ESCOLA TÈCNICA SUPERIOR ENGINYERIA
UNIVERSITAT ROVIRA I VIRGILI
Departament d'Enginyeria Electrònica Elèctrica i Automàtica

Control de módem GSM desde microcontrolador.

TITULACIÓN: Ingeniería Técnica Industrial en Electrónica Industrial

AUTOR: Christian Paniagua Martín
DIRECTOR: José Luís Ramirez Falo

FECHA: Junio del 2008.

Índice

Índice figuras

Índice tablas

1 Memoria Descriptiva.

1.1 Objeto del proyecto.

El objeto del presente proyecto es diseñar y montar un circuito que controle un MODEM GSM con el fin de enviar mensajes cortos (SMS) a un terminal GSM externo, así como recibir mensajes SMS y actuar según proceda. El circuito se basará en un sistema controlado por un microcontrolador de la serie PIC, que gestionará un módem GSM y las comunicaciones necesarias con el PC. El prototipo incluirá dos LEDs para mostrar al usuario el estado de las comunicaciones con el PC y con el MODEM. Para dar un uso alternativo, se programará el microcontrolador para controlar sus múltiples E/S, activándolas o desactivándolas mediante un SMS enviado desde cualquier teléfono móvil en cualquier parte del mundo.

1.2 Antecedentes.

Existen aplicaciones que requieren comunicarse con un terminal remoto para visualizar y controlar el sistema. Un ejemplo de aplicación podría ser un dispositivo conectado a un sistema automático cualquiera, que avisara al usuario de una incidencia grave y le permitiera a éste tomar una decisión y actuar a distancia.

Dado que, según un estudio del Ministerio de Industria de 2006, el 98% del territorio poblado tiene cobertura GSM, resulta interesante aprovechar la red de telefonía móvil para nuestras comunicaciones.

1.3 Posibles soluciones y solución adoptada.

1.3.1 Elección del Microcontrolador.

Los microcontroladores son circuitos integrados que incluyen como mínimo una CPU, unidades E/S y memoria de programa, además de una serie de componentes que dotan al microcontrolador de diversas funcionalidades.

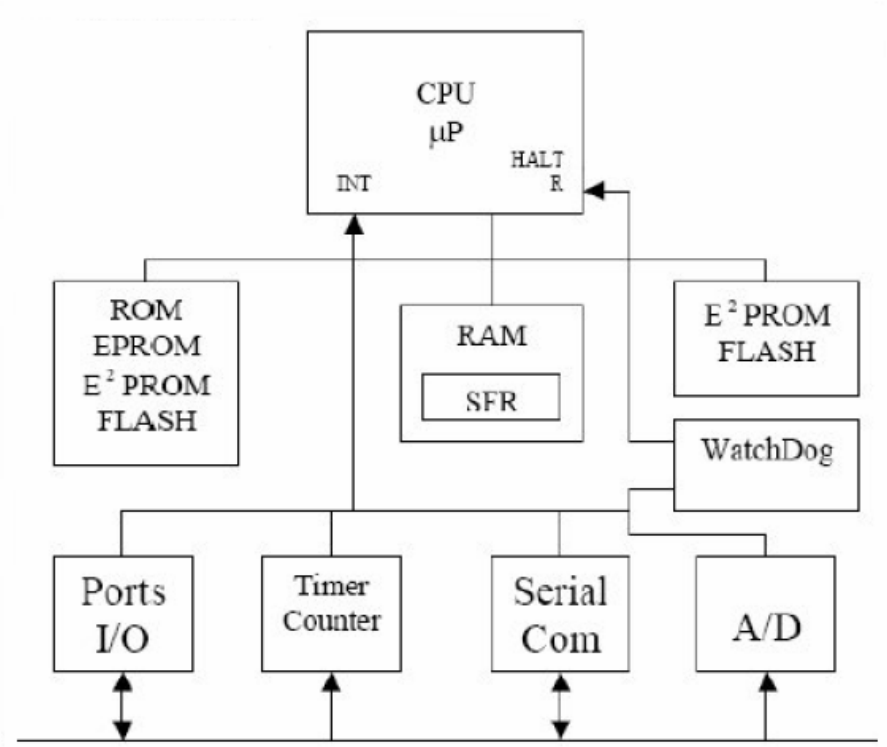


Figura 1.1 Ejemplo de esquema interno de un microcontrolador.

De los microcontroladores existentes, es con los PIC de MICROCHIP con los que estoy más familiarizado. Por lo tanto, siendo los PIC unos microcontroladores suficientes para la tarea a llevar a cabo y poseyendo varias unidades, se decide hacer uso de ellos.

Por requisitos de diseño, se necesita un microcontrolador con conectividad USB y un puerto de E/S, por lo que con el **PIC18F2550**, un microcontrolador de 28 patas satisfaceremos nuestras expectativas.

Figura 1.2 Pic 18F2550.

Las funciones y características que han sido utilizadas son las siguientes:

- Memoria FLASH =32 KBytes, 16384 Instrucciones
- EEPROM =256 Bