   LER,3          ; ver instrucciones
goto  LETCON1
btfss   LER,4          ;identificación de la instrucción
goto  FINCONT
btfss   LER,5
goto  TEST1            ;modo 1 de test RA2
btfss   LER,6
goto  SET2             ;más instrucciones posibles set 2
btfss LER,7
goto  ENABLE           ;habilitar el MODEM
bcf   PORTC,5
bcf   PORTC,4          ;acabar con los modos test
goto  BUCLE
TEST1 bsf   PORTC,4          ;activa modo test
goto BUCLE
SET2  clrwdt                 ;se identifica dentro del set 2
btfss   PIR1,RCIF      ;bucle de espera en captura
goto SET2
bcf   PIR1,RCIF
movf  RCREG,W
movwf LER
btfss   LER,0
goto  FREC0
btfss   LER,1
goto  FREC1
btfss   LER,2
goto  FRECP
btfss   LER,3
goto  FREC3
btfss   LER,4
goto  RESET
```

```
btfss   LER,5
goto  TRANSF
btfss   LER,6
goto  NULO
btfss   LER,7
goto  TESESC
goto  FRESET
TESESC      bcf   PORTC,5     ;activa modo test-esc
goto  BUCLE
FREC0 clrwdt                  ; frecuencia 0
btfss   PIR1,RCIF       ;se espera a coger el dato de config
goto  FREC0
bcf   PIR1,RCIF
movf  RCREG,w
movwf DATMOD      ;se guarda el dato de frecuencia para el MODEM
FREC01      clrwdt
btfss   PIR1,RCIF
goto  FREC01      ;se captura el valor para configurar el timer
bcf   PIR1,RCIF
movf  RCREG,w
movwf PORTN
FREC02      clrwdt
btfss   PIR1,RCIF
goto  FREC02                  ;se captura el modo
bcf   PIR1,RCIF
btfss   RCREG,0
goto  CF0
movf  PORTN,W           ;se configura y activa el timer 0
movwf TMR0
      bsf   INTCON,5
CF0   movlw .52               ;se escribe en el MODEM
movwf ADDRESS
call  ESCRT
goto  BUCLE
FREC1 clrwdt
btfss   PIR1,RCIF    ;se espera a coger la frecuencia del MODEM
goto  FREC1
bcf   PIR1,RCIF
movf  RCREG,w
movwf DATMOD
FREC11 btfss   PIR1,RCIF
goto  FREC11      ; se captura el valor para configurar el timer
bcf   PIR1,RCIF
movf  RCREG,w
movwf FRECUEN
FREC12      clrwdt
btfss   PIR1,RCIF
goto  FREC12                  ;se captura el modo
bcf   PIR1,RCIF
btfss   RCREG,0
goto  CF1
movf  FRECUEN,w
bsf   STATUS,RP0        ;se configura el PWM
movwf PR2
bcf   STATUS,RP0        ;el valor de configuración es el periodo
incf  FRECUEN,1         ;según la formula del datasheet del PIC
rlf   FRECUEN,1         ; se configura el tiempo activo a partir
bcf   FRECUEN,0         ; del mismo valor
btfsc FRECUEN,1
goto  OTRO
movlw b'00001100'
```

```
movwf CCP2CON
goto  SIGUE
OTRO  movlw b'00101100'
movwf CCP2CON           ;se terminan de configurar los registros
SIGUE rrf   FRECUEN,1         ;del PWM
rrf   FRECUEN,1
movlw b'00111111'
andwf FRECUEN,W
movwf CCPR2L
bsf   STATUS,RP0
bcf   TRISC,1
bcf   STATUS,RP0
movlw b'00000100'             ;activación del PWM
movwf T2CON
bsf   CCP2CON,3
bsf   CCP2CON,2
CF1   movlw .53
movwf ADDRESS
call ESCRT              ; se escribe en el MODEM la frecuencia
goto BUCLE

FRECP clrwdt
btfss PIR1,RCIF         ;se captura la frecuencia para el MODEM
goto  FRECP
bcf   PIR1,RCIF
movf  RCREG,w
movwf DATMOD
FRECP1      clrwdt
btfss PIR1,RCIF   ;se captura el dato de configuración del timer
goto  FRECP1
bcf   PIR1,RCIF
movf  RCREG,w
movwf TMR1L       ;se configura el timer 1
movwf PORTM
movlw b'11111111'
movfw TMR1H
FRECP2      clrwdt
btfss   PIR1,RCIF
goto  FRECP2                  ;se captura el modo
bcf   PIR1,RCIF
btfss RCREG,0
goto  CFP
bsf   T1CON,0           ;se activa el timer 1
bsf   STATUS,RP0
bsf   PIE1,0
bcf   STATUS,RP0
CFP   movlw .54
movwf ADDRESS
call ESCRT              ; se escribe en el MODEM la frecuencia
goto BUCLE
FREC3 clrwdt
btfss PIR1,RCIF         ;captura el primer dato
goto  FREC3
bcf   PIR1,RCIF
movf  RCREG,w
movwf DATMOD
FREC31 clrwdt
btfss   PIR1,RCIF       ;captura el segundo dato
goto  FREC31
bcf   PIR1,RCIF
movf  RCREG,w
```

```
movwf DATMOD2
FREC32      clrwdt
btfss   PIR1,RCIF       ;captura el tercer dato
goto  FREC32
bcf   PIR1,RCIF
movf  RCREG,w
movwf DATMOD
CFT   clrwdt                ;escribe el primer dato en el MODEM
movlw .52
movwf ADDRESS
call  ESCRT
clrwdt
movlw .53
movwf ADDRESS
movf  DATMOD2,w
movwf DATMOD            ;escribe el segundo dato en el MODEM
call  ESCRT
clrwdt
movlw .54
movwf ADDRESS
movf  DATMOD3,w
movwf DATMOD            ;escribe el tercer dato en el MODEM
call  ESCRT
goto  BUCLE
RESET bcf   PORTB,5           ;activa el reset
goto  BUCLE
FRESET      bsf   PORTB,5     ;desactiva el reset
goto  BUCLE
ENABLE bcf  PORTB,4           ;activa el MODEM
goto BUCLE
LECTU clrwdt
btfss   PIR1,RCIF       ;ejecución de una lectura en el MODEM
goto LECTU              ;captura dirección a leer
bcf   PIR1,RCIF
movf  RCREG,w
movwf ADDRESS
ESP   btfss   PIR1,RCIF       ;captura el modo
goto ESP
bcf   PIR1,RCIF
btfss   RCREG,0
goto  NORMAL
btfss   RCREG,1
goto  ESPEC
bsf   PORTE,1     ;termina el ciclo de lectura
bsf STATUS,RP0
clrf TRISD        ;se config de nuevo el puerto D como salida
bcf STATUS,RP0
bcf STATUS,RP1
goto BUCLE
NORMAL call LECT  ;lectura normal
;almacenándose en la variable DATMOD el valor del
; registro leído
goto BUCLE
ESPEC call LECT2              ;vuelca el registro en puerto D
goto BUCLE              ;al no completar el ciclo de lectura
ESCRITU clrwdt
btfss   PIR1,RCIF       ;instrucción de escritura
goto  ESCRITU
bcf   PIR1,RCIF
movf  RCREG,w           ; se espera a coger la dirección
movwf ADDRESS
```

```
ESCRITE     btfss   PIR1,RCIF       ;se espera a coger el dato
goto  ESCRITE
bcf   PIR1,RCIF
movf  RCREG,w
movwf DATMOD
call ESCRT              ; se escribe en el MODEM
goto BUCLE              ;ciclo de escritura normal
LETCON0 movlw     .4
movwf    ADDRESS
movlw       .1
movwf    DATMOD
call  ESCRT
movlw .0                ; se activa el modo continuo de lectura
; del canal 0
movwf    ADDRESS        ; se mete la dirección del registro , en
; este caso 0
movwf TEMPO             ; se habilitan las interrupciones  del
; MODEM , son  a nivel alto
bsf   FLAG,2            ;se activa la lectura continua L=1
goto BUCLE
LETCON1 movlw     .4
movwf    ADDRESS
movlw       .8
movwf    DATMOD
call  ESCRT
movlw .1                ; lo mismo para el canal 1
movwf    ADDRESS
movwf TEMPO
bsf   FLAG,2
goto BUCLE
FINCONT     bcf   FLAG,2      ; se desactiva modo continuo L=0
movlw       .4
movwf    ADDRESS
movlw       .0
      movwf    DATMOD         ;se desactivan las interrupciones en
; el MODEM
call  ESCRT
goto BUCLE
TRANS btfss       FLAG,1
goto LEER
TRANS1      btfss   PIR1,4          ;transmisión de datos por la USART
goto BUCLE              ;si esta el buffer vacío
                              ; y hay datos a tx en DATMOD
movf DATMOD,w           ;se mete el dato en el buffer de tx y se
; desactiva la bandera de FLAG
movwf TXREG
bcf FLAG,1
      goto BUCLE
LEER  btfss FLAG,2            ; si hay datos que leer del MODEM
goto  TRANST
btfss INTCON,INTF
goto BUCLE              ;y si hay interrupción del MODEM
bcf   INTCON,INTF
movf  TEMPO,W
movwf ADDRESS                 ;se lee el canal de datos del MODEM
call LECT   ;almacenándose en la variable DATMOD el valor del
; registro leído
goto BUCLE
TRANST      btfss INTCON,INTF       ;si hay interrupción por tx
goto BUCLE
bcf   INTCON,INTF             ;se envía un 3 por el puerto serie
```

```
movlw       .3
movwf TXREG

goto  BUCLE

;fin de transmisión de rafaga
TRANSF      movlw       .38         ;desactiva el transmisor
movwf    ADDRESS
call  LECT                   ;sin modificar la configuración
movf  DATMOD,W               ;del receptor ni del transmisor
movlw b'11111011'
andwf DATMOD,W
movwf    DATMOD
call  ESCRT
      goto  BUCLE
NULO  goto  BUCLE                   ;no hace nada , en reserva por si
; hiciera falta hacer una rutina adicional
end
```

```
LIST P=16F877
errorlevel -302
INCLUDE "P16F877.INC"   ;inclusión del archivo de direcciones

__CONFIG _BODEN_OFF & _CP_OFF & _PWRTE_OFF & _WDT_ON & _WRT_ENABLE_ON &
_HS_OSC & _DEBUG_OFF & _CPD_OFF & _LVP_OFF
;configuración para el compilador

LER   EQU 0x6F
DATMOD EQU 0x61
ADDRESS EQU 0x62
TEMPO EQU 0x60
TTOP  EQU 0x63
TBOT  EQU 0x64
RTOP  EQU 0x65
RBOT  EQU 0x66
```

```
DATPIL      EQU 0x67
PILATX      EQU 0x20
PILARX      EQU 0x40
FLAG  EQU 0x74                      ;reserva de espacio para registros
FRECUEN     EQU 0x77                ;del programa
SEGUR EQU 0x72
COPIA EQU 0x79
PORTM EQU 0x7A
PORTN EQU 0x7D
DATMOD2     EQU 0x7B
DATMOD3     EQU 0x7C
CONTRX      EQU 0x6A
CONTTX      EQU 0x6B
COMPW EQU 0x6C
ORG 0x00                ;fijo la dirección de reset
goto INICIO             ;dirección donde comienza a ejecutarse
; después del reset
ORG 0x04                ;fijo la dirección de interrupción
goto INTER              ;dirección de la rutina de interrupción
org 0x05                ;escribo después el programa

INTER movwf COPIA                   ;rutina de interrupción
movf  STATUS,W
movwf BANCO                 ;salva los registros
bcf  STATUS,RP0
btfss PIR1,TMR1IF           ;identifica la causa de la
; interrupción
goto INTER2
movf  PORTM,W
movwf TMR1L                 ;configura timer 1
movlw b'11111111'
movwf TMR1H
bcf PIR1,TMR1IF
btfss PORTB,6
goto CLAR
bcf PORTB,6       ;  cambio en la señal para generación de
;  frecuencia
FINAL movf  BANCO,W
movwf STATUS      ;devuelve los registros

movf  COPIA,W

retfie                      ;vuelve de la rutina
CLAR  bsf PORTB,6       ;  cambio en la señal para generación de
;  frecuencia
movf  BANCO,W
movwf STATUS      ;devuelve los registros
movf  COPIA,W
retfie            ;vuelve de la rutina
INTER2 bcf  INTCON,2
movf  PORTN,W
movwf TMR0              ;configura timer 0
btfss PORTB,7     ; cambio en la señal para generación de
;  frecuencia
goto CLAR2
bcf PORTB,7
movf  BANCO,W     ;devuelve los registros
movwf STATUS
movf  COPIA,W
retfie            ;vuelve de la rutina
CLAR2 bsf PORTB,7       ; cambio en la señal para generación de
```

```
;  frecuencia
movf  BANCO,W
movwf STATUS      ;devuelve los registros
movf  COPIA,W
retfie            ;vuelve de la rutina

LECT  movf ADDRESS,W          ;Función de lectura del MODEM
movwf PORTD
movlw .2          ;se pasa primero la dirección del registro
movwf PORTE
nop               ;al final en DATMOD queda el dato leído del
; MODEM
movlw .7          ;se sigue el protocolo de lectura
movwf PORTE
bsf STATUS,RP0          ;se cambia el Puerto D a lectura
movlw b'11111111'
movwf TRISD
bcf STATUS,RP0
movlw .5
movwf PORTE       ;se lee el dato
movf PORTD,W
movwf DATMOD
bsf  PORTE,1
bsf  FLAG,1       ;se activa la bandera de dato para tx
bsf STATUS,RP0
clrf TRISD        ;se configura  el puerto D como salida
bcf STATUS,RP0
bcf STATUS,RP1
return            ;se vuelve
LECT2 movf ADDRESS,W
movwf PORTD
movlw .2          ;se pasa primero la dirección del registro
movwf PORTE       ;siguiendo el protocolo de lectura
nop               ;al final en DATMOD queda el dato leído del
; modem
movlw .7
movwf PORTE
bsf STATUS,RP0
movlw b'11111111' ; se configura el puerto D como entrada
movwf TRISD
bcf STATUS,RP0
movlw .5          ;se interrumpe el protocolo de lectura
movwf PORTE       ;quedando la lectura del registro continua
return            ;se vuelve

ESCRT movf ADDRESS,W          ;función de escritura del MODEM
movwf PORTD
movlw .2
movwf PORTE             ;se pasa la dirección al MODEM
nop
movlw .7                ;cumpliendo con el protocolo de escritura
movwf PORTE
movf DATMOD,W           ;se pasa el valor a escribir  en el MODEM
movwf PORTD
bcf  PORTE,0
nop
nop
bsf  PORTE,0            ;se hace que el MODEM capture el dato
return                  ; se vuelve

COMP0 btfsc COMPW,0                ;compara el valor COMPW con 0
```

```
goto  FCMP                    ;bit a bit
btfsc COMPW,1
goto  FCMP                    ;al primero que no sea 0 ya
btfsc COMPW,2                 ;tenemos el resultado
goto  FCMP
btfsc COMPW,3
goto  FCMP
btfsc COMPW,4
goto  FCMP
btfsc COMPW,5
goto  FCMP
btfsc COMPW,6
goto  FCMP
btfsc COMPW,7
goto  FCMP
bsf   FLAG,7
return                  ;si es igual a 0 se pone un 1 en el bit C
FCMP  bcf   FLAG,7
return

MTX   movf  TTOP,W            ;mete datos en cola de transmisión
movwf FSR
movf  DATPIL,W          ;usa direccionamiento indirecto
movwf INDF
incf  FSR,W             ;después de meter el dato incrementa
btfss FSR,6
goto  FMT2
addlw 32                ;el índice de manera circular
andlw 63                ;así no nos salimos de la cola
FMT2  movwf TTOP
incf  CONTTX,1          ;incrementa el contador de datos en cola
return
MRX   movf  RTOP,W            ;mete datos en cola de recepción
movwf FSR
movf  DATPIL,W
movwf INDF              ;usa direccionamiento indirecto
incf  FSR,w             ;después de meter el dato incrementa
btfss FSR,5
goto  FMR
andlw 64                ;el índice de manera circular
;así no nos salimos de la cola
FMR   movwf
incf  CONTRX,1          ;incrementa el contador de datos en cola
return
STX   movf  TBOT,W            ;saca dato de la cola de transmisión
movwf FSR               ;usando direccionamiento indirecto
movf  INDF,W
movwf DATPIL            ;el dato lo mete en DATPIL
incf  FSR,W             ;incrementa el indice de manera circular
btfss FSR,6
goto  FMT
addlw 32                ;el indice de manera circular
andlw 63                ;así no nos salimos de la cola
FMT   movwf TBOT        ;decrementa el contador de datos en cola
decf  CONTTX,1
return
SRX   movf  RBOT,W            ;saca dato de la cola de recepción
movwf FSR               ;usando direccionamiento indirecto
movf  INDF,W
movwf DATPIL            ;el dato lo mete en DATPIL
incf  FSR,W             ;incrementa el indice de manera circular
```

```
btfss FSR,5
goto  FMR2
andlw 64                ;el indice de manera circular
;así no nos salimos de la cola
FMR2  movwf RBOT
bcf   FLAG,1            ;desactivamos la lectura R=0
decf  CONTRX,1          ;decrementa el contador de datos en cola
return

INICIO      clrf FLAG         ;todas las banderas a 0
clrf PORTB
movlw b'01000000'
movwf RTOP              ;csn a nivel alto , MODEM no seleccionado
movwf RBOT
movlw b'00100000'
movwf TTOP
      movwf TBOT
clrf PORTC
clrf PORTD


clrf PORTE
bsf  STATUS,RP0
movlw .6
movwf ADCON1            ;configuración de los registros
clrf  CONTTX            ;para el correcto funcionamiento
clrf  CONTRX            ;del programa
clrf  TRISD
clrf  TRISE
clrf  TRISB
bsf   TRISB,0
bcf   TRISC,6
bcf   TRISC,1
bcf   TRISC,2
bsf   TXSTA,5
bsf   TXSTA,2           ;USART sin inter , se opera por banderas
bsf   SPBRG,0
bsf   SPBRG,7           ; funciona asíncrona a 9,6 kb
bcf   OPTION_REG,5

bcf  STATUS,RP0
bcf  PORTC,5            ;modos tests desactivados
bcf   PORTC,4
bsf   INTCON,7          ;interrupciones solo de periféricos
bsf   RCSTA,7
bsf   RCSTA,4           ;configuración de la USART
bsf   INTCON,6

bsf   PORTB,5
bsf   PORTB,4           ;reset y enable desactivados
movlw .7
movwf PORTE
BUCLE clrwdt                        ;bucle principal
btfss   PIR1,RCIF
goto TRANS        ; lo primero q se recibe es una instrucción
bcf   PIR1,RCIF   ; según esta se hacen varias cosas
movf  RCREG,W
movwf LER
btfss   LER,0           ; se procede a identificar la instrucción
goto  LECTU       ; dentro del set 1
btfss   LER,1
```

```
goto  ESCRITU
btfss   LER,2
goto  LETCON0
btfss   LER,3
goto  LETCON1
btfss   LER,4
goto  FINCONT
btfss   LER,5
goto  TEST1
btfss   LER,6
goto  SET2              ;se busca en el set 2
btfss LER,7
goto  ENABLE
bcf   PORTC,5
bcf   PORTC,4           ;acabar con el modo test  todo a 1
goto  BUCLE
TEST1 bsf   PORTC,4

goto BUCLE
SET2  clrwdt
btfss   PIR1,RCIF
goto SET2
bcf   PIR1,RCIF
movf  RCREG,W
movwf LER
btfss   LER,0           ; identificación de instrucción
goto  FREC0             ;entre el set 2
btfss   LER,1
goto  FREC1
btfss   LER,2
goto  FRECP
btfss   LER,3
goto  FREC3
btfss   LER,4
goto  RESET
btfss   LER,5
goto  TRANSF
btfss   LER,6
goto  NULO
btfss   LER,7
goto  TESESC
goto  FRESET
TESESC      bcf   PORTC,5
goto  BUCLE
FREC0 clrwdt
btfss   PIR1,RCIF       ;frecuencia canal 0
goto  FREC0
bcf   PIR1,RCIF         ;captura el dato para el MODEM
movf  RCREG,w
movwf DATMOD
FREC01      clrwdt
btfss   PIR1,RCIF
goto  FREC01                ;captura la frecuencia
bcf   PIR1,RCIF
movf  RCREG,w
movwf PORTN
FREC02      clrwdt
btfss   PIR1,RCIF
goto  FREC02                ;captura el modo
bcf   PIR1,RCIF
btfss   RCREG,0
```

```
goto  CF0
movf  PORTN,W          ;configura y activa el timer 0
movwf TMR0
      bsf   INTCON,5
CF0   movlw .52              ;escribe el dato en el MODEM
movwf ADDRESS
call  ESCRT
goto  BUCLE
FREC1 clrwdt
btfss   PIR1,RCIF      ;frecuencia canal 1
goto  FREC1
bcf   PIR1,RCIF        ;captura del dato para el MODEM
movf  RCREG,w
movwf DATMOD
FREC11      btfss   PIR1,RCIF
goto  FREC11
bcf   PIR1,RCIF              ;captura del periodo de la señal
movf  RCREG,w
movwf FRECUEN
FREC12      clrwdt
btfss   PIR1,RCIF
goto  FREC12                 ;captura del modo
bcf   PIR1,RCIF
btfss   RCREG,0
goto  CF1
movf  FRECUEN,w
bsf   STATUS,RP0
movwf PR2
bcf   STATUS,RP0
incf  FRECUEN,1        ;configuración y activación del PWM
rlf   FRECUEN,1
bcf   FRECUEN,0        ;a periodo suministrado
btfsc FRECUEN,1
goto  OTRO
movlw b'00001100'
movwf CCP2CON
goto  SIGUE
OTRO  movlw b'00101100'
movwf CCP2CON
SIGUE rrf   FRECUEN,1        ;configuración del ciclo activo
rrf   FRECUEN,1        ;en el PWM
movlw b'00111111'      ;a partir del dato suministrado
andwf FRECUEN,W
movwf CCPR2L
bsf   STATUS,RP0
bcf   TRISC,1
bcf   STATUS,RP0
movlw b'00000100'
movwf T2CON
bsf   CCP2CON,3
bsf   CCP2CON,2
CF1   movlw .53
movwf ADDRESS
call ESCRT             ; se escribe en el MODEM la frecuencia
goto BUCLE

FRECP clrwdt
btfss   PIR1,RCIF      ;frecuencia piloto
goto  FRECP
bcf   PIR1,RCIF        ;se captura el dato para el MODEM
movf  RCREG,w
```

```
movwf DATMOD
FRECP1      clrwdt
btfss   PIR1,RCIF
goto  FRECP1      ; se captura la frecuencia de generación
bcf   PIR1,RCIF
movf  RCREG,w           ;se configura el timer 1
movwf TMR1L
movwf PORTM
movlw b'11111111'
movfw TMR1H
FRECP2      clrwdt
btfss   PIR1,RCIF
goto  FRECP2                ;se captura el modo
bcf   PIR1,RCIF
btfss   RCREG,0
goto  CFP
bsf   T1CON,0
bsf   STATUS,RP0        ;se activa el timer 1
bsf   PIE1,0
bcf   STATUS,RP0
CFP   movlw .54
movwf ADDRESS
call ESCRT              ; se escribe en el MODEM la frecuencia
goto BUCLE
FREC3 clrwdt            ;frecuencia total
btfss   PIR1,RCIF
goto  FREC3             ;se captura el primer dato
bcf   PIR1,RCIF
movf  RCREG,w
movwf DATMOD
FREC31      clrwdt
btfss   PIR1,RCIF
goto  FREC31      ; se captura el segundo dato
bcf   PIR1,RCIF
movf  RCREG,w
movwf DATMOD2
FREC32      clrwdt
btfss   PIR1,RCIF
goto  FREC32      ; se captura el tercer dato
bcf   PIR1,RCIF
movf  RCREG,w
movwf DATMOD3
CFT   clrwdt
movlw .52         ;se escribe el primer dato
movwf ADDRESS
call  ESCRT
clrwdt
movlw .53
movwf ADDRESS
movf  DATMOD2,w         ;se escribe el segundo dato en el MODEM
movwf DATMOD
call  ESCRT
clrwdt
movlw .54
movwf ADDRESS
movf  DATMOD3,w         ;se escribe el tercer dato en el MODEM
movwf DATMOD
call  ESCRT
goto  BUCLE
RESET bcf   PORTB,5           ;se activa el reset
goto  BUCLE
```

```
FRESET      bsf   PORTB,5     ;se desactiva el reset
goto  BUCLE
ENABLE      bcf   PORTB,4     ;se activa el MODEM
goto BUCLE
LECTU clrwdt
btfss   PIR1,RCIF       ;instrucción de lectura
goto LECTU
bcf   PIR1,RCIF
movf  RCREG,w
movwf ADDRESS
ESP   btfss   PIR1,RCIF
goto ESP          ; se captura la dirección
bcf   PIR1,RCIF
btfss   RCREG,0         ; se captura el modo
goto  NORMAL
btfss   RCREG,1
goto  ESPEC             ;se termina el ciclo de lectura
bsf   PORTE,1
bsf STATUS,RP0
clrf TRISD        ;se config de nuevo el puerto D como salida
bcf STATUS,RP0
bcf STATUS,RP1
goto BUCLE
NORMAL call LECT        ;lectura normal del registro
goto BUCLE
ESPEC call LECT2        ;se realiza el volcado en el puerto D
goto BUCLE        ;al no acabar el ciclo de lectura
ESCRITU     clrwdt            ;escritura
btfss   PIR1,RCIF
goto  ESCRITU                 ;se captura la dirección
bcf   PIR1,RCIF
movf  RCREG,w
movwf ADDRESS
ESCRITE     btfss   PIR1,RCIF       ;se captura el dato
goto  ESCRITE
bcf   PIR1,RCIF
movf  RCREG,w
movwf DATMOD
ESCRIT2     btfss   PIR1,RCIF       ;se captura el modo
goto  ESCRIT2
bcf   PIR1,RCIF
btfss RCREG,0
goto  ESSIG
btfss CONTTX,5          ;cola llena?
goto  METP
goto  BUCLE
METP  movf  DATMOD,w
movwf DATPIL
call  MTX               ;se mete data en cola
goto  BUCLE
ESSIG call ESCRT              ; se escribe en el MODEM
goto BUCLE
LETCON0 movlw     .4
movwf    ADDRESS
movlw       .1
movwf    DATMOD
call  ESCRT
movlw .0      ; se activa el modo continuo de lectura del canal 0
movwf    ADDRESS        ; se mete la dirección del registro
movwf TEMPO             ;a leer con cada interrupción
bsf   FLAG,2            ;se activa L ,L=1
```

```
goto BUCLE
LETCON1 movlw     .4
movwf    ADDRESS
movlw       .8
movwf    DATMOD
call  ESCRT
movlw .1               ; lo mismo para el canal 1
movwf    ADDRESS

movwf TEMPO
bsf   FLAG,2
goto BUCLE
FINCONT     bcf   FLAG,2      ;fin de captura de datos del canal de
; recepción del MODEM
movlw       .4
movwf    ADDRESS        ;se deshabilitan las interrupciones
movlw       .0          ;en el modem
movwf    DATMOD
call  ESCRT
bsf   FLAG,3      ;se activa el bit V ,V=1, para vaciar cola
TRX   movf  CONTRX,W          ;bloque funcional de vaciado de cola
call COMP0
btfsc FLAG,7
goto  VCIA        ;cola vacia?
call  SRX         ;saca dato de cola recepción
TXESP btfss   PIR1,4
goto  TXESP
movf  DATPIL,W
movwf TXREG             ;transmite el dato por puerto serie
goto BUCLE
VCIA  bcf   FLAG,3            ;desactiva V ,V=0
goto BUCLE

TRANS btfss       FLAG,1
goto LEER
TRANS1      btfss   PIR1,4          ;transmisión de datos por la USART
goto BUCLE              ;si esta el buffer vacío
; y hay datos a tx en DATMOD
movf DATMOD,w
movwf TXREG
bcf FLAG,1              ; se desactiva la bandera de FLAG
goto BUCLE
LEER  btfss FLAG,2            ;L=1?
goto  TRANST
btfss INTCON,INTF       ;hay interrupción?
goto  BUCLE
bcf   INTCON,INTF
movf  TEMPO,W
movwf ADDRESS           ;se lee el canal de recepción del MODEM
call LECT               ;almacenándose el dato en DATMOD
bcf FLAG,1              ;se desactiva R , R=0
movf  DATMOD,W
movwf DATPIL
call  MRX               ;se mete el dato en la cola
btfsc CONTRX,5
goto  TRX         ;si esta medio llena la cola se ejecuta TRX
goto  BUCLE
TRANST btfss INTCON,INTF      ;hay interrupción?
goto VACIA
bcf   INTCON,INTF
movf  CONTTX,W          ;cola de transmisión vacia?
```

```
movwf COMPW
      call COMP0
btfsc FLAG,7
goto  TRANSF
call  STX               ;se saca un dato a transmitir
movlw       .43
movwf    ADDRESS
movf  DATPIL,W
movwf DATMOD            ;se escribe en el registro del MODEM
call  ESCRT             ;para transmitirlo
goto BUCLE
TRANSF      movlw       .38   ;si la cola esta vacia se apaga el tx
movwf    ADDRESS
call  LECT
movlw b'11111011'
andwf DATMOD,1
call  ESCRT
goto  BUCLE
NULO  goto  BUCLE
VACIA btfss FLAG,3      ;si hay datos en la cola de recepción se sacan
goto  BUCLE
goto  TRX         ; y transmiten(TRX)

end
```

```
LIST
; P16F877.INC  Standard Header File, Version 1.00    Microchip Technology,
Inc.
NOLIST

; This header file defines configurations, registers, and other useful bits of
; information for the PIC16F877 microcontroller.  These names are taken to
match
; the data sheets as closely as possible.

; Note that the processor must be selected before this file is
; included.  The processor may be selected the following ways:

;       1. Command line switch:
;               C:\ MPASM MYFILE.ASM /PIC16F877
;       2. LIST directive in the source file
;               LIST   P=PIC16F877
;       3. Processor Type entry in the MPASM full-screen interface

;==========================================================================
;
;       Revision History
;
;==========================================================================
```

```
;Rev:   Date:    Reason:

;1.12   01/12/00 Changed some bit names, a register name, configuration bits
;                 to match datasheet (DS30292B)
;1.00   08/07/98 Initial Release

;==========================================================================
;
;       Verify Processor
;
;==========================================================================

IFNDEF __16F877
MESSG "Processor-header file mismatch.  Verify selected processor."
ENDIF

;==========================================================================
;
;       Register Definitions
;
;==========================================================================

W                           EQU     H'0000'
F                           EQU     H'0001'

;----- Register Files------------------------------------------------------

INDF                        EQU     H'0000'
TMR0                        EQU     H'0001'
PCL                         EQU     H'0002'
STATUS                      EQU     H'0003'
FSR                         EQU     H'0004'
PORTA                       EQU     H'0005'
PORTB                       EQU     H'0006'
PORTC                       EQU     H'0007'
PORTD                       EQU     H'0008'
PORTE                       EQU     H'0009'
PCLATH                      EQU     H'000A'
INTCON                      EQU     H'000B'
PIR1                        EQU     H'000C'
PIR2                        EQU     H'000D'
TMR1L                       EQU     H'000E'
TMR1H                       EQU     H'000F'
T1CON                       EQU     H'0010'
TMR2                        EQU     H'0011'
T2CON                       EQU     H'0012'
SSPBUF                      EQU     H'0013'
SSPCON                      EQU     H'0014'
CCPR1L                      EQU     H'0015'
CCPR1H                      EQU     H'0016'
CCP1CON                     EQU     H'0017'
RCSTA                       EQU     H'0018'
TXREG                       EQU     H'0019'
RCREG                       EQU     H'001A'
CCPR2L                      EQU     H'001B'
CCPR2H                      EQU     H'001C'
CCP2CON                     EQU     H'001D'
ADRESH                      EQU     H'001E'
ADCON0                      EQU     H'001F'
```

```
OPTION_REG                      EQU     H'0081'
TRISA                           EQU     H'0085'
TRISB                           EQU     H'0086'
TRISC                           EQU     H'0087'
TRISD                           EQU     H'0088'
TRISE                           EQU     H'0089'
PIE1                            EQU     H'008C'
PIE2                            EQU     H'008D'
PCON                            EQU     H'008E'
SSPCON2                         EQU     H'0091'
PR2                             EQU     H'0092'
SSPADD                          EQU     H'0093'
SSPSTAT                         EQU     H'0094'
TXSTA                           EQU     H'0098'
SPBRG                           EQU     H'0099'
ADRESL                          EQU     H'009E'
ADCON1                          EQU     H'009F'

EEDATA                          EQU     H'010C'
EEADR                           EQU     H'010D'
EEDATH                          EQU     H'010E'
EEADRH                          EQU     H'010F'

EECON1                          EQU     H'018C'
EECON2                          EQU     H'018D'

;----- STATUS Bits --------------------------------------------------------

IRP                             EQU     H'0007'
RP1                             EQU     H'0006'
RP0                             EQU     H'0005'
NOT_TO                          EQU     H'0004'
NOT_PD                          EQU     H'0003'
Z                               EQU     H'0002'
DC                              EQU     H'0001'
C                               EQU     H'0000'

;----- INTCON Bits --------------------------------------------------------

GIE                             EQU     H'0007'
PEIE                            EQU     H'0006'
T0IE                            EQU     H'0005'
INTE                            EQU     H'0004'
RBIE                            EQU     H'0003'
T0IF                            EQU     H'0002'
INTF                            EQU     H'0001'
RBIF                            EQU     H'0000'

;----- PIR1 Bits ----------------------------------------------------------

PSPIF                           EQU     H'0007'
ADIF                            EQU     H'0006'
RCIF                            EQU     H'0005'
TXIF                            EQU     H'0004'
SSPIF                           EQU     H'0003'
CCP1IF                          EQU     H'0002'
TMR2IF                          EQU     H'0001'
TMR1IF                          EQU     H'0000'

;----- PIR2 Bits ----------------------------------------------------------
```

```
EEIF                            EQU     H'0004'
BCLIF                           EQU     H'0003'
CCP2IF                          EQU     H'0000'

;----- T1CON Bits --------------------------------------------------------

T1CKPS1                         EQU     H'0005'
T1CKPS0                         EQU     H'0004'
T1OSCEN                         EQU     H'0003'
NOT_T1SYNC                      EQU     H'0002'
T1INSYNC                        EQU     H'0002'    ; Backward compatibility only
T1SYNC                          EQU     H'0002'
TMR1CS                          EQU     H'0001'
TMR1ON                          EQU     H'0000'

;----- T2CON Bits --------------------------------------------------------

TOUTPS3                         EQU     H'0006'
TOUTPS2                         EQU     H'0005'
TOUTPS1                         EQU     H'0004'
TOUTPS0                         EQU     H'0003'
TMR2ON                          EQU     H'0002'
T2CKPS1                         EQU     H'0001'
T2CKPS0                         EQU     H'0000'

;----- SSPCON Bits -------------------------------------------------------

WCOL                            EQU     H'0007'
SSPOV                           EQU     H'0006'
SSPEN                           EQU     H'0005'
CKP                             EQU     H'0004'
SSPM3                           EQU     H'0003'
SSPM2                           EQU     H'0002'
SSPM1                           EQU     H'0001'
SSPM0                           EQU     H'0000'

;----- CCP1CON Bits ------------------------------------------------------

CCP1X                           EQU     H'0005'
CCP1Y                           EQU     H'0004'
CCP1M3                          EQU     H'0003'
CCP1M2                          EQU     H'0002'
CCP1M1                          EQU     H'0001'
CCP1M0                          EQU     H'0000'

;----- RCSTA Bits --------------------------------------------------------

SPEN                            EQU     H'0007'
RX9                             EQU     H'0006'
RC9                             EQU     H'0006'    ; Backward compatibility only
NOT_RC8                         EQU     H'0006'    ; Backward compatibility only
RC8_9                           EQU     H'0006'    ; Backward compatibility only
SREN                            EQU     H'0005'
CREN                            EQU     H'0004'
ADDEN                           EQU     H'0003'
FERR                            EQU     H'0002'
OERR                            EQU     H'0001'
RX9D                            EQU     H'0000'
RCD8                            EQU     H'0000'    ; Backward compatibility only

;----- CCP2CON Bits ------------------------------------------------------
```

```
CCP2X                           EQU     H'0005'
CCP2Y                           EQU     H'0004'
CCP2M3                          EQU     H'0003'
CCP2M2                          EQU     H'0002'
CCP2M1                          EQU     H'0001'
CCP2M0                          EQU     H'0000'

;----- ADCON0 Bits --------------------------------------------------------

ADCS1                           EQU     H'0007'
ADCS0                           EQU     H'0006'
CHS2                            EQU     H'0005'
CHS1                            EQU     H'0004'
CHS0                            EQU     H'0003'
GO                              EQU     H'0002'
NOT_DONE                        EQU     H'0002'
GO_DONE                         EQU     H'0002'
ADON                            EQU     H'0000'

;----- OPTION_REG Bits ----------------------------------------------------

NOT_RBPU                        EQU     H'0007'
INTEDG                          EQU     H'0006'
T0CS                            EQU     H'0005'
T0SE                            EQU     H'0004'
PSA                             EQU     H'0003'
PS2                             EQU     H'0002'
PS1                             EQU     H'0001'
PS0                             EQU     H'0000'

;----- TRISE Bits ---------------------------------------------------------

IBF                             EQU     H'0007'

OBF                             EQU     H'0006'
IBOV                            EQU     H'0005'
PSPMODE                         EQU     H'0004'
TRISE2                          EQU     H'0002'
TRISE1                          EQU     H'0001'
TRISE0                          EQU     H'0000'

;----- PIE1 Bits ----------------------------------------------------------

PSPIE                           EQU     H'0007'
ADIE                            EQU     H'0006'
RCIE                            EQU     H'0005'
TXIE                            EQU     H'0004'
SSPIE                           EQU     H'0003'
CCP1IE                          EQU     H'0002'
TMR2IE                          EQU     H'0001'
TMR1IE                          EQU     H'0000'

;----- PIE2 Bits ----------------------------------------------------------

EEIE                            EQU     H'0004'
BCLIE                           EQU     H'0003'
CCP2IE                          EQU     H'0000'

;----- PCON Bits ----------------------------------------------------------
```

```
NOT_POR                         EQU     H'0001'
NOT_BO                          EQU     H'0000'
NOT_BOR                         EQU     H'0000'

;----- SSPCON2 Bits --------------------------------------------------

GCEN                            EQU     H'0007'
ACKSTAT                         EQU     H'0006'
ACKDT                           EQU     H'0005'
ACKEN                           EQU     H'0004'
RCEN                            EQU     H'0003'
PEN                             EQU     H'0002'
RSEN                            EQU     H'0001'
SEN                             EQU     H'0000'

;----- SSPSTAT Bits --------------------------------------------------

SMP                             EQU     H'0007'
CKE                             EQU     H'0006'

D                               EQU     H'0005'
I2C_DATA                        EQU     H'0005'
NOT_A                           EQU     H'0005'
NOT_ADDRESS                     EQU     H'0005'
D_A                             EQU     H'0005'
DATA_ADDRESS                    EQU     H'0005'
P                               EQU     H'0004'
I2C_STOP                        EQU     H'0004'
S                               EQU     H'0003'
I2C_START                       EQU     H'0003'
R                               EQU     H'0002'
I2C_READ                        EQU     H'0002'
NOT_W                           EQU     H'0002'
NOT_WRITE                       EQU     H'0002'
R_W                             EQU     H'0002'
READ_WRITE                      EQU     H'0002'
UA                              EQU     H'0001'
BF                              EQU     H'0000'

;----- TXSTA Bits ----------------------------------------------------

CSRC                            EQU     H'0007'
TX9                             EQU     H'0006'
NOT_TX8                         EQU     H'0006'    ; Backward compatibility only
TX8_9                           EQU     H'0006'    ; Backward compatibility only
TXEN                            EQU     H'0005'
SYNC                            EQU     H'0004'
BRGH                            EQU     H'0002'
TRMT                            EQU     H'0001'
TX9D                            EQU     H'0000'
TXD8                            EQU     H'0000'    ; Backward compatibility only

;----- ADCON1 Bits ---------------------------------------------------

ADFM                            EQU     H'0007'
PCFG3                           EQU     H'0003'
PCFG2                           EQU     H'0002'
PCFG1                           EQU     H'0001'
PCFG0                           EQU     H'0000'

;----- EECON1 Bits ---------------------------------------------------
```

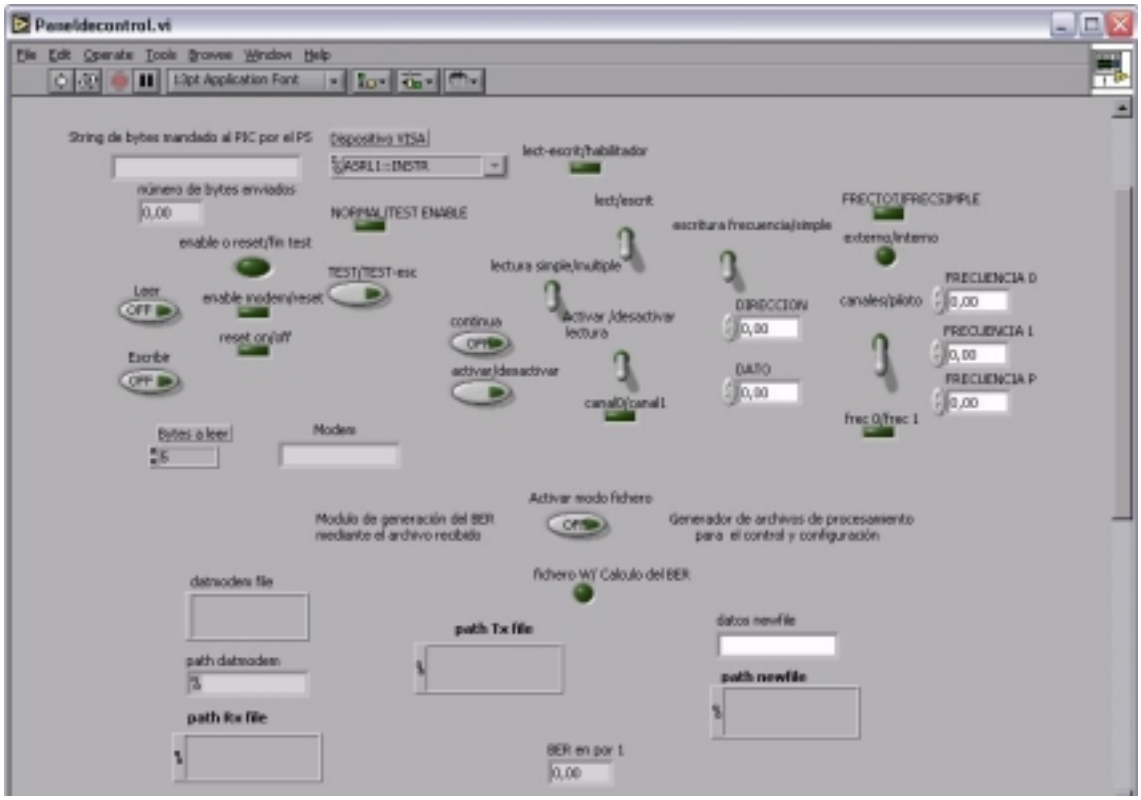
```
EEPGD                           EQU     H'0007'
WRERR                           EQU     H'0003'
WREN                            EQU     H'0002'
WR                              EQU     H'0001'
RD                              EQU     H'0000'


TOIF              EQU   H'0002'
;==========================================================================
;
;       RAM Definition
;
;==========================================================================

__MAXRAM H'1FF'
__BADRAM H'8F'-H'90', H'95'-H'97', H'9A'-H'9D'
__BADRAM H'105', H'107'-H'109'
__BADRAM H'185', H'187'-H'189', H'18E'-H'18F'

;==========================================================================
;
;       Configuration Bits
;
;==========================================================================

_CP_ALL                         EQU     H'0FCF'
_CP_HALF                        EQU     H'1FDF'
_CP_UPPER_256                   EQU     H'2FEF'
_CP_OFF                         EQU     H'3FFF'
_DEBUG_ON                       EQU     H'37FF'
_DEBUG_OFF                      EQU     H'3FFF'
_WRT_ENABLE_ON                  EQU     H'3FFF'
_WRT_ENABLE_OFF                 EQU     H'3DFF'
_CPD_ON                         EQU     H'3EFF'
_CPD_OFF                        EQU     H'3FFF'
_LVP_ON                         EQU     H'3FFF'
_LVP_OFF                        EQU     H'3F7F'
_BODEN_ON                       EQU     H'3FFF'
_BODEN_OFF                      EQU     H'3FBF'
_PWRTE_OFF                      EQU     H'3FFF'
_PWRTE_ON                       EQU     H'3FF7'
_WDT_ON                         EQU     H'3FFF'
_WDT_OFF                        EQU     H'3FFB'
_LP_OSC                         EQU     H'3FFC'
_XT_OSC                         EQU     H'3FFD'
_HS_OSC                         EQU     H'3FFE'
_RC_OSC                         EQU     H'3FFF'

LIST
```

5. APLICACIONES EN LABVIEW

Mediante el uso de estas aplicaciones podremos gobernar el PIC previamente programado, permitiendo un fácil manejo de la tarjeta controladora desde un PC.

5.1 PANEL DE CONTROL

Esta herramienta básica nos proporciona un control paso a paso del MODEM mediante la ejecución de instrucciones de manera secuencial sobre el controlador, pudiendo tanto activar las señales básicas de este como la fácil lectura o escritura de cualquiera de sus registros.



Panel de control configurable

5.1.1 Funciones y modos de trabajo

Esta aplicación nos permite dos modos de trabajo compatibles (se pueden ejecutar hasta dos funciones de una misma ejecución, siendo una de cada tipo):

- Modo control: En este modo operamos directamente sobre el MODEM.
- Modo fichero: En este modo realizamos operaciones con archivos de datos recibidos del MODEM o creamos ficheros para configurarlo.

Las funciones que nos permite realizar esta aplicación son:

- En modo control:

    - Habilitar MODEM.
    - Resetear MODEM(activando el reset y luego desactivándolo).
    - Escribir en registro del MODEM.
    - Leer de registro del MODEM.
    - Volcar un registro del MODEM en el puerto D del PIC para su lectura continua (usando osciloscopio), y su desactivación.
    - Leer un canal de recepción del MODEM de manera continuada(siempre que le lleguen datos).
    - Generar frecuencias externas para las referencias del MODEM.
    - Configurar las frecuencias del transmisor del MODEM.
    - Activar , desactivar los modos Test del MODEM.
    - Escribir en cola de transmisión *

- En modo fichero:
    - Crear archivo de instrucciones para configurar el MODEM , o datos.
    - Calcular el BER de un archivo recibido disponiendo de una copia original de lo transmitido.

* Solo en la segunda versión , utilizada en el 2º modo de programación del PIC, en el que están implementadas las colas de transmisión y de recepción.

5.1.2 Diagrama del programa

Al tratarse de la aplicación más compleja analizaremos su estructura por bloques que luego se detallaran , poniendo especial atención al bloque de creación de string de instrucción .

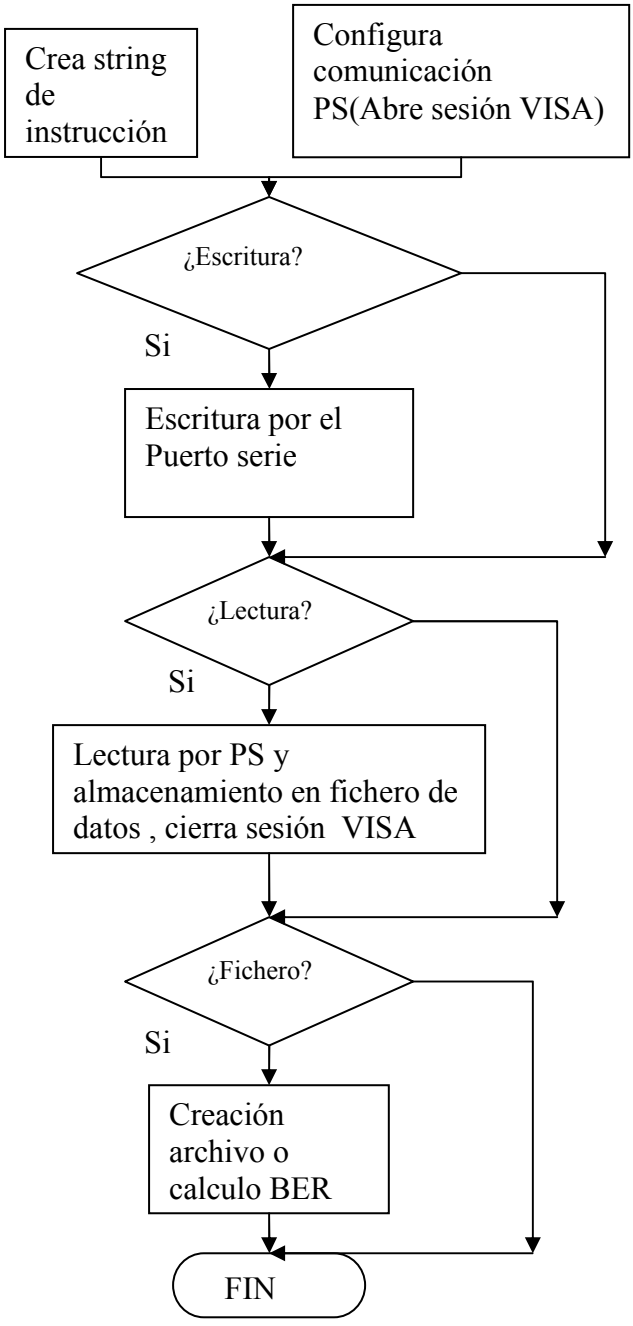


Diagrama funcional del programa

5.1.3 Descripción funcional

Esta aplicación al igual que el resto de aplicaciones desarrolladas en este proyecto que llevan a cabo comunicación por el puerto serie del PC utiliza sesiones VISA para una fácil comunicación , así pues lo primero que hace es abrir una sesión y configurarla a 9600 bps , asíncrona ,8 bits de datos ,sin paridad y con 1 bit de stop que además de ser los valores por defecto es la configuración que usaremos en todas las aplicaciones ya que el PIC así lo hemos configurado dado que es el valor para el que nos da menor error en la tasa de bits.Al mismo tiempo se ira componiendo la cadena de bytes de instrucción en función de las entradas que hayamos puesto en los controles lógicos y los parámetros que queramos pasarle al PIC(dirección , dato , modo).
Si se solicito la escritura (activa la escritura hacia el PIC), se mandaran por el puerto serie estos bytes de la cadena , empezando por el que identifica la instrucción .
Del mismo modo si se solicito la lectura (del PIC al PC) , se capturaran los datos requeridos(lectura de un registro del MODEM) diseccionándose esos datos hacia un indicador en nuestra pantalla y hacia un archivo en el cual se almacenaran para posterior uso.
Por ultimo si se solicito hacer uso de los modos de fichero disponibles se realizaran las operaciones pertinentes , ya sea crear o modificar un archivo de configuración como calcular la BER de los archivos seleccionados.

5.1.3.1Diagrama del bloque de generación de instrucción

Este bloque a pesar de parecer complejo solo es un anidamiento de estructuras If , gobernadas por controladores lógicos , de manera que en función de la combinación de entradas lógicas obtenemos una instrucción u otra , siendo necesarios los parámetros pertinentes(dato, dirección) en el caso que convenga.
En el diagrama se detalla la instrucción como su valor en hexadecimal y los parámetros donde son necesarios , el parámetro de modo lo genera el mismo , siendo * un valor que no importa del parámetro que corresponda (dato o dirección), ya que no sirve a la hora de ejecutar luego la instrucción en el PIC.

5.1.3.2 Diagrama del bloque de modo fichero

Este bloque es bastante simple , se reduce a realizar las funciones de modo fichero , crea o modifica un archivo , escribiendo en hexadecimal para su posterior carga por el puerto serie en el PIC , o realiza el calculo de tasa de error de bit sobre un archivo de datos recibidos por el MODEM y otro en el que se encuentran los datos realmente transmitidos por este.

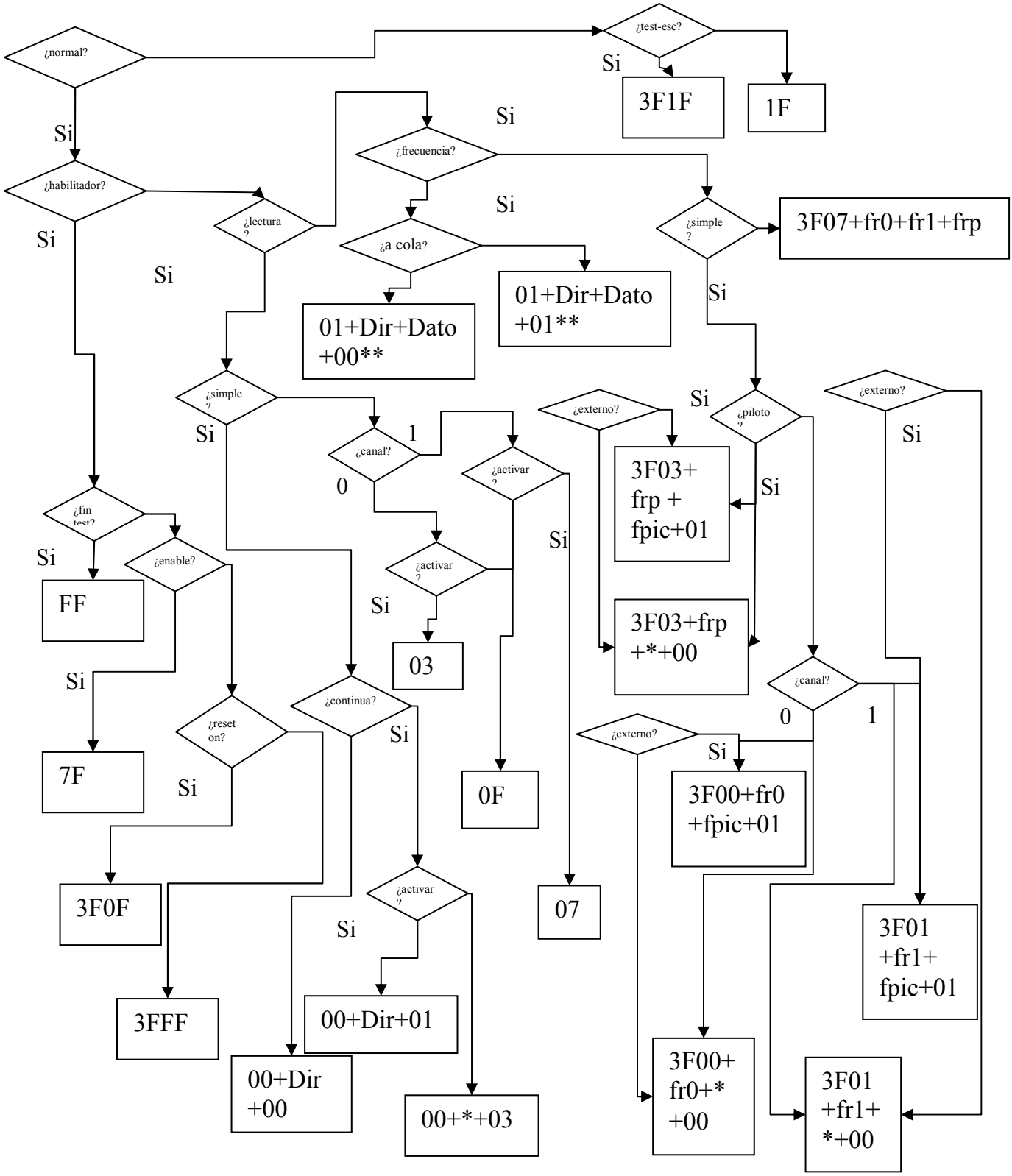


Diagrama de generación de instrucciones

* No importa el valor

** Solo en el 2º modo de programación del PIC.

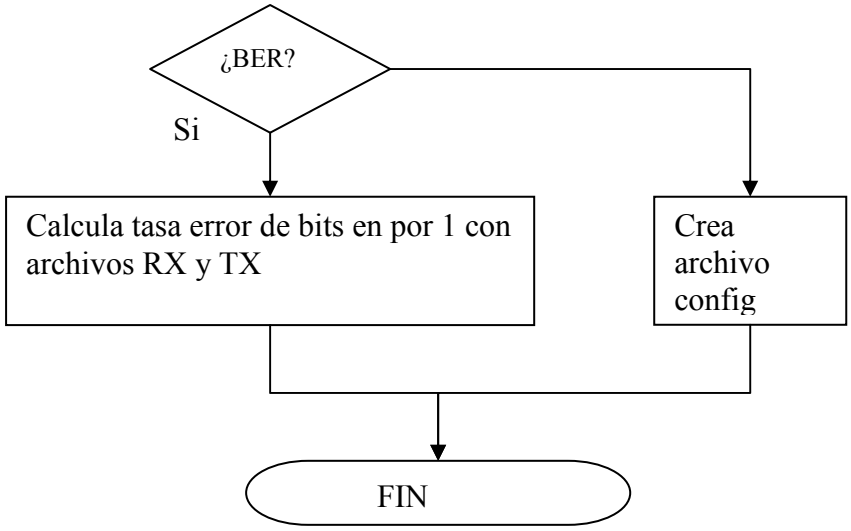


Diagrama de modo fichero

5.1.4 Parámetros del programa: controles e indicadores

Esta aplicación dispone de los siguientes controles e indicadores dada la versatilidad de la misma , los dividiremos entre los lógicos(interruptores) que nos permiten seleccionar el modo y función a realizar por la aplicación , y los no lógicos , que se encargan de los demás aspectos de la misma.

5.1.4.1 Lógicos

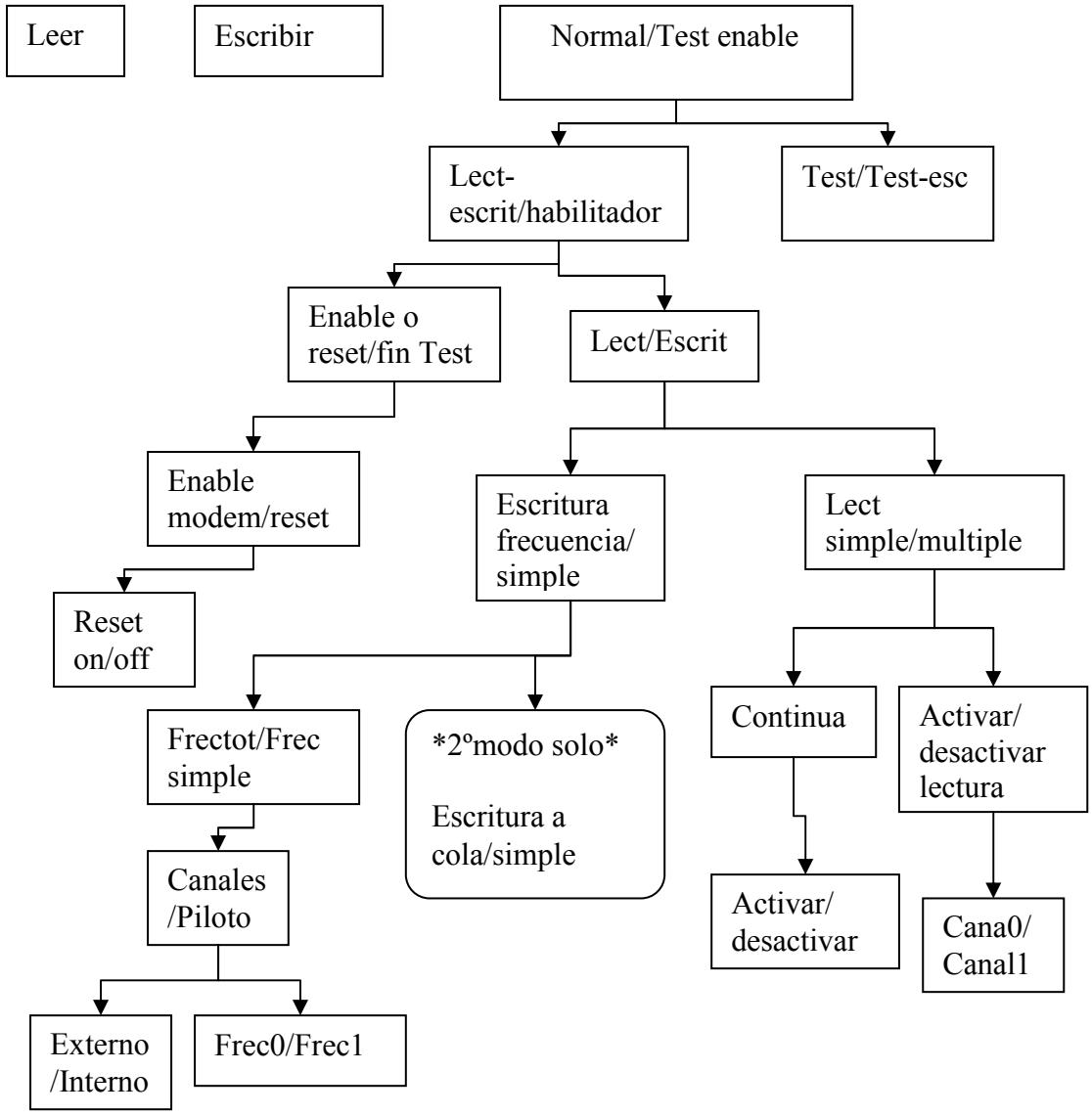
En la siguiente tabla podemos observar los interruptores necesarios para activar la función deseada del programa , estos deben situarse en la posición deseada antes de ejecutarla.

Nombre del control lógico	Tipo	Valor	Comentarios y funcionalidad
Normal/Test enable	Pulsador	On/off	Elige entre modo test y normal
Enable o reset/fin Test	Pulsador	On/off	Termina los test
Leer	Pulsador	On/off	Permite o prohíbe la lectura del PS
Escribir	Pulsador	On/off	Permite o prohíbe la escritura en el PS
Continua	Pulsador	On/off	Elige entre lectura simple de un registro y la opción de volcado en el puerto D del registro
Activar/desactivar	Pulsador	On/off	Activa o desactiva el volcado del registro al puerto D
Test/Test-esc	Pulsador	On/off	Elige entre los dos tipos de test

Lect-escrit/habilitador	Pulsador	On/off	Elige entre acceso a registro y habilitación de señales
Lect/escrit	Palanca	Arriba/abajo	Elige entre leer o escribir el registro del MODEM seleccionado por DIRECCION
Lect simple/múltiple	Palanca	Arriba/abajo	Elige entre el tipo de lectura simple y el que lee cada vez que llega un dato por el canal seleccionado del MODEM
Escritura frecuencia/simple	Palanca	Arriba/abajo	Selecciona el tipo de escritura a realizar , a los de frecuencia de Tx o a uno general
Frectot/frec simple	Pulsador	On/off	Elige entre si escribir todas las frecuencias o solo una de ellas
Externo/interno	Pulsador	On/off	Elige entre generar o no las frecuencias del PLL
Activar/desactivar lectura	Palanca	Arriba/abajo	Activa o desactiva la lectura continua de un canal de Rx del MODEM
Canal0/canal1	Pulsador	On/off	Selecciona el canal de Rx del MODEM
Canales/piloto	Palanca	Arriba/abajo	Selecciona entre las frecuencias a configurar
Frec0/frec1	Pulsador	On/off	Selecciona entre los canales de Tx
Enable MODEM/reset	Pulsador	On/off	Elige entre las señales
Reset on/off	Pulsador	On/off	Activa o desactiva el reset
Activar modo fichero	Pulsador	On/off	Activa el modo de operaciones con ficheros
Fichero W/calculo BER	Pulsador	On/off	Elige la operación a realizar con los ficheros
Escritura normal/a cola *	Pulsador	On/off	Elige el modo de escritura en cola de Tx o el normal

* Solo en la segunda versión, utilizada en el 2º modo de programación del PIC, en el que están implementadas las colas de transmisión y de recepción.

Vistos así no parece fácil conseguir distinguir que pulsador o que palanca hace falta para realizar determinada función en esta aplicación, por lo que los organizamos en estructura jerárquica, y posteriormente se vera en una tabla los interruptores necesarios para seleccionar las instrucciones (algunos interruptores, como los del modo fichero son fáciles de comprender y no necesitan de esto).



Esquema jerárquico de interruptores

Como se puede apreciar, los interruptores escribir y leer se refieren a la comunicación entre PC y PIC por el puerto serie, y deben habilitarse según proceda, ya que si después de generar una instrucción no la cargamos en el PIC, o si no leemos del puerto serie el valor que nos manda el PIC no podremos llevar a cabo con éxito la ejecución de la instrucción o no recibiremos sus consecuencias (datos leídos del MODEM).
En el caso que no se espere dato del MODEM leer debe estar apagado porque sino el programa seguirá esperando un dato que nunca le llega.
En el modo fichero solo hay 2 interruptores; 1 para saber si lo activamos, y el otro para saber que función deseamos realizar, por su simpleza se han omitido aquí .además no interfieren para nada con los otros.

5.1.4.2 No lógicos

Son los controles necesarios para configurar los parámetros de las instrucciones (dato,dirección), las rutas y nombres de los archivos que usamos(Paths) y el dispositivo serie de comunicaciones(COM1 ,COM2),pero también los indicadores de lo que esta sucediendo en el programa(el string que le mandamos al PIC , el que nos llega del PIC, la BER calculada).

Nombre del parámetro	Tipo	Valor	Comentarios y funcionalidad
Dispositivo VISA	control	ASRLi:: INSTR con i=1,2	Tipo de dispositivo para la sesión de comunicaciones, usaremos ASRL1 y ASRL2 Refiriéndonos a los puertos COM1 y COM2
Bytes a leer	control	Entero	Número de bytes que se desean recibir del PIC provenientes del MODEM
DIRECCION	control	00-82(Hex)	Dirección del registro a leer o escribir del MODEM
DATO	control	00-FF(Hex)	Dato a escribir en registro del MODEM
FRECUENCIA 0	control	20-145(KHz)	Frecuencia del canal 0 para Tx del MODEM Para generación solo acepta 20-34,5(KHz)
FRECUENCIA 1	control	20-145(KHz)	Frecuencia del canal 1 para Tx del MODEM
FRECUENCIA P	control	20-145(KHz)	Frecuencia del canal P para Tx del MODEM Para generación solo acepta 20-31(KHz)
Número de bytes enviados	indicador	Entero	Nos permite saber el número de bytes de la ultima instrucción enviada al PIC
String de bytes enviados al PIC por el PS	indicador	Cadena hex	Nos permite saber la ultima instrucción enviada al PIC incluidos los parámetros
MODEM	indicador	Cadena hex	Ultima cadena de bytes (en hexadecimal)recibida
Datmodem file	control	Nombre de archivo	Nombre del archivo donde se almacenan los datos recibidos
Path datmodem	control	Path normal	Path del archivo donde se almacenan los datos recibidos
Datos newfile	control	Cadena hex	Cadena de bytes en hexadecimal a escribir en un archivo de configuración o datos
Path Tx file	control	Path normal	Path del archivo que se transmite desde el otro MODEM

Path Rx file	Control	Path normal	Path del archivo donde se han almacenado los datos
Path newfile	Control	Path normal	Path del archivo de configuración o datos creado o modificado
BER en por 1	indicador	0-1	Calculo del BER del Rx file utilizando el Tx file , que es el que debería recibirse

5.1.5 Selección de función

Una vez definidos los controles que utilizaremos en el programa podremos seleccionar la función que queramos realizar con él simplemente realizando los siguientes ajustes en los controles:

5.1.5.1 Instrucciones de activación/desactivación

Control ---------- Instrucción	Leer	Escribir	Normal /Test enable	Test/ Test-esc	Lect-escrit/ habilitador	Enable MODEM/ reset	Reset on/off	Enable o reset/fin Test
Habilitar MODEM	off	On	on	*	off	On	*	on
Habilitar Test	off	On	off	On	*	*	*	*
Habilitar Test esc	off	On	off	off	*	*	*	*
Fin Test	off	On	on	*	*	*	*	off
Reset MODEM	off	On	on	*	Off	off	on	on
Fin reset	Off	On	On	*	on	off	off	on

Para este grupo de funciones no hacen falta parámetros adicionales de dato o dirección.

* significa que no importa el valor.

5.1.5.2 Instrucciones de lectura

Las funciones(que realmente generan una instrucción , como en la tabla de arriba) de lectura necesitan los siguientes controles comunes :

Parámetro	Escribir	Normal/Test enable	Lect-escrit/habilitador	Lect/escrit

Valor	On	On	On	arriba

Y además para seleccionar entre una y otra :

Control ---------- Instrucción	Leer	Lect simple/ múltiple	Activar/ desactivar lectura	Canal0/ canal1	Continua	Activar/ desactivar
Lect. Reg 0	On	Abajo	Arriba	On	*	*
Lect. Reg 1	On	Abajo	Arriba	Off	*	*
Lectura simple (modo=00)	On	Arriba	*	*	Off	*
Lectura continua activada (modo=01)	Off	Arriba	*	*	On	Arriba
Lectura continua desactivada (modo=03)	Off	Arriba	*	*	On	Abajo
Fin Lect. Reg	Off	Abajo	Abajo		*	*

Estas funciones necesitan según el caso un valor apropiado de dirección.

5.1.5.3 Instrucciones de escritura

Estas funciones necesitan los siguientes controles comunes:

Parámetro	Escribir	Leer	Normal/Test enable	Lect-escrit/habilitador	Lect/escrit
Valor	On	Off	On	On	Off

Y además:

Control ---------- Instrucción	Escritura frecuencia /simple	Escritura a pila *	Frectot /frec simple	Externo/ interno	Canales/ piloto	Frec0/frec1
Escritura ** Normal (modo=00)	Abajo	Abajo	*	*	*	*
Escritura**	Abajo	Arriba	*	*	*	*

A pila (modo=01)						
Frecuencia total	Arriba	*	Arriba	*	*	*
Frecuencia 0 externa (modo=01)	Arriba	*	Abajo	On	Arriba	On
Frecuencia 0 interna (modo=00)	Arriba	*	Abajo	Off	Arriba	On
Frecuencia 1 externa (modo=01)	Arriba	*	Abajo	On	Arriba	Off
Frecuencia 1 interna (modo=00)	Arriba	*	Abajo	Off	Arriba	Off
Frecuencia P externa (modo=01)	Arriba	*	Abajo	On	Abajo	*
Frecuencia P interna (modo=00)	Arriba	*	Abajo	Off	Abajo	*

* No importa el valor.

** Solo en el 2º modo de programación del PIC, modo avanzado.

Estas funciones(generación de instrucciones realmente) necesitan los parámetros pertinentes (dato, dirección).

5.1.5.4 Funciones de fichero

Las siguientes funciones se pueden ejecutar por separado o en una misma ejecución de la aplicación:

Parámetro Función	Activar modo fichero	Fichero W/calculo BER	Paths	String
Escribe fichero	On	On	Newfile	Datos newfile
Calcula BER	On	Off	Tx ,Rx files	*

5.1.6 Utilización de la aplicación

Para el uso efectivo de esta aplicación se procederá según el tipo de función a realizar con ella:

- Si se quiere usar una función del modo fichero , que es independiente del otro modo y que no es incompatible(se puede realizar en la misma ejecución que una función en modo control).
- Si se quiere realizar una función en modo control.

5.1.6.1 Pasos a seguir en modo fichero:

Si se quiere crear o modificar un archivo , se seleccionan los controles lógicos pertinentes (ver arriba) y se introduce el string de bytes que queremos en el control string de datos a newfile , y el path completo(con nombre de fichero) que queramos en Path newfile.
Para la otra función se procede de manera idéntica pero introduciendo en este caso los paths completos del archivo Tx y el Rx , el resultado en tanto por 1 nos será dado en el indicador correspondiente.

5.1.6.2 Pasos a seguir en modo control:

Ante todo se procederá a configurar el dispositivo VISA que es fundamental para la comunicación(en este caso por el puerto serie , así que será el COM1 o el COM2, ver arriba).Luego se seleccionaran los controles lógicos pertinentes(ver arriba) , y se procederá dependiendo de la instrucción que queramos ejecutar:

- Instrucciones habilitadoras(reset , enable , tests) ; ya no hace falta hacer nada más , se ejecuta la aplicación.

- Instrucciones de lectura:
    - Volcado de archivo y Lectura normal; se introduce la dirección pertinente(en el caso de lectura normal también se introduce el path completo del archivo donde guardamos los datos y un 1 en los bytes a recibir , si hay más de un 1 se quedara en ejecución continua a menos que anteriormente hayamos activado la lectura continua) y se ejecuta.
    - Lectura continua de registro ; se introduce el número de bytes que queremos recibir (si se activa , si se desactiva no hace falta , de hecho es necesario desactivarla una vez recibidos los datos que queremos pues si no siguen llegando datos al PC del canal del MODEM) y el path completo de donde queremos guardar los datos recibidos , y se ejecuta la aplicación.

- Instrucciones de Escritura:

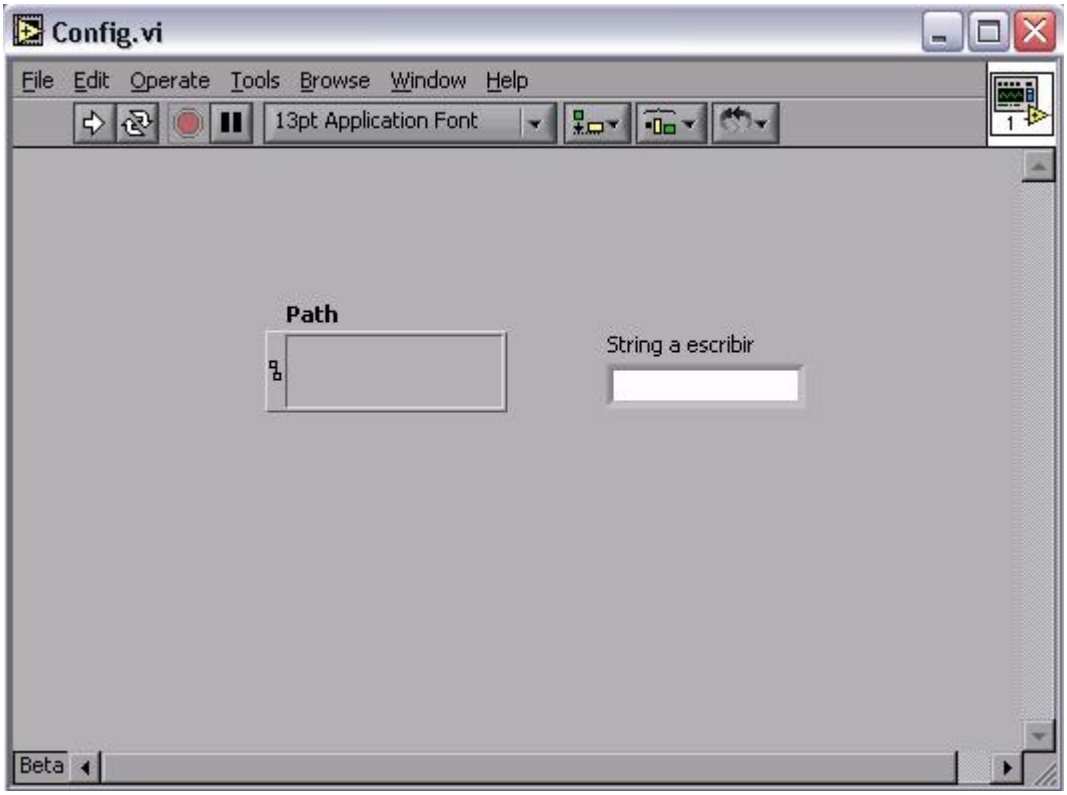
    - Escritura simple; se introducen , dato y dirección y se ejecuta.

- Escritura de frecuencia total e internas(en el caso de internas se tienen en cuenta los limites de frecuencia soportados) de los demás canales ; se introduce el o los datos (3 o 1) y se ejecuta.
- Escritura de frecuencia externa , se introduce el dato y se ejecuta.
- Escritura en cola de transmisión * ; se introduce el dato y se ejecuta la aplicación.

* solo en el 2º caso de programación del PIC.

5.2 CONFIG

Esta aplicación nos permite crear un archivo hexadecimal de instrucciones y/o datos que luego posteriormente usaremos con otra aplicación para cargarlo por el puerto serie en el PIC.



Aplicación Config

5.2.1 Funcionalidad

Mediante el uso de esta aplicación creamos un archivo de configuración o un archivo de datos para posterior volcado en el PIC o en el registro de transmisión del MODEM respectivamente.

5.2.2 Diagrama del programa y descripción funcional

Se trata de un programa muy simple como se puede observar, de manera secuencial se abre o crea el archivo (dependiendo si existía antes o no), se introducen los datos y se cierra el archivo. Esta aplicación crea el archivo indicado por el path completo y mete la cadena en la última posición escrita (sin introducir espacios), de manera que si el archivo existe se escribirá al final de lo que había antes (sin crear un archivo nuevo) y si no existe se creara el archivo con la cadena al principio

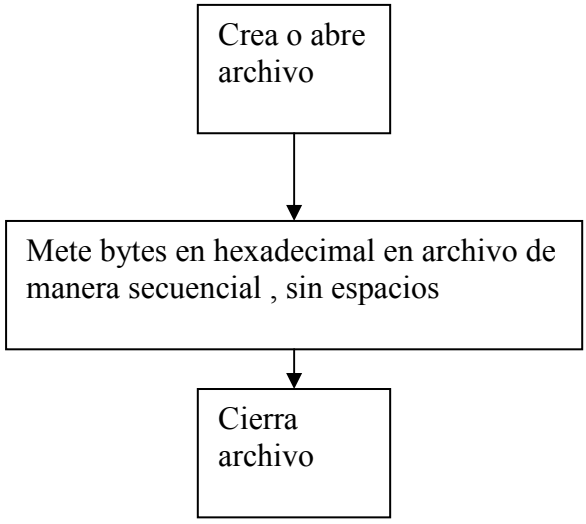


Diagrama funcional del programa

5.2.3 Parámetros del programa: controles e indicadores

Este programa solo dispone de un par de controles, no necesita indicadores al ser el archivo el producto final que requerimos:

Nombre del parámetro	Tipo	Valor	Comentarios y funcionalidad
Path	Control	Path normal	Es el Path completo del archivo a crear o escribir
String a escribir	control	Cadena bytes(Hex)	Los datos a escribir en el archivo

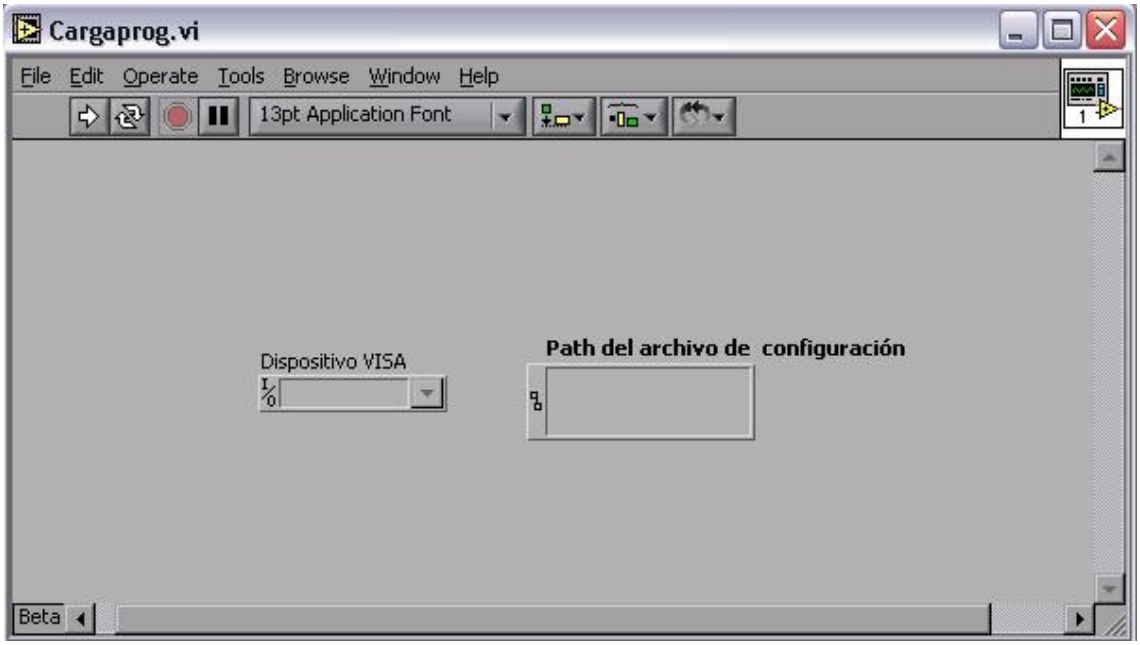
5.2.4 Utilización de la aplicación

Para el uso efectivo de la aplicación se tendrá en cuenta lo siguiente:

- Las instrucciones se escriben como cadenas de bytes en hexadecimal puestas de manera secuencial, empezando con la identificación de la instrucción y luego los parámetros pertinentes.
- Se pondrá el path completo(incluido nombre del fichero que queremos crear o modificar), sabiendo que si ya existe , la información se introducirá al final de este.

5.3 CARGAPROG

Esta aplicación nos permite cargar una serie de instrucciones ordenadas de manera secuencial en el PIC por medio del puerto serie , enviándose byte a byte , ya que las instrucciones son identificadas por el PIC con los dos primeros bytes recibidos , y no dan lugar a confusiones , sabiéndose luego el número de parámetros que debe recibir también.



Aplicación Cargaprog

5.3.1 Funcionalidad

Cargar secuencias de instrucciones en el PIC por el puerto serie , de manera que este realice la configuración del MODEM , e incluso configurar y activar el transmisor en modo interno (2º modo de programación del PIC).

5.3.2 Diagrama del programa y descripción funcional

Se trata de un esquema bastante simple , se abre y configura una sesión VISA para comunicación por el puerto serie con los parámetros estándar , es decir ; 9600 bps , asíncrona ,8 bits de datos ,sin paridad y con 1 bit de stop.
Luego se van leyendo del archivo donde tenemos las instrucciones los bytes 1 a 1 y mandando por el puerto serie al PIC , esperando un pequeño tiempo cada vez para no sobrescribir el buffer de recepción del PIC antes de que haya podido ejecutar la anterior instrucción , y cuando se acaba de leer todo el archivo se cierra la sesión.

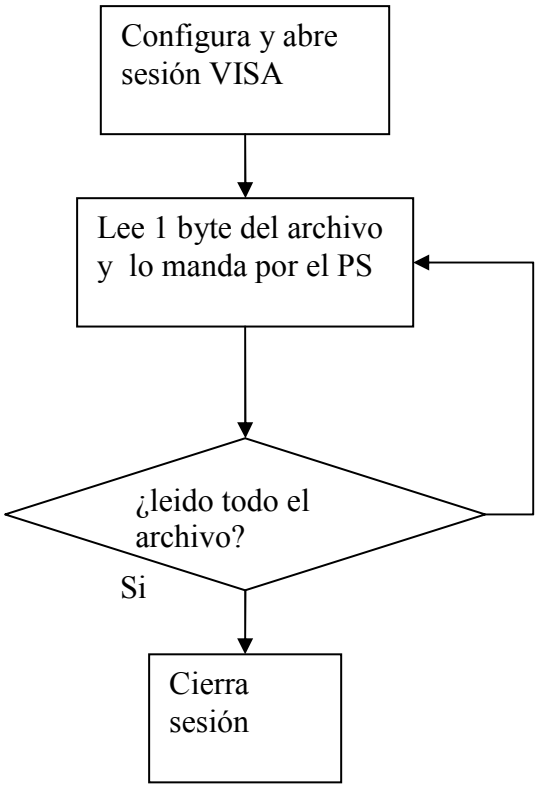


Diagrama funcional

5.3.3 Parámetros del programa: controles e indicadores

Debido a que el resultado de la aplicación observa directamente en el controlador del MODEM , no utilizamos indicadores :

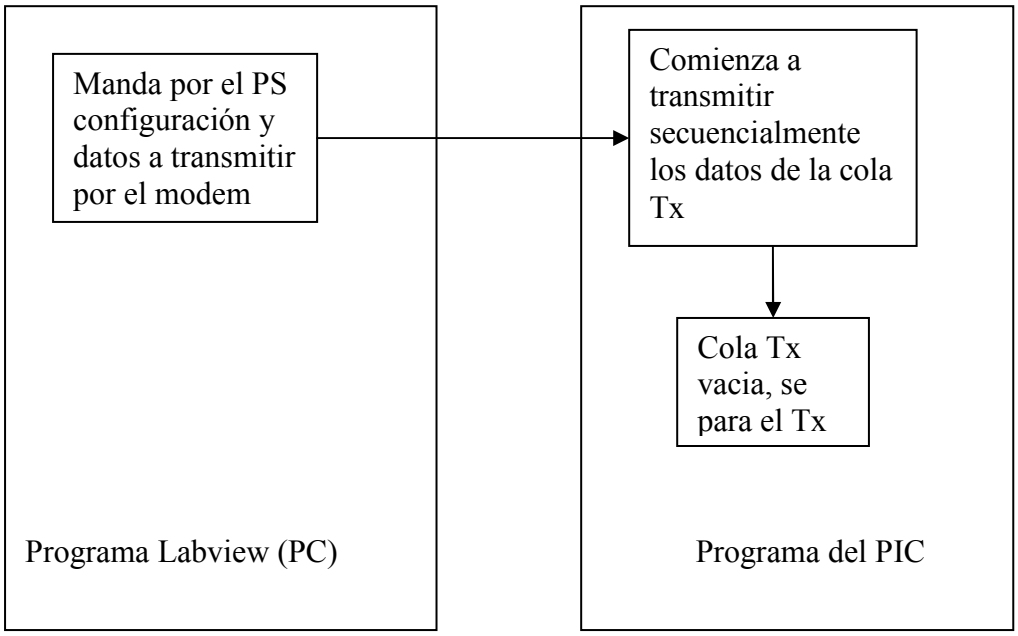
Nombre del parámetro	Tipo	Valor	Comentarios y funcionalidad
Dispositivo VISA	control	ASRLi:: INSTR con i=1,2	Tipo de dispositivo para la sesión de comunicaciones, usaremos ASRL1 y ASRL2 Refiriéndonos a los puertos COM1 y COM2
Path del archivo de configuración	control	Path normal	Indicamos donde esta y cual es el archivo que contiene las instrucciones y datos para configurar el MODEM al transmitirlos a este por el puerto serie de manera secuencial

5.3.4 Utilización de la aplicación

Debido al carácter especial de esta aplicación y el 2º modo de programación del PIC este programa nos permite llevar a cabo la trasmisión automática del MODEM o activar el modo en el que cargando un simple fichero por el puerto serie , el PIC se encarga de controlar de manera automática la transmisión desde que se activa , hasta que se desactiva.
así pues explicamos los dos modos de trabajo:

- Modo simple ; se introduce el valor del dispositivo VISA y el Path completo del archivo de configuración(instrucciones) y se ejecuta normalmente , dejando el MODEM en el estado deseado.

- Modo transmisor automático ; se procede igual que antes , pero el resultado es una Transmisión completa de 1 ráfaga de datos desactivándose luego el transmisor, con lo que cargando posteriores archivos , se podrían realizar otras ráfagas (configurando el transmisor de la misma o diferente manera).
  Vemos esto en el siguiente esquema de interacción PC-PIC (para más detalle consultar el capitulo referente al 2º modo de programación del PIC):

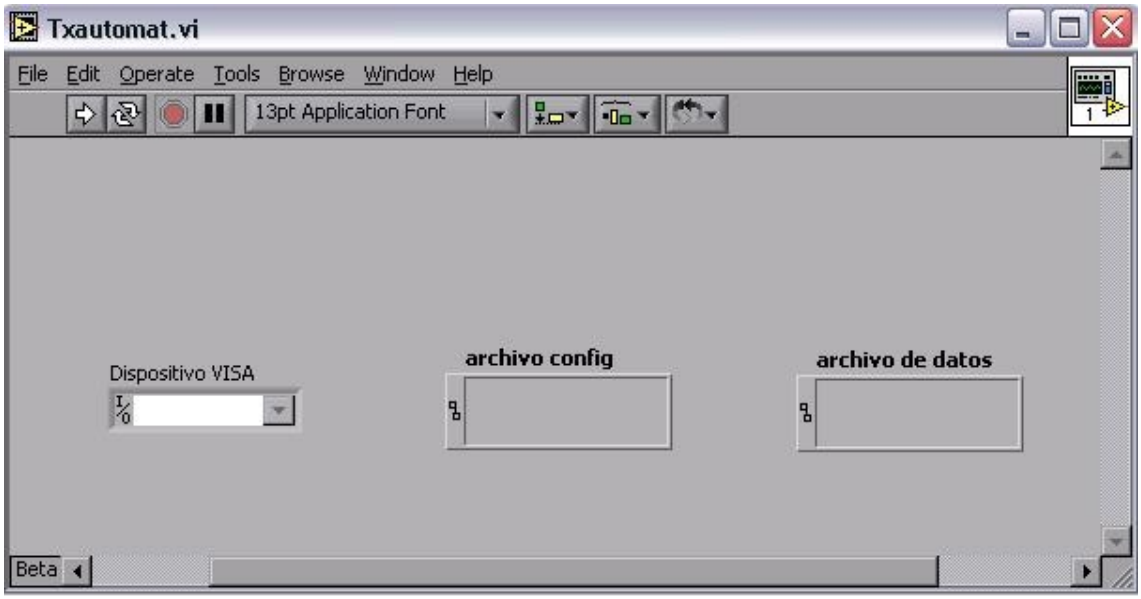


Modo de transmisión automático (control interno) a ráfagas.

Como se puede apreciar , gracias a esta aplicación podemos realizar una transmisión a ráfagas por el MODEM , configurándolo convenientemente y cargando los datos a transmitir en la cola de transmisión antes de activar el transmisor.De esta manera , el transmisor seguirá enviando datos hasta que la cola este vacía, momento en el que el transmisor se desactivara.
Esto es especialmente útil , pero hay que considerar que los primeros datos a meter en la cola de transmisión deben ser los que componen la cabecera , pues sin esta no se identifica una ráfaga de datos.

5.4 TXAUTOMAT

Es la automatización de una transmisión a ráfagas (con el modo normal de programación el PIC , es decir control externo).Nos permite realizar la transmisión siendo el PC el encargado de controlar el proceso.



Aplicación Txautomat

5.4.1 Funcionalidad

Controla y lleva a cabo el proceso de transmisión a ráfagas por el MODEM, para ello utiliza:

- El archivo de instrucciones pertinente o de configuración.
- El archivo de datos que queremos transmitir.
- Protocolo de paso de datos a transmitir por demanda del MODEM.

Nos permite de manera automática transmitir ráfagas mucho más grandes que en el otro modo (visto anteriormente) , pero si la comunicación por el puerto serie tiene una velocidad parecida a la de transmisión del MODEM no es un modo útil , pues puede ocasionar retrasos en el paso de datos.El protocolo de paso de datos entre PC y MODEM es bastante simple.

5.4.2 Diagrama del programa y descripción funcional

Al utilizar una sesión VISA para comunicación lo primero es abrirla y configurarla a los valores apropiados ; 9600 bps , asíncrona ,8 bits de datos ,sin paridad y con 1 bit de stop.Luego se procede a mandar el archivo de configuración al PIC de manera secuencial (como hacíamos en la anterior aplicación) , teniendo en cuenta que ponemos en el registro a transmitir el valor del primer byte de la cabecera , esto es 55 (hexadecimal) .
Mediante el protocolo de paso de datos el PC le pasa al PIC los siguientes datos a transmitir ; el otro byte de la cabecera(54) y el contenido del archivo de datos (secuencialmente) hasta que este

haya sido leído y enviado entero , momento en el cual el PC manda por el puerto serie la instrucción de desactivar transmisor y se cierra la sesión.

El protocolo de paso de datos entre PC y MODEM vía PIC se basa en lo siguiente:

1. El PIC recibe una interrupción del MODEM de fin de transmisión del byte (registro de transmisión vacío).
2. El PIC manda un 3 por el puerto serie al PC
3. El programa de labview lee el siguiente dato del archivo y lo manda al PIC por el puerto serie
4. El PIC lo escribe en el registro de transmisión.

Esto funciona si la velocidad de transmisión del MODEM es mucho menor que la de la comunicación por el puerto serie y si el MODEM genera la interrupción cuando el dato se ha volcado del registro al buffer de transmisión desde el registro de transmisión , es decir aun se esta transmitiendo , teniendo continuidad.

Suposiciones que hago sobre el transmisor:

- El MODEM genera la interrupción de transmisión cuando el dato se ha volcado del registro 43(hexadecimal) al buffer de transmisión desde el registro de transmisión, asegurándonos la continuidad.
- Una vez activo el transmisor solo se apaga cuando ha terminado de transmitir el byte que tiene en el buffer de transmisión.

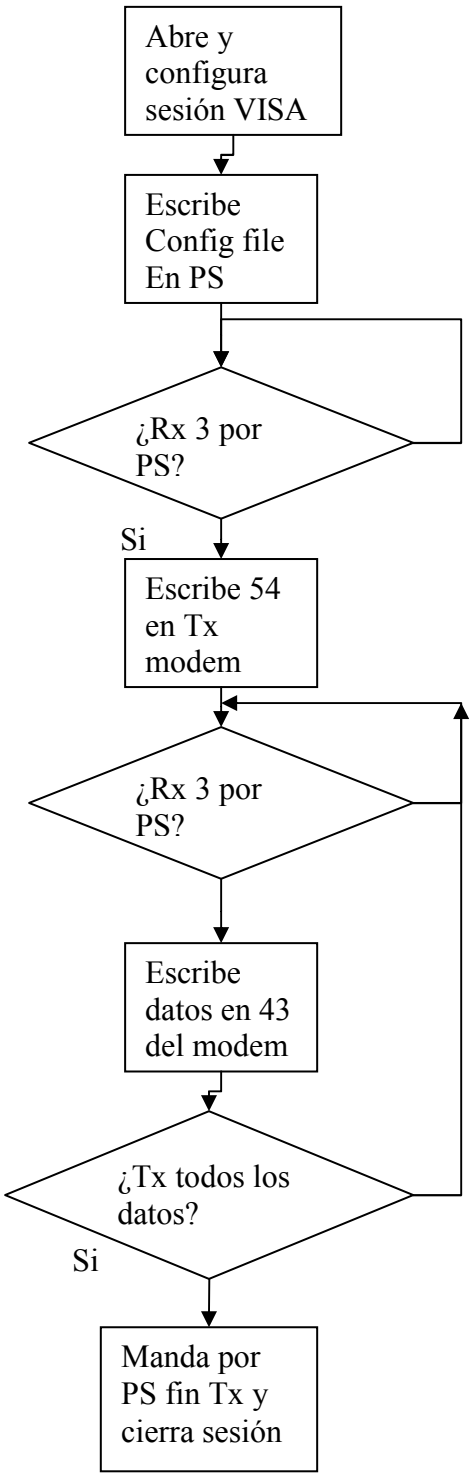


Diagrama de funcionamiento

5.4.3 Parámetros del programa: controles e indicadores

Utilizaremos varios controles para configurar los archivos a cargar y el puerto a usar :

Nombre del parámetro	Tipo	Valor	Comentarios y funcionalidad
Dispositivo VISA	control	ASRLi:: INSTR con i=1,2	Tipo de dispositivo para la sesión de comunicaciones, usaremos ASRL1 y ASRL2 Refiriéndonos a los puertos COM1 y COM2
Archivo config	Control	Path normal	Es el Path completo del archivo en el que están las instrucciones de configuración (con sus parámetros) de manera secuencial
Archivo de datos	control	Path normal	Es el Path completo del archivo en el que están los datos a transmitir por el MODEM

5.4.4 Utilización de la aplicación

Simplemente hay que introducir los Paths adecuados de los archivos y el puerto a utilizar. Comentaremos sin embargo la interacción entre PC-PIC que permite que esta aplicación funcione:

Como podemos ver en el diagrama en este caso el PC es el encargado de pasar los datos al PIC , para ello usa el protocolo anteriormente explicado , y además es el encargado de parar el transmisor , por lo que hay que tener en cuenta si se cumplen las condiciones de velocidad anteriormente citadas.Esto nos crea una cierta dependencia del PC que puede ser conflictiva en algunos casos , por los que se desarrollo el anterior método de control (el PIC es el que gobierna la transmisión en todo momento y tiene los datos a transmitir en una cola) , si bien con este método se pueden realizar transmisiones de una manera más simple y con archivos de configuración más simples.

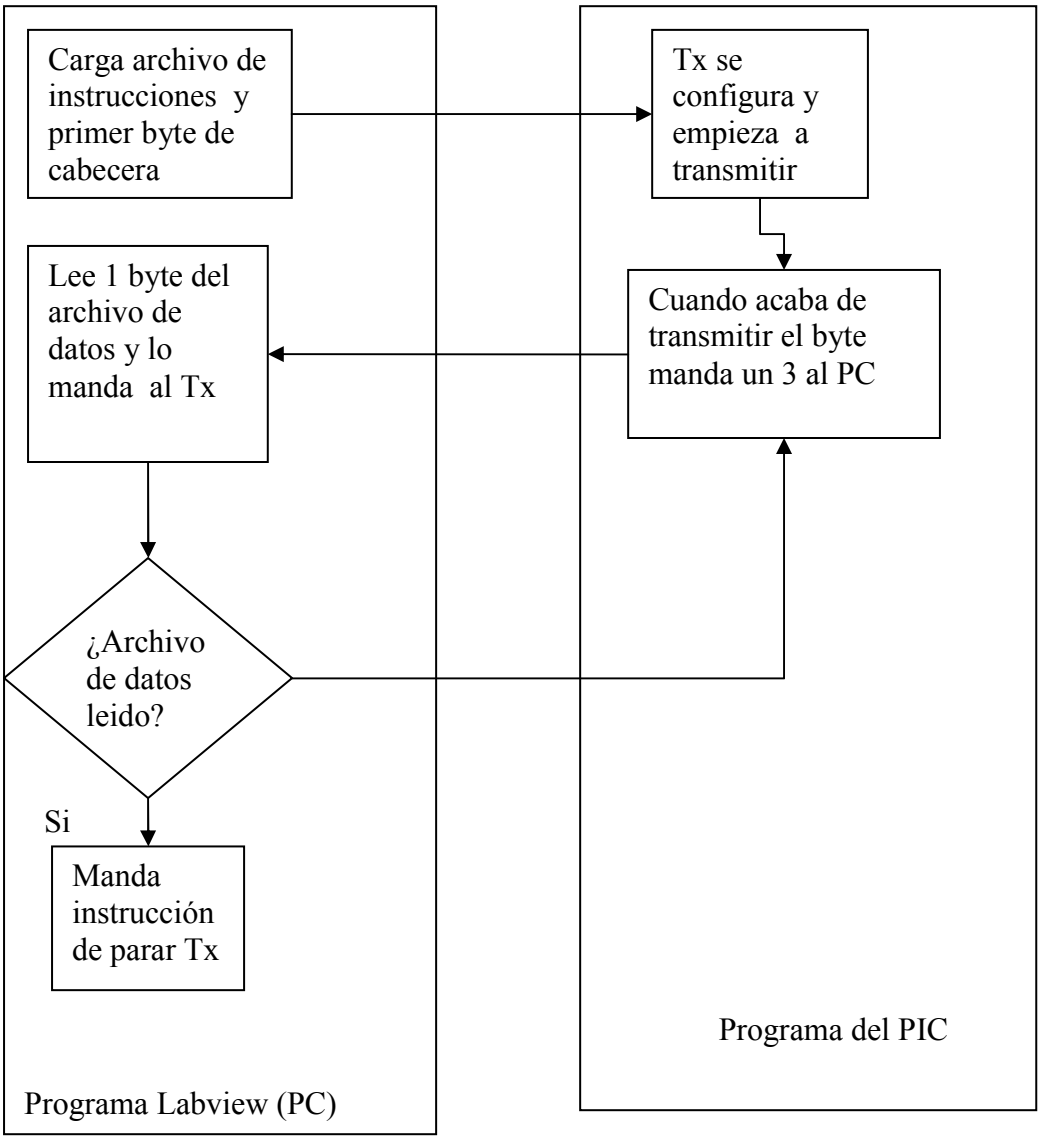
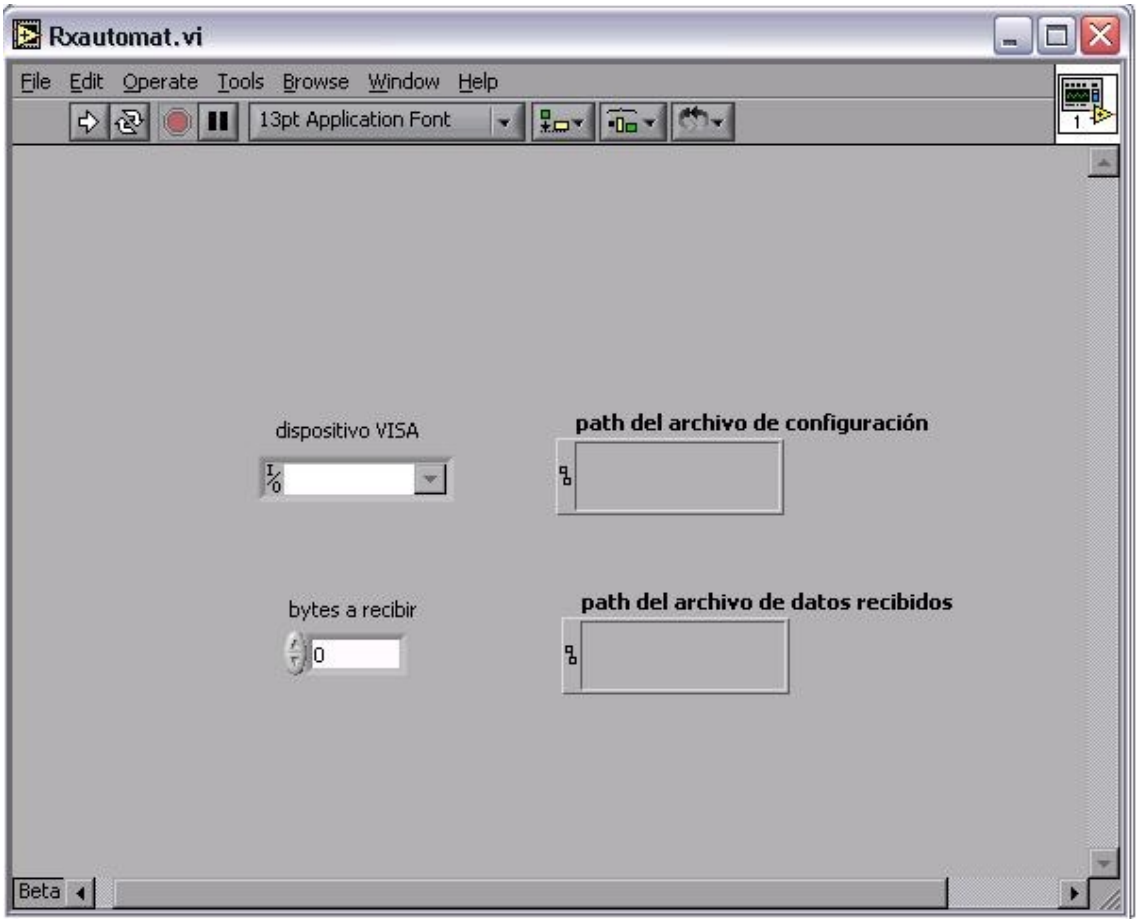


Diagrama Transmisión automática externa

5.5 RXAUTOMAT

Esta aplicación nos permite realizar recepciones del MODEM de un número especifico de bytes de manera automática, tanto para el modo con buffer interno(cola de Rx) como para el normal , ya que solo cambia la forma en que el PIC manda los datos al PC por el puerto serie.



Aplicación Rxautomat

5.5.1 Funcionalidad

Lleva a cabo una recepción automática mediante el MODEM , para ello utiliza:

- Carga de archivo de configuración del MODEM.
- Selección del archivo utilizado para guardar los datos recibidos.
- Número de bytes a recibir configurable.

Es decir mediante esta aplicación podemos configurar al MODEM para recibir por el canal elegido(0 o 1) y guardar los datos en un registro para posterior procesado.
El programa se queda en espera hasta que el MODEM recibe el número de bytes que nosotros solicitamos , estos son enviados por el PIC vía puerto serie (con almacenamiento previo en un registro o en la cola de Rx , según el modo que estemos usando de programación del PIC) .

5.5.2 Diagrama del programa y descripción funcional

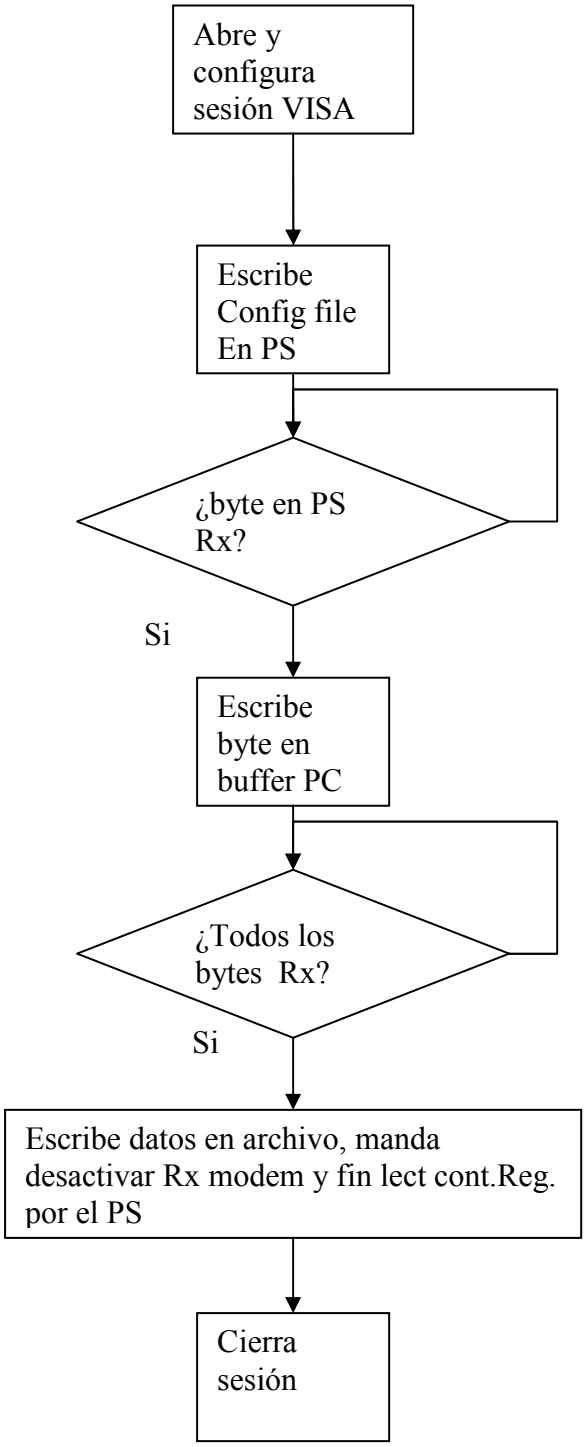


Diagrama funcional

Esta aplicación abre y configura una sesión VISA como es habitual ; 9600 bps , asíncrona ,8 bits de datos ,sin paridad y con 1 bit de stop.

Se procede a mandar secuencialmente las instrucciones del archivo de configuración(entre las que estará la de habilitar lectura continua de registro del canal por el que se espera recibir datos) por el

puerto serie , con lo que el PIC configura adecuadamente el MODEM , habilitando la recepción.Posteriormente se espera hasta recibir el número de bytes requeridos en e puerto serie , enviados previamente por el PIC al leerlos este del canal de recepción del MODEM que esta generando las interrupciones.Cuando se tienen todos los datos , se escriben en el archivo seleccionado por el control apropiado y se mandan al PIC las instrucciones de parar la lectura continua y el receptor, quedando este en reposo de nuevo(ya que si no seguiría recibiendo datos , puesto que una vez capturada la cabecera debemos parar nosotros el receptor cuando se han recibido los datos previstos).Cerrándose la sesión.

5.5.3 Parámetros del programa: controles e indicadores

Estos son los controles utilizados:

Nombre del parámetro	Tipo	Valor	Comentarios y funcionalidad
Dispositivo VISA	control	ASRLi::INSTR con i=1,2	Tipo de dispositivo para la sesión de comunicaciones, usaremos ASRL1 y ASRL2 Refiriéndonos a los puertos COM1 y COM2
Bytes a recibir	control	Nº entero	Parámetro que indica el número de bytes que esperamos recibir por el MODEM y que queremos capturar y almacenar en el PC para posterior procesado
Path del archivo de configuración	control	Path normal	Indicamos donde esta y cual es el archivo que contiene las instrucciones y datos para configurar el MODEM al transmitirlos a este por el puerto serie de manera secuencial
Path del archivo de datos recibidos	control	Path normal	Indicamos el path completo del archivo donde se almacenarán los datos recibidos

5.5.4 Utilización de la aplicación

Se procede a introducir los valores oportunos de:

- Puerto de comunicaciones.
- Paths con nombre de archivo para configuración y para guardar datos del MODEM.
- Número de bytes a recibir por el MODEM.

Una vez configurados estos controles , se ejecuta la aplicación , aunque hay que tener en cuenta que en la configuración debe activarse el modo lectura continua de registro 0 o 1 según el canal de datos a recibir por el MODEM , puesto que el paso de datos del PIC al PC provenientes del MODEM se basa en la captura de los valores del registro pertinente cuando hay una interrupción en recepción.

Explicamos ahora el modo de recepción (interna o externa , en este caso da igual) automática , desde el punto de vista de la interacción entre PC y PIC , ya que el gobierno del MODEM se debe a este ultimo , aunque el que decide cuando para el Receptor es el PC :

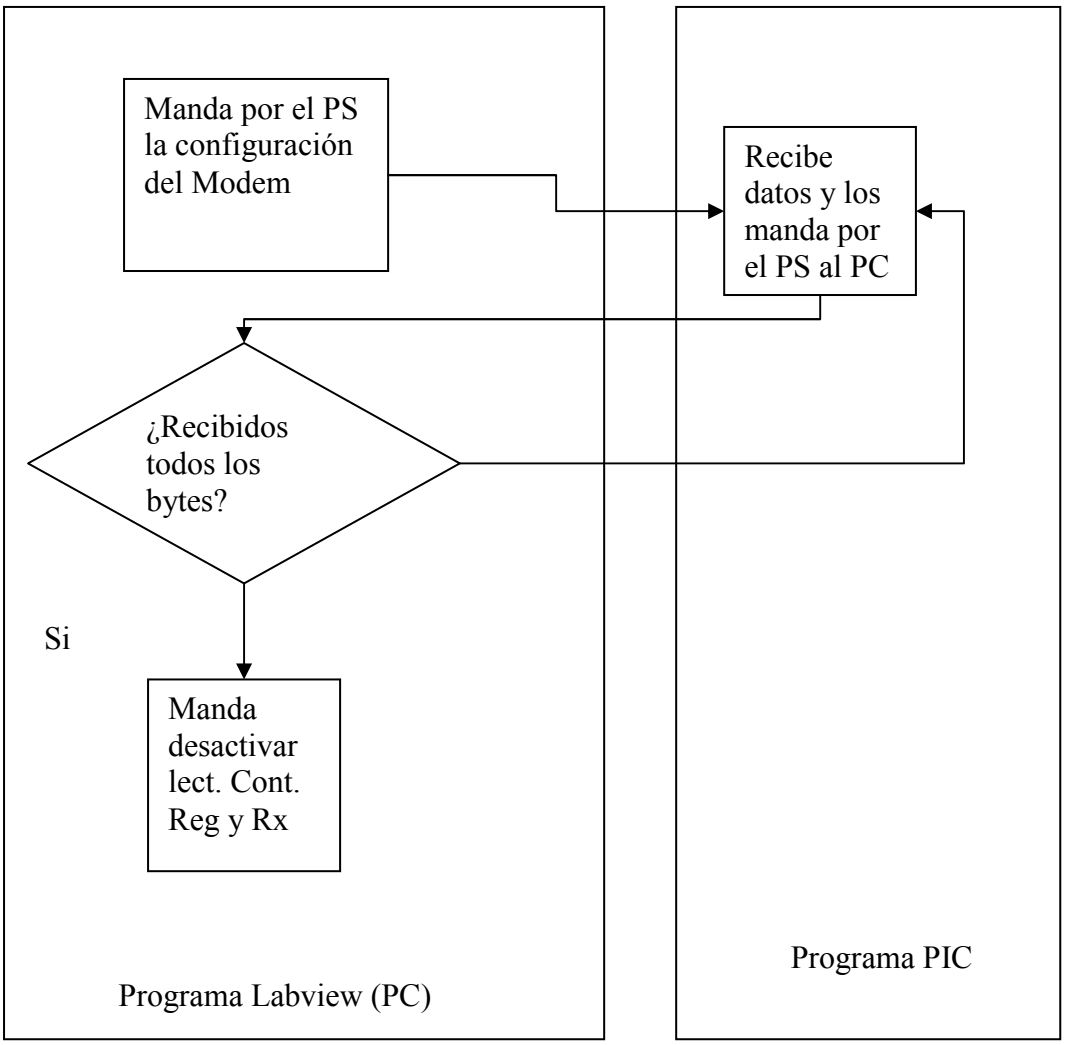


Diagrama de interacción en Recepción automática

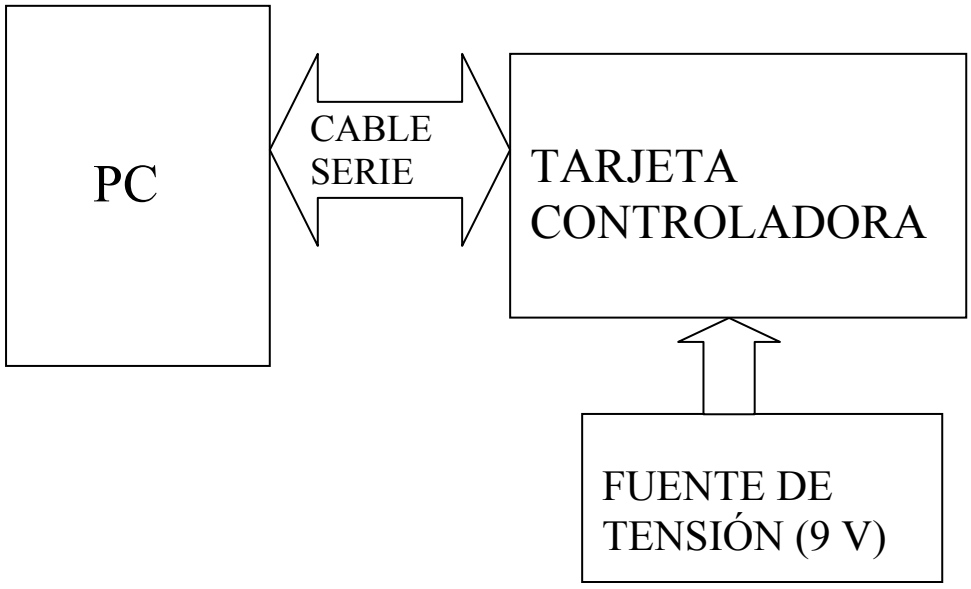
Como puede verse el PC a pesar de gobernar el proceso esta a merced del PIC , y este depende de si el receptor recibe o no.De manera que el programa estar esperando hasta que el número de datos recibidos sea el apropiado (ejecución continua en espera) , si bien este es el que finaliza el proceso desactivando el las lecturas continuas de registro efectuadas por el PIC ,y el receptor del MODEM.

6. FUNCIONAMIENTO Y PRUEBAS

En este capitulo describiremos como utilizamos la placa controladora y las aplicaciones software para realizar pruebas con el MODEM siguiendo con el esquema de control anteriormente explicado; las aplicaciones de Labview ejecutadas sobre el PC nos proporcionan el medio de control directo del MODEM a través del controlador de la placa.

6.1 Funcionamiento de la tarjeta controladora

Una vez conectada la placa al PC mediante el cable serie y alimentada mediante un transformador a la red eléctrica o mediante una pila de 9V como se puede ver en el esquema de montaje:



Esquema de montaje de la placa controladora

Se procederá a resetear manualmente la placa con lo que el PIC estará esperando instrucciones por el puerto serie ,estas serán proporcionadas mediante el interfaz de la aplicación Labview pertinente ejecutada en el PC:

- Si queremos controlar paso a paso la configuración del MODEM y comprobar el funcionamiento de la placa utilizamos la aplicación Panel de control , para ello solo tenemos que seguir el manual adjunto de la aplicación contenido en el capitulo dedicado a las aplicaciones en Labview , teniendo en cuenta que debemos seguir los siguientes pasos:
  - Habilitar el MODEM.
  - Resetearlo durante el tiempo deseado.

    - Realizar las configuraciones y comprobaciones que se deseen teniendo en cuenta el manual de la aplicación Labview.
- Podemos cargar un archivo de configuración creado previamente mediante la aplicación Cargaprog (el archivo contiene la secuencia de instrucciones y datos necesarios dispuestos para su envío secuencial por el puerto serie), de manera que el MODEM quedara configurado y realizara lo que le hallamos ordenado , incluidas las transmisiones a ráfagas si estamos utilizando el 2º modo de programación del PIC .
- Si queremos realizar una transmisión a ráfagas en el primer modo de programación utilizaremos la aplicación Txautomat teniendo en cuenta su funcionamiento (ver manual en aplicaciones Labview) ya que en este modo el PC va cargando uno a uno los datos a transmitir y para la transmisión.
- Para llevar a cabo recepciones automáticas con el MODEM utilizamos la aplicación Rxautomat (ver manual en aplicaciones Labview) que nos deja los datos recibidos en un archivo almacenado en el PC.

Esto es en cuanto al funcionamiento de una de las placas controladoras , veremos ahora en función a esto y según su asociación (sola o con otra) las pruebas realizadas .

6.2 Pruebas del interfaz

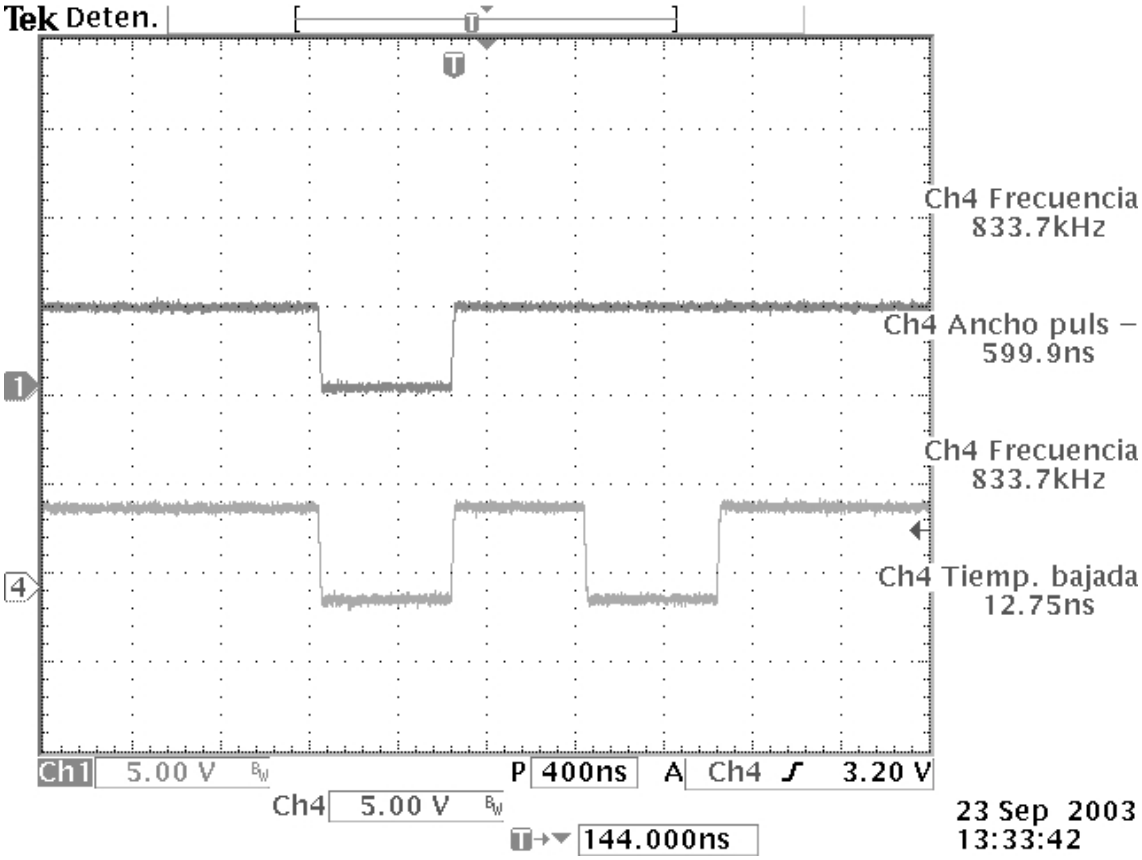
Para realizar estas pruebas se utilizo solamente una de las placas , si bien se probaron cada una por separado , se trata de confirmar el buen funcionamiento del control del MODEM desde su carácter más básico , es decir:

- Activación y desactivación de señales de control del MODEM como habilitación o modos test.
- Reset software del MODEM(ordenado desde el PC).
- Accesos de lectura y escritura a los registros del MODEM (básico para la configuración del MODEM).

Estas pruebas se realizaron utilizando el montaje antes explicado de una sola placa y un osciloscopio digital para comprobar el estado de las señales y sus cambios como se puede ver en las siguientes capturas.
Se utilizo la aplicación Labview Panel de control , pues se debía operar paso a paso. Como se puede observar del hecho que funcionan los accesos a los registros del MODEM(lectura y escritura) , las activaciones de las señales de control funcionan perfectamente , al igual que el reset software(desde el PC) , que se emplea continuamente cada vez que queremos variar las condiciones de trabajo del MODEM volviéndolo a su estado inicial.
Se probó el interfaz hardware diseñado para el control del MODEM con resultados satisfactorios como se puede ver en las siguientes graficas capturadas en el osciloscopio que se utilizo para comprobar los resultados:

- Ciclo de Escritura:

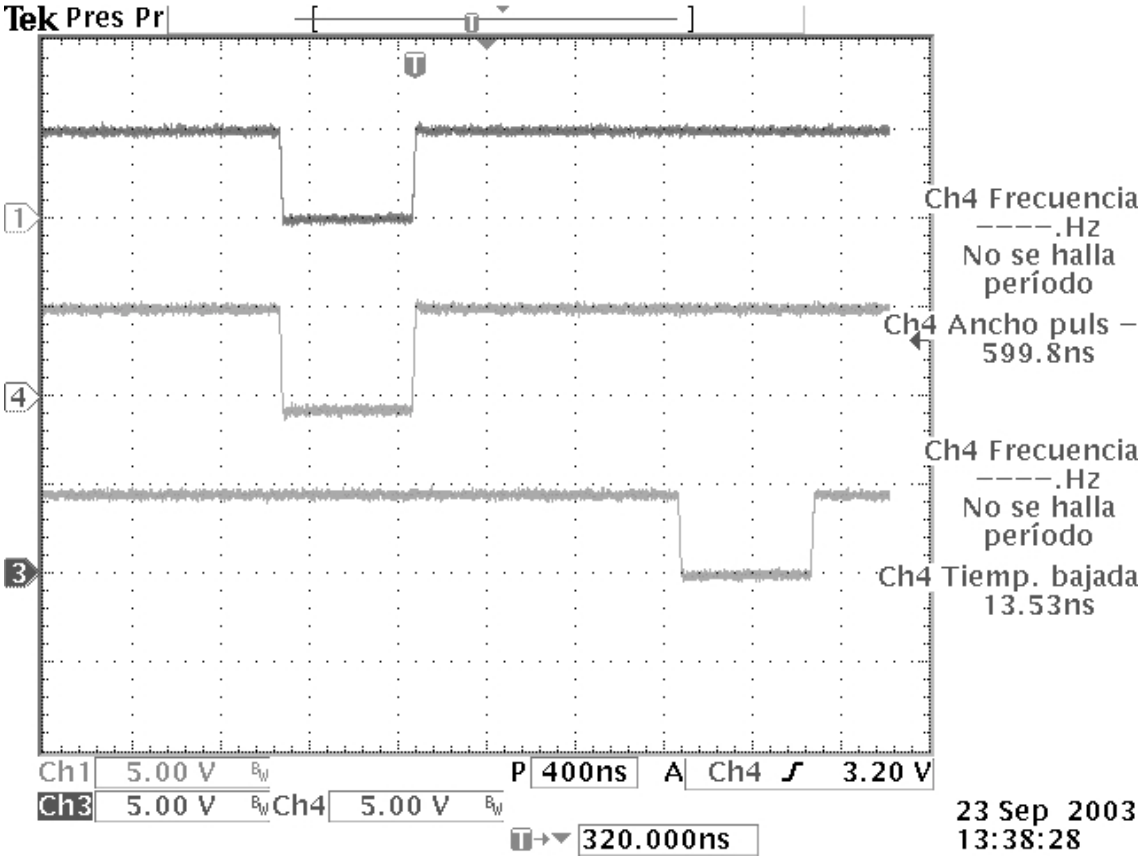
Se observa en la captura el cumplimiento de los requisitos en el ciclo de escritura del modem.En el canal 1 tenemos la señal ASN y en el canal 4 WEN , se cumple con los tiempos establecidos para el acceso , habilitándose primero la dirección valida del registro y escribiendo los datos validos (capturados por el MODEM).
Además se pudo comprobar que al escribir un dato en un registro accesible en escritura almacenaba ese valor, al luego leer el registro y obtener ese valor, por lo que los accesos de escritura funcionan perfectamente.



Captura del ciclo de escritura controlado por el PIC

- Ciclo de lectura:

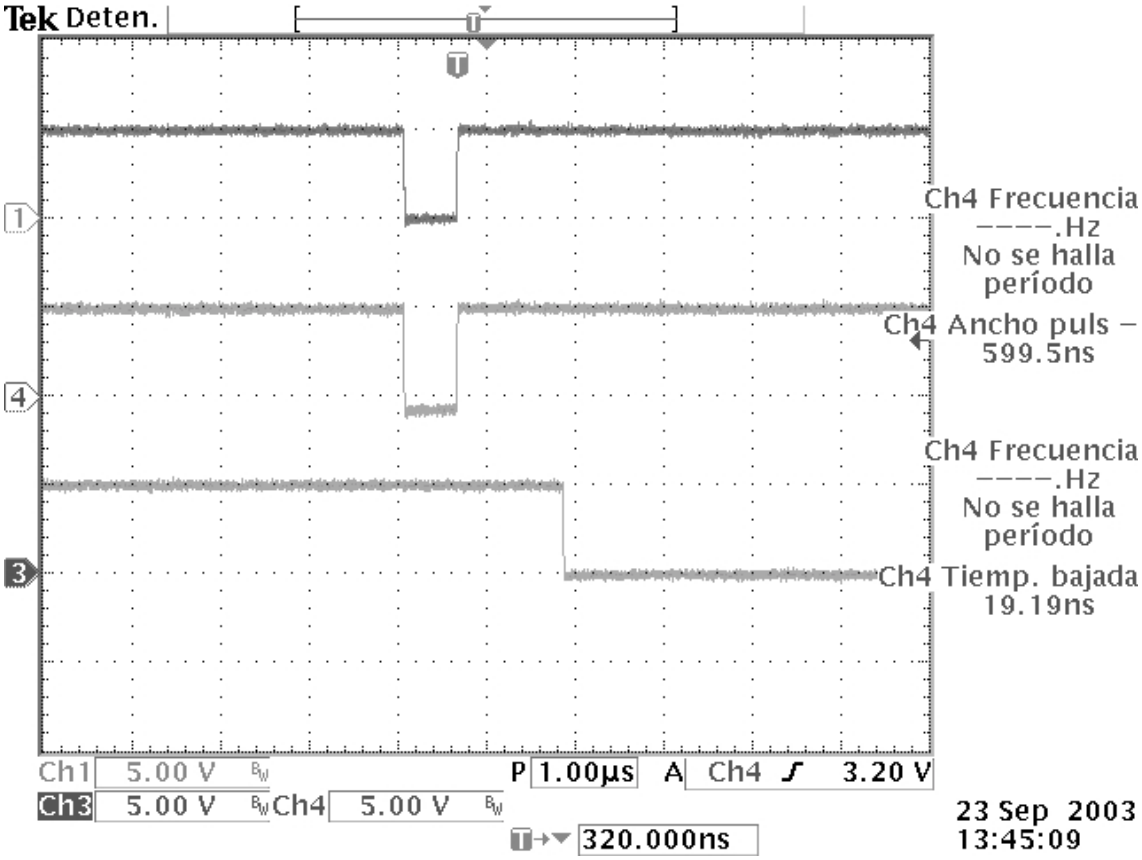
Se observa en la captura el cumplimiento de los requisitos en el ciclo de lectura del MODEM. En el canal 1 tenemos la señal ASN , en el canal 4 WEN y en el canal 3 OEN, se cumple con los tiempos establecidos para el acceso , habilitándose primero la dirección valida del registro y capturándose los datos validos, aportados por el MODEM, al subir de nuevo la señal OEN(el puerto D se vuelve entrada del PIC para la captura y luego vuelve a ser salida).Se comprobó que los accesos de lectura funcionan perfectamente al leer valores de registros después de un reset , coincidiendo con los valores previstos .A demás también se comprobó escribiendo y luego leyendo registros de manera que no se confundiera el valor obtenido como el almacenado en el puerto D , es decir ; escribir en registro A , escribir en registro B , leer en registro A y leer en registro B obteniendo los valores introducidos , por lo que no se trata de un valor almacenado en el puerto D durante todo el proceso.



Captura del ciclo de lectura controlado por el PIC

- Volcado de registro (Lectura interrumpida):

Se observa en la captura el cumplimiento de los requisitos del ciclo de lectura del MODEM hasta la captura de datos por el PIC(poniendo el puerto D como entrada), ya que se interrumpe aquí el ciclo para tener acceso continuo de los datos del registro del MODEM sobre el puerto D para su visionado mediante el osciloscopio de los bits pertinentes.En el canal 1 tenemos la señal ASN , en el canal 4 WEN y en el canal 3 OEN, se cumple con los tiempos establecidos para el acceso , habilitándose primero la dirección valida del registro y luego dejándose la captura de datos sin terminar.Hace falta una vez acabado con el estudio del registro elegido terminar con el ciclo para poder seguir accediendo al MODEM (lecturas o escrituras) , esto ultimo se lleva a cabo con la instrucción de lectura con los parámetros pertinentes , que lo único que hace es subir OEN y volver a configurar el puerto D como salida del PIC.
En la practica con este método se estudiaron los registros pertenecientes al bloque analógico de recepción como son las salidas de los sigma-delta y las señales PLLs internas utilizadas.



Captura del ciclo de lectura interrumpido para volcado de registro

6.3 Pruebas del MODEM

Para realizar pruebas de configuración , transmisión y recepción del MODEM hay que tener en cuenta los aspectos funcionales básicos del mismo .
A continuación se explican los principales registros necesarios para la configuración del MODEM , para más información ver el anexo adjunto al final de la memoria.

6.3.1 Breve Descripción del MODEM

Lo primero que debe hacerse con el módem antes de poder comenzar las pruebas , es configurar el mismo con las características que más nos interesen, Por tanto para poder dominar un poco mejor el módem vamos a ver a continuación los registros que son necesarios conocer para la configuración del módem.

Registros necesarios para la configuración del módem.

Básicamente veremos en este apartado el nombre del registro, tamaño, dirección, así como las posibilidades que nos ofrece en cuanto a la configuración del módem.

NOTA: Los nombres de los registros se han mantenido con la nomenclatura original.

Dir. (hex.)	*Dir. (dec.)*	*Nombre*	*Descripción*	*Acceso*	*Tam. (bits)*	*Estado de reset*
00	0	IDR0	Receive Buffer Register for channel 0	Lectura	8	
01	1	IDR1	Receive Buffer Register for channel 1	Lectura	8	
02-03	2-3	IIR	Interrupt Indentification Register	Lectura	9	
04	4	IER	Interrupt Enable Register	Lect/escrit	5	
05-06	5-6	M0alterna	Base Band Signal Module received in channel 0 (suitable signal level)	Lectura	12	
07-08	7-8	M1alterna	Base Band Signal Module received in channel 1 (suitable signal level)	Lectura	12	
09	9	C0	PLL error in reception channel 0	Lectura	8	
0A	10	C1	PLL error in reception channel 1	Lectura	8	
0B-0C	11-12	SLR0	Line Status Register for reception channel number 0	Lectura	10	
0D-0E	13-14	SLR1	Line Status Register for reception channel number 1	Lectura	10	
0F-11	15-17	AcrB0	Noise-Signal Estimation in reception channel number 0	Lectura	17	
12-14	18-20	AcrB1	Noise-Signal Estimation in reception channel number 0	Lectura	17	
15-17	21-23	AcrA0	Noise-Signal Estimation in reception channel number 1	Lectura	17	
18-1A	24-26	AcrA1	Noise-Signal Estimation in reception channel number 1	Lectura	17	
1B-1C	27-28	Nb_regmd0	DC value detected in the base band by Stadix	Lectura	14	
1D-1E	29-30	Nb_regmd1	DC value detected in the base band by Stadix	Lectura	14	
1F	31	Desp0	Shifts performed to transform the base band signal in a suitable signal. Channel 0	Lectura	8	
20	32	Desp1	Shifts performed to transform the base band signal in a suitable signal. Channel 0	Lectura	8	
21-22	33-34	VCO0	VCO counter used in PLL for channel 0 processing.	Lectura	15	
23-24	35-36	VCO1	VCO counter used in PLL for channel 1 processing	Lectura	15	
25-26	37-38	sbmCCR	Communication Control Register	Lect/escrit	10	
27-28	39-40	SbmBDT	Bit Detection Threshold	Lect/escrit	15	
29-2A	41-42	SbmBPT	Base-Band Power Threshold	Lect/escrit	16	
2B	43	SbmTHR	Transmisión Holding Register	Lect/escrit	8	
2C-2D	44-45	M0	Stadix module output. Channel 0	Lectura	13	
2E-2F	46-47	M1	Stadix module output. Channel 1	Lectura	13	

30	48	BFF		Lectura	8	
31-32	49-50	SLR01		Lectura	11	
33	51	Mreg	Receptor Decimator Register	Lect/escrit	8	00110011
34	52	TXFA	Transmitter frequency channel 0	Lect/escrit	8	00110000
35	53	TXFB	Transmitter frequency channel 1	Lect/escrit	8	01010000
36	54	TXFP	Transmitter pilot frequency	Lect/escrit	8	01000000
37	55	Txpcfg	Transmitter pilot configuration	Lect/escrit	5	11110
38	56	PLL0div	PLL0 divider bits 8 downto 1	Lect/escrit	8	00010100
39	57	PLL0cfg	PLL0 LSB and configuration register	Lect/escrit	3	100
3A	58	PLL1div	PLL1 divider bits 8 downto 1	Lect/escrit	8	00010100
3B	59	PLL1cfg	PLL1 LSB and configuration register	Lect/escrit	3	100
3C	60	PLLpdiv	PLLp divider bits 8 downto 1	Lect/escrit	8	00010100
3D	61	PLLpcfg	PLLp LSB and configuration register	Lect/escrit	3	100
3E	62	Attreg	Attenuation Register	Lect/escrit	6	000000
3F	63	Pwrdreg	Powerdown Register	Lect/escrit	6	000000
40-63	64-99	GPIO	Por definir	Lect/escrit	8	00000000
64-65	100-101	PPWR	Pilot Reception power	Lectura	14	
66-67	102-103	SINC0	Output of CHOi SincM filter	Lectura	13	
68-69	104-105	FIR0	Output of CH0i first FIR filter	Lectura	13	
6A-6B	106-107	FIR1	Output of CH0i second FIR filter	Lectura	13	
6C	108	SDREG	Output of CH0 and CH1 SD modulators	Lectura	6	
6D	109	PSDREG	Output of pilot SD modulator	Lectura	3	
6E	110	PLLsout	Output of the three PLLs	Lectura	3	
80	128	DACTEST	Test configuration register for DACs	Lect/escrit	8	00010000
81	129	TXDACTEST	8 most significant bits for test mode	Lect/escrit	8	00000000
82	130	PDACTEST	8 most significant bits for test mode	Lect/escrit	8	00000000

Tabla de registros básicos de configuración del Modem

6.3.1.1 Interrupt Enable Register (IER)

Registro de habilitación de interrupciones.

Dirección : 04 h (04 d)

Acceso: Lectura y escritura

Tamaño: 5 bits

Este registro nos permite habilitar, poniendo a 1 el bit correspondiente los 5 tipos de interrupciones que posee el chip. Veamos las distintas posibilidades en la siguiente tabla.

Bit	Identificación	Descripción
0	ERDI_0	Enable Received Data in Channel 0 Interrupt
1	ETHRE	Enable Transmitter Holding Register Empty
2	ELSI_0	Enable Receiver Line Status Channel 0 Interrupt
3	ERDI_1	Enable Received Data in Channel 1 Interrupt
4	ELSR_1	Enable Receiver Line Status Channel 1 Interrupt

Tabla : Interrupt Enable Register

6.3.1.2Interrupt Identification Register (IIR)

Registro de identificación de interrupciones.

Dirección: 02-03 h (02-03 d)

Acceso: Sólo lectura.

Tamaño: 9 Bits

Este registro nos informa sobre el estado de las interrupciones, no es necesario para la configuración inicial del módem pero si necesitamos leerlo para saber que es lo que ha provocado la interrupción y poder actuar en consecuencia.

En la tabla siguiente se puede observar con más detalle las distintas posibilidades de este registro.

IIR Bit	Interrupt Type	Interrupt Source	Reset Control
0..2	Bytes in FIFO of channel 1	New byte available in channel 0	

3..5	Bytes in FIFO of channel 0	New byte available in channel 1	
6	Transmitter empty	Transmitter is ready to accept one byte more	
7	Channel 0 Line Status	Reception status for channel 0 has changed. (See LSR)	
8	Channel 1 Line Status	Reception status for channel 1 has changed. (See LSR)	

Tabla: Interrupt Identification Register

6.3.1.3 Communication Control Register (CCR)

Registro de control de comunicaciones.

Dirección: 25-26 h (37-38 d)

Acceso: Lectura y escritura

Tamaño: 10 bits

En este registro se encuentran los parámetros necesarios para la configuración tanto del transmisor como del receptor. Entre las opciones que nos ofrece cabe destacar:

- Habilitar transmisión en el canal 0, canal 1 o en ambos
- Habilitar el transmisor
- Fijar el tipo de modulación (FSK ó SFSK)
- Fijar la velocidad de transmisión y de recepción.

Para comprobar todas las posibles configuraciones del transmisor y el receptor basta con consultar la tabla siguiente

Bit	Identification	Description
0	RXEN_0	Enable channel 0 reception
1	RXEN_1	Enable channel 1 reception
2	TXEN	Enable Transmitter
3	SFSK/ FSK	Modulation (0 = FSK)
4..6	RXBR	Reception Baud Rate
7..9	TXBR	Transmisión Baud Rate

Tabla: Communication Control Register

Puede observarse que se puede programar diferentes tasas de bit tanto para la transmisión como para la recepción, para conocer las distintas tasas y como configurarlas basta con consultar la tabla siguiente.

Baud Rate	**Codificación Hexadecimal**
300	0
600	1
1200	2
2400	3
4800	4
9600	5

Tabla: Configuración de la tasa de bit

6.3.1.4 Transmmiter frequency registers: TXFA, TXFB y TXFP

Registros de configuración de las frecuencias de transmisión

Direcciones: TXFA 34 h (52 d)

TXFB 35 h (53 d)

TXFP 36 h (54 d)

Acceso: Lectura y escritura.

Tamaño: 8 bits.

Estos tres registros permiten configurar la frecuencia de transmisión para el canal 0 (TXFA), canal 1 (TXFB) y para la señal piloto (TXFP). Los posibles valores de la frecuencia así como su programación se detallan en la tabla siguiente

Código Binario	**Frecuencia (KHz)**

´d0	Reset
´d1	20
´d2	20.5
´d251	145
>´d251	Reservado

Tabla: Códigos de las frecuencias de transmisión

6.3.1.5 Reception Base-Band Power Threshold for channel 0 and 1 (BPT)

Nivel de potencia de recepción para canal 0 y canal 1

Dirección: 29-2A h (41- 42 d)

Acceso: Lectura y escritura.

Tamaño: 16 bits.

En este registro se especifica el valor umbral que se utilizará en el receptor para comparar con la potencia de la señal que le está llegando. Si la potencia que le llega alcanza el valor umbral entonces el receptor ha detectado suficiente potencia para comenzar el proceso de detección de cabecera. Si no alcanza este valor umbral el receptor rechazará todo lo que escuche de la línea.

6.3.1.6 Reception Bit Detection Threshold (BDT)

Nivel de potencia de decisión para recepción de bit.

Dirección: 27-28 h (39- 40 d)

Acceso: Lectura y escritura.

Tamaño: 15 bits.

El valor de este registro ajusta el umbral de decisión utilizado para distinguir entre el símbolo “0” y el “1” (se especifica en complemento 2)

6.3.1.7 Line Status Registers (LSR_0 and LSR_1)

Registros del estado de la transmisión. Estos registros nos proporcionan información acerca del estado en el que se encuentra la transferencia de información. Existen 2 registros, uno para el canal 0 y otro para el canal 1.

Bit	***Identif***	***Descripción***
0	ROE	Reception Over-run Error has occurred in channel 0
1	RBF	Reception Buffer Full
2	RBE	Reception Buffer for channel 0 Empty
3	OPD	Over-Threshold Power Detected in channel 0
4	CHD	Correct Header Detected in channel 0
5..7	RSI	Reception Status Information (channel 0)
8..9	Reserved	

Tabla: Line Status Registers (LSR_0, LSR_1)

A continuación se verá con detalle el significado de cada uno de los bits de estos registros

Bit 0: Un error "over-run" ocurre cuando un nuevo byte ha sido demodulado y no hay espacio libre en la FIFO de recepción para almacenarlo.

Bit 1: El buffer de recepción para el canal 0 está lleno

Bit 2: No hay dato disponible en el FIFO de recepción (para el canal 0)

Bit 3: El receptor ha detectado suficiente potencia en la banda base para comenzar la detección de símbolo en el canal 0.

Bit 4: El receptor ha detectado una cabecera correcta.

Bits 5..7: Nos informa sobre el estado del receptor. La codificación de esta información se muestra en la siguiente tabla.

Hex code.	***State***
0	INITIAL
1	IDLE
2	LISTENING
3	SEARCHING HEADER
4	RECEIVING

Tabla: Codificación del estado del receptor

Una vez recibida una ráfaga debera desactivarse el receptor mediante el controlador , ya que sino se seguiran recibiendo datos aunque no los haya.con el transmisor pasara lo mismo , debe controlarse el envio de bits según la ráfaga que se quiera enviar y después volver el transmisor al reposo.

6.3.1.8Receptor Decimator Register (Mreg)

Este registro nos configura el diezmado de los filtros digitales del receptor. Veamos en la siguiente tabla el significado de sus bits.

Bit	***Ident.***	***Description***
2..0	M0	Value of M for the channel 0 filter 2 ^(M0+1)
3	HBIT0	Enable final HB filter for channel 0
6..4	M1	Value of M for the channel 1 filter 2 ^(M1+1)
7	HBIT1	Enable final HB filter for channel 1

Tabla: Receptor Decimator Register

*M se obtiene en cada canal dependiendo de la frecuencia del canal y la tasa de bits configurada en recepción , siguiendo lo siguiente ; M +1 es la parte entera aproximada por abajo del logaritmo en base 2 de FREC/Tasabits.** aunque el valor apropiado que funciona es menor como se vera en las pruebas.*

6.3.1.9 PLL divider registers (PLL0div, PLL1div y PLLpdiv)

Estos registros nos proporcionan los bits más significativos (desde el bit 8 al bit 1) del registro divisor del PLL en cada uno de los tres canales de entrada (canal 0, canal 1 y piloto).

6.3.1.10 PLL configuration registers (PLL0cfg, PLL1cfg y PLLpcfg)

Estos registros nos proporcionan los bits menos significativos del divisor PLL y dos bits de configuración. Veamos a continuación el significado de sus bits.

Bit	*Ident.*	*Description*
0	DIV LSB	Bit 0 if he divider
1	FU	Divide by 2 on oscillator output
2	FSEL	External PLL enable

Tabla: PLL configuration registers

El último bit de este registro nos permitirá seleccionar que la frecuencia externa de los PLL´s nos la proporcione los filtros externos de los propios PLL´s o bien se le introduzca desde los pines correspondientes (*FEXPLL0,1,2*)

6.3.1.11Attenuation Control Register (Attreg)

Este registro nos permite programar el control de ganancia de los filtros de entrada. En la siguiente tabla se muestra el significado de sus bits.

Bit	*Ident.*	*Description*
0	Att20dB_1	Channel 0
1	Att20dB_2	Channel 1
2	Att20dB_p	Pilot
3	Gan20dB_1	Channel 0
4	Gan20dB_2	Channel 1
5	Gan20dB_p	Pilot

Tabla: Attenuation Control Register (Attreg)

Las atenuaciones estan activas a 0 y las ganancias se activan a 1.

6.3.1.12 Powerdown Control Register (Pwrdreg)

Este registro nos permite configurar el valor de power-down de los filtros de entrada, de los PLLs y de los moduladores sigma-delta. En la siguiente tabla se muestra el significado de cada uno de sus bits.

Bit	*Ident.*	*Description*
0	PTF_1	Channel 0 (filter)
1	PTF_2	Channel 1 (filter)
2	PTF_P	Pilot (filter)
3	Pdown_1	Channel 0 (Sigma-delta modulator)
4	Pdown_2	Channel 1 (Sigma-delta modulator)
5	Pdown_P	Pilot (Sigma-delta modulator)
6	PLLpd1	PLL group 1
7	PLLpd2	PLL group 2

Tabla: Powerdown Control Register (Pwrdreg)

Para un buen funcionamiento del receptor todos los bits deben estar activados salvo los dos mas significativos , que solo deben activarse si utilizamos los PLLs internos.

6.3.1.13 Sigma-Delta observation registers (SDREG y PSDREG)

El registro *SDREG* nos proporciona monitorización permanente de los moduladores Sigma-Delta para los canales 0 y 1, mientras que el registro *PSDREG* nos proporciona la misma información para el canal piloto. En la siguiente tabla se muestra el contenido de los bits del registro *SDREG,* los del registro *PSDREG* son análogos.

Bit	*Ident.*	*Description*
0	asd1clk	Trigger for SD modulators on CH1
1	asd1q	Output of Q channel modulator on CH1
2	asd1i	Output of I channel modulator on CH1
3	asd0clk	Trigger for SD modulators on CH0
4	asd0q	Output of Q channel modulator on CH0
5	asd0i	Output of I channel modulator on CH0

Tabla: Contenido del registro SDREG

6.3.1.14 PLL Observation Register (PLLsout)

Este registro nos proporciona permanente monitorización de los PLLs del chip. Veamos a continuación una tabla donde se detalla el contenido de sus bits.

Bit	***Ident.***	***Description***
0	PLLP	Pilot channel oscillator
1	PLL1	CH1 channel oscillator
2	PLL0	CH0 channel oscillator

Tabla: PLL Observation Register (PLLsout)

6.3.1.15 DAC Test registers DACs: DACTEST, TXDACTEST, PDACTEST

El registro DACTEST nos proporciona la posibilidad de colocar los DACs (Digital-Analog Converters) en modo test. También nos proporciona los 2 bits menos significativos pasados a los DACs en modo test. Veamos a continuación una tabla en la que se detalla el contenido de los bits del registro DACTEST.

Bit	***Ident.***	***Description***
0-1	TXDAC_L	The bits 1 and 0 for the transmitter DAC test
2-3	PDAC_L	The bits 1 and 0 for the pilot DAC test
4	Enable TX 0	High enables tx0 transmitter
5	Enable TX 1	High enables tx1 transmitter
6	TXDACTEST	Test enable for transmitter DAC
7	PDACTEST	Test enable for pilot DAC

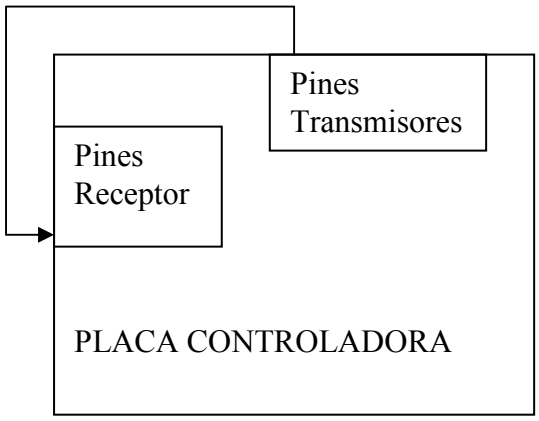
Figura : Registro DACTEST

Los registros *TXDACTEST* y *PDACTEST* nos proporcionan los 8 bits más significativos en el modo test del convertidor Digital-analógico. (DAC)

6.4 Montajes

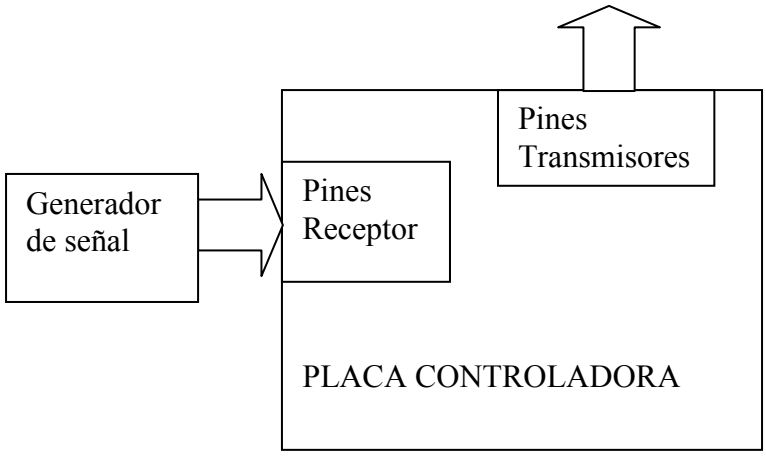
Utilizaremos distintos montajes dependiendo de las pruebas que queramos realizar.
Básicamente se usaran los tres montajes descritos a continuación:

- Para pruebas de recepción básicas utilizaremos el montaje en bucle cerrado siguiente que solo consta de una placa controladora en la que conectamos uno de sus pines transmisores a uno de los pines de entrada del receptor , conectando el otro a tierra.Tambien se puede pasar la señal por un circuito estabilizador mediante un seguidor de tensión(mediante un amplificador operacional) y un divisor para que la entrada al MODEM llegue atenuada.



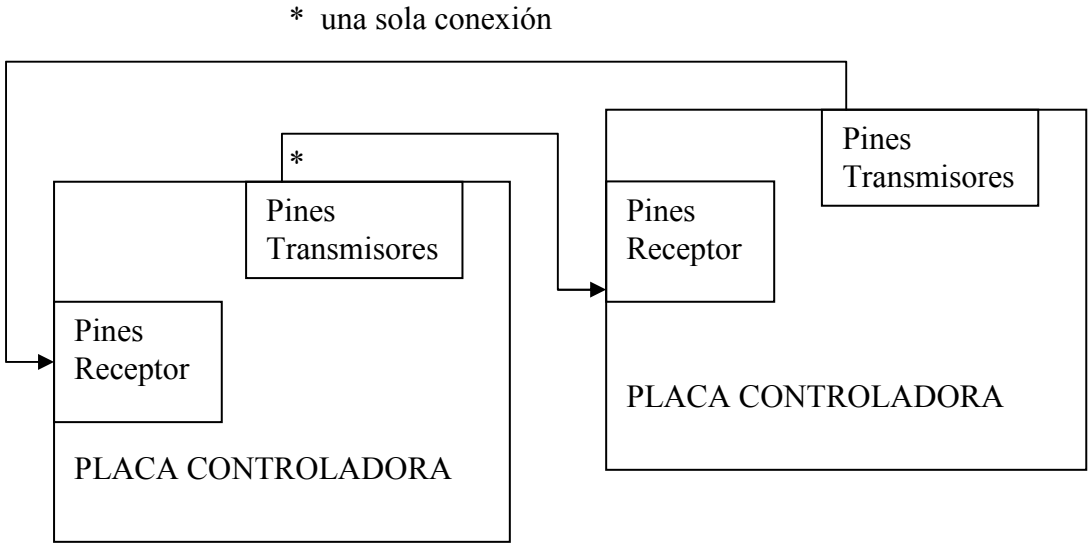
Montaje en bucle cerrado de una placa

- Para realizar pruebas sobre el transmisor y sobre los bloques del receptor por separado , como son los PLLs internos , los sigma-deltas y ver la actuación de los atenuadores en la recepción utilizamos el siguiente montaje que solo requiere una placa , utilizando un osciloscopio digital para analizar las señales a observar.



Montaje auxiliar para probar el bloque analógico de recepción y el transmisor

- En las pruebas de transmisión y recepción por ráfagas utilizamos el montaje siguiente de dos placas controladas desde el mismo PC , cada una por un puerto COM distinto:



Montaje de las dos placas para comunicación

6.5 Procedimientos de configuración

Para llevar a cabo transmisiones a ráfagas o recepciones se necesita configurar el MODEM previamente y ademas realizar según el modo de programación del PIC una carga o recogida de datos de distinta manera.Las instrucciones y datos que se le deben proporcionar al PIC por el puerto serie se describen a continuación expuestas de forma secuencial , ya sea para su carga desde un fichero o para la ejecución desde el modo paso a paso del Panel de control.

6.5.1 Transmisión

Dependiendo del tipo de programación que le hayamos grabado al PIC utilizaremos un método u otro , ya que los archivos de configuración y las aplicaciones de Labview empleadas difieren de un modo a otro:

- Modo externo (primer modo de programación del PIC):
  Utilizamos la aplicación Txautomat de manera que el paso de datos a transmitir se controla por el PC , cargándose primero el archivo de configuración del MODEM , luego transmitiéndose la cabecera necesaria , 5554 (Hex),y posteriormente los datos cogidos uno a uno del archivo correspondiente y pasados al MODEM vía PIC cuando son solicitados , terminándose la transmisión por desactivación del transmisor debido a que no hay más datos que transmitir(el archivo de datos se ha leído entero).
  El archivo de configuración debe atender a los siguientes pasos:
    - Activar el MODEM.
    - Resetear MODEM: Se activa el reset , luego se ponen varias instrucciones de fintest(que no hace nada realmente al no estar en test , FF en hexadecimal)y se desactiva el reset.
    - Configurar los siguientes registros:
      - IER
      - TXFA
      - TXFB
      - TXFP
      - DACTEST para activar los transmisores
      - Introducir en sbmTHR el primer byte de la cabecera(55).
      - Primer byte de sbmCCR
      - Ultimo byte de sbmCCR , activando la transmisión.

  En el archivo de datos primero se colocara el ultimo byte de la cabecera (54) y luego los datos que se vayan a transmitir.

- Modo interno(segundo modo de programación del PIC):
  Mediante la aplicación Cargaprog ,o ejecutando todas las instrucciones de manera secuencial mediante el Panel de control, configuramos el MODEM para realizar la transmisión de la ráfaga cuyos datos y cabecera a transmitir se almacenan en la cola de transmisión implementada en el PIC , de manera que una vez cargado el archivo de configuración, que además contiene los datos , el proceso se realiza automáticamente bajo el control del PIC , desactivándose el transmisor al terminar de transmitir el ultimo dato de la cola.
  El archivo de configuración y datos debe atender a lo siguiente:
    - Activar el MODEM.
    - Resetear MODEM: Se activa el reset , luego se ponen varias instrucciones de fintest(que no hace nada realmente al no estar en test , FF en hexadecimal)y se desactiva el reset
    - Configurar los siguientes registros:
      - IER
      - TXFA
      - TXFB
      - TXFP
      - DACTEST para activar los transmisores
      - Introducir en sbmTHR el primer byte de la cabecera(55).

- Meter los datos a trasmitir en la cola de transmisión empezando por el segundo byte de la cabecera(54) y luego los demas datos por orden (hasta 31 datos mas), para introducir los datos se realiza la escritura en cola(ver manual de programación en el capitulo de Programa del PIC).
- Primer byte de sbmCCR
- Ultimo byte de sbmCCR , activando la transmisión

Esto vale para los modos de transmisión a ráfagas , para comprobar simplemente el funcionamiento del transmisor se puede utilizar el Panel de control , siguiendo los siguientes pasos para configurarlo:

- Activar el MODEM.
- Resetear MODEM.
- Configurar los siguientes registros:
  - TXFA
  - TXFB
  - TXFP
  - DACTEST para activar los transmisores
  - Introducir en sbmTHR el primer byte de la cabecera(55).
  - Primer byte de sbmCCR
  - Ultimo byte de sbmCCR , activando la transmisión
- Posteriormente cambiamos el dato a transmitir en sbmTHR al valor que queramos, así como las frecuencias de los canales, viendo su funcionamiento.

Mediante este procedimiento paso a paso se puede ver si se activa la recepción (montaje en bucle cerrado), configurando previamente el bloque de recepción y graduando mediante el osciloscopio las frecuencias de transmisor y receptor.

6.5.2 Recepción

Para los dos modos de programación se opera de la misma manera ya que una vez configurado el MODEM en espera de recibir datos la aplicación Labview utilizada espera a que se reciban los datos del MODEM vía PIC y es el que ordena luego al receptor parar de recibir (tanto automáticamente como manualmente), por lo que da igual si se reciben más datos de los requeridos pues se descartan.

Para esto se pueden utilizar dos métodos, según la aplicación Labview empleada:

- Panel de control: para controlar paso a paso el proceso y parar manualmente el receptor cuando nos convenga, ignorando los datos sobrantes.
  La cadena de instrucciones básicas a tener en cuenta es:
  - Activar el MODEM.
  - Resetear MODEM.
  - Configurar los siguientes registros:
    - sbmBDT , con un valor de al menos 100 para evitar reconocer ruido.
    - sbmBPT con valor bajo
    - Attreg
    - Pwrdreg , en función de los PLLs que se utilicen se hará:
      - PLLs internos: PLL0div,PLL1div,PLLpdiv

- ❖ PLLs externos:PLL0cfg,PLL1cfg,PLLpcfg y activación de las señales PLLs externas pertinentes generadas por el PIC(ver manual de Programación).
- Mreg
- Primer byte de sbmCCR
- Ultimo byte de sbmCCR , activando la transmisión

En la última configuración (sbmCCR 2º byte) se le comunica a la aplicación el numero de datos que queremos recibir y donde almacenarlos , para de esa manera dejar el receptor en espera y cuando se reciban los datos se almacenan , posteriormente se desactivara el receptor , ordenandolo nosotros(escribir en sbmCCR 2º byte , la configuración correspondiente).

- Rxautomat : carga el archivo de configuración en el MODEM(lo configura para recibir) y lo deja en estado de espera hasta que se reciben el número de datos requerido , tras lo cual automáticamente se desactiva el receptor.
  El archivo de configuración debe tener en cuenta las siguientes instrucciones ordenadas como sigue:
  - Activar el MODEM.
  - Resetear MODEM: Se activa el reset , luego se ponen varias instrucciones de fintest(que no hace nada realmente al no estar en test , FF en hexadecimal)y se desactiva el reset
  - Configurar los siguientes registros:
    - IER
    - sbmBDT , con un valor de al menos 100 para evitar reconocer ruido.
    - sbmBPT con valor bajo
    - Attreg
    - Pwrdreg , en función de los PLLs que se utilicen se hará:
      - ❖ PLLs internos: PLL0div,PLL1div,PLLpdiv
      - ❖ PLLs externos:PLL0cfg,PLL1cfg,PLLpcfg y activación de las señales PLLs externas pertinentes generadas por el PIC(ver manual de Programación).
    - Mreg
    - Primer byte de sbmCCR
    - Ultimo byte de sbmCCR , activando la transmisión

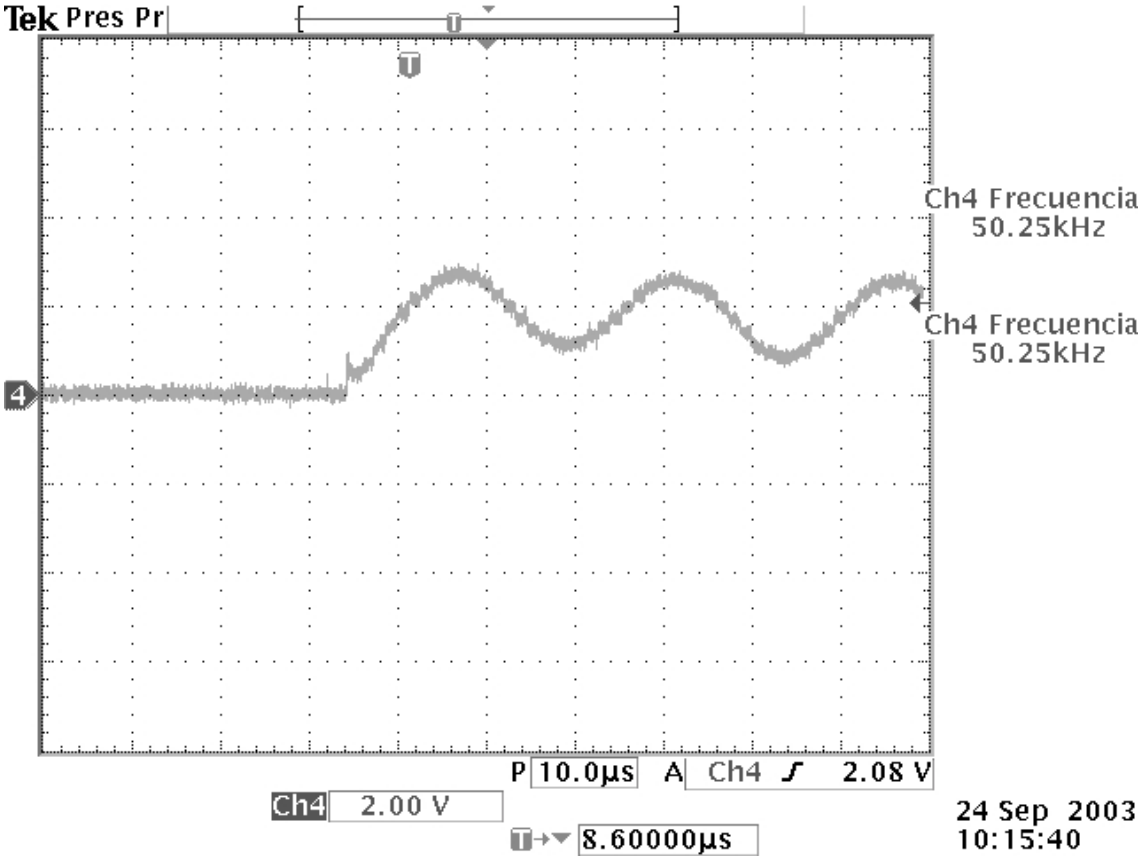
Especificando en Rxautomat el número de datos que queremos recibir y donde almacenarlos estos son almacenados en el mencionado archivo y se termina la ejecución de la aplicación.

6.6 Resultados experimentales

Mediante los procesos de configuración se realizaron pruebas sobre la funcionalidad del MODEM , obteniéndose los siguientes resultados en cada uno de sus partes testadas:

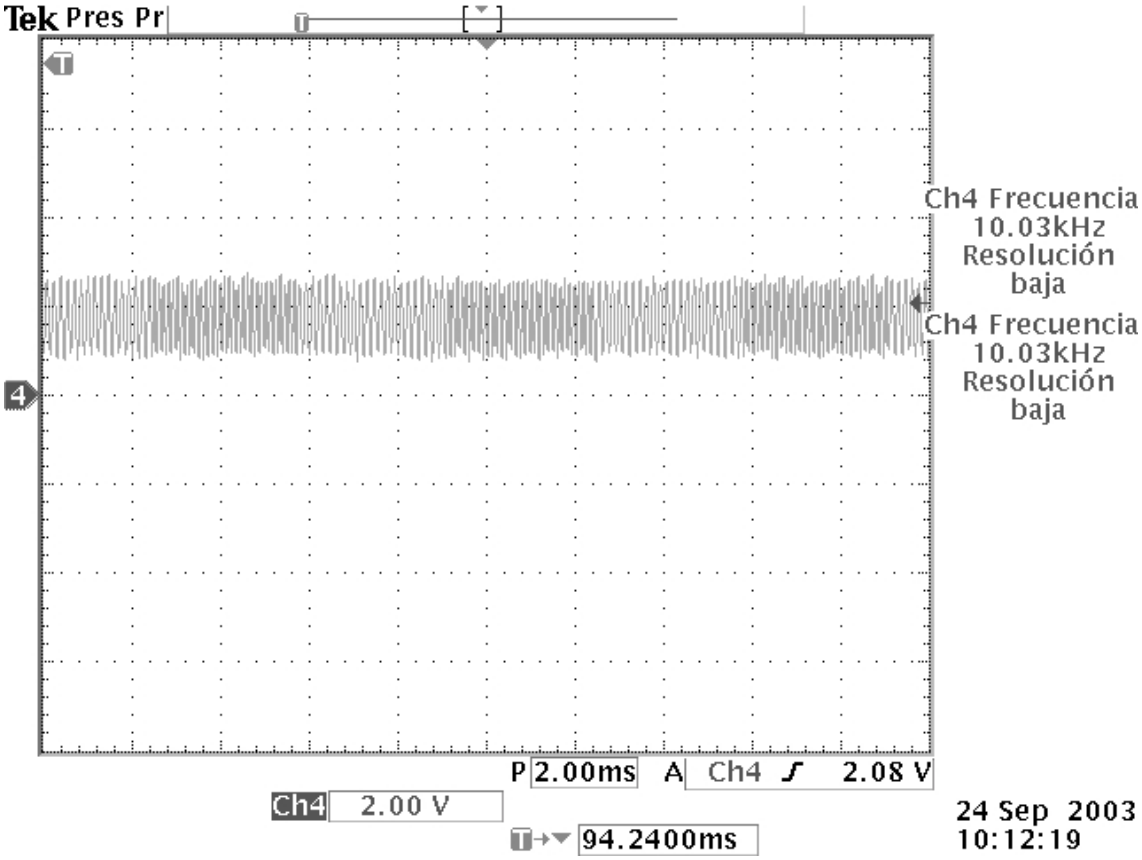
- Bloque de transmisión:

Configurando el MODEM para una transmisión a ráfagas se procedió a comprobar la realización de estas, en el montaje segundo. Así como las interrupciones del MODEM al PIC para el paso de datos a transmitir y el funcionamiento de los transmisores.
Se utilizo la aplicación de Txautomat y la Cargaprog(modos externo e interno) con identicos resultados:



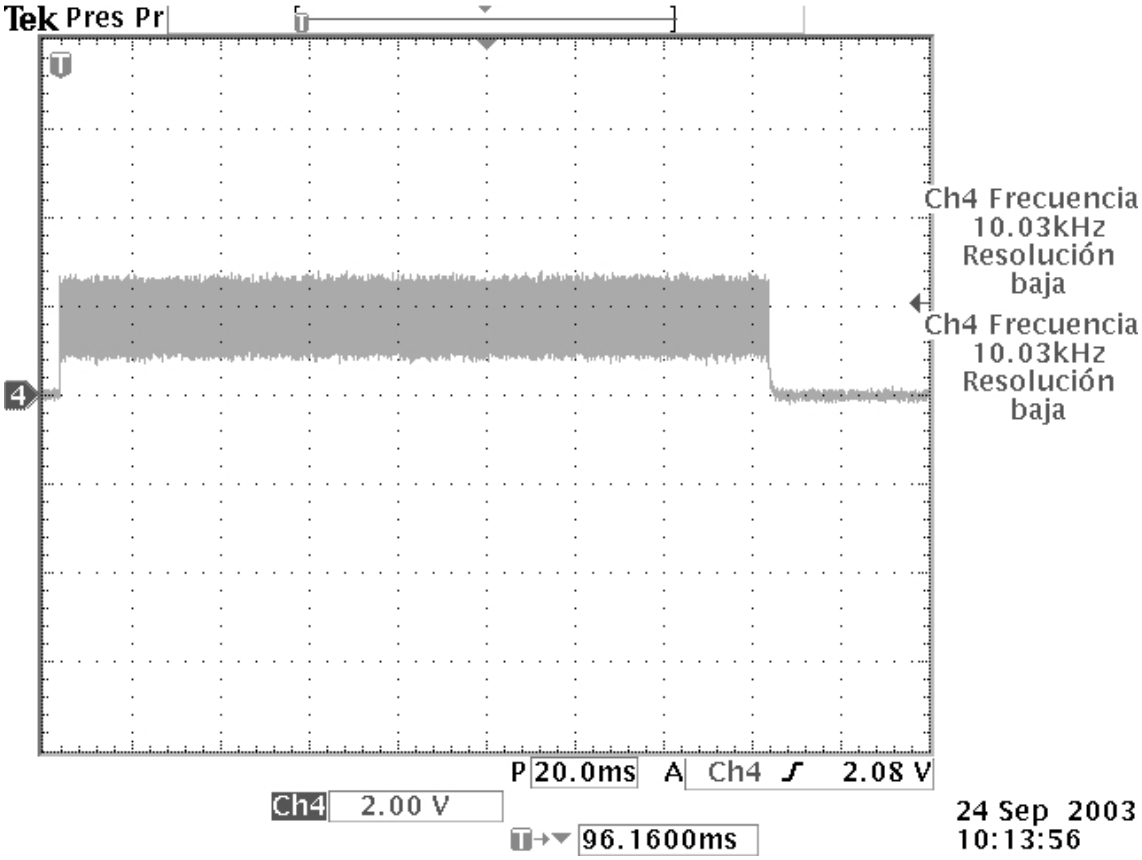
Captura del comienzo de la transmisión de una ráfaga

Como se puede apreciar en la captura la señal transmitida proviene de una cuantización de niveles (escalonado) tal y como debe ser al provenir de un DAC .
Las frecuencias de transmisión de los canales (0 y 1) se ciñen a las programadas previamente en los registros de configuración , siendo como deben ser.

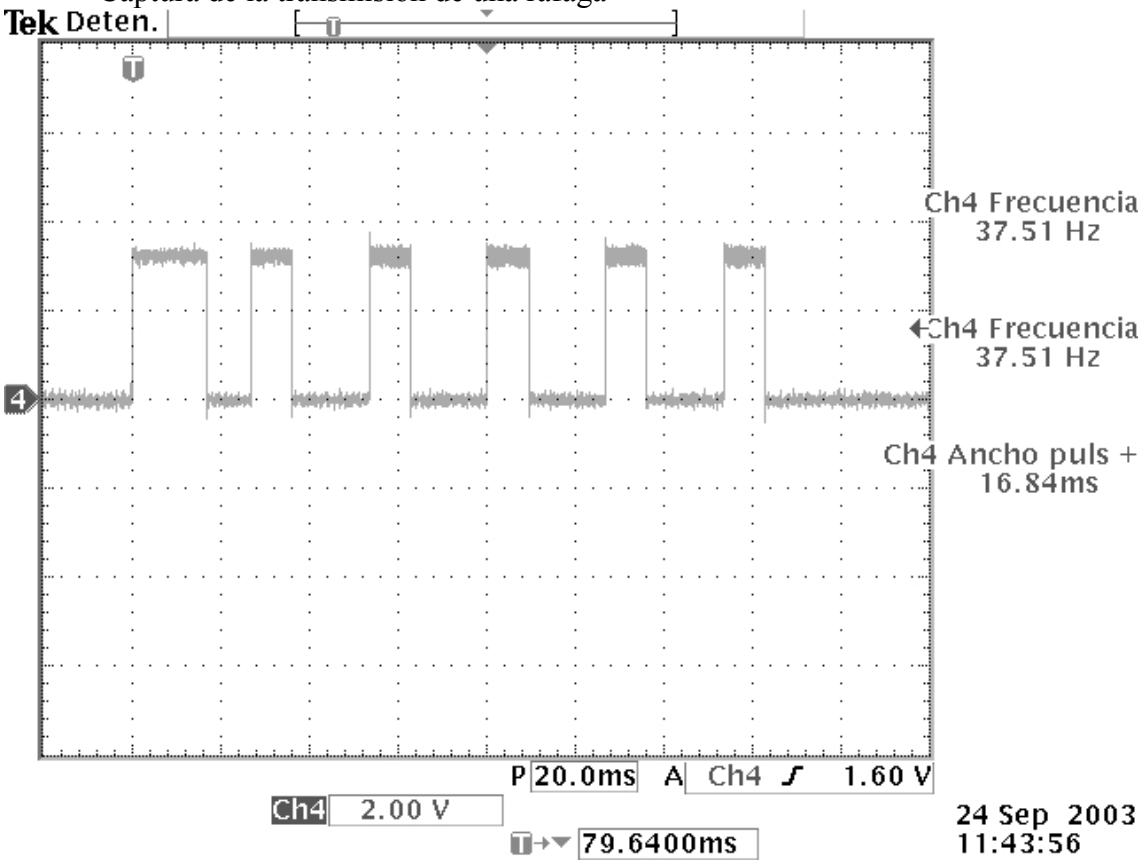


Captura de la señal transmitida durante la ráfaga

Podemos observar en la anterior captura la transmisión un byte de la cabecera compuesto por unos y ceros consecutivos , ya que se pueden apreciar perfectamente los cambios de frecuencia en la señal en intervalos que duran lo mismo , analizamos durante el experimento las frecuencias y vimos que coincidían con las programadas , y que no se desvían mas de la cuenta , funcionando perfectamente el bloque de transmisión.
En la siguiente captura podemos apreciar la transmisión de una ráfaga completa , si bien al ser bastante extensa no podemos apreciar los bytes , si podemos ver que el proceso se lleva a cabo como debe , ya que una vez terminado de transmitir el ultimo dato se desactiva el transmisor.En la captura podemos ver que se transmiten 6 bytes siendo los dos primeros la cabecera y los otros 4 los datos usando una tasa de transmisión de 300 bps :



Captura de la transmisión de una ráfaga



Captura de las interrupciones del MODEM solicitando dato a transmitir en ráfaga de 4

Con la configuración utilizada se transmite un byte cada 26,6ms por lo que al iniciar la transmisión de uno ya pide el siguiente , el primer pulso es más grande ya que se tarda mas en atender la petición del segundo byte de la cabecera por el programa al estar configurando todavía el MODEM(activando el transmisor).La ultima petición hace que se desactive el transmisor , no obstante este termina de transmitir el byte que estaba transmitiendo antes de desactivarse (esto es una hipótesis que parece ser cierta debido a los tiempos , además seria lo lógico pues sino no hay manera de parar una transmisión sin controlar aparte el tiempo de transmisión de los bytes).

- Bloque de Recepción:

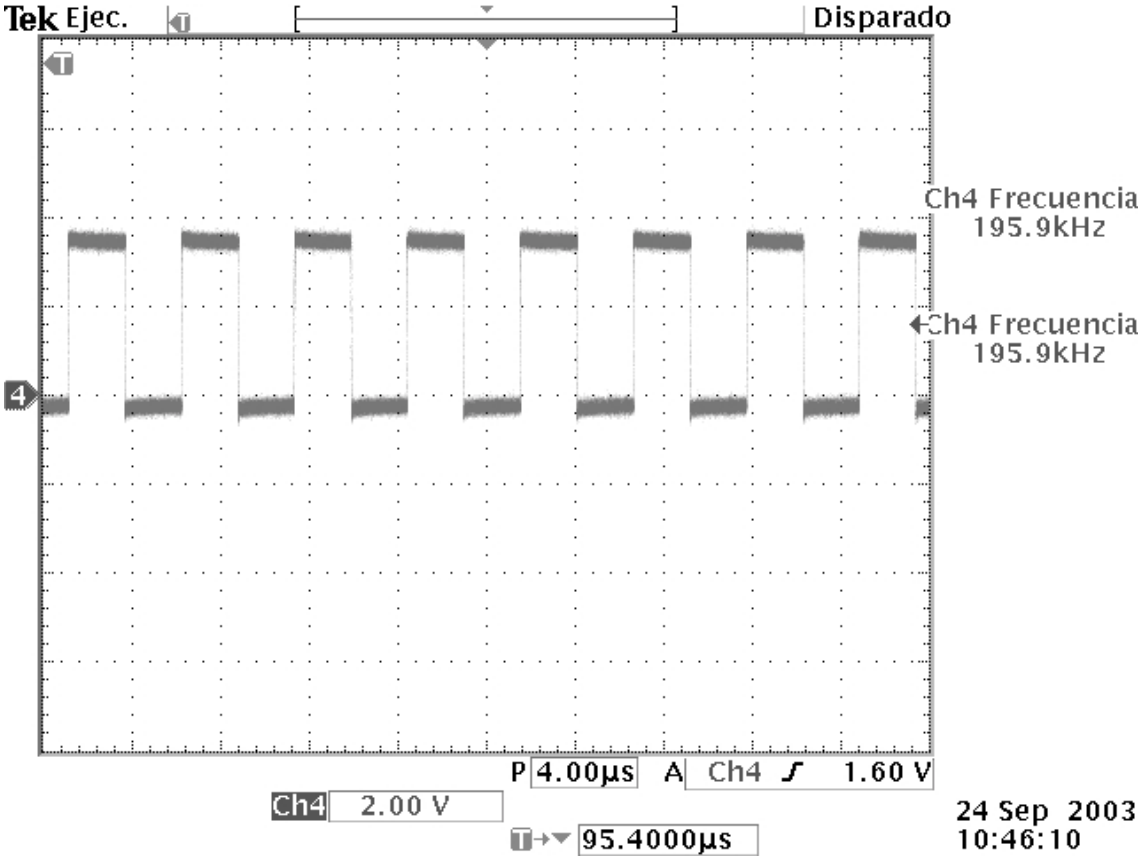
Se procedió mediante el segundo montaje a comprobar las caracteristicas de los bloques internos del receptor , así como las señales proporcionadas por el PIC para su utilización como señales PLLs externas.Para ello se utilizo el Panel de control.
Los resultados fueron satisfactorios en cuanto al funcionamiento de los bloques del receptor y la generación de las frecuencias por el PIC como se puede ver a continuación:

- Bloques PLLs internos del receptor:

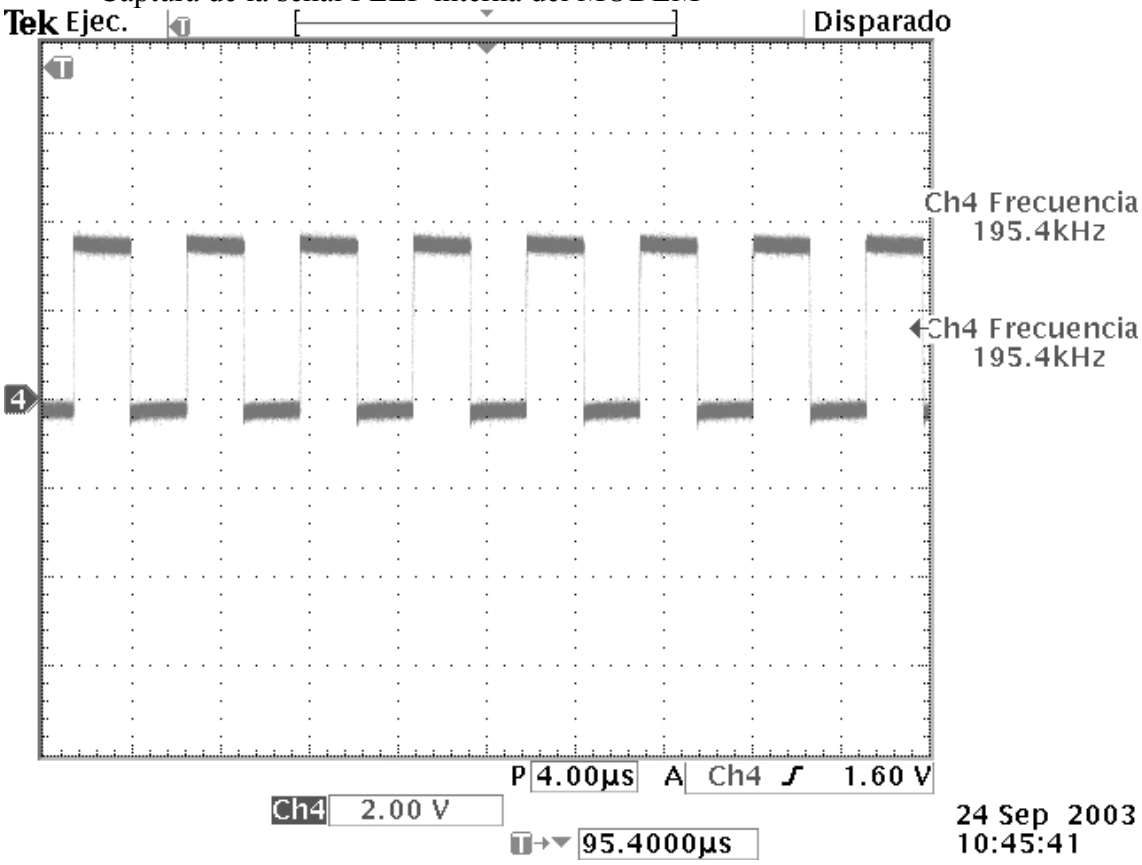
Para ver el funcionamiento de los PLLs internos del receptor se procedió de la siguiente manera:

- Activar el MODEM.
- Resetear MODEM.
- Configurar los siguientes registros:
  - Pwrdreg
  - PLLidiv , según el PLL a observar
- Realizar una lectura continua (volcado de registro) del registro PLLsout , observando los bits correspondientes a las señales PLL internas que queremos estudiar.
- Terminar con la lectura continua del registro.

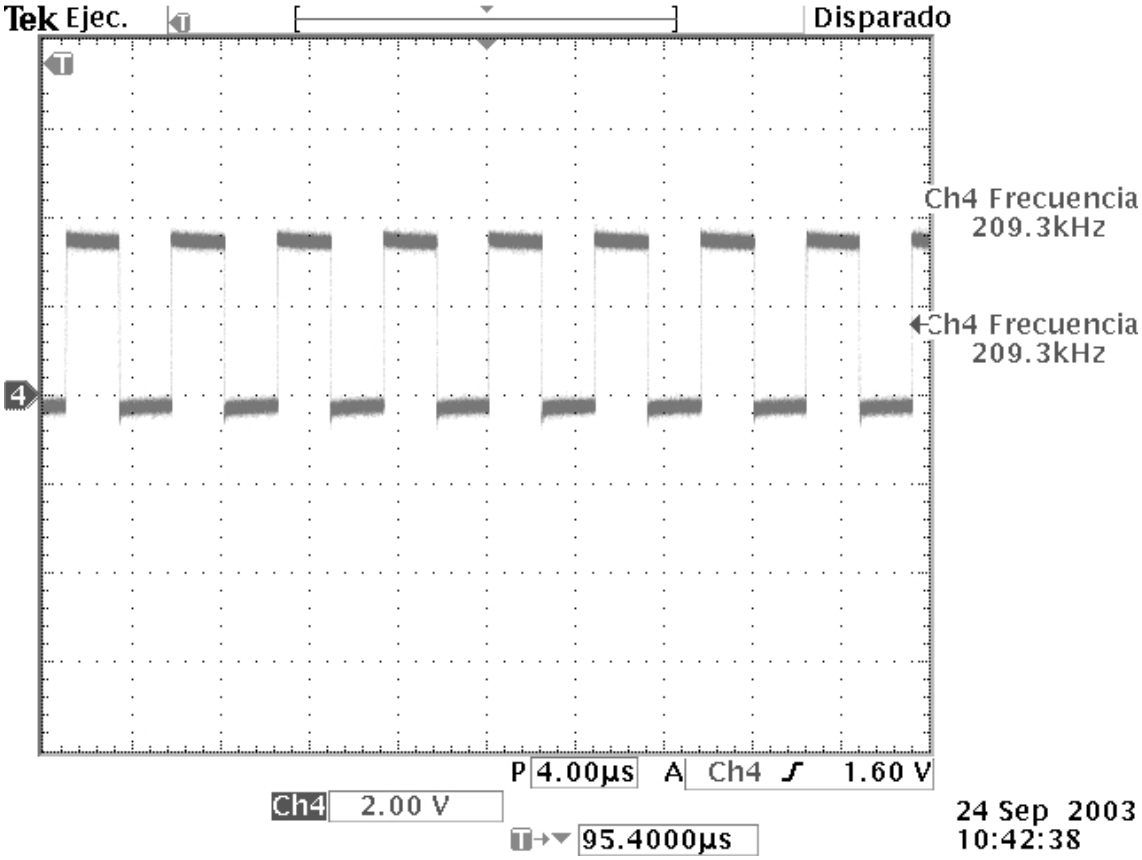
Desgraciadamente al no tener documentación precisa sobre los registros PLLidiv no se sabe la tabla de programación de frecuencias exacta de los PLLs internos , por lo que al usarlos en recepción se mirara la frecuencia a la que están trabajando y se programara en el transmisor (nunca al revés).
Las capturas se realizaron utilizando las frecuencias programadas por defecto(desde el reset) de los PLLs.



Captura de la señal PLLP interna del MODEM

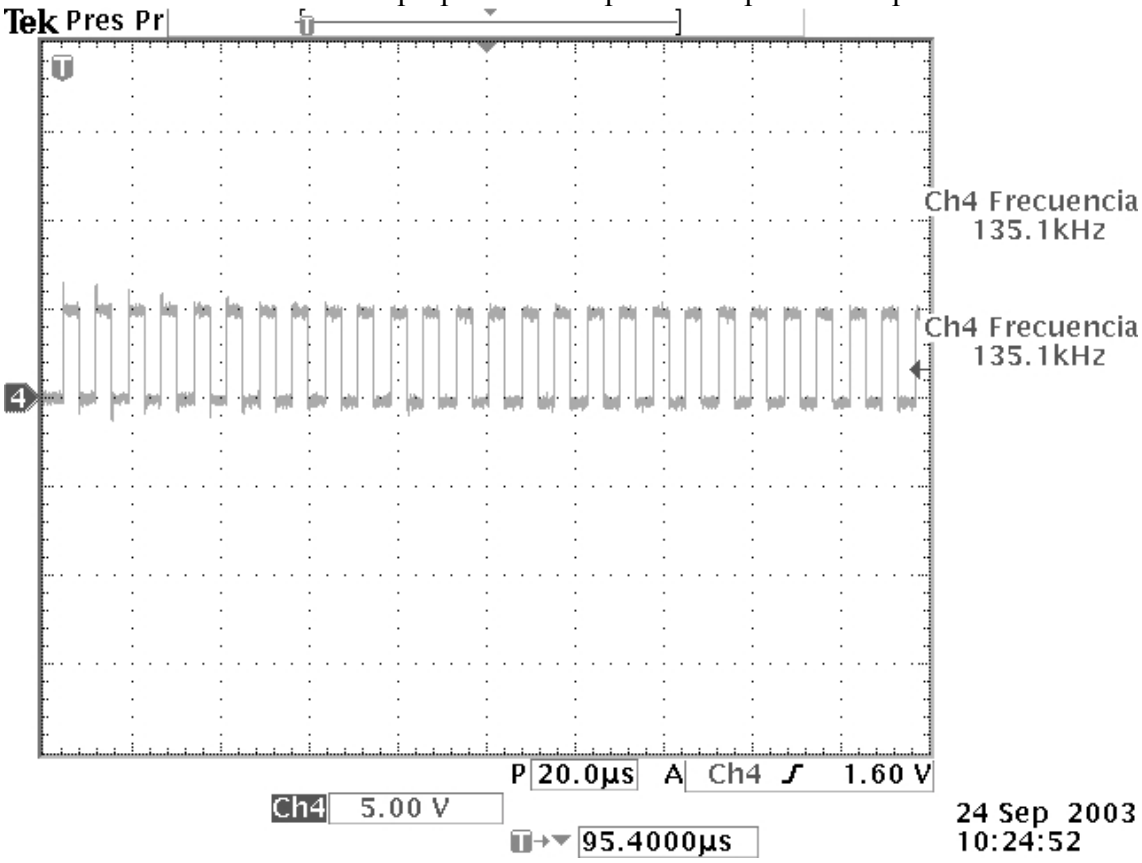


Captura de la señal PLL1 interna del MODEM

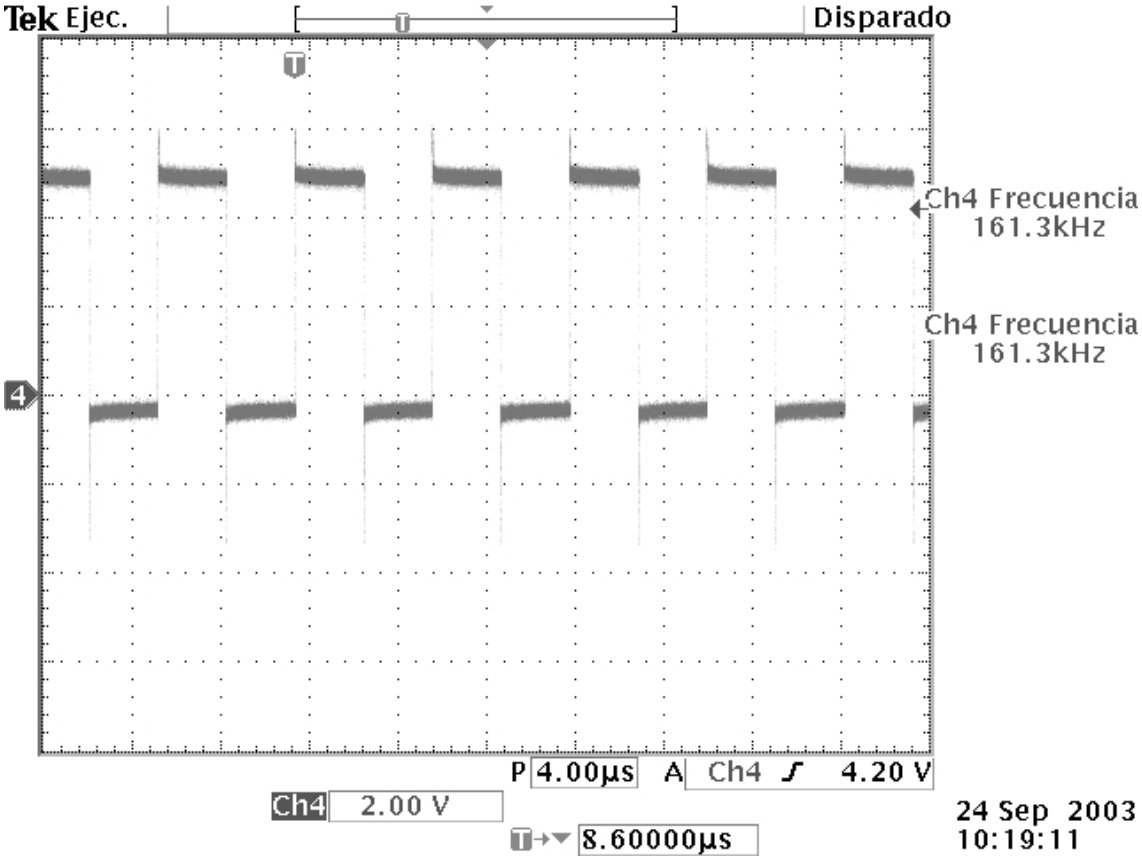


Captura de la señal PLL0 interna del MODEM

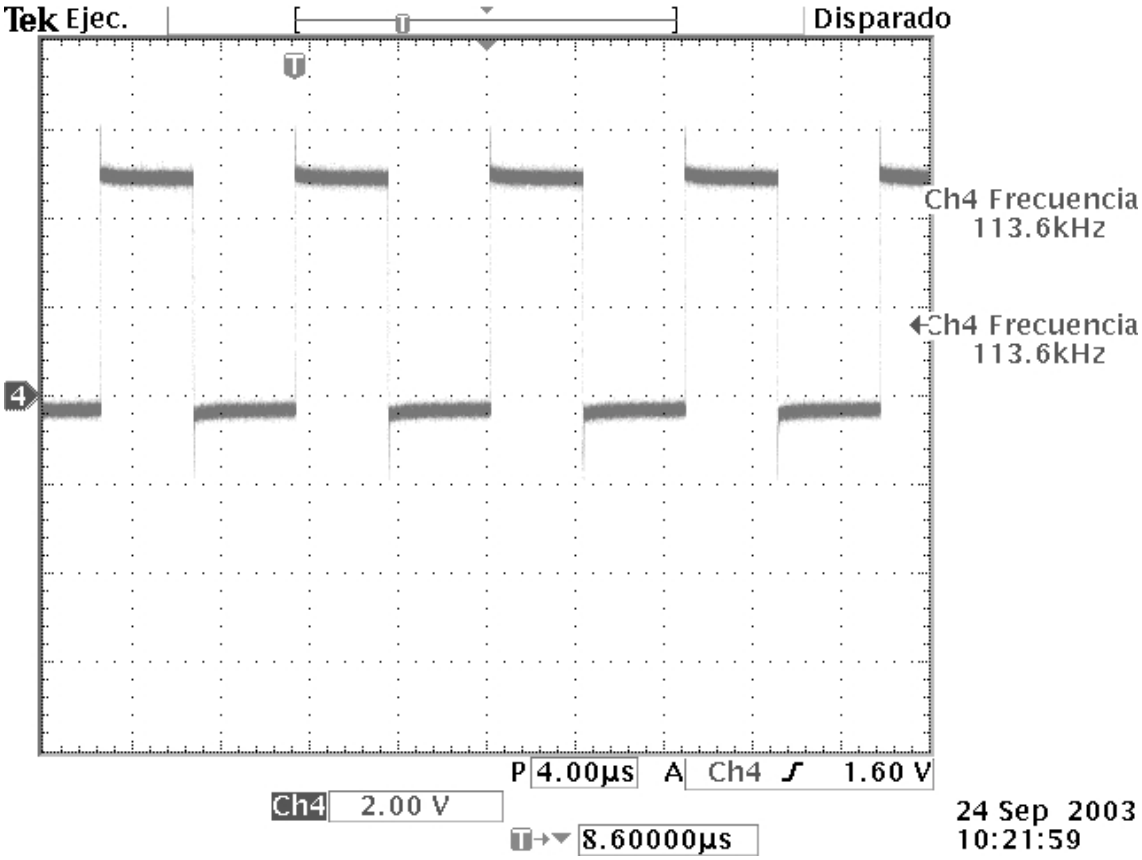
- Señales PLLs externas proporcionados por el PIC para el receptor:



Captura de la señal PLL0 externa generada por el PIC



Captura de la señal PLL1 externa generada por el PWM del PIC



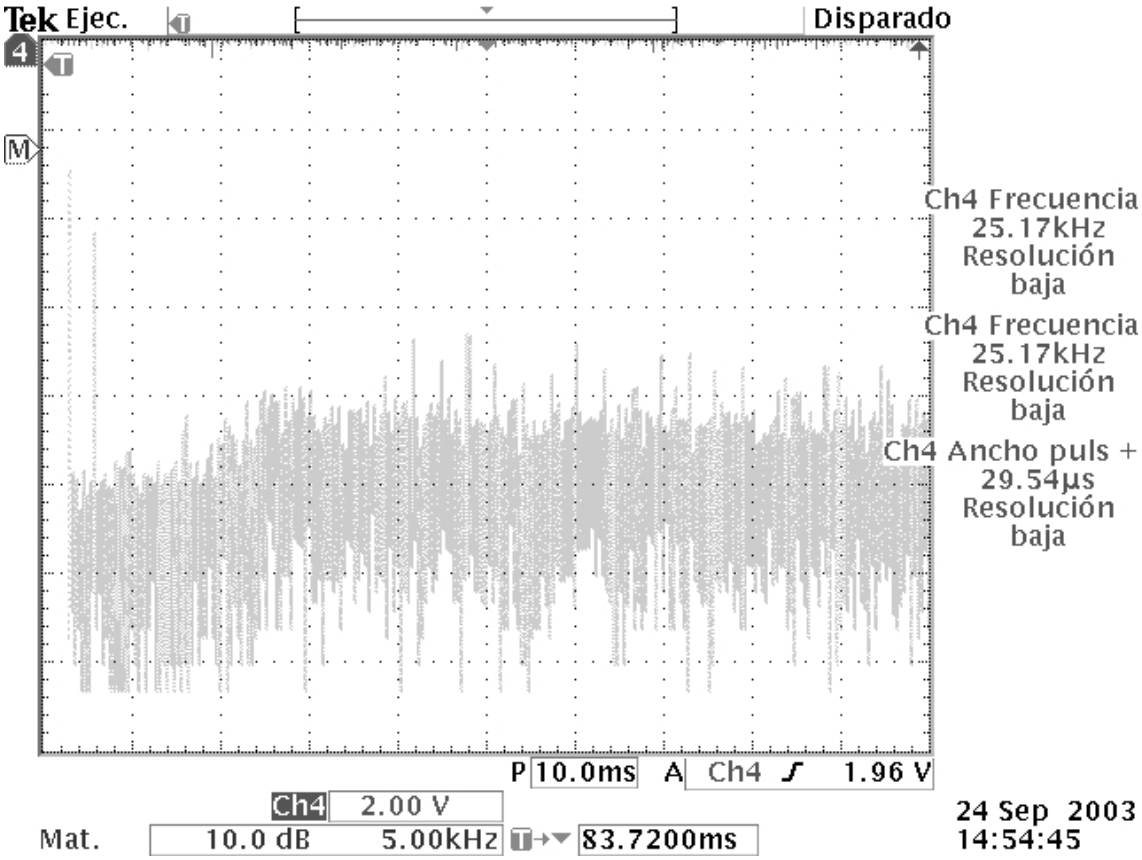
Captura de la señal PLLP externa generada por el PIC

Las señales PLLs externas se activaron por separado y de dos en dos salvo la Piloto y el canal 0 a la vez ya que el PIC no nos permite ese modo de trabajo.Para ello simplemente se ejecutaron las instrucciones pertinentes seleccionadas mediante el Panel de control (ver el manual en el capitulo de Aplicaciones Labview).
El resultado fue satisfactorio , obteniéndose frecuencias precisas para ciertos valores(el resto debido a la forma de obtenerlas sufre cierta variación) , por lo que para las pruebas de recepción utilizando frecuencias PLLs externas se utilizaron valores de frecuencia de transmisión acordes a estas , ya que las frecuencias de transmisión son más flexibles de programar.

- Bloque analógico de recepción :

  Prueba del funcionamiento del bloque analógico de recepción al tener a la entrada una señal desviada 1KHz de la utilizada en el PLL interno(generada externamente por el generador de señales), probando en ella el uso de los registros de atenuación , conforme se aleja de la frecuencia a recibir decrece en potencia (segundo pico observado desde la izquierda, desplazado de la frecuencia del PLL 1 KHz).los registros de atenuación funcionan perfectamente , atenuando cuando los bits correspondientes están a 0 y amplificando cuando los bits pertinentes están a 1.

Los convertidores sigma-delta funcionan . Como se pudo ver al observarse la típica característica de los sigma-delta con el osciloscopio.
Para su estudio se procedió de igual manera que para el estudio de los PLLs internos , pero observando los bits pertinentes del registro SDREG , configurando los PLLs internos en función a la señal desplazada que le proporcionamos a la entrada.



Captura de la FFT del canal en el que se probo el funcionamiento de los sigma-delta

- Recepciones :
  Distinguimos varios casos probados por separado de la recepción de datos transmitidos por otro o por el mismo MODEM.
  Los casos son los siguientes:

    - Recepción en bucle cerrado:Utilizamos el primer montaje (1 sola placa) y controlamos el proceso mediante la aplicación Panel de control , ya que se trata de operar de manera iterativa hasta encontrar la manera de que el receptor se active , comprobando la cabecera que se va a utilizar en las siguientes pruebas y las frecuencias de los canales.
      En este caso se probo la recepción mediante la utilización de frecuencias externas con resultado satisfactorio(dentro de unos limites) para las tasas de 300 y 600 bps , no lográndose recibir para mayores tasas(ver en detalle mas adelante).También se probo a recibir con las frecuencias internas de PLL sin resultados satisfactorios ya que no se consiguió recibir (usando la misma configuración salvo el detalle de la frecuencia del PLL , que de por si funciona bien) .
    - Recepción de un MODEM a otro usando PLLs internos: Utilizando el tercer montaje anteriormente expuesto(2 placas) y mediante el control de cada una mediante la aplicación pertinente; es decir , el que vaya a recibir con Rxautomat o el Panel de control , y el que vaya a transmitir con Txautomat o con el Panel de control(ya que se trata de comprobar la recepción primero y una vez asegurada la configuración se automatiza el proceso), llevamos a cabo el proceso. Como en el caso de una sola placa no se consiguió recibir , esto puede ser por una mala sintonización entre frecuencias (aunque se probo un amplio rango sobre la frecuencia de trabajo)o por una mala conexión interna dentro del propio MODEM de la señal interna de PLL y el bloque de transmisión (esto solo es una hipótesis debido a que con señal externa el bloque funciona).
    - Recepción de un MODEM a otro usando PLLs externos: Utilizamos el mismo montaje y aplicaciones que en el caso anterior pues solo cambia la configuración .Asegurando la configuración (basada en el caso de una sola placa pero modificando el umbral de detección de bit) conseguimos operando con el modo manual activar el receptor, es decir que reciba , pero con consecuencias tan imprevistas e incluso más extrañas que en el caso de bucle cerrado.

Se detalla el comportamiento del receptor del MODEM en los dos casos que llego a funcionar , mostrando la configuración y los datos transmitidos y los recibidos y una posible hipótesis de la causa de este funcionamiento:

- Recepción en bucle cerrado usando frecuencias de PLL externas:
    - Utilizamos el canal 1 para transmitir y recibir , por lo que configuramos el PIC según la instrucción de generación de frecuencia.
    - Las configuraciones del transmisor y el receptor son:

Registro	Valor decimal	Valor hexadecimal
Mreg	204 M=4 aunque deberia ser M=6	CC
TXFB	42	2A
sbmCCR(byte 1)	0	00
sbmTHR	85	55

sbmBDT(byte 1)	30	1E
sbmBDT(2º byte)	0	00
sbmBPT(byte 1)	0	00
sbmBPT(2º byte)	65	41
PLL1cfg	0	00
Pwrdreg	63	3F
Attreg	248	F8
DACTEST	48	30
sbmCCR(2º byte)	15	0F

Tabla de registros utilizados

Al activar el transmisor y el receptor a la vez , estando enviando el primero de los bytes de la cabecera el estado del canal 1 ,que puede verse al leer el registro correspondiente a la dirección 14(decimal) del MODEM, pasa a ser de 24(hex) a 64(hex) que es esperando cabecera . De manera que cuando se envía el segundo byte de la cabecera , metiendo 84 en el registro sbmTHR, el estado de la línea pasa a ser 9B , es decir empieza a recibir , por lo que se concluye que el bloque de recepción funciona bajo estas condiciones , recibiéndose exactamente el valor transmitido(84) , sin embargo al tratar de recibir otros datos ocurre un extraño fenómeno , que no puede deberse al ruido ya que utilizamos un umbral relativamente alto en la configuración y al ser siempre el mismo resultado no se trata de algo aleatorio , los datos se reciben "codificados " , así si se transmite un 16 se recibe un 16 siempre pero si se transmite un 64 se recibe siempre un 4 (así con cada dato posible a recibir entre 0 y 255) .esto no puede deberse a una mala configuración del registro decimador Mreg ya que se configuro a base de experimentación a un valor menor que el indicado(según las indicaciones que tenemos de ese registro) ya que es la única manera que reciba , también se escogió el valor mayor con el que funcionaba y que permitía obtener una recepción estable(como hemos visto antes , se transmite un dato y se recibe otro siendo los mismos siempre en cada experimento) , ya que para valores menores de M se obtiene una cadena de valores que se repiten cíclicamente basados en el valor estable que se consigue con el valor correcto de M.
Al repetir el experimento con una tasa mayor de bits , 600 bps, M se reduce esta vez a 2 , en cuyo caso (procediendo de la misma manera) se obtienen los mismos resultados de datos recibidos codificados.
A continuación se detalla en una tabla algunos de estos datos "codificados" que se obtuvieron , siendo siempre los mismos resultados :

Dato Transmitido (hex)	Dato Recibido (hex)
54	54
10	10
40	04
03	C1
03	81
80	02
FF	FF
00	00
37	D9
36	D8
35	59
34	58
33	99
32	98

31	19
30	18
C0	06
08	20
04	40
02	80
01	01
20	08

Tabla de datos “codificados”

Es curioso comprobar que el número de bits activos coinciden en los datos recibidos y los transmitidos , si bien su orden a veces no es el mismo , aunque tienen una correspondencia fija , si bien no se trata de una desplazamiento de todos los datos o una inversión , sino de una especie de codificación .La causa de esta es desconocida .

- Recepción de placa 1 a placa 2 utilizando frecuencia externa del canal 1:
  - Utilizamos el canal 1 para transmitir y recibir , por lo que configuramos el PIC según la instrucción de generación de frecuencia.
  - Las configuraciones del transmisor y el receptor son:

Registro	Valor decimal	Valor hexadecimal
Mreg	204 M=4 aunque debería ser M=6	CC
TXFB	42	2A
sbmCCR(byte 1)	0	00
sbmTHR	85	55
sbmBDT(byte 1)	10	0A
sbmBDT(2º byte)	0	00
sbmBPT(byte 1)	0	00
sbmBPT(2º byte)	65	41
PLL1cfg	0	00
Pwrdreg	63	3F
Attreg	248	F8
DACTEST	48	30
sbmCCR(2º byte)	15	0F

Tabla de registros utilizados

En la placa 1 se configuran los registros del transmisor y se van cambiando los valores a transmitir metiéndolos en sbmTHR y en la placa 2 se configura el receptor dejándolo en espera . Utilizamos la misma configuración que en caso de bucle cerrado pero con la peculiaridad de tener que bajar el umbral de bit para la detección , ya que al tratarse de placas diferentes aun con tierra común los niveles son distintos(por construcción , tolerancias y valores poco afortunados de resistencias en algunos caso junto con el extraño efecto de subida de tensión del segundo regulador , aunque esto afecta a todas las tensiones a la vez , pero puede ser diferente en cada placa) , de hecho el transmisor tiene una señal un poco más débil debido a la construcción de la placa.

Los resultados son los mismos en cuanto a que se consigue recibir en las mismas tasas de bit , pero lo que se recibe ya no esta codificado unívocamente , sino que además presenta un cierto desplazamiento cíclico(no se puede estabilizar con un valor mayor de M ya que entonces dejamos

de recibir), si bien los ciclos si son constantes , se trata pues de una degeneración del comportamiento anterior provocada posiblemente por las diferencias entre las placas y algún factor adicional como pueda ser el deterioro de la primera placa que se construyo hace más tiempo con peor técnica.

Podemos ver como ejemplo de recepción uno de estos ciclos , el del segundo byte de la cabecera 54(hex) , al activarse la recepción se nota claramente que en algún momento(al estar transmitiendo continuamente el mismo byte) se recibe el 54 , leyendo los sucesivos cambios en la reopción nos encontramos con algo así:

8A-45-2A-54-A8-51-

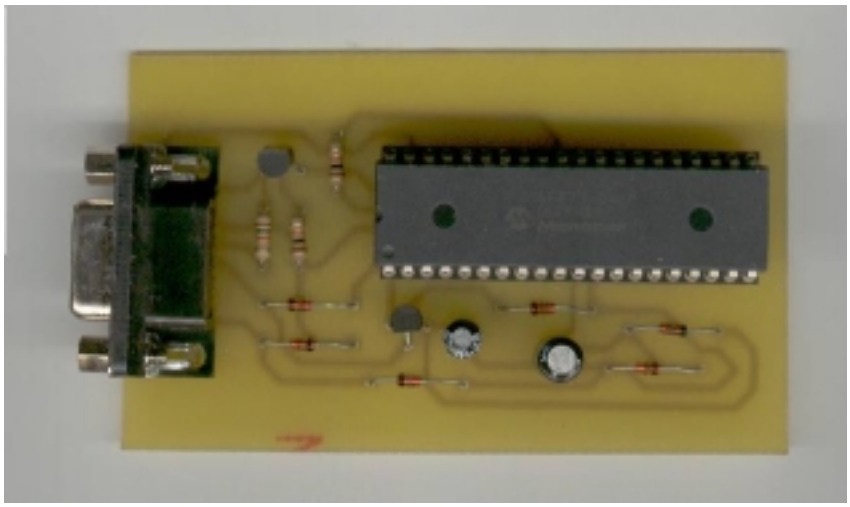
Que puede no ser la cadena completa pero nos da una idea de lo que esta sucediendo en el receptor.
De nuevo se observa la conservación del número de bits activos en dato transmitido y recibido , lo que nos impulsa a creer que se trata de algún tipo de fallo de sincronismo , junto con una posible codificación.

7. ANEXOS

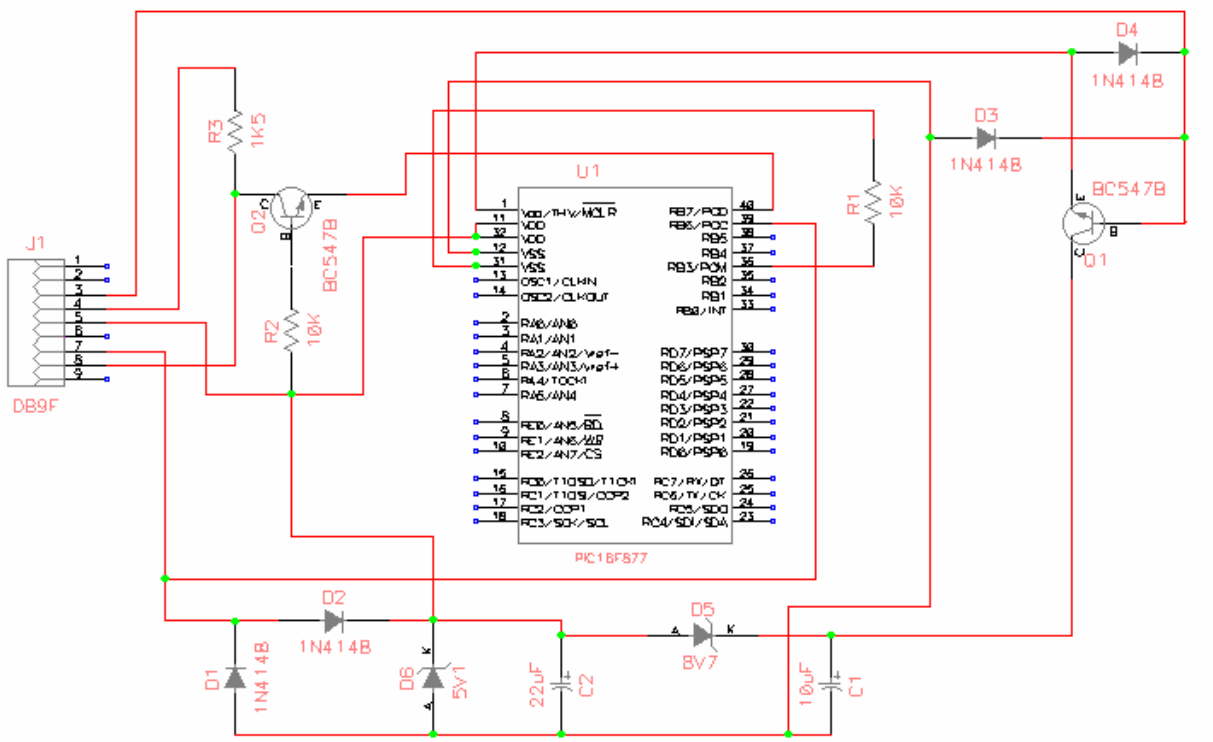
7.1 Placa programadora de PIC por puerto serie

Está placa es un programador JDM, y sus dos principales características son:

- Alimentación necesaria para la grabación de alto voltaje (comprendida entre 12 y 14 voltios), proporcionada por el PC.
- Grabación serie síncrona, utilizando la patita **RB6/PGC** para la introducción desde el PC de los impulsos de reloj y la patita **RB7/PGD** para los datos.



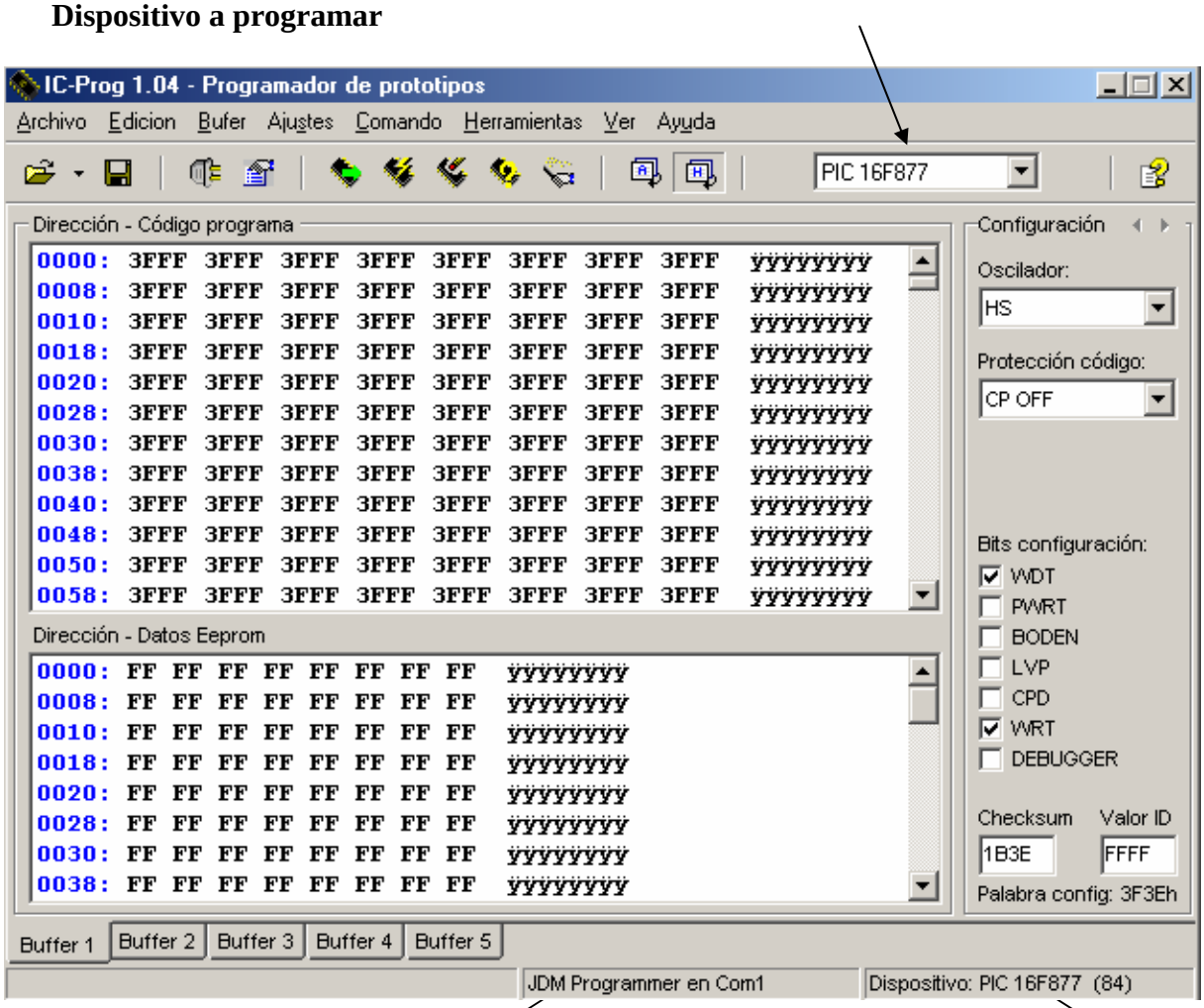
ESQUEMA DEL CIRCUITO GRABADOR



SOFTWARE DE GRABACIÓN

El software de grabación utilizado es un programa de sencilla utilización y de uso gratuito, disponible en la pagina web **ic-prog.net** , para su correcto funcionamiento hay que seleccionar el tipo de dispositivo que se va a grabar y las características hardware que se van a utilizar.

Dispositivo a programar



Tipo de hardware **Dispositivo a programar**

Después de la configuración, hay que cargar el archivo con el programa fuente, es importante que el archivo este compilado (MPLAB) con formato hexadecimal INHX8M, que tienen extensión .HEX.

Hay que tener en cuenta que el programa está diseñado para funcionar como grabador del modelo PIC16F84 (gama baja), por eso si introducimos el código fuente sin compilar este programa transforma las líneas de código adaptándolas al modelo. Si trabajamos con el PIC16F877 (gama media), tenemos que cargarlo ya compilado y en formato hexadecimal.

Antes de grabar el programa hay que seleccionar los bits de configuración:

- **Oscilador**:
  - ✓ LP ⇒ Oscilador de bajo consumo (32-200 KHz).

- ✓ XT ⇒ Oscilador estándar (100 KHz – 4 MHz).
- ✓ HS ⇒ Oscilador de alta velocidad (8- 20 MHz).
- ✓ RC ⇒ Oscilador de bajo costo.

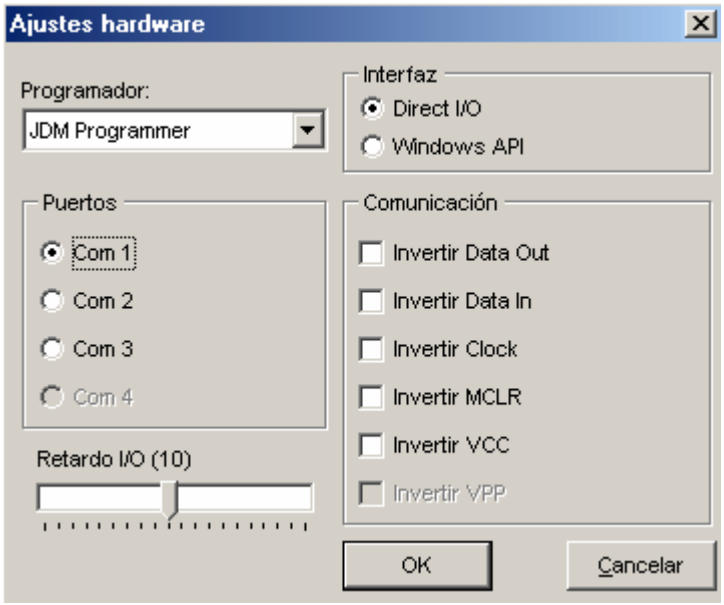
- **Protección Código**: Para el PIC16F877 que tiene una memoria de código de 8K.

  - ✓ CP 0000h-1FFFFh ⇒ Protección total del código.
  - ✓ CP 1000h-1FFFFh ⇒ Protección para ½ del código (la parte alta).
  - ✓ CP 1F00h-1FFFFh ⇒ Protección para ¾ del código (la parte alta).
  - ✓ CP OFF ⇒ Sin protección para el código.

- **Bits de configuración**:

  - ✓ WDT ⇒ Activación del perro guardián.
  - ✓ PWRT ⇒ Activación del Timer de Conexión de Alimentación.
  - ✓ BODEN ⇒ Activación del Reset por Caída de Tensión.
  - ✓ LVP ⇒ Activación de la programación en Bajo Voltaje.
  - ✓ CPD ⇒ Activación del código de protección de la memoria EEPROM.
  - ✓ WRT ⇒ Activación del permiso de escritura en la memoria FLASH.
  - ✓ DEBUGGER ⇒ Activación del Modo Depurador en Circuito.

Para la selección del tipo de hardware y del puerto de comunicaciones, con los que se va a comunicar la placa grabadora y el PC, ir a **Ajustes > Tipo de hardware**, o pulse el botón **F3:**



7.2 MPLAB

El MPLAB es un entorno de desarrollo integrado que le permite escribir y codificar los programas de los microcontroladores PIC de Microchip para ejecutarlos. El MPLAB incluye un editor de texto, funciones para el manejo de proyectos, un simulador interno y una variedad de herramientas que lo ayudarán a mantener y ejecutar su aplicación. También provee una

interfase de usuario para todos los productos con lenguaje Microchip, programadores de dispositivos, sistemas emuladores y herramientas de tercer orden.

La guía MPLAB le permitirá realizar las siguientes tareas:

- Manejar el escritorio MPLAB.
- Crear un nuevo archivo de código fuente para el ensamble e ingresarlo a un nuevo proyecto para el microcontrolador PIC.
- Identificar y corregir los errores simples.
- Ejecutar el simulador interno.
- Marcar puntos de interrupción.
- Crear ventanas de observación.
- Manejar ventanas para el seguimiento de errores.

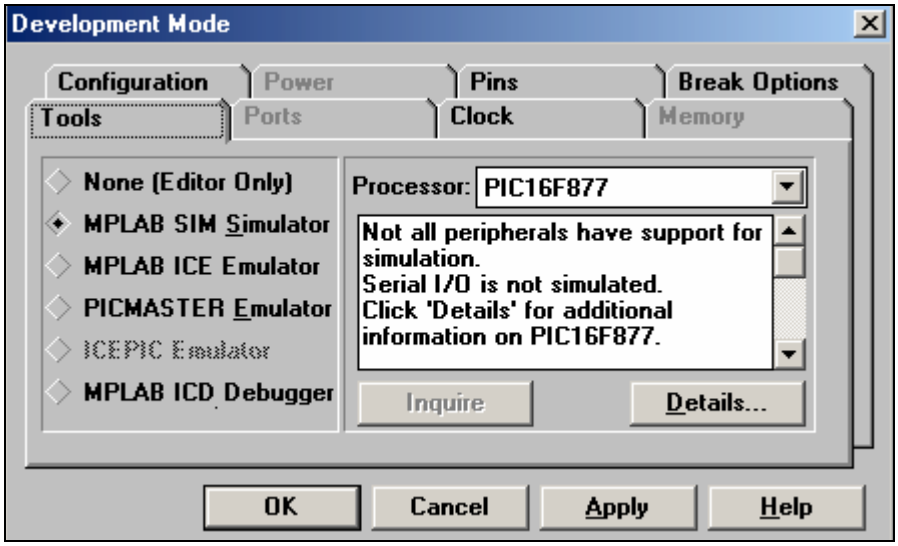
CONFIGURAR EL MODO DE DESARROLLO

El escritorio básico del MPLAB se asemeja al de las aplicaciones de Windows. Tiene una barra de menú en el margen superior, una barra de herramientas, y también una barra de estado en el margen inferior. Podrá advertir que la barra de estado incluye información sobre cómo se ha configurado el sistema.

Nota: El "modo de desarrollo" determina la herramienta, si elige alguna, que ejecutará el código. Para esta guía, usaremos el simulador de software MPLAB-SIM. Si tiene un emulador, más tarde podrá cambiar a una de sus operaciones. La operación será similar.

El modo "Sólo Editor" no permite la ejecución del código, y sólo es útil en caso de no haber instalado el simulador, no disponer de un emulador, y que sólo cree el código para programar un micro PIC.

Al seleccionar el ítem del menú "**`Options>Development Mode`**", aparecerá una caja de diálogo semejante a la siguiente:



Pantalla 1

El MPLAB es un producto en constante evolución, de modo que pueden aparecer sutiles diferencias entre la pantalla que usted vea y la que se muestra aquí. Seleccione el botón radio próximo al Simulador MPLAB-SIM y elija el modelo de microcontrolador en la lista de procesadores disponibles que pueden ser resistidos por el simulador. Seleccione por ejemplo el microcontrolador PIC16F877 y luego presione el botón "OK". De este modo se iniciará el simulador, y debería aparecer en la barra de estado "PIC16F877" y "Sim". Se encuentra así en el modo simulador para el 16F877.

CREAR UN NUEVO PROYECTO SIMPLE

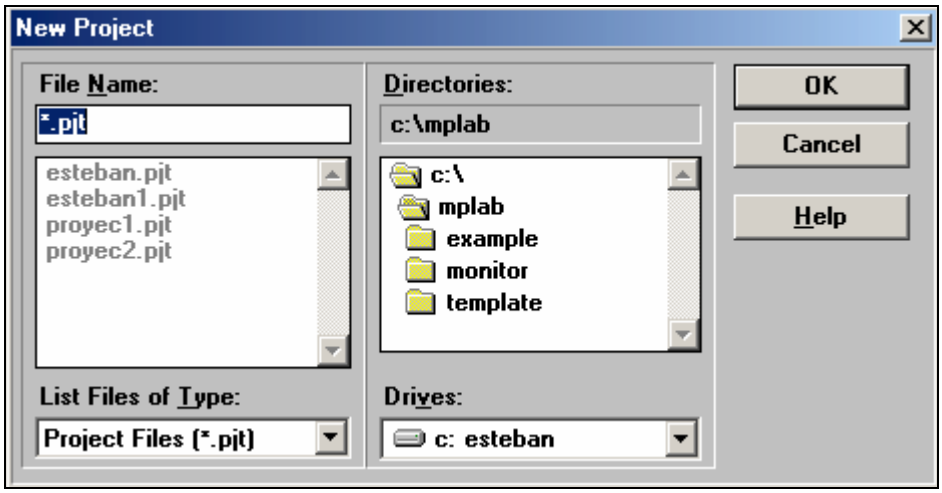
El simulador se ejecutará desde el mismo archivo, llamado "**nuevo.hex**", el cual puede ser programado en el micro PIC. Para que se ejecute el simulador, primero deberá crear un archivo de código fuente y realizar el montaje del código fuente.

Nota: El lenguaje ensamblador produce, entre otros elementos, un archivo **.hex**. Más adelante este archivo puede ser cargado directamente en el programador del dispositivo sin usar el ensamblador o un proyecto del MPLAB. Este archivo también puede ser cargado por otros programadores de tercer orden.

Seleccione "**File>New**" en el menú y seleccione el botón **OK** y seguidamente aparecerá un diálogo de exploración de Windows estándar. Decida dónde desea crear su proyecto y recuerde dónde lo ubicó. Más tarde necesitará esta información.

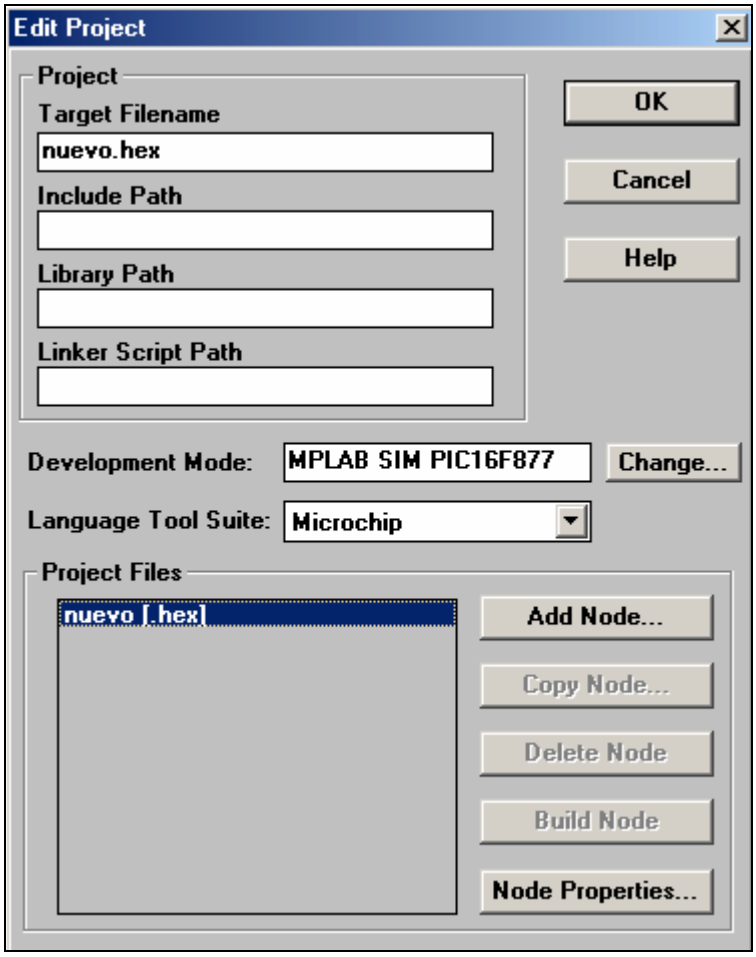
"**PJT**" es el sufijo estándar para los archivos de proyectos en el MPLAB. El prefijo del nombre de archivo del proyecto, en este caso "nuevo", será el prefijo por defecto de muchos de los archivos que el MPLAB usará o creará para esta guía.

La pantalla de exploración de creación de un nuevo proyecto en MPLAB es:



Pantalla 2

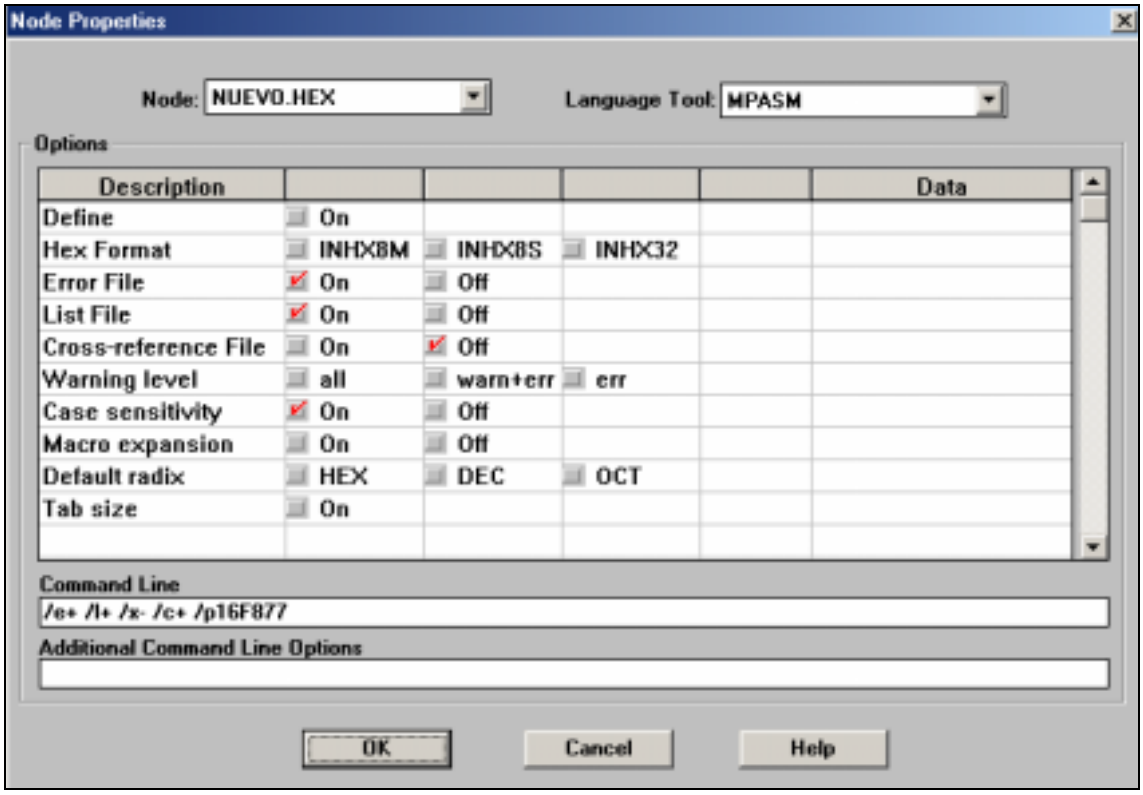
Seleccione el botón **OK** para que aparezca el diálogo Proyecto MPLAB. Este diálogo puede parecer confuso, pero en realidad es muy simple. Este diálogo "**Edit Proyect**" será semejante al siguiente:



Pantalla 3

Nota: El simulador, los programadores y los sistemas emuladores que operan con el MPLAB usan un archivo hex creado por el ensamble, la compilación y/o el linking del código fuente. Algunas herramientas diferentes pueden crear archivos hex, y estas herramientas forman parte de cada proyecto. Los proyectos le dan la flexibilidad para describir cómo se construirá la aplicación y qué herramientas se usarán para crear el archivo .hex. Para modificar estas características, deberá usar el botón "**Node Properties**". Vea la Guía del Proyecto v3.40 del MPLAB para más información sobre proyectos complejos.

Advierta que el nombre del archivo de destino ya ha sido completado. Ya conoce el modo de desarrollo que configuramos previamente y asume que usaremos la serie de herramientas de lenguaje Microchip. En la ventana "**Proyect Files**", encontrará *nuevo. [hex]*. Al destacar este nombre el botón "**Node Properties**" se tornará utilizable. Seguidamente debe indicarle al MPLAB cómo crear el archivo hex. Hágalo seleccionando el botón anterior y aparecerá el siguiente dialogo:



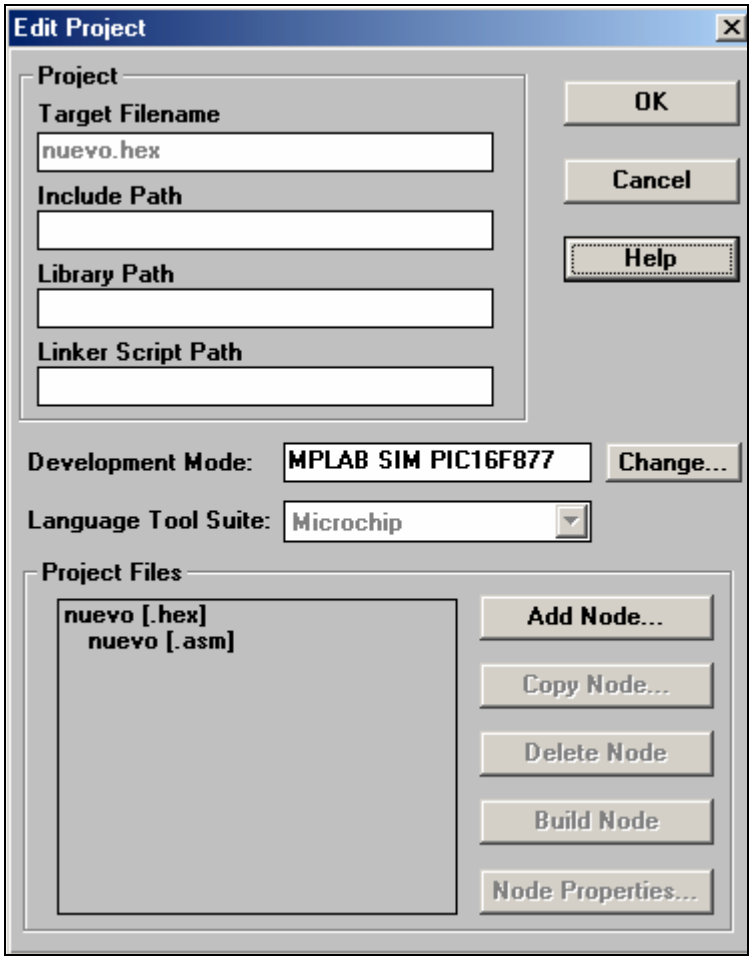
Pantalla 4

Este diálogo contiene todas las configuraciones por defecto para una herramienta de lenguaje - en este caso MPASM, como podrá ver en el ángulo superior derecho del diálogo. En su forma más simple, el proyecto contiene un archivo hex creado desde un archivo fuente de ensamble. Esta será la configuración por defecto cuando aparezca el diálogo "**Node Properties**".

Nota: Como puede ver, hay una cantidad de filas y columnas en este diálogo. Cada fila usualmente corresponde a un "cambio", aquellos elementos que se establecen en la línea de comando cuando se invoca una herramienta. De hecho, la configuración de estos cambios se refleja en la ventana "**Command Line**", próxima al margen inferior de la pantalla. Esta es la línea de comando que se usará cuando se invoque el MPASM desde el MPLAB. Por el momento, puede usar las configuraciones por defecto, pero cuando ya sepa construir una aplicación, probablemente deseará cambiar algunas.

Al seleccionar el botón **OK**, aplicará estas configuraciones, y retornará al diálogo "**Edit Proyect**", con el botón "**Add Node**" disponible.

Presione el botón "**Add Node**". Aparecerá el diálogo de exploración de Windows estándar, con el mismo directorio usado para el proyecto. Ingrese el nombre de archivo, *nuevo.asm*, y presione **OK**. Retornará al diálogo "**Edit Proyect**", donde podrá ver *"nuevo.asm"* añadido debajo del archivo hex, indicando que es un nodo concurrente.



Pantalla 5

Al presionar **OK**, retornará al escritorio MPLAB con un archivo de código fuente abierto y aún sin nombre.

CREAR UN NUEVO ARCHIVO FUENTE SIMPLE

Presione dentro del espacio en blanco de la ventana de archivo creada. Seguramente se llamará "**Untitled1**". De este modo accederá al "foco" de la ventana. Use la opción de menú "**File>Save As..**", y guarde el archivo vacío como *nuevo.asm*. Cuando abra el diálogo de exploración estándar, encontrará su ubicación en el directorio del proyecto. Ingrese el nombre de archivo y presione **OK**.

Ahora estarán disponibles el escritorio MPLAB y la ventana de archivo vacío, pero el nombre de la ventana de archivo reflejará su nuevo nombre.

Nota: El nombre del archivo fuente y el nombre del proyecto ("nuevo" en esta guía) deben ser iguales en este tipo de proyectos. Hay otros proyectos de archivos múltiple que usan el linker y permiten que el nombre del archivo de salida sea diferente al del archivo de

entrada (hay una guía aparte para los proyectos de archivos múltiples que usan el linker). En esta guía, para el tipo de archivo fuente del proyecto, el MPASM siempre creará una archivo hex de salida con el mismo nombre que el archivo fuente, y esta configuración no puede modificarse. Si cambia el nombre del archivo fuente, también deberá cambiar el nombre del proyecto.

INGRESAR EL CÓDIGO FUENTE

Use el ratón para ubicar el cursor al comienzo de la ventana de archivo vacío *nuevo.asm*, e ingrese el siguiente texto, exactamente como esta escrito en cada línea. Los comentarios no son necesarios para el correcto funcionamiento del programa, pero son notas aclaratorias de los parámetros utilizados.

```
LIST        P=16F877          ;Indica el tipo de microcontrolador
INCLUDE     "P16F877.INC"     ;Archivo de características del PIC

contador    EQU   0x0c        ; Establece la variable contadora

            ORG   0x00        ;Vector de reset
            goto  INICIO      ;Salta al principio del programa
            ORG   0x05        ;Salva el vector de interrupción

INICIO      mov1w .5          ;Inicializa el contador a un valor
                              ;arbitrario mayor que cero
movwf contador    ;Lo guarda  en la variable definida

BUCLE       decfsz contador,F ;Decrementa el contador, y ubica los
                              ;resultados en el registro de archivos
goto    BUCLE     ;Lo repite hasta que el contador acabe
goto    INICIO    ;Cuando el contador se completa,
                              ;se reinicia
end
```

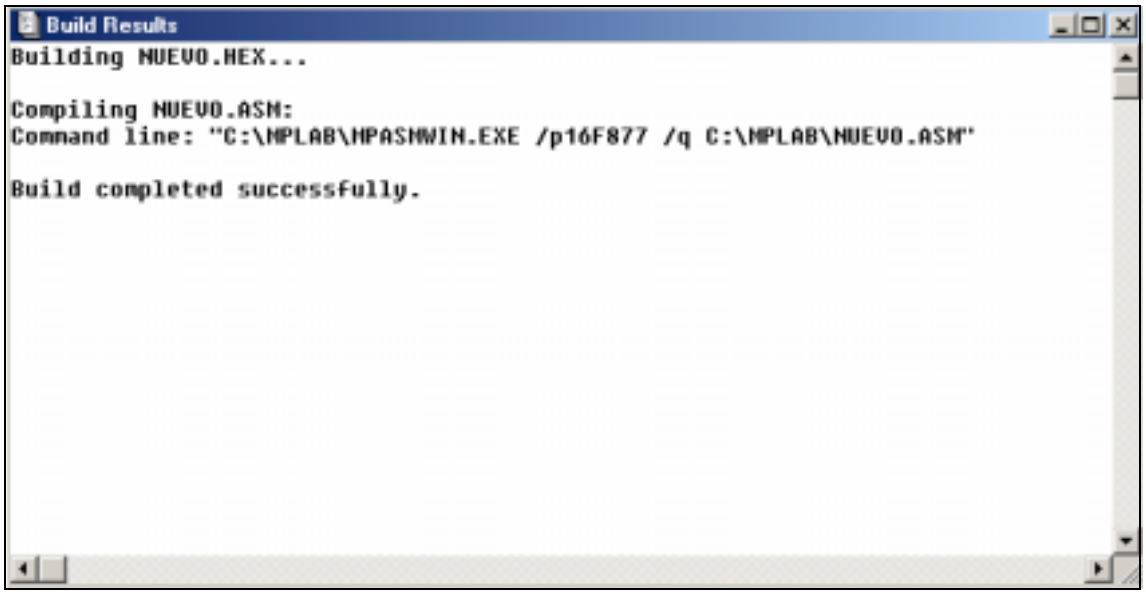
Este código es un programa muy simple que decrementa un contador y lo "resetea" a un valor predeterminado cuando el contador vuelve a cero.

Nota: Todos las etiquetas comienzan en la primera columna, y la última línea tiene una directiva "end". Busque en la Guía del Usuario MPASM con MPLINK y MPLIB más información sobre directivas. Las páginas de datos del micro PIC contienen información completa sobre instrucciones con ejemplos para su uso.

Guarde el archivo usando la función de menú "**`File>Save`**".

ENSAMBLE DEL ARCHIVO FUENTE

El ensamble del archivo puede realizarse de varias maneras. Aquí describiremos un método. Use el ítem de menú "**Proyect>Build All**". De este modo ejecutará el lenguaje ensamblador MPASM en el trasfondo usando las configuraciones guardadas con el proyecto anteriormente. Una vez completado el proceso de ensamble, aparecerá la siguiente ventana "**Build Results**":



Pantalla 6

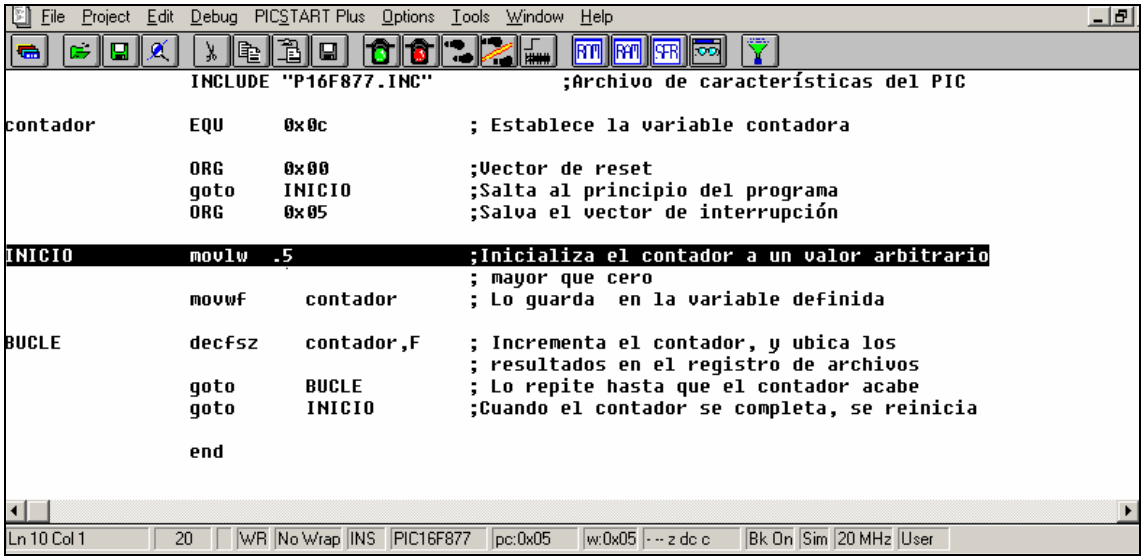
Haga un doble clic sobre el mensaje de error. De este modo ubicará el cursor en la línea que contiene el error en el código fuente. Use la ventana " **Build Results**" para hacer una búsqueda de errores, y reparar los que aparecieran en el código fuente. Re-ensamble el archivo ejecutando la función de menú "**Proyect>Build All**". Este procedimiento puede demandar un par de repeticiones.

Luego de reparar todos los problemas en el código fuente, la ventana " **`Build Results`**" mostrará el mensaje "**`Build completed successfully`**". Ya ha completado un proyecto que puede ejecutarse usando el simulador.

EJECUCIÓN DE SU PROGRAMA

Use "**Debug>Run>Reset**" para iniciar el sistema. El contador del programa se reseteará a cero, que es el vector de reset en el 16F877. La línea del código fuente en esta dirección será destacada con una barra oscura. También advertirá que en la barra de estado, la PC se establecerá en 0x00.

Use el ítem de menú "**Debug>Run>Step**". Al hacerlo, el contador del programa avanzará hasta la siguiente ubicación de instrucción. La barra oscura seguirá al código fuente y el contador del programa desplegado en la barra de estado avanzará.



Pantalla 7

Cuando ejecute el ítem de menú " **Debug>Run>Step**", advierta la aparición de un texto en el lado derecho del ítem de menú que dirá "**F7**". El mismo equivale a "tecla de función siete" en su teclado. Muchas funciones del MPLAB se asignan a "teclas-especiales". Estas teclas cumplen la misma función que los ítems de menú a los cuales corresponden. Presione F7 varias veces y podrá ver al contador del programa y a la barra avanzar a través del programa.

Ejecute el ítem de menú "**Debug>Run>Run**" o presione F9 para iniciar la ejecución del programa desde la ubicación actual del contador. Los colores de la barra de estado cambiarán, indicando que el programa está ejecutando las instrucciones. Ninguno de los campos de la barra de estado se actualizará mientras el programa esté en ejecución.

Detenga el programa ejecutando el ítem de menú "**Debug>Run>Halt**" o presionando F5. La barra de estado volverá a su color original, y el contador del programa y otras informaciones de su estado serán actualizados.

Nota: Otra manera de ejecutar funciones es usar la barra de herramientas ubicada en el margen superior de la pantalla. Si ubica el cursor sobre los ítems de la barra de herramientas, podrá ver el nombre de su función en la barra de estado. El botón de la izquierda es un botón estándar "cambiar barra de herramientas" que le permite desplegar

las barras de herramientas disponibles. En la barra de herramientas de debug, la luz verde es equivalente a F9 (Ejecución) y la luz roja equivale a F5 (Detención).

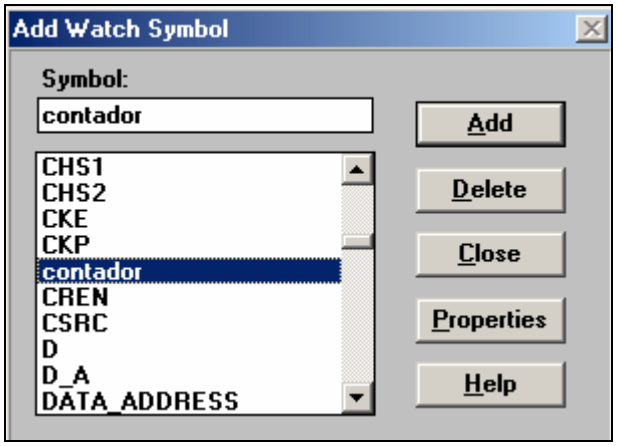
ABRIR OTRAS VENTANAS PARA EL SEGUIMIENTO DE ERRORES

Hay muchas maneras de visualizar el programa y su ejecución usando el MPLAB. Por ejemplo, este programa está destinado a decrementar una variable contador pero, ¿cómo puede asegurarse que se está produciendo dicho decremento?. Una manera es abrir e inspeccionar una ventana de registro de archivo. Puede hacerlo ejecutando el ítem de menú "**`Window>File Registers`**". Aparecerá una pequeña ventana con todos los registros de archivo o la memoria RAM del 16F877.

Presione F7 (ejecutando instrucción por instrucción si piensa anularlo) varias veces y observe la actualización de valores en la ventana de registro de archivo. Hemos colocado la variable del contador en la ubicación de dirección 0x0c. Mientras el contador se decremente, su decremento se reflejará en la ventana de registro de archivo. Los registros de archivo cambian de color cuando su valor cambia, de modo que los cambios puedan advertirse fácilmente en la inspección. De todos modos, en muchos programas complejos, varios valores pueden cambiar, resultando más difícil focalizar las variables que le interesan. Este problema puede solucionarse usando una ventana de observación especial.

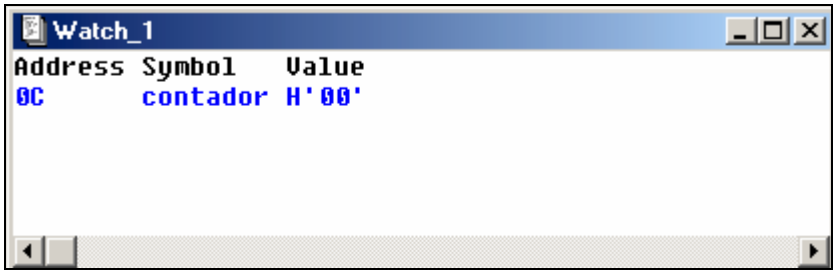
CREAR UNA VENTANA DE OBSERVACIÓN

Ejecute el ítem de menú "**`Window>Watch Windows>New Watch Window..`**". Aparecerá el diálogo "**`Add Watch Symbol`**".



Pantalla 8

Seleccione la variable contador, presione el botón "**`Add`**", y luego el botón "**`Close`**". Aparecerá en su escritorio MPLAB una ventana de observación desplegando el valor actual de la variable "contador".



Pantalla 9

Presione F7 varias veces para advertir cómo se actualiza la ventana de observación mientras el valor del contador se decrementa. Si dejó la ventana de registro de archivo abierta, la misma también será actualizada.

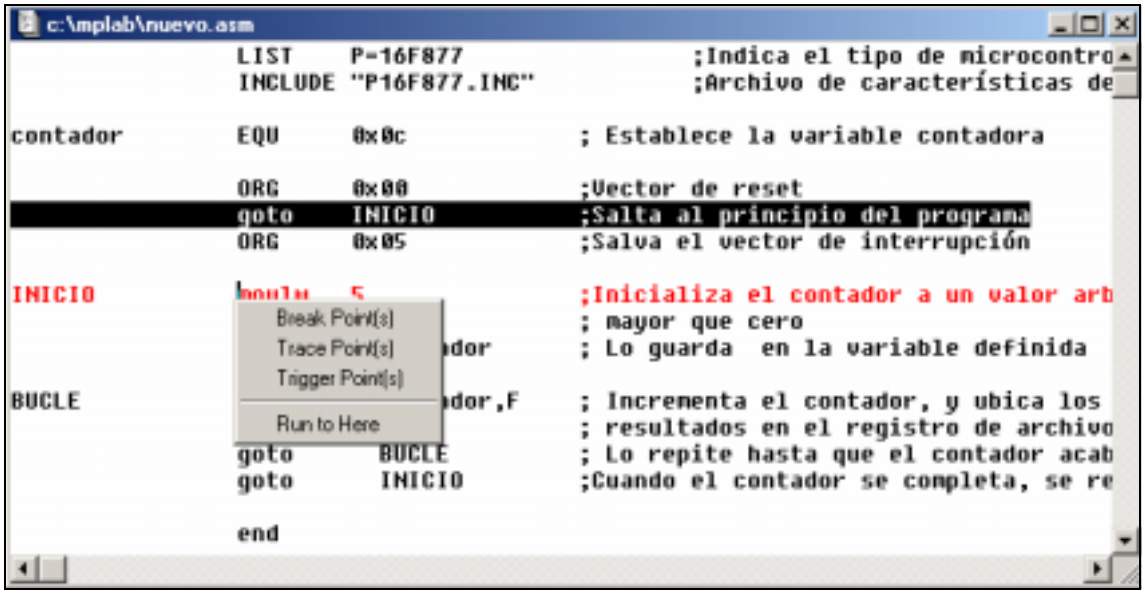
Puede guardar la ventana de observación y sus configuraciones ejecutando el ítem "**Window>Watch Windows>Save Active Watch..**". Aparecerá el diálogo de exploración standard ubicado en el directorio del proyecto. Elija cualquier nombre arbitrario y presione **OK**.

Si no nombra la ventana de observación, el MPLAB lo hará por usted. La ubicación y el estado en la pantalla de la ventana abierta o cerrada serán guardados con el proyecto, de modo que la próxima vez que abra su proyecto, sus ventanas de observación aparecerán restauradas.

Nota: También puede editar ventanas de observación luego de crearlas. Use el botón del sistema y seleccione "**Add To Active Watch..**" para que aparezca un diálogo mediante el cual podrá agregar más ítems. Con la tecla "Ins" podrá hacer lo mismo. Si desea borrar un ítem, selecciónelo y presione la tecla Suprimir; la observación referida desaparecerá de la ventana. Puede seleccionar "**Edit Active Watch..**" en el menú del sistema para cambiar el modo en el cual se muestra el ítem (en hex, binario, como una variable de 16-bit en vez de 8-bit, etc.).

MARCAR UN PUNTO DE INTERRUPCIÓN

Presione F5 ("**Debug>Run>Halt**") para asegurarse que el procesador del simulador se ha detenido. Entre dentro de la ventana del código fuente la línea siguiente al rótulo "INICIO", que dice "*movlw .5*'". Presione el botón derecho del mouse para que aparezca el siguiente menú:



Pantalla 10

Seleccione el ítem de menú "**Break Point(s)**". El menú desaparecerá y la línea donde se ubicó el cursor cambiará de color, indicando que ha sido establecido un punto de interrupción en dicha ubicación.

Presione F6 o ejecute el ítem de menú "**Debug>Run>Reset**" para resetear el sistema. Luego ejecute el sistema presionando F9. El programa se ejecutará y se detendrá en la instrucción ubicada luego del punto de interrupción.

7.3 Información del PIC

Este dispositivo pertenece a la subfamilia de microcontroladores PIC de la gama media, que se identifica por tener como memoria de programa una de tipo FLASH, que se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos y es más pequeña. A diferencia de la ROM, la memoria FLASH es programable en el circuito. Es más rápida y de mayor densidad que le memoria EEPROM, tolerando más ciclos de escritura / borrado que esta . La alternativa FLASH está recomendada cuando se precisa gran cantidad de memoria no volátil.

Este microcontrolador está encapsulado en 40 patitas, por lo que posee una gran cantidad de líneas de entrada / salida.

Los principales recursos que posee el microcontrolador PIC16F877 son:

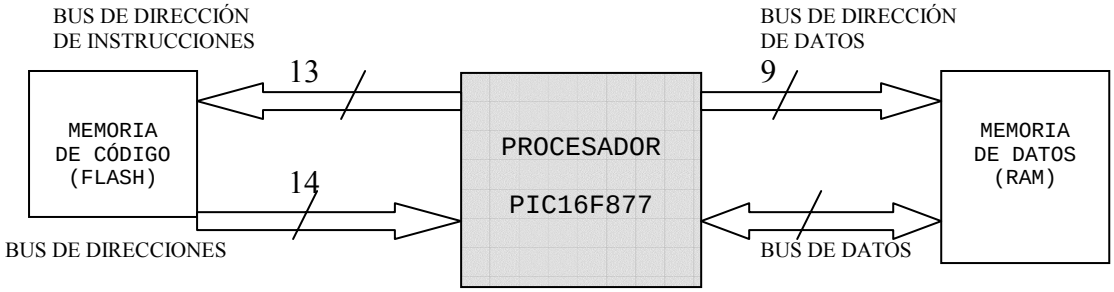
- Procesador de arquitectura RISC avanzada.
- Juego de 35 instrucciones con 14 bits de longitud. Todas ellas se ejecutan en un ciclo de instrucción, menos las de salto que tardan dos.
- Frecuencia máxima de funcionamiento de 20Mhz.
- Hasta 8 K(8192) palabras de 14 bits para la Memoria de Código, tipo FLASH.
- Hasta 368 bytes de Memoria de Datos RAM.
- Hasta 256 bytes de Memoria de Datos EEPROM.
- Hasta 14 fuentes de interrupción internas y externas.
- Pila con 8 niveles.
- Modos de direccionamiento directo, indirecto y relativo.
- Perro Guardián (WDT).
- Código de protección programable.
- Modo SLEEP de bajo consumo.
- Programación serie en circuito con dos patitas.
- Voltaje de alimentación comprendido entre 2 y 5.5 V.
- Bajo consumo (menos de 2 mA a 5 V y 5 Mhz).

- Diversos periféricos como:

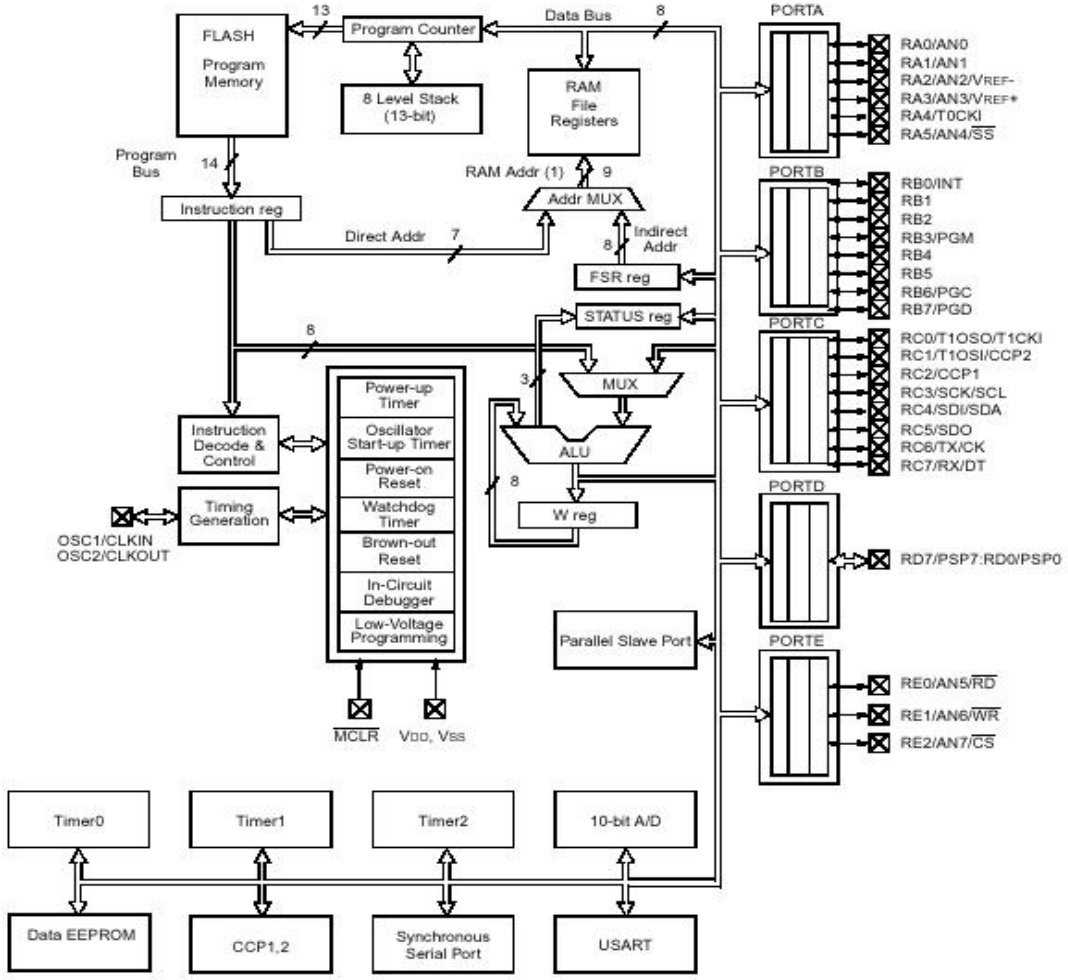
  - ✓ Timer 0 : Temporizador-Contador de 8 bits con predivisor.
  - ✓ Timer 1 : Temporizador-Contador de 16 bits con predivisor.
  - ✓ Timer 2 : Temporizador-Contador de 8 bits con predivisor y postdivisor.
  - ✓ Dos módulos de Captura-Comparación-PWM.
  - ✓ Convertidor A/D de 10 bits de precisión.
  - ✓ Modulo de comunicación Serie Síncrona (MSSP), con modo SPI e I2C.
  - ✓ Interface de comunicación Serie SCI, también llamado USART.
  - ✓ Puerta Paralela Esclava (PSP).

PROCESADOR RISC CON ARQUITECTURA HARVARD

La arquitectura aplicada en este microcontrolador se caracteriza por la independencia entre la memoria de código y la de datos, esto facilita el trabajo en paralelo de las dos memorias, obteniendo unas altas cotas de rendimiento.



La memoria de código (FLASH) está direccionada por el contador de programa (PC) en conexión con la Pila de 8 niveles. La memoria de datos (RAM) contiene el Banco de Registros Específicos y el Banco de los Registros de Propósito General y transfiere información bidireccional por el bus de datos de 8 líneas que interconecta todos los elementos.



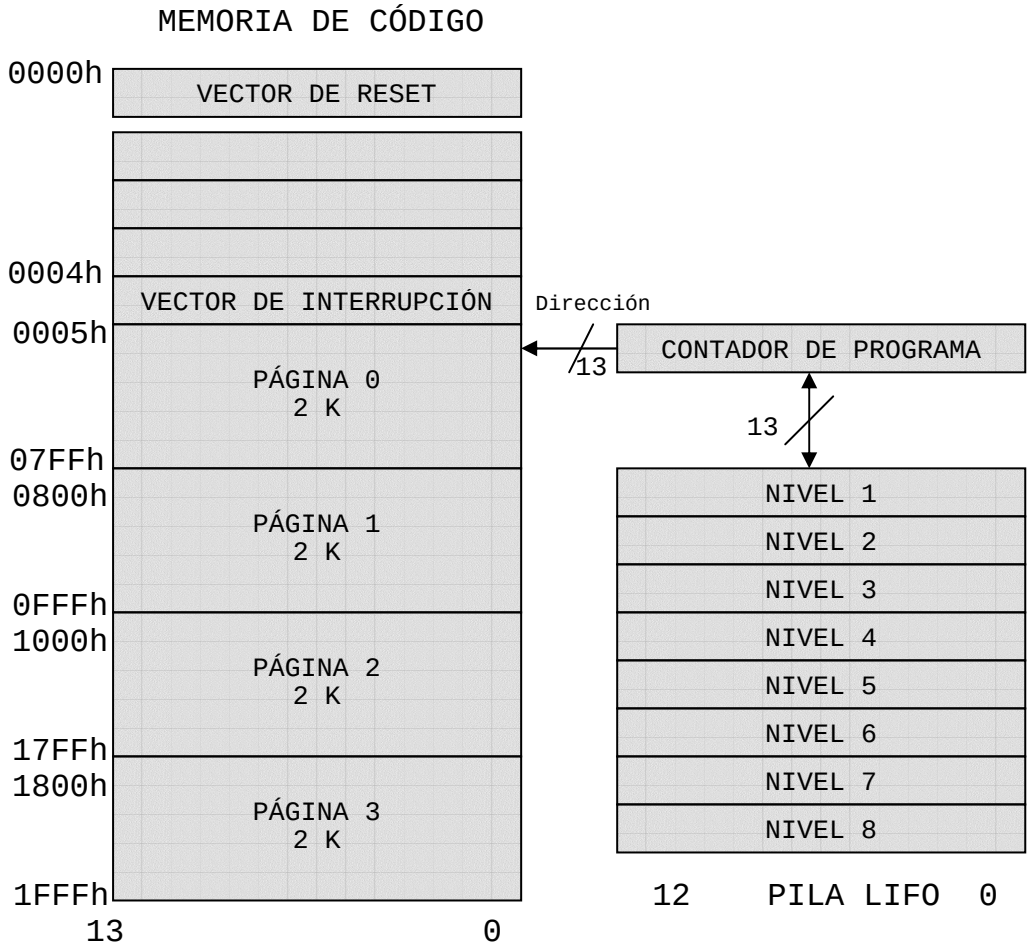
ORGANIZACIÓN DE LA MEMORIA DE PROGRAMA

La memoria FLASH, en la que se graba el programa de la aplicación, tiene una capacidad de 8K(8192) palabras de 14 bits cada una. Dicha memoria está dividida en 4 paginas de 2K(2048) y está direccionada con el contador de programa (PC), que tiene un tamaño de 13 bits.

La pila, que tiene 8 niveles de profundidad, es transparente para el usuario, por lo que funciona automáticamente y no dispone de instrucciones para guardar o sacar de ella información. Al poseer la pila sólo 8 niveles le corresponde al programador preocuparse por los anidamientos en las subrutinas para no sobrepasar dicho valor.

Con la instrucción **`CALL`** y con las interrupciones el valor del Contador de Programa (PC) se salva en el nivel superior de la pila. Con las instrucciones **`RETURN, RETFIE, RETLW`** el valor contenido en la cima de la pila se carga en Contador de Programa.

El vector de Reset ocupa la dirección 0000h y el vector de Interrupción la 0004h.



ORGANIZACIÓN DE LA MEMORIA DE DATOS RAM

La memoria de datos tiene posiciones implementadas en RAM y en EEPROM. En la RAM se alojan los registros operativos fundamentales en el funcionamiento del procesador y en el manejo

de todos sus periféricos, además de registros que el programador puede usar para información de trabajo propia de la aplicación.

La memoria RAM del microcontrolador consta de 4 bancos de 128 bytes cada uno. En las posiciones iniciales de cada banco se ubican los Registros Específicos que gobiernan al procesador y sus recursos. Para seleccionar el banco al que se desea acceder se usan los bits 6 y 5 del Registro de Estado (**RP1** y **RP0**).

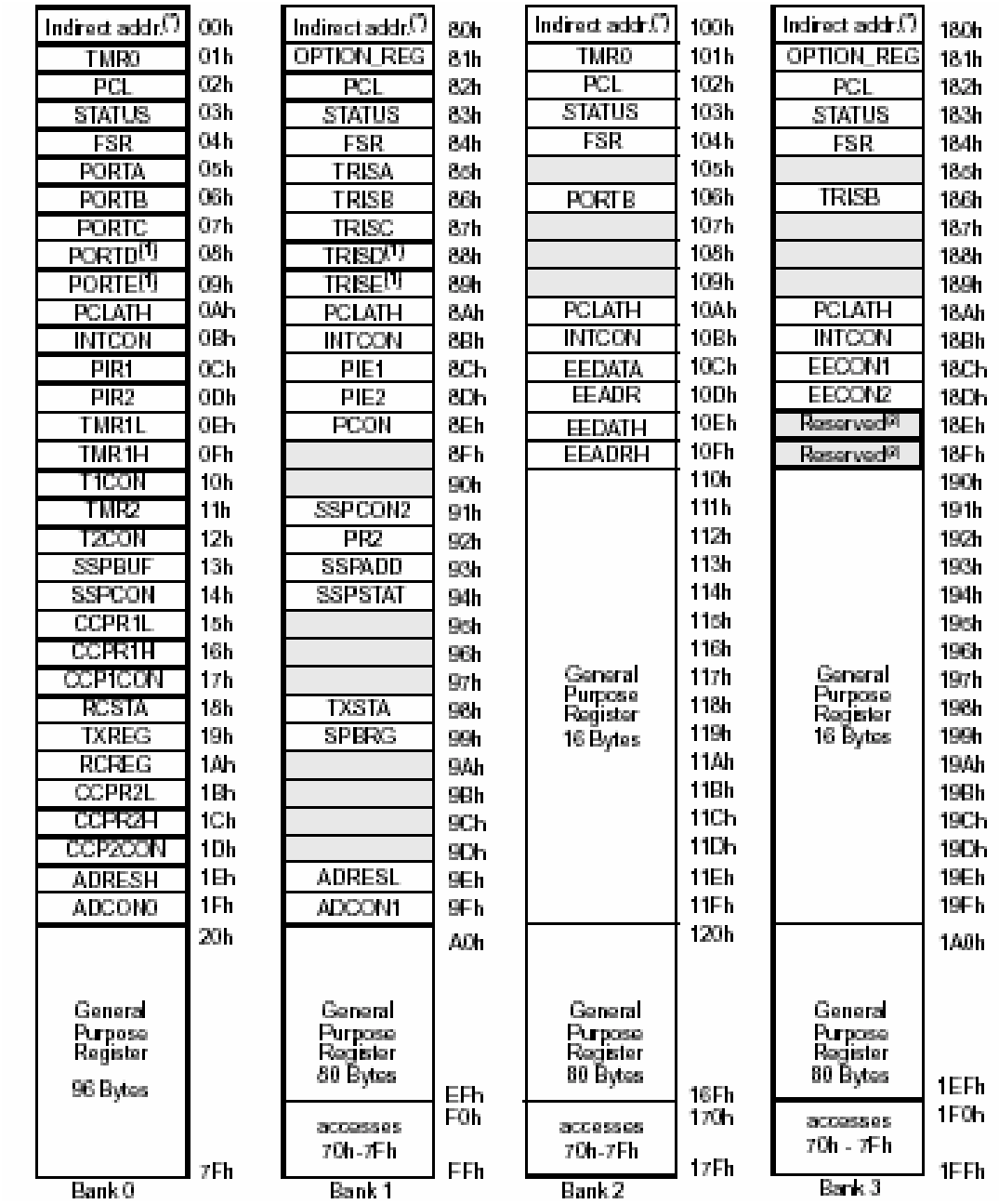
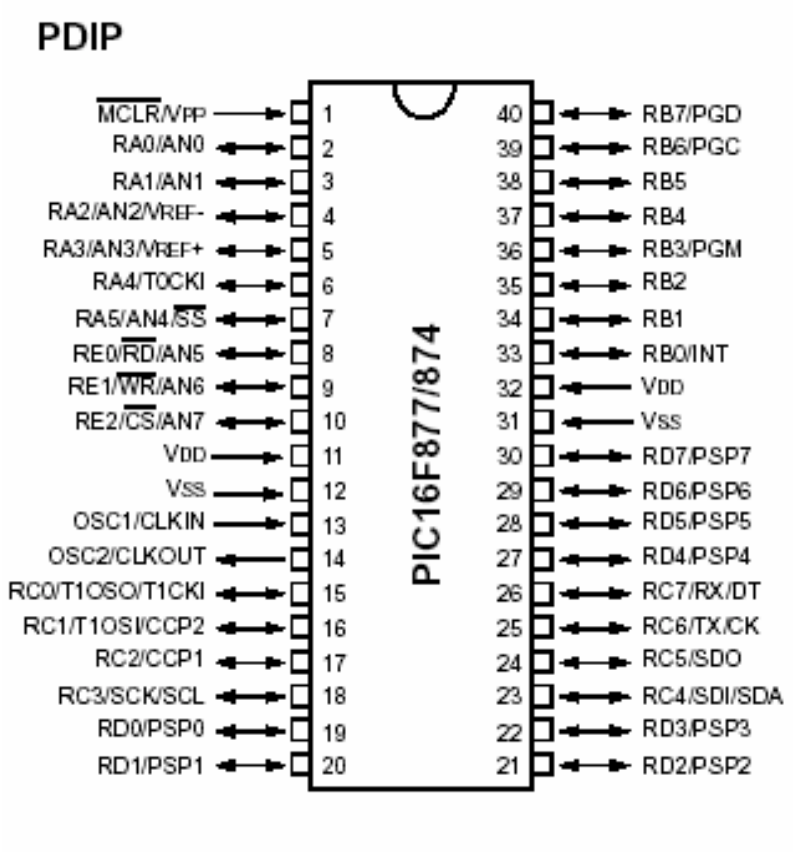


DIAGRAMA DE CONEXIONADO

Microchip posee diversos encapsulados para la protección del microcontrolador PIC16F877; aquí se representa el encapsulado PDIP que se puede conectar a la placa microcontroladora sin necesidad de soldarlo a ella, mediante zócalos.



La asignación de funciones para las diferentes patitas es la siguiente:

PATITAS DE PROPÓSITO GENERAL

- **OSC1/CLKIN (patita 13)** ⇒ Entrada del cristal de cuarzo o del oscilador externo.
- **OSC2/CLKOUT (patita 14)** ⇒ Salida del cristal de cuarzo. En modo RC la patita **OSC2** saca la cuarta parte de la frecuencia que se introduce por la patita **OSC1**, que determina el ciclo de instrucción.
- **VSS (patitas 12 y 31)** ⇒ Conexión a tierra.
- **VDD (patitas 11 y 32)** ⇒ Entrada de la alimentación positiva.
- **MCLR#/VPP/THV (patita 1)** ⇒ Entrada de RESET o entrada del voltaje de programación (voltaje alto en el modo test).

PATITAS DE LA PUERTA A

- **RA0/AN0 (patita 2)** ⇒ Esta patita puede funcionar como una línea digital de E/S o como una entrada analógica (canal 0) del convertidor A/D.
- **RA1/AN1 (patita 3)** ⇒ Esta patita puede funcionar como una línea digital de E/S o como una entrada analógica (canal 1) del convertidor A/D.

- **RA2/AN2/V_{REF-} (patita 4)** ⇒ Esta patita a parte de las dos funciones anteriores, también puede funcionar como entrada del voltaje de referencia negativo, necesario para la realización de algunas aplicaciones con el convertidor A/D.
- **RA3/AN3/V_{REF+} (patita 5)** ⇒ Esta patita puede funcionar como una E/S digital, como entrada analógica (canal 3) o como un voltaje de referencia positivo de entrada para el convertidor A/D.
- **RA4/T0CKI (patita 6)** ⇒ Esta patita puede funcionar como una E/S digital o como una entrada de reloj externo para el Timer 0.
- **RA5/AN4/SS# (patita 7)** ⇒ Esta patita puede funcionar como una E/S digital, como una entrada analógica (canal 4) o como selección de la puerta serie síncrona para su funcionamiento como esclavo.

PATITAS DE LA PUERTA B

- **RB0/INT (patita 33)** ⇒ Esta patita puede funcionar como una E/S digital o como una entrada de petición de interrupción externa, para el control de diversos periféricos externos.
- **RB1 (patita 34)** ⇒ Esta patita solo puede funcionar como E/S digital.
- **RB2 (patita 35)** ⇒ Esta patita solo puede funcionar como E/S digital.
- **RB3/PGM (patita 36)** ⇒ Esta patita puede funcionar como una E/S digital o como la entrada del voltaje bajo para programación.
- **RB4 (patita 37)** ⇒ Esta patita solo puede funcionar como E/S digital.
- **RB5 (patita 38)** ⇒ Esta patita solo puede funcionar como E/S digital.
- **RB6/PGC (patita 39)** ⇒ Esta patita puede funcionar como una E/S digital o como la entrada de la señal de reloj en la programación serie del microcontrolador.
- **RB7/PGD (patita 40)** ⇒ Esta patita puede funcionar como una E/S digital o como la entrada de datos en la programación serie del microcontrolador.

PATITAS DE LA PUERTA C

- **RC0/T1OSO/T1CKI (patita 15)** ⇒ Esta patita puede funcionar como una E/S digital, como una salida de reloj desde el Timer 1 o como una entrada de reloj externa para el Timer 1.
- **RC1/T1OSI/CCP2 (patita 16)** ⇒ Esta patita puede funcionar como una E/S digital, como una entrada al oscilador del Timer 1 o como:
  - ✓ La entrada al módulo Captura nº2.
  - ✓ La salida del módulo Comparación nº2.
  - ✓ La salida del módulo PWM nº2.
- **RC2/CCP1 (patita 17)** ⇒ Esta patita puede funcionar como una E/S digital o como:
  - ✓ La entrada del módulo de captura nº1 .
  - ✓ La salida del módulo de comparación nº1.
  - ✓ La salida del módulo de PWM nº1.
- **RC3/SCK/SCL (patita 18)** ⇒ Esta puede funcionar como una E/S digital o como:
  - ✓ La señal de reloj para la comunicación serie síncrona en modo SPI.
  - ✓ La señal de reloj de para la comunicación serie síncrona en modo I2C.
- **RC4/SDI/SDA (patita 23)** ⇒ Esta puede funcionar como una E/S digital, o como:
  - ✓ La Entrada de datos en modo SPI.

- ✓ La E/S de datos en modo I2C.

- **RC5/SDO (patita 24)** ⇒ Esta patita puede funcionar como una E/S digital o como la salida de datos en modo SPI.

- **RC6/TX/CK (patita 25)** ⇒ Esta patita puede funcionar como una E/S digital o como:
  - ✓ La patita del transmisor del USART asíncrono.
  - ✓ El reloj del USART síncrono.

- **RC7/RX/DT (patita 26)** ⇒ Esta patita puede funcionar como una E/S digital o como:
  - ✓ La patita del receptor del USART asíncrono.
  - ✓ La línea de datos del USART síncrono.

PATITAS DE LA PUERTA D

- **RD0/PSPO-RD7/PSP7** ⇒ Las 8 patitas de esta puerta pueden funcionar como líneas de E/S digitales o como líneas para la transferencia de información en la comunicación de la puerta paralela esclava, para la comunicación en paralelo con dispositivos como por ejemplo otros tipos de procesadores, FPGA´s, etc. Útil cuando la velocidad de la comunicación serie no es suficiente.

PATITAS DE LA PUERTA E

- **RE0/RD#/AN5 (patita 8)** ⇒ Esta patita puede funcionar como una E/S digital o:
  - ✓ Como la señal de lectura para la puerta paralela esclava.
  - ✓ Como el canal 5 del convertidor A/D.

- **RE1/WR#/AN6 (patita 9)** ⇒ Esta patita puede funcionar como una E/S digital o:
  - ✓ Como la señal de escritura para la puerta paralela esclava.
  - ✓ Como el canal 6 del convertidor A/D.

- **RE2/CS#/AN7 (patita 10)** ⇒ Esta patita puede funcionar como una E/S digital o:
  - ✓ Como la señal de activación /desactivación de la puerta paralela esclava.
  - ✓ Como el canal 7 del convertidor A/D.

REPERTORIO DE INSTRUCCIONES

Al ser un procesador con una arquitectura RISC (computadores de juego de instrucciones reducido), las 35 instrucciones que posee el microcontrolador de la gama media PIC16F877 son simples, y generalmente, se ejecutan en ciclo de instrucción (cuatro periodos del reloj principal) excepto las de salto que precisan dos. Esta sencillez y rapidez permite optimizar el hardware y el software del procesador.

Otra característica es la ortogonalidad de las instrucciones, donde la ubicación de los operandos es muy flexible; por lo que cualquier objeto del procesador puede actuar como fuente o como destino.

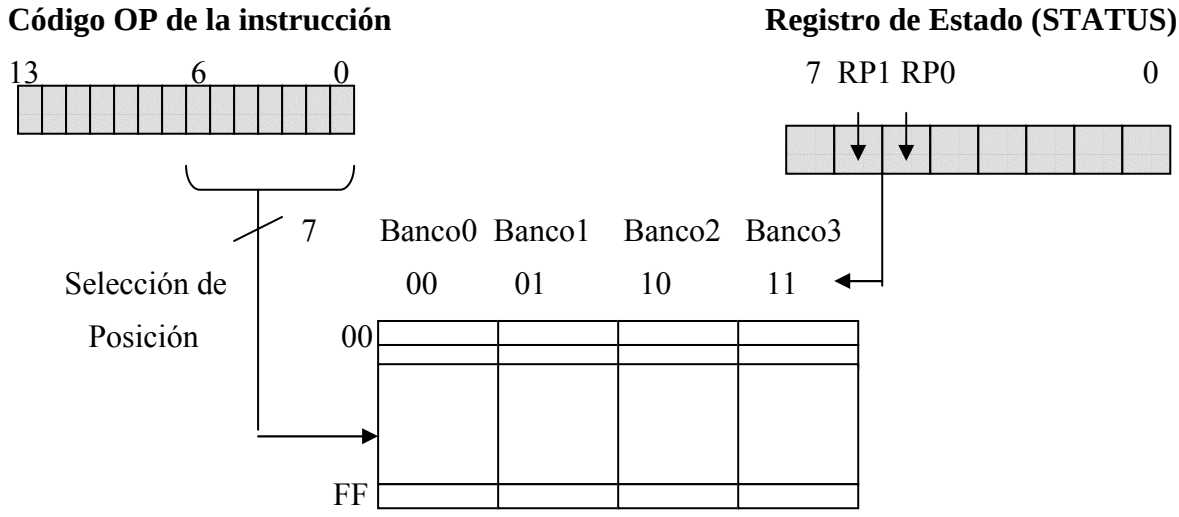
El formato utilizado por este tipo de microcontroladores PIC de la gama media es, el de los datos y operandos de 8 bits, y el de las instrucciones de 14 bits. Esto supone un notable ahorro de la memoria de código y una facilidad en la construcción de compiladores.

Respecto a los direccionamientos de la memoria de código, es el Contador de Programa (PC) el que establece el flujo de control. En la mayoría de las instrucciones el PC se incrementa automáticamente para apuntar la siguiente instrucción del programa. En las instrucciones de salto, cuando es del tipo directo, el valor que se carga en el PC proviene de una parte de los bits del código OP de la propia instrucción. En los saltos relativos, la ALU suma al valor actual del PC el de salto, almacenando el resultado en el PC.

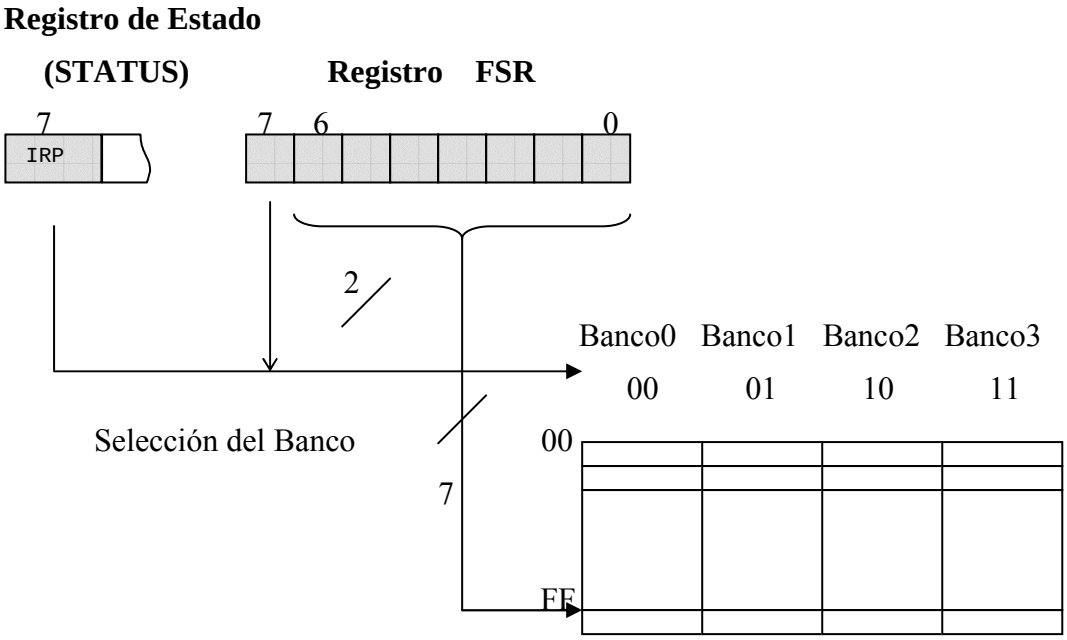
Existen tres modos de direccionar los datos u operandos que manejan las instrucciones:

1. **Inmediato** .- El valor del dato está contenido en el código OP de la propia instrucción.

2. **Directo** .- La dirección del área de la memoria de datos donde se halla el operando está contenida en el código OP de la instrucción. Los 7 bits de menos peso del código OP proporcionan la dirección de la posición de un banco. Los bits RP1 y RP0 del registro de estado seleccionan el banco.



3. Indirecto .- La dirección de la memoria de datos que guarda el operando está contenida en un registro. En este caso el operando hace referencia al registro **INDF**, que ocupa la dirección 0 del área de datos. Se accede a la posición que apunta el registro **FSR**, que está en la posición 4 del banco0 de la memoria RAM. Los 7 bits de menos peso del registro **FSR** seleccionan la posición y su bit de más peso junto con el bit **IRP** del registro de estado (**STATUS**), seleccionan el banco.



DEFINICIONES Y ABREVIATURAS

- **PC** ⇒ Contador de programa que direcciona la memoria de instrucciones. Tiene un tamaño de 13 bits, de los cuales los ocho de menos peso configuran el registro **PCL** que ocupa la dirección 02h del área de datos.
- **TOS** ⇒ Cima de la pila.
- **WDT** ⇒ Watchdog Timer (Perro Guardián).
- **W** ⇒ Registro acumulador.
- **f** ⇒ Es un campo de bits que contiene la dirección del Banco de registros.
- **d** ⇒ Es el bit del código OP de la instrucción. Si d=0, el destino es W y si d=1 el destino es f.
- **b** ⇒ Es un campo de bits que determina la posición de un bit, dentro de un registro de 8 bits.
- **k** ⇒ Es un campo de 8 bits que representa un dato inmediato. También pueden constar de más bits en las instrucciones de salto que cargan el PC.
- **dest** ⇒ Registro de destino (W o f).
- **TO#** ⇒ Bit de Time Out del registro de estado.
- **PD#** ⇒ Bit de Power Down del registro de estado.
- **Z** ⇒ Bit señalizador de Cero en el registro de estado.
- **C** ⇒ Bit señalizador del acarreo en el 8º bit del registro de estado.

- **DC** ⇒ Bit señalizador del acarreo en el 4º bit del registro de estado.
- **Label** ⇒ Nombre de una etiqueta.

INSTRUCCIONES QUE MANEJAN REGISTROS

Este grupo de instrucciones incluyen las que realizan operaciones aritméticas a través de la ALU: Sumas, restas, complementos, incrementos, decrementos y rotaciones. También incluyen operaciones lógicas como: AND, OR y XOR. Por ultimo incluyen instrucciones de transferencia y de puesta a cero.

NEMÓNICOS	PARÁMETROS	OPERACIÓN	CICLOS	FLAG
addwf	f,d	Suma del acumulador W con f	1	C,DC,Z
andwf	f,d	Operación lógica AND de W con f	1	Z
clrf	f	Borrado de f	1	Z
clrw	-	Borrado del acumulador W	1	Z
comf	f,d	Complemento de f	1	Z
decf	f,d	Decremento de f	1	Z
incf	f,d	Incremento de f	1	Z
iorwf	f,d	Operación lógica OR de W con f	1	Z
movf	f,d	Movimiento de f	1	Z
movwf	f	Movimiento de W a f	1	-
nop	-	No realiza ninguna operación	1	-
rlf	f,d	Rota f a la izquierda con Carry	1	C
rrf	f,d	Rota f a la derecha con Carry	1	C
subwf	f,d	Resta de f a W (f-W)	1	C,DC,Z
swapf	f,d	Intercambia 4 bits +peso con -peso	1	-
xorwf	f,d	Operación OR exclusiva de W con f	1	Z

INSTRUCCIONES QUE MANEJAN BITS Y DE SALTO

Las instrucciones que manejan bit individualmente son dos, y tienen una gran importancia en la elaboración de aplicaciones; ya que podemos alterar el valor de un bit, sin alterar el valor de los demás.

NEMÓNICOS	PARÁMETROS	OPERACIÓN	CICLOS	FLAG
bcf	f,b	Puesta a 0 del bit b de f	1	-
bsf	f,b	Puesta a 1 del bit b de f	1	-

Las instrucciones de brinco o salto, normalmente tardan dos ciclos de instrucción para realizarse:

NEMÓNICOS	PARÁMETROS	OPERACIÓN	CICLOS	FLAG
btfsc	f,b	Testea el bit b de f y salta si 0	2	-
btfss	f,b	Testea el bit b de f y salta si 1	2	-

decfsz	f,d	Decremento de f y salta si es 0	2	-
incfsz	f,d	Incremento de f y salta si es 0	2	-

INSTRUCCIONES QUE MANEJAN OPERANDOS INMEDIATOS

NEMÓNICOS	PARÁMETROS	OPERACIÓN	CICLOS	FLAG
addlw	k	Suma de literal con W	1	C,DC,Z
andlw	k	Operación AND de literal con W	1	Z
iorlw	k	Operación OR de literal con W	1	Z
movlw	k	Movimiento de literal a W	1	-
sublw	k	Resta W de literal (k-W)	1	C,CD,Z
xorlw	k	Operación OR exclusiva de literal con W	1	Z

INSTRUCCIONES DE CONTROL Y ESPECIALES

NEMÓNICOS	PARÁMETROS	OPERACIÓN	CICLOS	FLAG
call	k	Llamada a una subrutina	2	-
clrwdt	-	Borrado del perro guardián	1	TO#,PD#
goto	k	Salto a una dirección	2	-
retfie	-	Retorno de una interrupción	2	-
retlw	k	Retorno devolviendo literal en W	2	-
return	-	Retorno de subrutina	2	-
sleep	-	Puesta del microprocesador en reposo	1	TO#,PD#

PRINCIPALES REGISTROS DE CONTROL

REGISTRO DE ESTADO

Es el registro más usado de todos, ya que sus bits están destinados a controlar las funciones vitales del procesador. Por ello, está duplicado en las 4ª posiciones de cada banco.

REGISTRO DE ESTADO (STATUS)

IRP	RP1	RP0	TO#	PD#	Z	DC	C
7							0

Los tres bits de menor peso son los señalizadores de ciertas condiciones en las operaciones lógico-aritméticas:

- **Z ⇒ Señalizador de cero**. Se pone a nivel alto cuando el resultado de una operación es 0.
- **C ⇒ Señalizador de acarreo(Carry) / de llevada del 8 bit**. Pasa a nivel alto cuando existe acarreo en el bit de más peso en las instrucciones de suma. También actúa como señalizador de llevada en las instrucciones de resta, pero en este caso la correspondencia es inversa (hay llevada cuando vale 0).
- **DC ⇒ Señalizador de acarreo / de llevada en el 4 bit**. Funciona igual que el anterior, pero para el 4 bit., muy útil en las operaciones con números en BCD.

Los señalizadores **PD# y TO#** son activos a nivel bajo y sirven para indicar la causa que ha provocado la reinicialización del procesador. Son bits solamente leíbles.

Los microcontroladores PIC se resetean al conectar la alimentación (**POR:** Power-on-reset) y también cuando la alimentación baja de 4 voltios (**BOR:** Brown-out-reset), aunque esta función puede desactivarse poniendo a 0 el bit **BODEN**, de la palabra de configuración.

- **PD# ⇒** Este señalizador se pone a 0 al ejecutarse la instrucción **SLEEP** (modo reposo). Se pone a 1 automáticamente tras conectarse la alimentación, o bien, al ejecutarse la instrucción **CLRWDT**(refrescar el perro guardián).
- **TO# ⇒** Este señalizador se pone a 0 al desbordarse el perro guardián y toma el valor 1 tras la conexión de la alimentación o al ejecutarse la instrucciones **SLEEP** y **CLRWDT**.

Los tres bits de más peso del Registro de Estado se emplean para seleccionar el banco de la memoria RAM a la que se desea acceder. En el direccionamiento directo, se utilizan los bits **RP1** y **RP0** de acuerdo con la siguiente codificación:

BANCO	RP1	RP0
0	0	0
1	0	1
2	1	0
3	1	1

El bit **IRP** se usa concatenado con el bit de más peso de registro **FSR** para la elección del banco de memoria mediante un direccionamiento indirecto. Si vale 0 se seleccionan los dos primeros bancos y si vale 1 los otros dos.

REGISTRO DE OPCIONES

El registro de opciones tiene las siguientes funciones:

1. Asigna el divisor de frecuencias al **TMR0** o al perro guardián (**WDT**).
2. Elige el rango en el que trabaja el divisor de frecuencia.
3. Selecciona el tipo de reloj del **TMR0**, que puede ser interno o externo, a través de la patita **T0CKI**. También se puede seleccionar el flanco activo.
4. Selecciona el flanco activo para la interrupción externa por **RB0/INT**.
5. Activa o desactiva las resistencias de pull-up de la puerta B.

REGISTRO DE OPCIONES (OPTION)

RBPU#	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

7 0

PSP2	PSP1	PSP0	DIVISIÓN DEL TMR0	DIVISIÓN DEL WDT
0 0	0		1:2	1:1
0 0	1		1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

- **PSPA** ⇒ Asignación del divisor de frecuencias.
  - ✓ 1= El divisor de frecuencias se le asigna al perro guardián (**WDT**).
  - ✓ 0= El divisor de frecuencias se le asigna al **TMR0**.

- **T0SE** ⇒ Tipo de flanco de entrada, para el **TMR0**, en **T0CKI**.
  - ✓ 1= Incremento del **TMR0** cada flanco descendente.
  - ✓ 0= Incremento del **TMR0** cada flanco ascendente.

- **T0CS** ⇒ Tipo de reloj para el **TMR0**.
  - ✓ 1= Pulsos introducidos a través de **T0CKI** (contador).
  - ✓ 0= Pulsos de reloj interno F_{OSC}.

- **INTEDG** ⇒ Flanco activo de la interrupción externa.
  - ✓ 1= Flanco ascendente.
  - ✓ 0= Flanco descendente.

- **RBPU#** ⇒ Resistencias de pull-up de la puerta B.

- ✓ 1= Desactivadas.
- ✓ 0= Activadas.

REGISTRO DE CONTROL DE INTERRUPCIONES

Este registro se encarga de controlar las interrupciones provocadas:

- ✓ Por desbordamiento del **TMR0**.
- ✓ Por el cambio de estado en las 4 líneas de más peso de la puerta B.
- ✓ Por activación de la patita **RB0/INT**.

REGISTRO DE CONTROL DE INTERRUPCIONES (INTCON)

GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

7 0

- ➢ **GIE** ⇒ Bit de permiso global de interrupciones (1=Permitido y 0=Prohibido).
- ➢ **PEIE** ⇒ Bit de permiso de los periféricos que no se controlan con **INTCON**. Este bit actúa como una segunda llave parcial de permiso o prohibición de las causas de interrupción que no están contempladas en el registro **INTCON** y que las provocan los restantes periféricos del microcontrolador.
- ➢ **T0IE** ⇒ Bit de permiso de interrupción del **TMR0**.
- ➢ **INTE** ⇒ Bit de permiso de la interrupción externa por **RB0/INT**.
- ➢ **RBIE** ⇒ Bit de permiso de la interrupción por cambio en **RB4-RB7**.
- ➢ **T0IF** ⇒ Bit señalizador del desbordamiento en **TMR0**.
- ➢ **INTF** ⇒ Bit señalizador de la activación de la patita **RB0/INT**.
- ➢ **RBIF** ⇒ Bit señalizador del cambio en **RB4-RB7**.

REGISTROS DE PERMISO DE INTERRUPCIONES 1 Y 2

Estos registros contienen los bits que permiten(1) o prohíben(0) las interrupciones provocadas por los periféricos internos del microcontrolador y que no estaban contempladas en el registro **INTCON**.

REGISTRO DE PERMISO DE INTERRUPCIONES (PIE1)

PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE

7 0

- ➢ **PSPIE** ⇒ Bit de permiso de interrupción para la puerta paralela esclava al realizar una operación de lectura / escritura.
- ➢ **ADIE** ⇒ Bit de permiso de interrupción para el conversor A/D al finalizar la conversión.

- **RCIE** ⇒ Bit de permiso de interrupción para el receptor del USART cuando el buffer se llena.
- **TXIE** ⇒ Bit de permiso de interrupción para el transmisor del USART cuando el buffer se vacía.
- **SSPIE** ⇒ Bit de permiso de interrupción para la puerta serie síncrona.
- **CCP1IE** ⇒ Bit de permiso de interrupción para el módulo **CCP1** cuando se produce una captura o comparación..
- **TMR2IE** ⇒ Bit de permiso de interrupción en el desbordamiento del **TMR2**.
- **TMR1IE** ⇒ Bit de permiso de interrupción en el desbordamiento del **TMR1**.

REGISTRO DE PERMISO DE INTERRUPCIONES (PIE2)

-	0	-	EEIE	BCLIE	-	-	CCP2IE

7 0

- **EEIE** ⇒ Bit de permiso de interrupción por fin de escritura en la EEPROM de datos.
- **BCLIE** ⇒ Bit de permiso de interrupción por colisión de bus en el SSP cuando dos o más maestros tratan de transferir al mismo tiempo.
- **CCP2IE** ⇒ Bit de permiso de interrupción para el módulo **CCP2** cuando se produce una captura o comparación.

REGISTROS SEÑALIZADORES DE INTERRUPCIONES 1 Y 2

Estos registros actúan de señalizadores del momento en el que se origina la causa que provoca la interrupción, independientemente de si está permitida o prohibida.

REGISTRO SEÑALIZADOR DE INTERRUPCIONES (PIR1)

PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF

7 0

REGISTRO SEÑALIZADOR DE INTERRUPCIONES (PIR2)

-	0	-	EEIF	BCLIF	-	-	CCP2IF

7 0

REGISTRO SEÑALIZADOR DEL TIPO DE RESET

REGISTRO SEÑALIZADOR DEL TIPO DE RESET (PCON)

-	-	-	-	-	-	POR#	BO#

7 0

- **POR#** ⇒ Bit señalizador del reset por conexión de la tensión alimentación (Power On Reset). Si está a 0 ha habido este tipo de reset.
- **BO#** ⇒ Bit señalizador del fallo en la alimentación (Brown-Out). Si está a 0 el reset ha sido provocado por un fallo en la tensión de alimentación.

MANEJO DE LAS MEMORIAS FLASH Y EEPROM

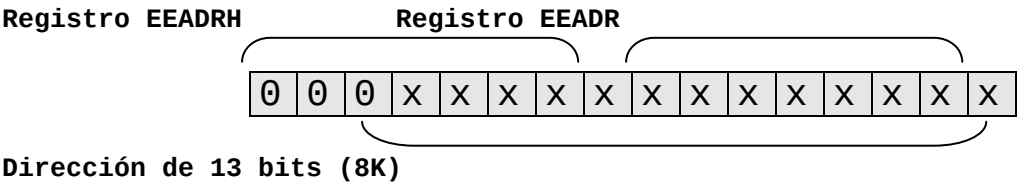
En este microcontrolador se puede leer y escribir en la memoria de código FLASH. Esto significa que un programa, de forma dinámica puede generar información que se puede grabar en esta memoria directamente, sin necesidad de utilizar el grabador externo.

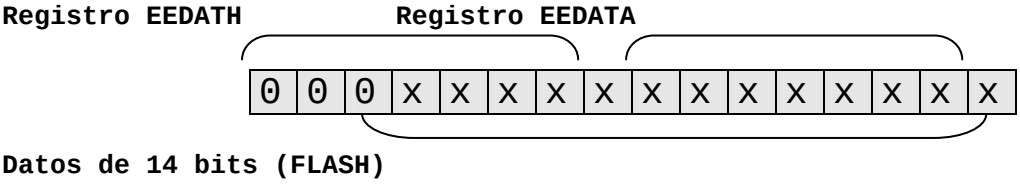
Utilizando esta característica, un microcontrolador incorporado al control de un motor de un automóvil permite que pueda modificarse el programa durante la rutina de mantenimiento periódico, compensando los desgastes, y otros factores como la instalación de piezas nuevas, compresión, etc.

La propia aplicación se puede reprogramar según las condiciones externas. Es posible ampliar el área de datos no volátiles (memoria EEPROM), con posiciones libres de la memoria de código FLASH.

La memoria FLASH de este microcontrolador puede alcanzar un tamaño de 8 K palabras de 14 bits cada una. No es suficiente con un solo registro para la dirección, que alcanza los 13 bits, y lo mismo sucede para el dato, que tiene una longitud de 14 bits.

Para solventar esta necesidad, el registro **EEADR** se concatena con el **EEADRH**, que contiene los 5 bits de más peso de la dirección. Por otra parte, el registro **EEDATH** se concatena con el **EEDATA** y contiene los 6 bits de más peso de la palabra a escribir en la memoria FLASH, o los 6 bits de más peso de la palabra leída de esta.





Para controlar la operación de lectura / escritura de las memorias EEPROM y FLASH, hay dos registros llamados **EECON1** y **EECON2**. Este segundo registro solo se utiliza en la delicada operación de escritura, que tiene una elevada duración de 2 milisegundos.

El registro **EECON1** se configura programando adecuadamente los siguientes bits:

REGISTRO EECON1

EEPGD	-	-	-	WRERR	WREN	WR	RD
7							0

- **EEPGD** ⇒ Bit que selecciona el acceso al tipo de memoria.
  - ✓ Puesto a 1 se selecciona la memoria FLASH.
  - ✓ Puesto a 0 se selecciona la EEPROM.
- **WRERR** ⇒ Bit señalizador de error en escritura.
- **WREN** ⇒ Bit de permiso de escritura.
- **WR** ⇒ Bit para iniciar la escritura, poniéndolo a 1 y pasa automáticamente a 0 cuando se finaliza dicha operación.
- **RD** ⇒ Bit para iniciar la operación de lectura, poniéndolo a 1.

Antes de iniciar la operación de escritura de una palabra en la memoria EEPROM se escribe en el registro **EECON2** primero el dato 55h y luego el AAh. Esta secuencia de inicialización está establecida por el fabricante del microcontrolador Microchip.

Para evitar escrituras indeseadas en la EEPROM motivadas por datos erróneos en la reinicialización del microcontrolador, se controla el bit **WREN**, prohibiendo cualquier operación de escritura mientras duran los 72 milisegundos que dura el Timer de power-up.

Para realizar la misma operación en la memoria FLASH se debe desactivar (poner a 0) el bit **WRT** de la palabra de configuración, que sólo puede escribirse desde un grabador externo. Dependiendo del valor de los bits de protección de código **CP1** y **CP0**, ubicados en la palabra de configuración, se consiguen diversas alternativas de protección contra lectura y escritura de la memoria FLASH.

```
;*******************************************************************
;**  Subrutina de lectura de la memoria EEPROM de datos
;*******************************************************************
bsf         STATUS,RP1
bcf         STATUS,RP0  ;Selecciona el banco2 de la memoria RAM
movf        ADDR_L,W    ;Se pone en EEADR la dirección a leer
movwf       EEADR
bsf         STATUS,RP0  ;Selecciona el banco3 de la memoria RAM
bcf         EECON1,EEPGD ; ;Se selecciona el acceso a la memoria EEPROM
bsf         EECON1,RD   ;Se da la orden de lectura
bcf         STATUS,RP0  ;Selecciona el banco2 de la memoria RAM
movf        EEDATA,W    ;Se coge el dato leído y se deja en DATA_L

;*******************************************************************
```

```
;**  Subrutina de escritura de la memoria EEPROM de datos
;*************************************************************************

bsf         STATUS,RP1
bcf         STATUS,RP0  ;Selecciona el banco2 de la memoria RAM
movf        ADDR_L,W    ;Se pone en EEADR la dirección a la que acceder
movwf       EEADR
movf        DATA_L,W    ;Se pone en EEDATA el dato a escribir
movwf       EEDATA
bsf         STATUS,RP0  ;Selecciona el banco3 de la memoria RAM
bcf         EECON1,EEPGD ;Se selecciona el acceso a la memoria EEPROM
bsf         EECON1,WREN ;Se habilita la escritura en la EEPROM
bcf         INTCON,GIE  ;se prohíben todas las interrupciones
movlw       55h
movwf       EECON2      ;Secuencia establecida por Microchip
movlw       AAh
movwf       EECON2
bsf         EECON1,WR   ;se da la orden de escritura
bsf         INTCON,GIE  ;se habilitan las interrupciones, para estar a la
sleep                   ;espera de la interrupción por fin de escritura.
bcf         EECON1,EEIF ;Se borra el flag de fin de escritura
bcf         EECON1,WREN ;Se prohíbe la escritura de nuevos datos
```

Las subrutinas para realizar las operaciones de escritura y lectura en la memoria FLASH de código son:

```
;*************************************************************************
;** Subrutina de escritura de la memoria FLASH
;*************************************************************************

bsf         STATUS,RP1
bcf         STATUS,RP0  ;Selecciona el banco2 de la memoria RAM
movf        ADDR_H,W    ;Se pone en EEADRH la parte alta de la dirección
movwf       EEADRH      ;a escribir
movf        ADDR_L,W    ;Se pone en ADDR la parte baja de la dirección
movwf       EEADR       ;a escribir
movf        DATA_H,W    ;Se pone EEDATH la parte alta del dato a escribir
movwf       EEDATH
movf        DATA_L,W    ;Se pone en EEDATA la parte alto del dato
movwf       EEDATA      ;a escribir
bsf         STATUS,RP0  ;Selecciona el banco3 de la memoria RAM
bsf         EECON1,EEPGD ;Se selecciona el acceso a la memoria FLASH
bsf         EECON1,WREN ;Se habilita la escritura en la memoria FLASH
bcf         INTCON,GIE  ;Se prohíben todas las interrupciones
movlw       55h
movwf       EECON2
movlw       AAh
movwf       EECON2      ;Secuencia recomendada por Microchip
bsf         EECON1,WR   ;Se da la orden de escritura
nop                     ;Con estas ordenes el microcontrolador no hace nada
nop                     ;y se espera a que se termine la escritura
bsf         INTCON,GIE  ;Se habilitan las interrupciones
bcf         EECON1,WREN ;Se prohíbe la escritura de nuevos datos

;*************************************************************************
;** Subrutina de lectura de la memoria FLASH
;*************************************************************************

bsf         STATUS,RP1
```

```
bcf         STATUS,RP0  ;Selecciona el banco2 de la memoria RAM
movf        ADDR_H,W    ;Se pone en EEADRH la parte alta de la dirección
movwf       EEADRH      ;a leer
movf        ADDR_L,W    ;Se pone en ADDR la parte baja de la dirección
movwf       EEADR       ;a leer
bsf         STATUS,RP0  ;Selecciona el banco3 de la memoria RAM
bsf         EECON1,EEPGD ;Se selecciona el acceso a la memoria FLASH
bsf         EECON1,RD   ;Se da la orden de lectura de la memoria FLASH
nop
bcf         STATUS,RP0  ;Selecciona el banco2 de la memoria RAM
movf        EEDATA,W    ;Se coge la parte baja del dato leído
movwf       DATA_L
movf        EEDATH,W    ;Se coge la parte alta del dato leído
movwf       DATA_H
```

PALABRAS DE CONFIGURACIÓN E IDENTIFICACIÓN

Existen 4 posiciones reservadas en la memoria de programa, para almacenar las palabras de identificación (ID). En estas palabras sólo se emplean los 4 bits de menos peso, en donde se almacena el numero de serie, códigos de identificación, numeraciones secuénciales o aleatorias, etc.

También existe la palabra de configuración, que en este PIC de la gama media se compone de 14 bits que se escriben durante el proceso de grabación del dispositivo. Estos bit ocupan la posición reservada 2007h de la memoria de programa.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
CP1	CP0	CP1	CP0	CP1	CP0	-	BODEN	CP1	CP0	PWRTE	WDTE	FOSC0	FOSC1

- **FOSC<1:0>** ⇒ Selección tipo de oscilador
  - ✓ 11 = Oscilador RC
  - ✓ 10 = Oscilador HS
  - ✓ 01 = Oscilador XT
  - ✓ 00 = Oscilador LP
- **WDTE** ⇒ Activación del perro guardián
  - ✓ 1 = Perro guardián activado
  - ✓ 0 = Perro guardián desactivado
- **BODEN** ⇒ Detección del fallo de alimentación (Brown-Out)
  - ✓ 1 = Detección activada
  - ✓ 0 = Detección desactivada.
- **PWRTE** ⇒ Activación del temporizador por conexión de la alimentación (Power-Up).
  - ✓ 1 = Desactivado
  - ✓ 0 = Activado.

ALGUNOS PERIFERICOS DESTACADOS

Nos ceñiremos a los más básicos y los utilizados por el PIC en su programción como controlador.

MÓDULOS DE CAPTURA, COMPARACIÓN Y PWM

El microcontrolador PIC16F877 dispone de 2 de estos módulos, denominados **CCP1 y CCP2**. que son idénticos excepto a la modalidad de Disparo Especial. Estos módulos pueden realizar tres funciones:

- **Captura**: Una pareja de registros de uno de estos dos módulos captura el valor que tiene el TMR1 cuando ocurre un evento especial en la patita **RC2/CCP1** (para el módulo **CCP1**) o en la patita **RC1/T1OSI/CCP2** (para el módulo **CCP2**).

- **Comparación**: Se compara el valor de 16 bits del TMR1 con otro valor cargado anteriormente en una pareja de registros de un módulo **CCPx** y cuando coinciden se produce un evento en la/s patita/s **RC2/CCP1** y/o **RC1/T1OSI/CCP2**.

- **Modulación de anchura de pulsos (PWM)**: En esta función se realiza dentro del intervalo del periodo de un impulso controlando la anchura en la que la señal está a nivel alto. Esta técnica es muy útil para el control de motores de corriente continua.

Los dos módulos **CCPx** utilizan un registro de trabajo de 16 bits que esta formado por la concatenación del los registros **CCPRxH-CCPRxL**, y unos registros de control **CCPxCON**; estos son los que hay que programar para que los módulos trabajen en el modo deseado en cada momento de la aplicación. Las parejas de registros de trabajo son las encargadas de capturar el valor del TMR1, de comparar el valor que tienen con el del TMR1 o, de modular la anchura del impulso.

REGISTRO CCPxCON

-	-	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
7							**0**

ASIGNACIÓN DE LOS BITS DE LOS REGISTROS CCPxCON

CCPxM3-0	MODO DE TRABAJO DEL MÓDULO
0000	Módulo CCPx desconectado.
0100	Modo Captura con cada flanco descendente en RCy/CCPx.
0101	Modo Captura con cada flanco ascendente en RCy/CCPx.
0110	Modo Captura cada 4 flancos ascendentes en RCy/CCPx.
0111	Modo captura cada 16 flancos ascendentes en RCy/CCPx.
1000	Modo Comparación que activa la RCy/CCPx al coincidir.
1001	Modo Comparación que desactiva la RCy/CCPx al coincidir

1010	Modo Comparación que genera una interrupción software.
1011	Modo Comparación en el que se produce un disparo especial
11xx	Modo PWM.

MODO DE CAPTURA

Como los módulos son idénticos excepto en la modalidad de Disparo Especial, de aquí en adelante se utiliza el módulo **CCP1**.

Funcionando el módulo en este modo, la pareja de registros **CCP1H-L** del módulo **CCP1** captura el valor de 16 bits que contiene el TMR1, cuando sucede un evento en la patita **RC2/CCP1** de la Puerta C, que anteriormente ha sido configurada como entrada.

Para la selección de los posibles eventos se programa el registro **CCP1CON**, cargando los valores adecuados en los bits **CCP1M3-0**. Los eventos que disparan esta captura son:

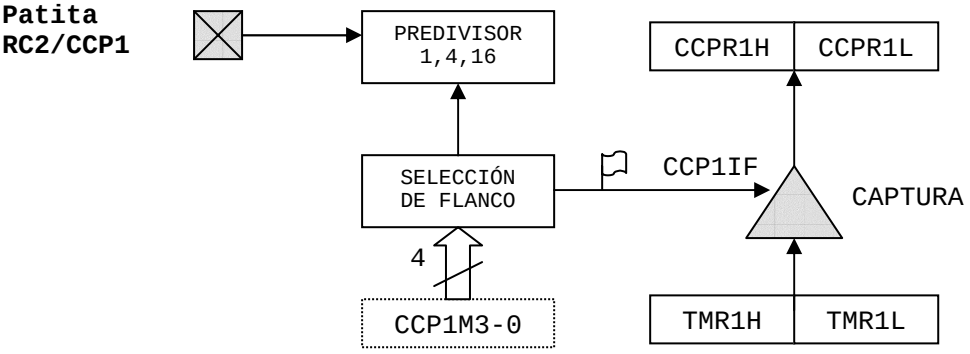
- Un flanco ascendente en dicha patita de entrada.
- Un flanco descendente.
- Cada 4 flancos ascendentes.
- Cada 16 flancos ascendentes.

Al efectuar la captura, se activa el bit señalizador **CCP1IF** del registro **PIR1**. Además, si se pone a nivel alto el bit de permiso de interrupción **CCP1IE** del registro **PIE1,** se genera una petición de interrupción cuando se carga en la pareja de registros **CCPR1H-L** el valor del TMR1. Este debe estar configurado para trabajar como un Temporizador o como un Contador Síncrono, nunca en modo asíncrono.

Si se van a cambiar las condiciones de funcionamiento, conviene desactivar el módulo **CCP1** para evitar que se produzcan falsas interrupciones durante la operación. Cuando se desactiva dicho módulo se borra la codificación del predivisor de frecuencia, por lo que habría que volver a codificarlo.

Si no se ha leído el contenido de los registros **CCPR1H-L** y se produce una nueva captura, dicha pareja de registros pasan a contener el nuevo valor.

Una aplicación interesante del modo de captura puede ser medir los intervalos de tiempo que existen entre los impulsos que llegan a la patita **RC2/CCP1**, que previamente ha sido configurada como entrada, donde el TMR1 debe trabajar con entrada de reloj externo sincronizada.



MODO DE COMPARACIÓN

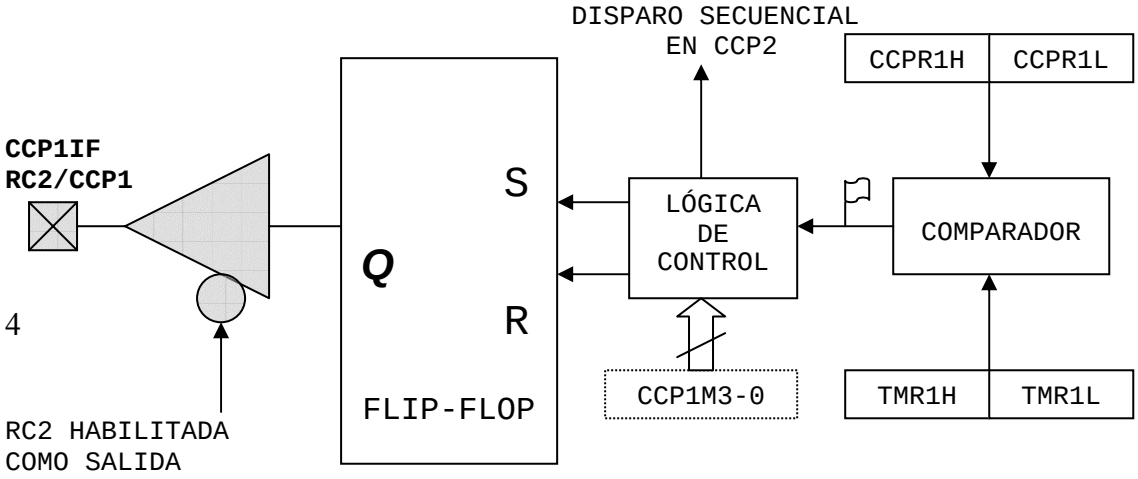
En este modo de trabajo, la pareja de registros **CCPR1H-L** compara su contenido, de forma continua, con el valor del TMR1. Cuando coinciden ambos valores, en la patita **RC2/CCP1** (para el módulo 1), que ha sido previamente configurada como salida, se produce uno de los siguientes eventos, en consonancia con la programación de los bits **CCP1M3-0**:

- Pasa a nivel alto.
- Pasa a nivel bajo.
- No cambia de estado, pero se produce una interrupción.

Al coincidir los valores del TMR1 con los de la pareja de registros, el bit señalizador **CCP1IF** del registro **PIR1** pasa a nivel alto, y si el bit de permiso de interrupción **CCP1IE** del registro **PIE1** está a nivel alto, cuando coinciden dichos valores, se origina una petición de interrupción.

El TMR1 debe estar configurado para trabajar como un Temporizador o como un Contador Síncrono, nunca en modo Asíncrono.

Si seleccionamos el Disparo Especial, el módulo **CCP1** pone a 0 el TMR1, y el **CCPR1** pasa a funcionar como un Registro de Período, capaz de provocar periódicamente interrupciones. En este modo, el módulo **CCP2**, pone a 0 el TMR1 y, además inicia una conversión A/D; por lo que es capaz de realizar conversiones periódicamente sin el control del programa de instrucciones.



MODO DE PWM

Con este modo de funcionamiento, se consiguen impulsos lógicos cuya anchura del nivel alto es de duración variable, que son de enorme aplicación en el control de motores de corriente continua, de triacs, etc.

La patita **RC2/CCP1** está configurada como salida y bascula entre los niveles lógicos 0 y 1, a intervalos de tiempo variables. Lo que se intenta es obtener un impulso cuyo nivel alto tenga una anchura variable (Duty Cycle) dentro del intervalo del periodo de trabajo.

Para lograr este basculado se usa un comparador que pone a 1 (Set) un flip-flop cuando el valor del registro **PR2** coincide con la parte alta del TMR2, momento en el que el TMR2 toma el valor 00h. Luego el flip-flop se resetea cuando otro comparador detecta la coincidencia del valor existente en **CCPR1H** con el de la parte alta del TMR2. De esta manera, variando los valores que se cargan en **PR2** y en **CCPR1L** (que luego se traspasa al **CCPR1H**) se varía el intervalo de tiempo en el que la patita está a 1 y a 0.

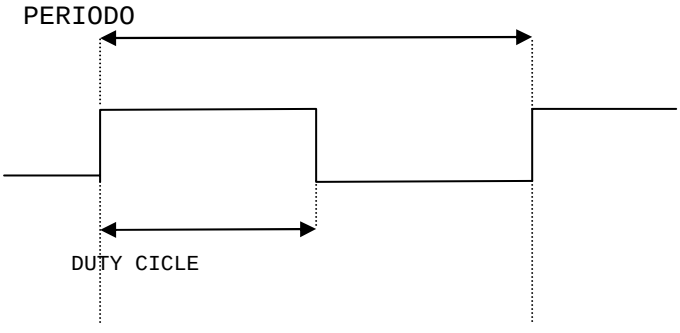
Cuando se trabaja con una precisión de 10 bits, los 2 bits **CCP1CON<5:4>** se concatenan con los 8 del registro **CCPR1L** y, de la misma forma, los 8 bits de más peso del TMR2 se concatenan con los dos bits de menos peso del reloj interno, haciendo que TMR2 cuente cada T_{OSC} en vez de cada $4 \cdot T_{OSC}$.

El tiempo que dura el período de la onda depende del valor cargado en el registro **PR2**, según la siguiente formula:

$$Periodo=[(PR2)+1] \cdot 4 \cdot T_{OSC} \cdot Predivisor\ TMR2$$

El tiempo que la patita de salida está a nivel alto, que es la anchura del impulso, depende del contenido cargado en **CCPR1L** y de los bits 5 y 4 del registro **CCP1CON**, cuando se trabaja con una precisión de 10 bits.

$$Anchura\ Impulso=(CCPR1L:CCP1CON<5:4>) \cdot T_{osc} \cdot Predivisor\ TMR2$$



Cuando el valor del TMR2 coincide con el del **PR2** suceden tres acontecimientos:

1) Se borra el TMR2.

2) La patita **RC2/CCP1** se pone a 1.

3) El valor del registro **CCPR1L**, que es el que determina la anchura del impulso, se carga en el registro **CCPR1H**.

El valor **CCPR1L:CCP1CON<5:4>** puede cargarse en cualquier momento, puesto que el mismo no se traspasa a **CCPR1H** y se compara hasta que coincidan **PR2** con TMR2. En este modo de funcionamiento el registro **CCPR1L** sólo puede ser leído.

Los paso a seguir para realizar la configuración del modo de modulación PWM son:

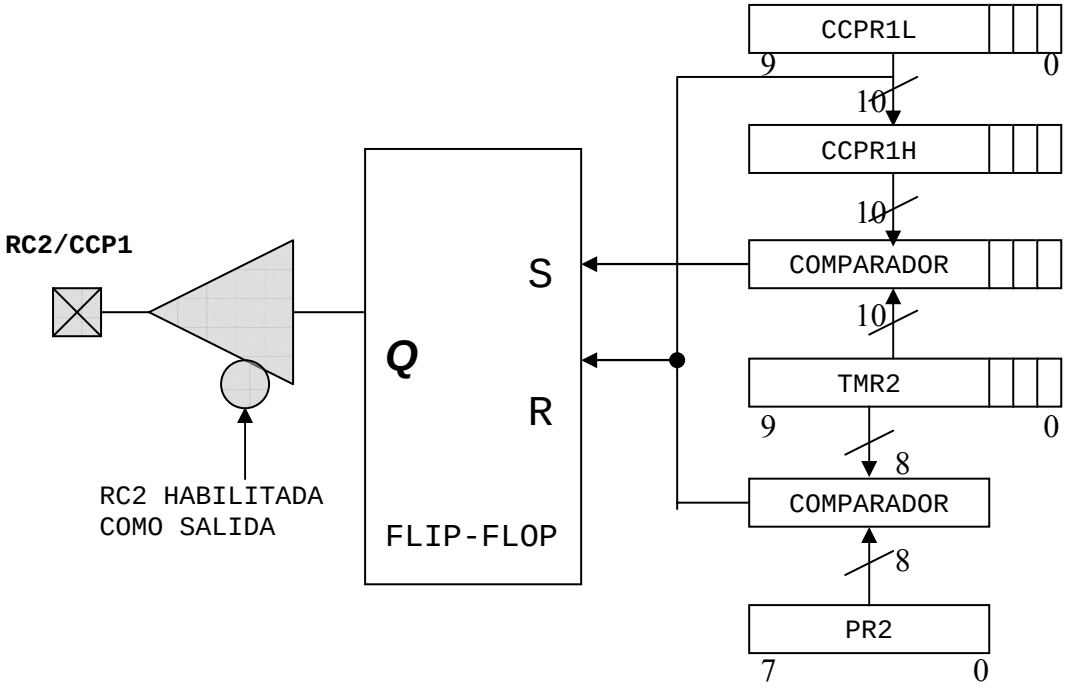
1) Asignar el periodo cargando el valor oportuno en el registro **PR2**.

2) Asignar la anchura del pulso cargando el registro **CCPR1L** y los bits del **CCP1CON<5:4>**.

3) Configurar la patita **RC2/CCP1** como una salida.

4) Asignar el valor del predivisor y activar el TMR2 programando el **T2CON**.

5) Configurar el módulo **CCP1** en modo PWM.



MÓDULO DE COMUNICACIÓN SERIE SINCRONA MSSP

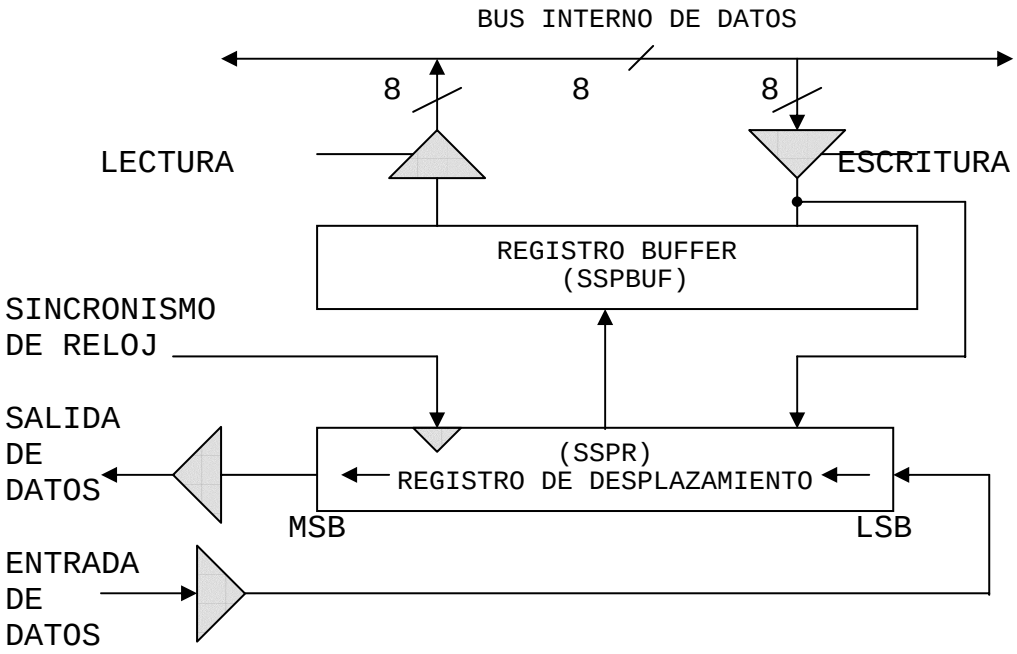
La comunicación serie es una forma muy apreciada de comunicación digital entre sistemas y circuitos integrados, debido al reducido numero de líneas que se necesitan. Este microcontrolador lo lleva implementado, proporcionando un excelente interfaz de comunicación con diversos periféricos.

El módulo MSSP (Master Synchronous Serial Port) admite dos de las alternativas más usadas en la comunicación serie síncrona:

- **SPI** (Serial Peripheral Interface): Este tipo de comunicación la utilizan principalmente las memorias (RAM y EEPROM) y utiliza tres líneas para ello.

- **I2C** (Inter-Integrated Circuit): Este tipo de comunicación implantada por la marca PHILIPS, tiene como principal característica la utilización de solo dos hilos y, recientemente, ha conseguido una importante implantación en la comunicación de circuitos integrados como memorias, controladores, relojes, convertidores, etc.

Este módulo consta básicamente de dos registros: el **SSPSR**, que un registro de desplazamiento que transforma la información serie en paralelo y viceversa, y el registro **SSPBUF**, que actúa como buffer de la información que se recibe o transmite.



En el funcionamiento del módulo en transmisión, el byte que se quiere transmitir se carga en el registro **SSPBUF** a través del bus de datos interno y automáticamente se traspasa al registro **SSPSR**, que va desplazando bit a bit el dato, sacándolo ordenadamente al exterior al ritmo de los impulsos de reloj. En recepción, los bits van entrando al ritmo del reloj por una patita y se van desplazando en el registro **SSPSR** hasta que lo llenan, en cuyo momento la información se traspasa al registro **SSPBUF**, donde queda lista para su lectura. Este doble almacenamiento del dato recibido permite iniciar la recepción de un nuevo dato antes de que se haya leído el último.

En la programación y control de este módulo intervienen los siguientes registros:

REGISTRO DE CONFIGURACIÓN SSPSTAT

SMP	CKE	D/A#	P	S	R/W#	UA	BF

7 0

- **SMP** ⇒ Determina el momento del muestreado del bit de entrada:
  - ✓ En modo maestro SPI, si vale 1 el dato de entrada se muestrea al final del impulso de reloj y si vale 0 se muestrea en la mitad del mismo.
  - ✓ En modo esclavo este bit no tiene ningún cometido y debe valer 0.
- **CKE** ⇒ Selecciona el flanco de reloj activo:
  - ✓ Si el bit **CKP** del registro **SSPCON** vale 0 significa que el estado de inactividad en la señal de reloj es el bajo y entonces si **CKE=1** el dato se transmite en el flanco ascendente de **SCK** y si **CKE=0** en el flanco descendente.
  - ✓ Si **CKP** vale 1 el estado de inactividad es el alto en la señal de reloj, por lo que cuando **CKE=1** el dato se transmite en el flanco descendente de **SCK** y si **CKE=0** en el ascendente.
- **D/A#** ⇒ Este bit (Dato/Dirección#) es utilizado en el modo I2C para indicar si el dato recibido es de información (1) o es una dirección (0).
- **P** ⇒ Este bit es utilizado en el modo I2C para la detección de llegada del bit de Parada o Stop poniéndose a 1.
- **S** ⇒ Este bit utilizado en el modo I2C detecta la llegada de la condición de Inicio o Start.
- **R/W#** ⇒ En el modo I2C indica si se trata de una Lectura / Escritura#.
- **UA** ⇒ Este bit utilizado en el modo I2C cuando vale 1 indica que la dirección es de 10 bits y que hay que cargar el byte alto, según la codificación expresada en el registro de dirección del bus **SSPADD**, y si vale 0 es de 7 bits.
- **BF** ⇒ Este bit indica cuándo vale 1 que el registro **SSPBUF** está lleno.

REGISTRO DE CONTROL SSPCON

WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

7 0

- **WCOL** ⇒ Este bit se pone a 1 cuando se ha producido una colisión. Esta colisión se produce cuando el registro **SSPBUF** carga en el registro **SSPSR** un byte sin haber dado tiempo a salir el anterior.
- **SSPOV** ⇒ Este bit se pone a 1 indicando sobrepasamiento, lo que significa que la información recibida en el registro **SSPSR** y que ha pasado al **SSPBUF**, cuando lega otro byte aún no se ha leído y se machacará la información.
- **SSPEN** ⇒ Es el bit de activación del módulo MSSP. Si vale 1 queda activada la puerta serie en modo SPI.

- **CKP** ⇒ Determina la polaridad o inactividad del reloj. Si vale 1 el estado de inactividad es alto, y si vale 0 es bajo.

SSPM3-0	MODO DE TRABAJO
0000	Modo maestro SPI, con reloj a $F_{OSC}/4$.
0001	Modo maestro SPI, con reloj a $F_{OSC}/16$.
0010	Modo maestro SPI, con reloj a $F_{OSC}/64$.
0011	Modo maestro SPI, con reloj igual a la salida del TMR2/2.
0100	Modo esclavo SPI, con reloj igual a la patita SCK y SS# a nivel bajo.
0101	Modo esclavo SPI, con reloj igual a la patita SCK y SS# no activo.
0110	Modo esclavo I2C, con dirección de 7 bits.
0111	Modo esclavo I2C, con dirección de 10 bits.
1000	Modo maestro I2C, con reloj = $F_{OSC}/(4\cdot(SSPADD + 1)$.
1011	Modo maestro I2C controlado por firmware.
1110	Modo maestro I2C controlado por firmware, con bits de Inicio y Stop, activada interrupción, y dirección de 7 bits.
1111	Modo maestro I2C igual que el anterior pero con dirección de 10 bits.

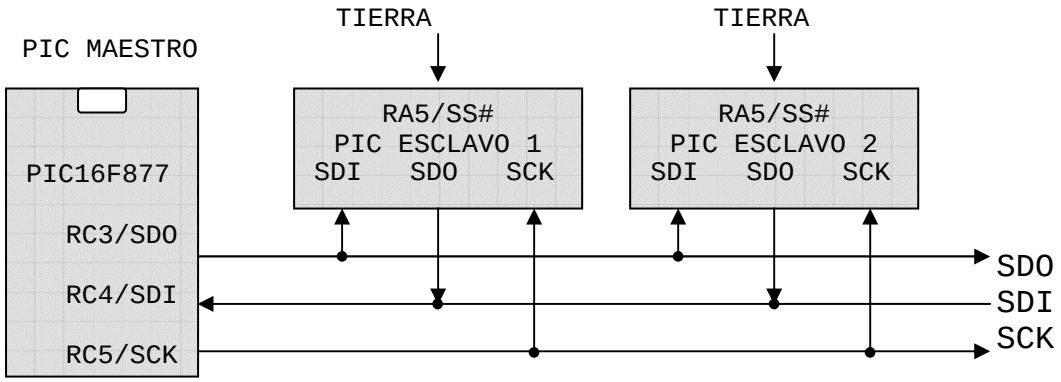
MODO DE FUNCIONAMIENTO SPI

Este modo de funcionamiento permite la transferencia de datos de 8 bits en serie, que pueden ser transmitidos y recibidos de forma síncrona y simultánea. Esta comunicación utiliza tres líneas del microcontrolador:

1. **SDO** (Serial Data Out): Esta patita es la salida de datos en serie.
2. **SDI** (Serial Data In): Esta patita es la entrada de datos en serie.
3. **SCK** (Serial Clock): Esta patita es el reloj de sincronización.

Cuando el microcontrolador trabaja en modo maestro hay que programar la patita **RC3/SDO** como salida, la patita **RC4/SDI** como entrada, y la patita **RC5/SCK** también como salida.

Cuando el microcontrolador trabaja en modo esclavo es necesario utilizar una cuarta línea, que seria la de selección del esclavo **SS#** y se pondría a tierra. La patita **RC5/SCK** debería configurarse como entrada.



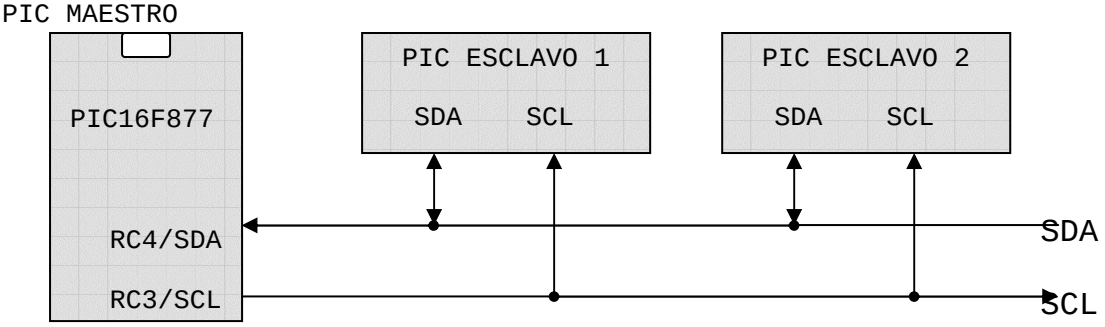
Al comenzar el funcionamiento en modo SPI, es necesario establecer las condiciones de trabajo programando adecuadamente los bits del registro **SSPCON** y los bits 7 y 6 del registro **SSPSTAT**, para determinar las siguientes especificaciones:

1. Si se trabaja en modo maestro, **SCK** debe estar programada como salida.
2. Si es en modo esclavo, **SCK** debe ser programada como entrada.
3. Hay que determinar la polaridad de la señal de reloj, es decir, su estado de inactividad.
4. Hay que programar es instante de muestreo(mitad o final del impulso).
5. Hay que programar el flanco de reloj activo(ascendente o descendente).
6. Hay que determinar el rango de trabajo del reloj en modo maestro.
7. Selección del modo esclavo, si se va ha trabajar de esta forma.

MODO DE FUNCIONAMIENTO I2C

Este modo de comunicación serie es un protocolo que utiliza solo dos líneas para la transferencia de información entre los elementos que se acoplan al bus. Una de dichas líneas se dedica a soportar los datos, es bidireccional y se llama **SDA**; la otra lleva los impulsos de reloj para la sincronización, es unidireccional y recibe el nombre de **SCL**. Estos impulsos siempre los genera el maestro y tienen la función de sincronizar las transferencias de datos con todos los dispositivos esclavos colgados a las dos líneas.

En el PIC16F877, el bus I2C está implementado en silicio tanto en el modo maestro como en el esclavo. En modo maestro es el PIC, el que inicia y finaliza la transferencia y genera los impulsos de reloj. También selecciona el esclavo al que se destina la información.



En el bus I2C, cada transferencia comienza con la condición de Inicio (Start) y termina con la condición de Parada (Stop). Ambas condiciones las genera el maestro, y la primera consiste en un flanco descendente en **SDA** mientras **SCL** tiene nivel alto. La condición de parada es un flanco ascendente mientras **SCL** tiene un nivel alto.

Los datos que se colocan sobre la línea **SDA** son bytes de 8 bits, que comienzan con el bit de mayor peso y termina con el menor peso, al cual sigue un noveno bit que es la condición de reconocimiento (**ACK**). Para este caso, el transmisor pone **SDA=1**, mientras que el receptor pone **SDA=0** en dicho impulso de reloj, prevaleciendo sobre la línea el nivel bajo.

El primer byte que envía el maestro tras la condición de inicio contiene la dirección del esclavo con el que se desea realizar la comunicación. El código 0 se usa para realizar una llamada general sobre todos los esclavos. La dirección consta de 7 bits, estando destinado el octavo bit a indicar la operación a realizar (**R/W#**), que puede ser de lectura o escritura. Tras este byte inicial de direccionamiento se manda otro que especifica las características de la operación a realizar.

Para activar el bus I2C hay que poner el bit **SSPEN** del registro **SSPCON** a 1. A partir de este momento las patitas **SDA** y **SCL** quedan configuradas para soportar este protocolo. Previamente estas patitas han de estar configuradas como entradas.

Para el control del bus I2C existen 6 registros: Dos registros de control **SSPCON** y **SSPCON2**; un registro de estado **SSPSTAT**; un registro de buffer para los datos **SSPBUF**; Un registro de desplazamiento no accesible directamente **SSPSR**; y un registro de dirección **SSPADD**.

Anteriormente se han descrito las características de todos los archivos menos el **SSPCON2**:

REGISTRO DE CONTROL SSPCON2

GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN

7 0

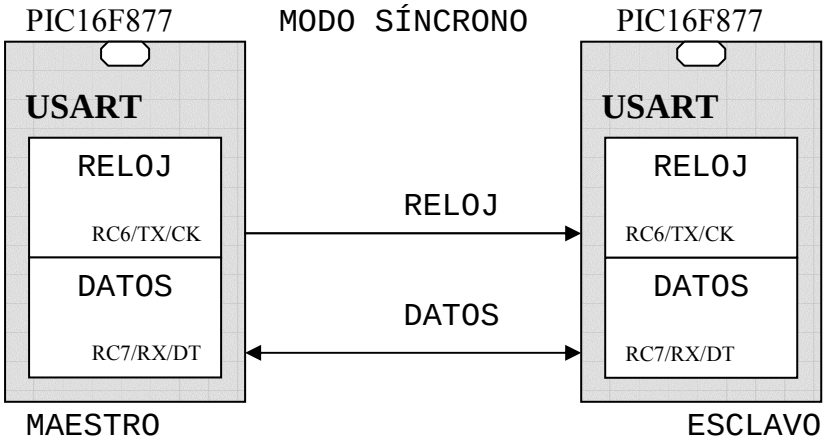
- **GCEN** ⇒ Este bit sólo se usa en el modo esclavo.
- **ACKSTAT** ⇒ Cuando este bit se pone a 1 significa que se ha recibido el bit de reconocimiento **ACK** del esclavo.
- **ACKDT** ⇒ Este bit es el de reconocimiento en el modo maestro en recepción. Si vale 0 significa que el maestro ha transmitido el bit de reconocimiento.
- **ACKEN** ⇒ Este bit indica si se pone a 1, que se ha iniciado la secuencia de generación de la condición de reconocimiento.

- **RCEN** ⇒ Este bit se utiliza para habilitar el modo de recepción del maestro.
- **PEN** ⇒ Este bit se usa para generar la condición de parada.
- **RSEN** ⇒ Este bit inicia la repetición de la condición de inicio cuando se pone a 1.
- **SEN** ⇒ Este bit hay que ponerlo a 1 para iniciar la condición de inicio.

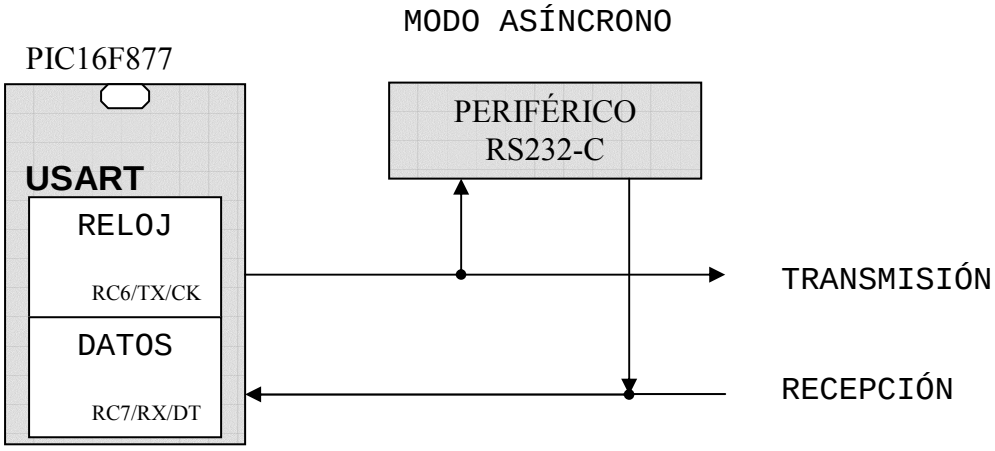
El USART: TRANSMISOR / RECEPTOR SÍNCRONO / ASÍNCRONO SERIE

El USART, llamado SCI (Serial Comunications Interface), puede funcionar como un sistema de comunicación full duplex o bidireccional asíncrono, adaptándose a multitud de periféricos y dispositivos que transfieren información de esta forma, como PC's, monitores CRT. También puede trabajar en modo síncrono unidireccional o half duplex para soportar periféricos como memorias, convertidores, etc.

En el USART en modo síncrono, la comunicación se realiza sobre dos líneas, la **DT** que traslada en los dos sentidos los bits a la frecuencia de los impulsos de reloj que salen por la línea **CK** desde el maestro.

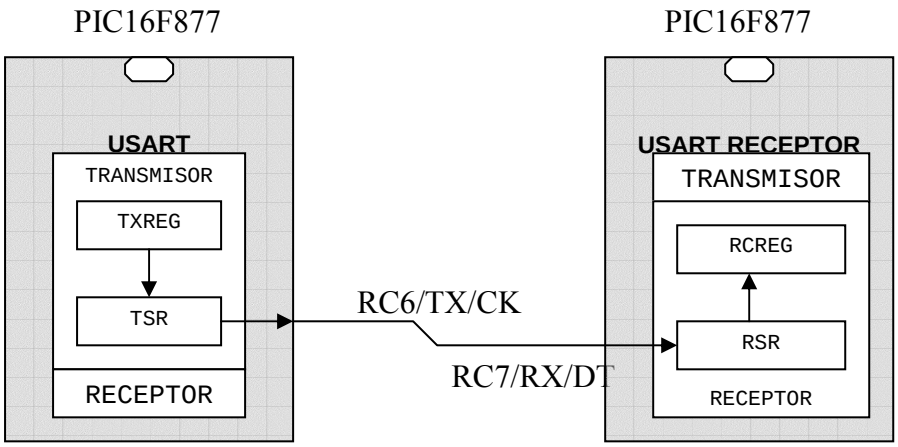


En modo asíncrono, las transferencias de información se realizan sobre dos líneas **TX** (transmisión) y **RX** (recepción), saliendo y entrando los bits por dichas líneas al ritmo de una frecuencia controlada internamente por el USART. Este tipo de comunicación usa la norma RS232-C, donde cada palabra de información o dato se envía independientemente de los demás. Suele constar de 8 o 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 (NonReturn-to-Zero). Los bits se transfieren a una frecuencia fija y normalizada.



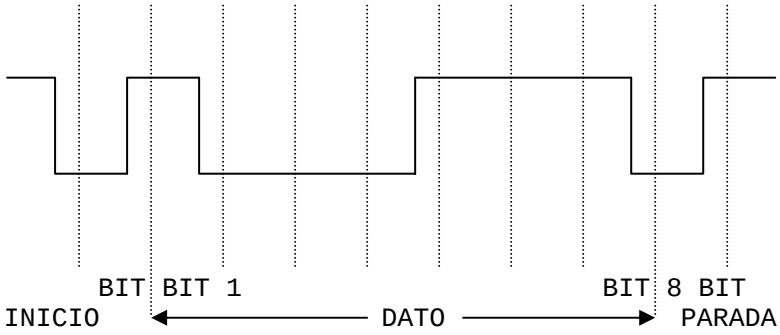
Los cuatro bloques que configuran la arquitectura del USART, en modo asíncrono, son:

- **Circuito de muestreo**.
- **Generador de baudios**.
- **Transmisor asíncrono**.
- **Receptor asíncrono**.



CIRCUITO DE MUESTREO

Este actúa sobre la patita **RC7/RX/DT**, que es donde se recibe el bit de información y control y se encarga de muestrear tres veces su valor, para decidir éste por mayoría.



GENERADOR DE BAUDIOS

En el protocolo RS232-C, la frecuencia en baudios (bit por segundo) a la que realiza la transferencia de datos, está normalizada: 330, 600,1200, 2400, 4800, 9600, etc. Para generar esta frecuencia, el USART dispone de un Generador de Frecuencia de Baudios (**BRG**), cuyo valor es controlado por el contenido grabado en el registro **SPBRG**. Además, la frecuencia en baudios del generador depende del bit **BRGH** del registro **TXSTA**, si vale 0 se trabaja en baja velocidad y si vale 1 en alta. Según este bit se obtendrá el valor de una constante K necesaria en la determinación de la frecuencia de funcionamiento.

$$\text{FRECUENCIA EN BAUDIOS} = F_{OSC} /(K\cdot(X+1))$$

$$X = (F_{OSC}/K\cdot F)-1$$

X es el valor cargado en SPBRG
Si BRGH=0, baja velocidad y K=64
Si BRGH=1, alta velocidad y K=16

TRANSMISOR ASÍNCRONO

El dato que se desea transmitir, se deposita en el registro **TXREG** y a continuación se traspasa al registro de desplazamiento **TSR**, que va sacando los bits secuencialmente y la frecuencia establecida, incluyendo los bits de Inicio y Parada. La transferencia entre los dos registros se realiza en un ciclo y entonces el señalizador **TXIF** del registro **PIR1** se pone a 1, para advertir que

el registro de transmisión se ha vaciado. También en ese momento puede producirse una interrupción si se ha habilitado programando el bit **TXIE=1** en el registro **PIE1**. Cuando se escribe de nuevo otro dato en el registro **TXREG**, el señalizador **TXIF** se pone a 0.

La secuencia a seguir para realizar una transmisión es la siguiente:

1. Configurar las patitas **RC6/TX/CK** como salida y **RC7/RX/DT** como entrada
2. Poner **SYNC=0** y **SPEN=1** para activar el USART en modo asíncrono.
3. Activar interrupción, si se desea, poniendo **TXIE=1**, además de habilitar las interrupciones en general.
4. Si el dato consta de 9 bits, poner el bit **TX9** a 1. El noveno bit se colocará en **TX9D** (**TXSTA**).
5. Se carga el valor adecuado en el registro **SPBRG**, para producir la frecuencia deseada.
6. Activar la transmisión, poniendo a 1 el bit **TXEN** del registro **TXSTA**. El bit **TXIF** del registro **PIR1** se encontrará a 1, ya que el registro **TXREG** está vacío.
7. Cargar en el registro **TXREG** el dato a transmitir. Comienza la transmisión.

REGISTRO DEL TRANSMISOR DEL USART (TXSTA)

CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D

7 0

- **CSRC** ⇒ Este bit selecciona el reloj a utilizar en el USART. En el modo asíncrono no influye; pero en el síncrono, si vale 1, se selecciona el modo maestro donde el reloj es generado internamente desde **BRG**, y si vale 0 el reloj es generado por una fuente externa.

- **TX9** ⇒ Habilita el bit 9 de transmisión.

- **TXEN** ⇒ Activa la transmisión.

- **SYNC** ⇒ Este es el bit de selección del modo del USART, con el que vamos a trabajar. si vale 1 es el modo síncrono y si vale 0 es el asíncrono.

- **BRGH** ⇒ Con este bit seleccionamos la velocidad de baudios. Si vale 1 es alta velocidad y si vale 0 es baja.

- **TRMT** ⇒ Este es el bit señalizador del estado del registro de desplazamiento de transmisión. Si vale 1 **TSR** está vacío y si vale 0 no está vacío.

- **TX9D** ⇒ Este es el bit 9 del dato a transmitir (puede ser el de paridad).

RECEPTOR ASÍNCRONO

Este recibe, uno a uno, los bits, elimina los dos de control, y los de información una vez que han llenado el registro de desplazamiento **RSR** los traslada automáticamente el registro **RCREG**, donde quedan disponibles para su posterior procesamiento. Estos bit de información van entrando secuencialmente en el registro de desplazamiento **RSR**, que funciona a una frecuencia 16 veces más rápida que la de trabajo.

Cuando el dato consta de 9 bits hay que programar el bit **RX9=1** y el noveno bit de información se colocará en el bit **RX9D** del registro **RCSTA**. Cuando el bit **CREN** del registro **RCSTA** vale 1, se habilita la recepción.

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 bit **RX9D** 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 **RSR** al buffer de recepción. Tras la interrupción, el esclavo deberá examinar la dirección y si coincide con la suya poner **ADDEN=0** para poder recibir datos del maestro. Si **ADDEN=1** como los datos son ignorados, el bit de Parada no se carga en **RSR**, por lo que este hecho no produce interrupción.

REGISTRO DEL RECEPTOR DEL USART (TXSTA)

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

7 0

- **SPEN** ⇒ Con este bit se habilita el puerto serie.
- **RX9** ⇒ Habilita el bit 9 de recepción.
- **SREN** ⇒ Configura la recepción sencilla:
  - ✓ Modo asíncrono: no influye.
  - ✓ Modo síncrono maestro: poniéndolo a 1 se habilita la recepción sencilla.
  - ✓ Modo síncrono esclavo: no se utiliza.
- **CREN** ⇒ Configura la recepción continua:
  - ✓ Modo asíncrono: poniéndolo a 1 se habilita la recepción continua.
  - ✓ Modo síncrono: poniéndolo a 1 se habilita la recepción continua hasta que el bit **CREN** es borrado.
- **ADDEN** ⇒ Con este bit se realiza la detección de dirección. En modo asíncrono con 9 bits (**RX9=1**):
  - ✓ 1 = activa la detección de dirección, activa la interrupción y descarga el buffer de recepción al activarse **RSR<8>**.
  - ✓ 0 = desactiva la detección de dirección, todos los bits son recibidos y el bit 9 puede ser utilizado como bit de paridad.
- **FERR** ⇒ Este es el bit de error de trama:
  - ✓ 1 = error de trama (puede ser actualizado leyendo el registro **RCREG** y recibir el siguiente dato válido).
  - ✓ 0 = no hay error de trama.
- **RX9D** ⇒ Este es el bit 9 del dato recibido (puede ser el de paridad).

La secuencia a seguir para realizar una recepción es la siguiente

1. Cargar con el valor **X** al registro **SPBRG** para trabajar con la frecuencia deseada, controlando además el valor del bit **BRGH**.
2. Habilitar el USART en modo asíncrono con **SPEN=1** y **SYNC=0**.
3. Si se desea generar una interrupción con la llegada de la señal de Parada, se pone **RCIE=1**, además de habilitar las interrupciones en general.
4. Poner RX9=1 para permitir la recepción del bit 9.
5. Para detectar la dirección, poner **ADDEN=1**.
6. Habilitar la recepción poniendo **CREN=1**.
7. Al terminarse la recepción **RCIF** se pondrá a 1 y se produce una interrupción si se había permitido.
8. Se lee el registro **RCSTA** y se averigua si se ha producido algún error.
9. Leer los 8 bits del registro **RCREG** para determinar si el dispositivo ha sido el direccionado.
10. Si se ha producido algún error, poner **CREN=0**.
11. Si ha sido direccionado el dispositivo, poner **ADDEN=0** para permitir la recepción de la información.

7.4 Información del Modem

Características principales del módem.

Las propiedades más importantes de este módem son:

- Apropiado para funcionamiento a 50 ó 60 Hz.
- Completo funcionamiento como módem para transmisión por línea de potencia.
- Técnica de modulación programable: S-FSK ó FSK.
- Frecuencia programable de la portadora en transmisión y recepción.
- Tasa de bit programable en transmisión y recepción: 300, 600, 1200, 2400, 4800 y 9600 baudios.
- Doble detección de canal.
- Ganancia de recepción programable.
- Bajo consumo, alimentación a 3.3 V
- Rigurosa información sobre el estado de transmisión y recepción del módem.
- Amplia administración de interrupciones para eficiencia de programación.

Descripción general del chip.

El chip CTBT es un módem que está indicado para la transmisión de datos por la línea de baja y media potencia. El módem trabaja a una tensión de alimentación de 3.3 V y su conexión a la línea de potencia se hará mediante un driver de potencia externo y un transformador. Un PLL interno se sintoniza a la frecuencia principal de 50 Hz y se utiliza para sincronizar la transmisión de datos con tasas de bit de 300, 600, 1200, 2400, 4800 y 9600 baudios para la frecuencia de 50 Hz, correspondiéndose a 3, 6, 12, 24, 48 y 96 bits de datos por cada medio ciclo de la frecuencia (50 Hz.)

Básicamente el módem consta de una celda con dos partes diferenciadas, una parte analógica y otra parte digital, además de un interfaz que nos permitirá leer, escribir y controlar los registros tanto de la parte analógica como de la digital para hacerlos controlables y leibles.

Gracias a la parte digital se podrá controlar toda la celda, ya que gracias al interfaz digital se podrá acceder a los registros tanto de la parte analógica como digital y por lo tanto podrá

hacerse la celda accesible y controlable, que es lo que realmente nos interesa, tener el mayor nivel de control sobre la celda y en definitiva sobre el módem.

Características generales de la celda :

- Interfaz A/D de bajo consumo sintonizable.
- Rango dinámico que abarca desde 20 kHz hasta 145 kHz con una resolución de frecuencia de 500 Hz.
- Control automático de ganancia de 40 dB más 80 dB del convertidor digital-analógico (Sigma-Delta) para 300 baudios, que nos proporciona un total de 120 dB en todo el rango dinámico (para 300 baudios).
- Tasa de bit programable desde 300 hasta 9600 baudios.
- Elevada potencia de salida (valor teórico 200 mA, valor real 400 mA)
- Soporta dos tipos de modulación, FSK y S-FSK.
- Tecnología digital CMOS
- Tensión de alimentación 3.3 V
- Aplicación: comunicaciones por línea de potencia (PLC communications).

Características eléctricas:

A continuación se muestra una tabla con las características eléctricas más importantes así como sus valores

Bloque Receptor

Todas las características que se muestran en la tabla siguiente son válidas para una tensión de alimentación de 3.3 V y temperatura ambiente entre - 40 y + 85 ºC

Parameter	***Symbol***	***Min***	***Typ***	***Max***	***Unit***
Power dissipation for the core (including voltage references) @ 3 Volts	Pd		2		mW
Resolution	N		16		bits
ADC full scale level	ADCFS		2		Vppdif
ADC max SNR for 300			80		dB

baud					
Total Harmonic Distortion for the ADC	THD			0.1	%
Sampling Frequency	Fs	80		580	kHZ
Filter centre frequency	Fo		Fs / 4		Hz
Filter bandwith	B		Fs / 50		Hz
Filter gain	G	0	20	40	dB

Tabla: Características eléctricas del bloque receptor

Bloque Transmisor :

Todas las características que se muestran en la tabla siguiente son válidas para una tensión de alimentación de 3.3 V y temperatura ambiente entre - 40 y + 85 ºC

Parameter	***Symbol***	***Min***	***Typ***	***Max***	***Unit***
Frequency resolution	Fr		10		Hz
DAC resolution	NDAC		10 **(9)**		bits
Maximun output current	Imax		200 **(400)**		mA
Total Harmonic Distortion at Imax	THD		3 **(1)**		%
Load Resistance	RL	5	20	100	Ohm

Tabla: Características eléctricas del bloque transmisor

Entre paréntesis y en negrita aparecen los valores que se han medido experimentalmente, el resto de los valores que se muestran en la tabla son los teóricos obtenidos durante el diseño de la celda.

Como puede observarse la máxima corriente de salida es el doble de la esperada, mientras que la distorsión harmónica total es bastante menor que la esperada.

Tanto el transmisor como el receptor presentan estas dos partes que se han comentado anteriormente bien diferenciadas (analógica y digital), empecemos viendo un esquema de la parte transmisora

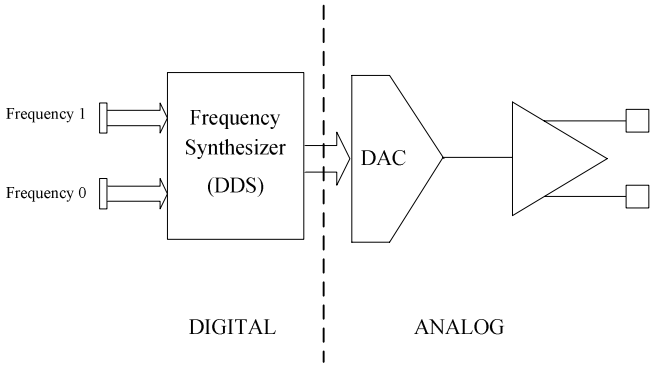


Figura: Esquema de la parte transmisora

Como puede observarse en la figura anterior el canal transmisor parte de las frecuencias y mediante un DDS genera la señal que mediante un convertidor digital-analógico se convierte en la señal analógica deseada, como puede verse también existe un amplificador de potencia de clase AB para una posible conexión directa del transformador a la línea.

Se incluye una capacidad de power-down para que en estado de stand-by se consuma muy poca potencia (casi despreciable).

Veamos ahora un esquema de la parte receptora en la que podremos observar que los canales de recepción emplean un filtro SC sintonizable de ganancia programable, un modulador Sigma-Delta de segundo orden con factor de sobremuestreo programable, realizando el filtrado los filtros SC a un cuarto de la frecuencia de muestreo.

El diezmado se realiza mediante un filtro diezmador SINC3 y una cascada de filtros diezmadores FIR de media banda.

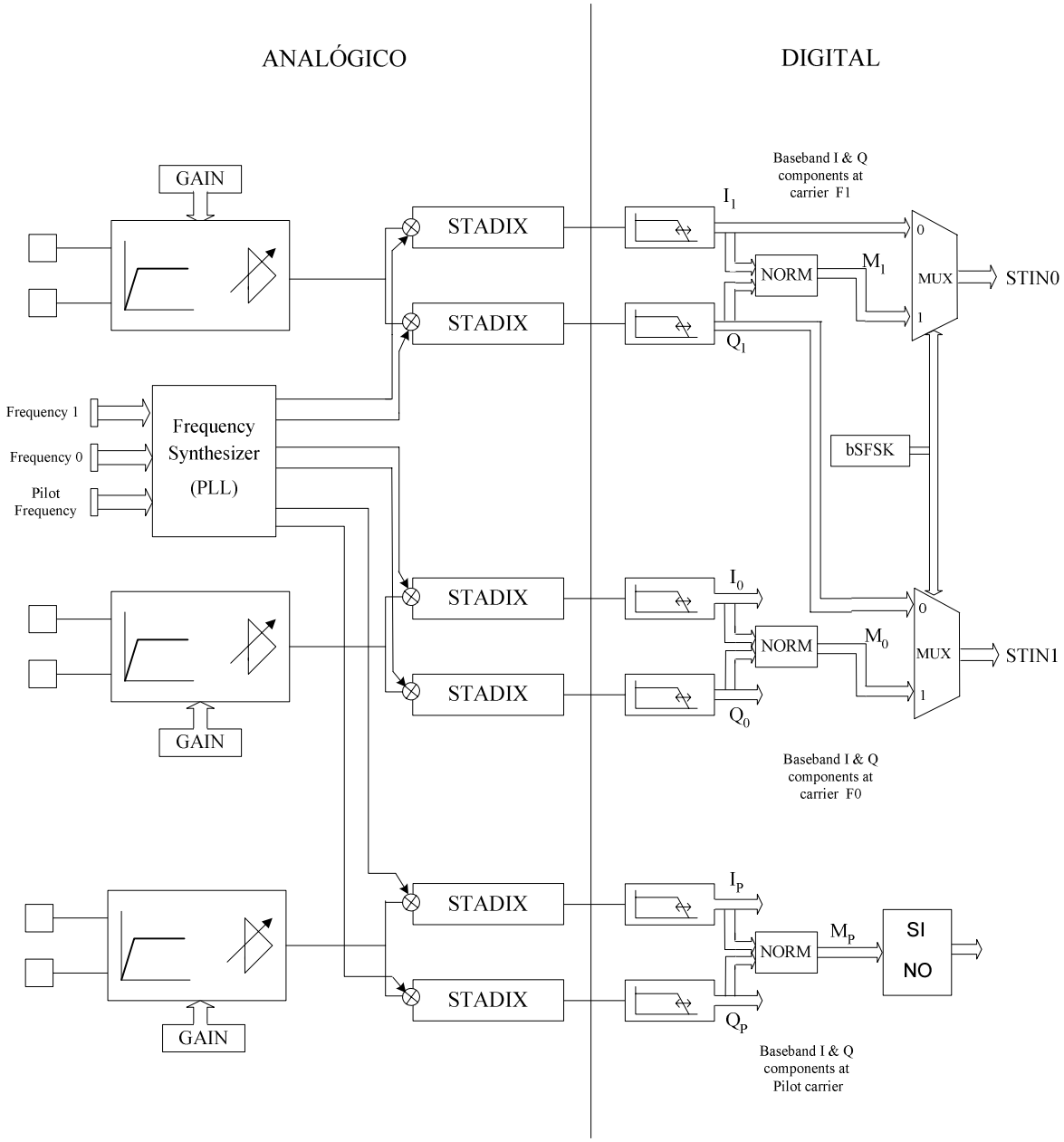


Figura: Esquema de la parte receptora

Una vez que ya se han mostrado los esquemas tanto de la parte transmisora como de la receptora pasaremos a comentar brevemente el interfaz digital que nos permite la monitorización de la celda comentada. A continuación se muestra un esquema de dicho interfaz

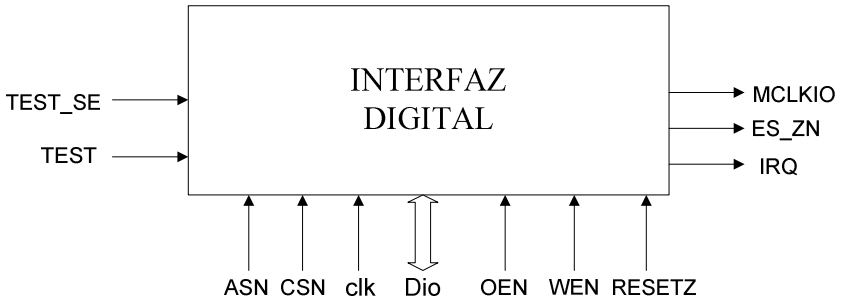


Figura: Esquema del interfaz digital de la celda

Para poder conocer un poco el significado de cada una de las señales se muestra a continuación una tabla con el nombre y una breve descripción de cada una de ellas

Nombre	***Dirección***	***Descripción***
CLK	Entrada	Reloj de sistema, activo en flanco de subida
RESETZ	Entrada	Reset asíncrono del sistema
TEST	Entrada	Señal para activar modo test, activa a nivel alto
TEST_SE	Entrada	Test escaneable
MCLKIO	Salida	Señal de sincronismo de canal 0 (para test)
ASN	Entrada	Address Select Enable
CSN	Entrada	Chip- Select
WEN	Entrada	Write Enable
OEN	Entrada	Output Enable
IRQ	Salida	Interrupción
DIO	Ent / Salida	Bus de entrada/salida (para pines tri-estado)
ES_ZN	Salida	Control de pines tri-estado (pad es salida a nivel bajo)

Tabla: Descripción de las señales del interfaz digital

Este interfaz también posee una serie de señales internas las cuales no vamos a estudiar porque no son necesarias para la utilización de la celda a nivel usuario.

Modo de funcionamiento.

En este apartado de describirá brevemente el modo de funcionamiento tanto del bloque transmisor como del bloque receptor.

Bloque Receptor.

La entrada a este bloque es diferencial y se aplica a dos canales (a tres si consideramos también la frecuencia piloto) sintonizado cada uno de ellos a las frecuencias centrales.

La señal de entrada se aplica en primer lugar a un Control Automático de Ganancia, esto nos proporciona una ganancia de -20, 0 ó +20 dB. En la configuración necesaria para ganancia de +20 dB se introduce un Biquad paso de alta con factor de calidad Q= 11.

El filtro tiene una ganancia de 20 dB y la siguiente función de transferencia:
El filtro está formado por un filtro paso de alta de 2º orden seguido de un filtro paso de baja de primer orden, formando el conjunto de ambos un filtro paso de banda con frecuencia central f_c A continuación se muestra un esquema de los filtros que componen la función de transferencia completa del mismo así como un esquema de ésta última.

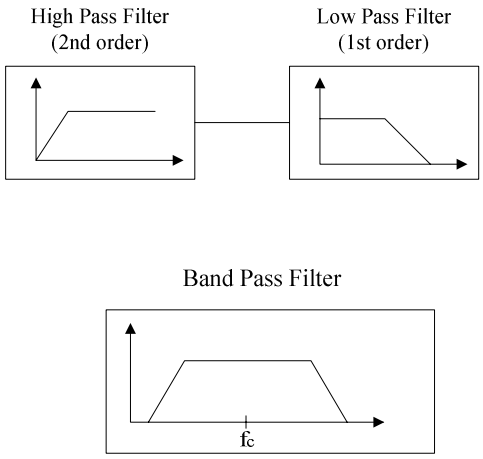


Figura: Función de transferencia del filtro

La frecuencia de muestreo es equivalente a 4 veces el valor de la frecuencia central que se pretende tener, ésta es sintetizada por un PLL que nos proporciona la frecuencia central con una resolución cercana a los 500 Hz.

A la entrada del convertidor Analógico- Digital la señal es multiplicada por un oscilador local en cuadratura a un cuarto de la frecuencia de muestreo (hay recordar que la frecuencia de muestreo es 4 veces la de la frecuencia central del canal)

Esta señal se convierte a una señal en la banda base con componentes en fase (I component) y en cuadratura (Q component). Dos moduladores Sigma-Delta se encargan de convertir la señal en una secuencia de bit la cual es filtrada por un filtro de diezmado.

Finalmente las componentes en fase (I) y en cuadratura (Q) son sumadas en potencia y aplicadas a un demodulador y un extractor del reloj.

Alternativamente sólo un canal puede estar activo y las componentes en fase y en cuadratura ser aplicadas al demodulador . Esta opción se permite para demodulación MSK y FSK.

Bloque transmisor.

La entrada binaria de datos se conduce hacia un Sintetizador Digital de frecuencias generando el tono en una de las dos frecuencias centrales programadas (tres si se considera la frecuencia piloto). La resolución en frecuencia es de 10 Hz.

Un convertidor analógico-digital nos conduce directamente a un amplificador de potencia clase AB capaz de proporcionar hasta 400 mA de corriente de salida con menos del
1 % de distorsión para cargas single-ended con valores comprendidos entre 5 y 100 Ohmios (también es posible con configuración diferencial)

Debido a la elevada eficiencia en potencia de este amplificador, también puede incluirse un amplificador de potencia externo con una disipación de potencia en el chip despreciable.

Encapsulado del chip.

Este módem aparece encapsulado en un chip de 68 pines (PLCC 68) de los cuales sólo 54 están rutados hacia señales internas del módem. El pinout del módem puede apreciarse en la figura que se muestra a continuación

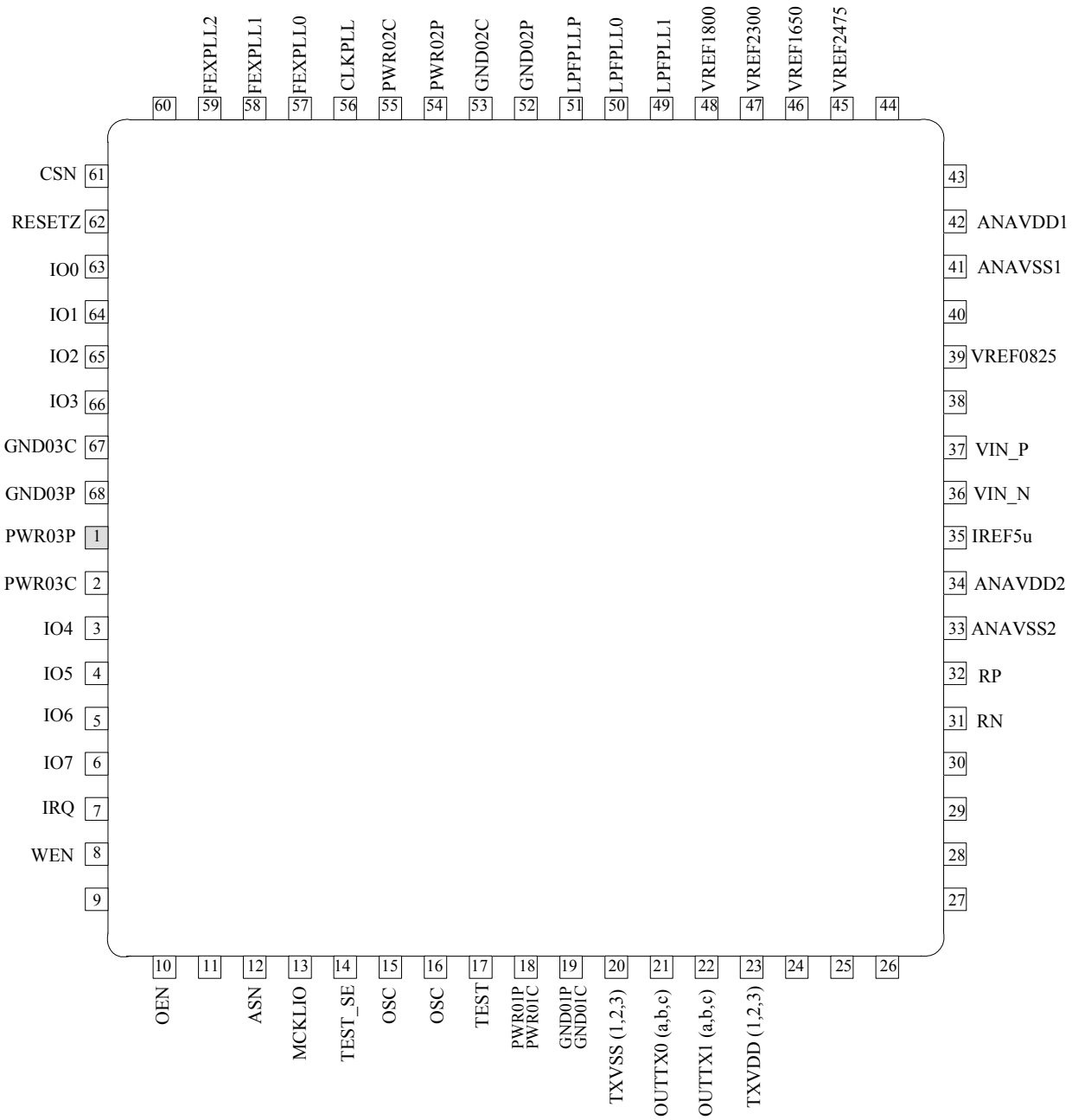


Figura: Esquema del pinout del módem

Para poder identificar los pines del chip con las señales del módem asociadas a cada uno de ellos se muestra a continuación una tabla con dicha correspondencia

Nº Pin	Señal
1	PWR03P
2	PWR03C
3	io4
4	io5
5	io6
6	io7
7	irq
8	wen
9	N.C.
10	oen
11	N.C
12	asn
13	mcklio
14	test_se
15	osc (out)
16	osc (in)
17	test
18	PWR01P, PWR01C
19	GND01P,GND01C
20	TXVSS (1,2,3)
21	OUTTX0 (a,b,c)
22	OUTTX1 (a,b,c)
23	TXVDD (1,2,3)
24	N.C.
25	N.C.
26	N.C.
27	N.C.
28	N.C.
29	N.C.
30	N.C.
31	RN
32	RP
33	ANAVSS2
34	ANAVDD2

Nº Pin	Señal
35	IREF5u
36	VIN_N
37	VIN_P
38	N.C.
39	VREF0825
40	N.C.
41	ANAVSS1
42	ANAVDD1
43	N.C.
44	N.C.
45	VREF_2475
46	VREF_1650
47	VREF_2300
48	VREF_1800
49	LPFPLL1
50	LPFPLL0
51	LPFPLLP
52	GND02P
53	GND02C
54	PWR02P
55	PWR02C
56	CLKPLL
57	FEXPLL0
58	FEXPLL1
59	FEXPLL2
60	N.C.
61	Csn
62	Resetz
63	io0
64	io1
65	io2
66	io3
67	GND03C
68	GND03P

Figura: Tabla de correspondencia entre señales y pines del módem

Señales asociadas al módem.

Para conocer con mayor detalle la naturaleza de las señales asignadas a cada pin del chip, el valor de referencia que debe aplicársele, así como su dirección y su descripción a continuación se muestra una tabla que contiene toda esta información

Nº	***Nombre***	***Dirección***	***Tipo***	***Descripción***
1	PWR03P		Ref. de tensión	Alimentación de Periferia (3.3 V)
2	PWR03C		Ref. de tensión	Alimentación de Core (3.3 V)
3	IO4	Entrada/salida	Señal digital	Pin del Bus de entrada/salida
4	IO5	Entrada/salida	Señal digital	Pin del Bus de entrada/salida
5	IO6	Entrada/salida	Señal digital	Pin del Bus de entrada/salida
6	IO7	Entrada/salida	Señal digital	Pin del Bus de entrada/salida
7	IRQ	Salida	Señal digital	Interrupción
8	WEN	Salida	Señal digital	Write Enable (activo nivel bajo)
10	OEN	Salida	Señal digital	Output Enable (activo nivel bajo)
12	ASN	Entrada	Señal digital	Address select enable
13	MCKLIO	Salida	Señal digital	Señal de sincronismo de canal 0 (para test)
14	TEST_SE	Entrada	Señal digital	Test escaneable
15	OSC	Salida	Señal analógica	Señal de reloj del oscilador (out)
16	OSC	Entrada	Señal analógica	Señal de reloj del oscilador (in)
17	TEST	Entrada	Señal digital	Señal para activar modo test, activa a nivel alto
18	PWR01P PWR01C		Ref. de tension	Alimentación de Periferia y Core (3.3 V)
19	GND01P GND01C		GND	Tierra de Periferia y Core (0 V)
20	TXVSS (1,2,3)		GND	Tierra del transmisor
21	OUTTX0 (a,b,c)		Señal analógica	Salida del transmisor 1 (D/A current Steering)
22	OUTTX1 (a,b,c)		Señal analógica	Salida del transmisor 1 (D/A de librería)
23	TXVDD (1,2,3)		Ref. de tensión	Alimentación del transmisor (3.3 V)
31	RN		Ref. de tensión	Referencia de tensión del D/A (0.6 V)
32	RP		Ref. de tensión	Referencia de tensión del D/A (2.5 V)
33	ANAVSS2		GND	Tierra del TX/RX (0 V)

34	ANAVDD2		Ref. de tensión	Referencia de tension del TX/RX (3.3V)
35	IREF5u		Ref. de corriente	Fuente de corriente (5µA)
36	VIN_N		Señal analógica	Entrada Negativa del RX
37	VIN_P		Señal analógica	Entrada positive del RX
39	VREF0825		Ref. de tensión	Referencia del A/D S-D (0.825 V)
41	ANAVSS1		GND	Tierra del TX/X
42	ANAVDD1		Ref. de tensión	Referencia de tensión del TX/RX (3.3V)
45	VREF2475		Ref. de tensión	Referencia del A/D S-D (2.475 V)
46	VREF1650		Ref. de tensión	Referencia del A/D S-D (1.650 V)
47	VREF2300		Ref. de tensión	Referencia del A/D S-D (2.3 V)
48	VREF1800		Ref. de tensión	Referencia del A/D S-D (1.8 V)
49	LPFPLL1			Conexión para el filtro del PLL1
50	LPFPLL0			Conexión para el filtro del PLL0
51	LPFPLLP			Piloto
52	GND02P		GND	Tierra de Periferia
53	GND02C		GND	Tierra del Core
54	PWR02P		Ref. de tensión	Alimentación de Periferia
55	PWR02C		Ref. de tensión	Alimentación del Core
56	CLKPLL			Referencia externa PLL
57	FEXPLL0			PLL0 externo (opcional)
58	FEXPLL1			PLL1 externo (opcional)
59	FEXPLL2			PLL2 externo (opcional)
61	CSN	Entrada	Señal digital	Chip Select (activo a nivel bajo)
62	RESETZ	Entrada	Señal digital	Reset asíncrono del sistema
63	IO0	Entrada/salida	Señal digital	Pin del Bus de entrada/salida
64	IO1	Entrada/salida	Señal digital	Pin del Bus de entrada/salida
65	IO2	Entrada/salida	Señal digital	Pin del Bus de entrad/salida
66	IO3	Entrada/salida	Señal digital	Pin del Bus de entrada/salida
67	GND03C		GND	Tierra del Core
68	GND03P		GND	Tierra de Periferia

Tabla: Descripción de las señales asociadas al módem

Registros internos del módem.

En capítulos anteriores ya se ha hablado de algunos de los registros del módem, pero sólo se hizo referencia a aquellos registros que estaban directamente relacionados con la configuración del mismo. En este apartado se describirán todos los registros del módem, indicando su dirección, tamaño, nombre, forma de acceso, y en algunos casos su detalle a nivel de bit. Esto nos permitirá tener un mayor control de la programabilidad del módem tanto en su configuración como en su monitorización (debido a que nos proporciona con total detalle el estado en el que se encuentra tanto el proceso de transmisión como el de recepción).

Para empezar la descripción de los registros internos del módem, vamos a mostrar en primer lugar una tabla en la que se nos indica la dirección que ocupa cada unos de los registros así como su nombre, su tamaño, su forma de acceso y una breve descripción.
NOTA: Los nombres de los registros se han mantenido con la nomenclatura original. La descripción de los mismos también se ha dejado en inglés puesto que prácticamente se trata de explicar el significado de las iniciales de las que consta el nombre del registro.

Dir. (hex.)	***Dir. (dec.)***	***Nombre***	***Descripción***	***Acceso***	***Tam. (bits)***	***Estado de reset***
00	0	IDR0	Receive Buffer Register for channel 0	Lectura	8	
01	1	IDR1	Receive Buffer Register for channel 1	Lectura	8	
02-03	2-3	IIR	Interrupt Indentification Register	Lectura	9	
04	4	IER	Interrupt Enable Register	Lect/escrit	5	
05-06	5-6	M0alterna	Base Band Signal Module received in channel 0 (suitable signal level)	Lectura	12	
07-08	7-8	M1alterna	Base Band Signal Module received in channel 1 (suitable signal level)	Lectura	12	
09	9	C0	PLL error in reception channel 0	Lectura	8	
0A	10	C1	PLL error in reception channel 1	Lectura	8	
0B-0C	11-12	SLR0	Line Status Register for reception channel number 0	Lectura	10	
0D-0E	13-14	SLR1	Line Status Register for	Lectura	10	

			reception channel number 1			
0F-11	15-17	AcrB0	Noise-Signal Estimation in reception channel number 0	Lectura	17	
12-14	18-20	AcrB1	Noise-Signal Estimation in reception channel number 0	Lectura	17	
15-17	21-23	AcrA0	Noise-Signal Estimation in reception channel number 1	Lectura	17	
18-1A	24-26	AcrA1	Noise-Signal Estimation in reception channel number 1	Lectura	17	
1B-1C	27-28	Nb_regmd0	DC value detected in the base band by Stadix	Lectura	14	
1D-1E	29-30	Nb_regmd1	DC value detected in the base band by Stadix	Lectura	14	
1F	31	Desp0	Shifts performed to transform the base band signal in a suitable signal. Channel 0	Lectura	8	
20	32	Desp1	Shifts performed to transform the base band signal in a suitable signal. Channel 0	Lectura	8	
21-22	33-34	VCO0	VCO counter used in PLL for channel 0 processing.	Lectura	15	
23-24	35-36	VCO1	VCO counter used in PLL for channel 1 processing	Lectura	15	
25-26	37-38	sbmCCR	Communication Control Register	Lect/escrit	10	
27-28	39-40	SbmBDT	Bit Detection Threshold	Lect/escrit	15	
29-2A	41-42	SbmBPT	Base-Band Power Threshold	Lect/escrit	16	
2B	43	SbmTHR	Transmisión Holding Register	Lect/escrit	8	
2C-2D	44-45	M0	Stadix module output. Channel 0	Lectura	13	
2E-2F	46-47	M1	Stadix module output. Channel 1	Lectura	13	
30	48	BFF		Lectura	8	
31-32	49-50	SLR01		Lectura	11	
33	51	Mreg	Receptor Decimator Register	Lect/escrit	8	00110011
34	52	TXFA	Transmitter frequency channel 0	Lect/escrit	8	00110000
35	53	TXFB	Transmitter frequency channel 1	Lect/escrit	8	01010000
36	54	TXFP	Transmitter pilot frequency	Lect/escrit	8	01000000
37	55	Txpcfg	Transmitter pilot configuration	Lect/escrit	5	11110
38	56	PLL0div	PLL0 divider bits 8 downto 1	Lect/escrit	8	00010100
39	57	PLL0cfg	PLL0 LSB and configuration register	Lect/escrit	3	100
3A	58	PLL1div	PLL1 divider bits 8 downto 1	Lect/escrit	8	00010100
3B	59	PLL1cfg	PLL1 LSB and configuration register	Lect/escrit	3	100
3C	60	PLLpdiv	PLLp divider bits 8 downto 1	Lect/escrit	8	00010100
3D	61	PLLpcfg	PLLp LSB and configuration	Lect/escrit	3	100

			register			
3E	62	Attreg	Attenuation Register	Lect/escrit	6	000000
3F	63	Pwrdreg	Powerdown Register	Lect/escrit	6	000000
40-63	64-99	GPIO	Por definir	Lect/escrit	8	00000000
64-65	100-101	PPWR	Pilot Reception power	Lectura	14	
66-67	102-103	SINC0	Output of CHOi SincM filter	Lectura	13	
68-69	104-105	FIR0	Output of CH0i first FIR filter	Lectura	13	
6A-6B	106-107	FIR1	Output of CH0i second FIR filter	Lectura	13	
6C	108	SDREG	Output of CH0 and CH1 SD modulators	Lectura	6	
6D	109	PSDREG	Output of pilot SD modulator	Lectura	3	
6E	110	PLLsout	Output of the three PLLs	Lectura	3	
80	128	DACTEST	Test configuration register for DACs	Lect/escrit	8	00010000
81	129	TXDACTEST	8 most significant bits for test mode	Lect/escrit	8	00000000
82	130	PDACTEST	8 most significant bits for test mode	Lect/escrit	8	00000000

Tabla: Registros internos del módem

Existen ciertos registros que debidos a su importancia en la configuración y el funcionamiento del módem deben estudiarse de manera más exhaustiva. A continuación detallaremos estos registros.

Interrupt Enable Register (IER)

Es el ***registro de habilitación de las interrupciones*** del módem, poniendo a 1 los bits correspondientes nos permite habilitar los 5 tipos de interrupciones que posee el chip. Veamos a continuación una tabla en la que se explica el significado de cada uno de los 5 bits que posee.

Bit	Identificación	Descripción
0	ERDI_0	Enable Received Data in Channel 0 Interrupt
1	ETHRE	Enable Transmitter Holding Register Empty
2	ELSI_0	Enable Receiver Line Status Channel 0 Interrupt
3	ERDI_1	Enable Received Data in Channel 1 Interrupt

4	ELSR_1	Enable Receiver Line Status Channel 1 Interrupt

Tabla: Interrupt Enable Register (IER)

Interrupt Identification Register (IIR)

Es el ***registro de identificación de interrupciones*** y nos permite, leyendo su contenido, conocer cual es la causa de ha provocado la interrupción en el módem. Este registro nos informa del estado de las interrupciones.
Veamos a continuación el significado de sus 9 bits

Bit	***Interrupt Type***	***Interrupt Source***	***Reset Control***
0..2	Bytes in FIFO of channel 1	New byte available in channel 0	
3..5	Bytes in FIFO of channel 0	New byte available in channel 1	
6	Transmitter empty	Transmitter is ready to accept one byte more	
7	Channel 0 Line Status	Reception status for channel 0 has changed. (See LSR)	
8	Channel 1 Line Status	Reception status for channel 1 has changed. (See LSR)	

Tabla: Interrupt Identification Register (IIR)

Communication Control Register (CCR)

Es el ***registro de control de la comunicación,*** y en él se especifican todos los parámetros necesarios para la configuración tanto del transmisor como del receptor. Veamos a continuación cuales son estos parámetros y cómo se encuentran ubicados en dicho registro

Bit	**Identification**	**Description**
0	RXEN_0	Enable channel 0 reception
1	RXEN_1	Enable channel 1 reception
2	TXEN	Enable Transmitter
3	SFSK/ FSK	Modulation (0 = FSK)

4..6	RXBR	Reception Baud Rate
7..9	TXBR	Transmisión Baud Rate

Tabla: Communication Control Register (CCR)

Como puede observarse en la tabla anterior, este registro nos ofrece varias posibilidades de configuración:

- Habilitar la recepción por el canal 0, canal 1 o por ambos
- Habilitar el transmisor
- Configurar la tasa de bits en transmisión y en recepción a 300, 600, 1200, 2400,
  4800 y 9600 bps.

La forma de codificar la tasa de bit deseada se muestra en la siguiente tabla

Tasa de Bit (bps)	**Codificación Hexadecimal**
300	0
600	1
1200	2
2400	3
4800	4
9600	5

Tabla: Codificación de la tasa de bit

Reception Base-Band Power Threshold for channel 0 and 1 (BPT)

Nivel de potencia de recepción para canal 0 y canal 1. En este registro se especifica el valor umbral que se utilizará en el receptor para comparar con la potencia de la señal que le está llegando. Si la potencia que le llega alcanza el valor umbral entonces el receptor ha detectado suficiente potencia para comenzar el proceso de detección de cabecera. Si no alcanza este valor umbral el receptor rechazará todo lo que escuche de la línea.

Reception Bit Detection Threshold (BDT)

Nivel de potencia de decisión para recepción de bit. El valor de este registro ajusta el umbral de decisión utilizado para distinguir entre el símbolo "0" y el "1" (se especifica en complemento 2)

Line Status Registers (LSR_0 and LSR_1)

Registros del estado de la transmisión. Estos registros nos proporcionan información acerca del estado en el que se encuentra la transferencia de información. Existen 2 registros, uno para el canal 0 y otro para el canal 1.

Bit	*Identif*	*Descripción*
0	ROE	Reception Over-run Error has occurred in channel 0
1	RBF	Reception Buffer Full
2	RBE	Reception Buffer for channel 0 Empty
3	OPD	Over-Threshold Power Detected in channel 0
4	CHD	Correct Header Detected in channel 0
5..7	RSI	Reception Status Information (channel 0)
8..9	Reserved	

Tabla: Line Status Registers (LSR_0, LSR_1)

A continuación se verá con detalle el significado de cada uno de los bits de estos registros

Bit 0: Un error "over-run" ocurre cuando un nuevo byte ha sido demodulado y no hay espacio libre en la FIFO de recepción para almacenarlo.

Bit 1: El buffer de recepción para el canal 0 está lleno

Bit 2: No hay dato disponible en el FIFO de recepción (para el canal 0)

Bit 3: El receptor ha detectado suficiente potencia en la banda base para comenzar la detección de símbolo en el canal 0.

Bit 4: El receptor ha detectado una cabecera correcta.

Bits 5..7: Nos informa sobre el estado del receptor. La codificación de esta información se muestra en la siguiente tabla.

Hex code.	***State***
0	INITIAL
1	IDLE
2	LISTENING
3	SEARCHING HEADER
4	RECEIVING

Tabla: Codificación del estado del receptor

Receptor Decimator Register (Mreq)

Este registro nos configura el diezmado de los filtros digitales del receptor. Veamos en la siguiente tabla el significado de sus bits.

Bit	***Ident.***	***Description***
2..0	M0	Value of M for the channel 0 filter 2 ^(M0+1)
3	HBIT0	Enable final HB filter for channel 0
6..4	M1	Value of M for the channel 1 filter 2 ^(M1+1)
7	HBIT1	Enable final HB filter for channel 1

Tabla: Receptor Decimator Register

Transmitter Frequency Registers: TXFA, TXFB and TXFP

Estos registros nos permiten configurar las frecuencias del transmisor para el canal 0 (*TXFA*), para el canal 1 (*TXFB*) y la señal piloto (*TXFP*). Veamos en la tabla siguiente la forma de configurar las distintas frecuencias.

Binary code	***Frequency (KHz)***
`d0	Reset
`d1	20
`d2	20.5
`d251	145
>`d251	Reserved

Tabla: Codificación de las frecuencias

Las frecuencias son programables en múltiplos de 500 Hz por tanto todos los valores comprendidos entre *´d2* y *´d250* nos programarán una frecuencia entre 21 y 144.5 KHz.

Transmitter Pilot Configuration (TXPCFG)

Este registro nos permite configurar la generación del transmisor piloto. Veamos a continuación una tabla donde se detalla el significado de sus bits

Bit	***Ident.***	***Description***
0	No pilot	Enable the pilot generation, `1´ enables carrier
1	dBcarrier_1	
2	dBcarrier_2	
3	dBcarrier_3	
4	dBcarrier_4	

Tabla. Transmitter Pilot Configuration (TXPcfg)

PLL divider registers (PLL0div, PLL1div y PLLpdiv)

Estos registros nos proporcionan los bits más significativos (desde el bit 8 al bit 1) del registro divisor del PLL en cada uno de los tres canales de entrada (canal 0, canal 1 y piloto).

PLL configuration registers (PLL0cfg, PLL1cfg y PLLpcfg)

Estos registros nos proporcionan los bits menos significativos del divisor PLL y dos bits de configuración. Veamos a continuación el significado de sus bits.

Bit	***Ident.***	***Description***
0	DIV LSB	Bit 0 if he divider
1	FU	Divide by 2 on oscillator output
2	FSEL	External PLL enable

Tabla: PLL configuration registers

El último bit de este registro nos permitirá seleccionar que la frecuencia externa de los PLL´s nos la proporcione los filtros externos de los propios PLL´s o bien se le introduzca desde los pines correspondientes (*FEXPLL0,1,2*)

Attenuation Control Register (Attreg)

Este registro nos permite programar el control de ganancia de los filtros de entrada. En la siguiente tabla se muestra el significado de sus bits.

Bit	***Ident.***	***Description***
0	Att20dB_1	Channel 0
1	Att20dB_2	Channel 1
2	Att20dB_p	Pilot

3	Gan20dB_1	Channel 0
4	Gan20dB_2	Channel 1
5	Gan20dB_p	Pilot

Tabla: Attenuation Control Register (Attreg)

Powerdown Control Register (Pwrdreg)

Este registro nos permite configurar el valor de power-down de los filtros de entrada, de los PLLs y de los moduladores sigma-delta. En la siguiente tabla se muestra el significado de cada uno de sus bits.

Bit	***Ident.***	***Description***
0	PTF_1	Channel 0 (filter)
1	PTF_2	Channel 1 (filter)
2	PTF_P	Pilot (filter)
3	Pdown_1	Channel 0 (Sigma-delta modulator)
4	Pdown_2	Channel 1 (Sigma-delta modulator)
5	Pdown_P	Pilot (Sigma-delta modulator)
6	PLLpd1	PLL group 1
7	PLLpd2	PLL group 2

Tabla: Powerdown Control Register (Pwrdreg)

Pilot Reception Power (PPWR)

Este registro nos proporciona el valor de potencia que está siendo recibida actualmente a la frecuencia piloto.

Registros de observación (SINC0, FIR0 y FIR1)

Los registros *SINC0, FIR0 y FIR1* nos permiten el acceso (podemos observarlo) a los valores intermedios de la estructura pipeline de los filtros digitales. Los valores están dados en complemento a 2.

Sigma-Delta observation registers (SDREG y PSDREG)

El registro *SDREG* nos proporciona monitorización permanente de los moduladores Sigma-Delta para los canales 0 y 1, mientras que el registro *PSDREG* nos proporciona la misma información para el canal piloto. En la siguiente tabla se muestra el contenido de los bits del registro *SDREG,* los del registro *PSDREG* son análogos.

Bit	***Ident.***	***Description***
0	asd1clk	Trigger for SD modulators on CH1
1	asd1q	Output of Q channel modulator on CH1
2	asd1i	Output of I channel modulator on CH1
3	asd0clk	Trigger for SD modulators on CH0
4	asd0q	Output of Q channel modulator on CH0
5	asd0i	Output of I channel modulator on CH0

Tabla: Contenido del registro SDREG

PLL Observation Register (PLLsout)

Este registro nos proporciona permanente monitorización de los PLLs del chip. Veamos a continuación una tabla donde se detalla el contenido de sus bits.

Bit	***Ident.***	***Description***

0	PLLP	Pilot channel oscillator
1	PLL1	CH1 channel oscillator
2	PLL0	CH0 channel oscillator

Tabla: PLL Observation Register (PLLsout)

DAC Test registers DACs: DACTEST, TXDACTEST, PDACTEST

El registro DACTEST nos proporciona la posibilidad de colocar los DACs (Digital-Analog Converters) en modo test. También nos proporciona los 2 bits menos significativos pasados a los DACs en modo test. Veamos a continuación una tabla en la que se detalla el contenido de los bits del registro DACTEST.

Bit	***Ident.***	***Description***
0-1	TXDAC_L	The bits 1 and 0 for the transmitter DAC test
2-3	PDAC_L	The bits 1 and 0 for the pilot DAC test
4	Enable TX 0	High enables tx0 transmitter
5	Enable TX 1	High enables tx1 transmitter
6	TXDACTEST	Test enable for transmitter DAC
7	PDACTEST	Test enable for pilot DAC

Figura : Registro DACTEST

Los registros *TXDACTEST* y *PDACTEST* nos proporcionan los 8 bits más significativos en el modo test del convertidor Digital-analógico. (DAC)

Funcionamiento aproximado.

En este apartado se describirá brevemente la secuencia de estados tanto de una operación de transmisión como de recepción.

Comencemos viendo como opera el ***receptor*** **:**

ESTADO 0 -> INITIAL

El estado INITIAL está provocado por un reset en el dispositivo. Los valores de configuración (tasa de bit, frecuencias, etc) no son válidas porque el usuario no ha programado todavía estos valores.
Si el usuario intenta comenzar la recepción (RXEN='1'), la respuesta del dispositivo puede ser impredecible.

ESTADO 1 -> IDLE

En este estado el receptor está configurado pero no está habilitado. Está preparado para comenzar la detección de potencia.

ESTADO 2 -> RECEIVER ACTIVE AND DETECTING BASE-BAND POWER

El receptor está escuchando de la línea y buscando la potencia de la señal que alcance el umbral programado en los registros de control. El módem permanecerá en este estado hasta que detecte esta potencia.

ESTADO 3 -> ACTIVE AND DETECTING HEADER

El receptor ha detectado suficiente potencia en la banda base y comienza el proceso de detección de la cabecera.

ESTADO 4 -> ACTIVE AND RECEIVING DATA

El receptor ha detectado una cabecera válida y comienza a recibir los datos de la línea de potencia. Estos datos serán almacenados en un registro de desplazamiento interno hasta que se reciba un byte completo, una vez que esto ocurra este byte será almacenado en el buffer de recepción y esperará a una orden de lectura por parte del usuario.

Veamos ahora el funcionamiento del ***transmisor*** :

ESTADO 0 -> INITIAL

Estado inicial del dispositivo tras un reset.

ESTADO 1 -> REST

El transmisor está deshabilitado para la transmisión. El buffer debe ser reseteado inicialmente. El usuario será capaz de cargar el buffer de transmisión para transmisiones futuras.

ESTADO 2 -> ACTIVE AND TRANSMITTING

Si el usuario habilita la transmisión con TXEN =’1’, el dispositivo comienza enviando bits de cabecera y continúa enviando datos del usuario. Los datos de usuario deben ser cargados antes de la habilitación de la transmisión para ser enviados después de la cabecera.

La transmisión de la cabecera puede comenzar estando el buffer de transmisión vacío. El usuario puede utilizar el bit de estado para monitorizar el buffer de transmisión, según sus necesidades.

8.BIBLIOGRAFIA

Libros de consulta:

- PIC16F87X Data Sheet de Microchip
- Labview User Manual de National instruments
- Mplab Manual de Microchip
- Proyecto fin de Carrera : PLACA DE ADQUISICIÓN DE DATOS BASADA EN EL MICROCONTROLADOR PIC16F877 de Esteban Carlos Crespo Díaz
- Documentación del MODEM de potencia
- Proyecto fin de Carrera : **Diseño de una interfaz hardware para módem por línea de potencia** de **Tomás R. Valdecantos Villa**
- Data Sheets: LM317 ,LM334,LM6132

Paginas de consulta:

http://www.microchip.com/1010/index.htm

http://www.ic-prog.net/

http://www.national.com/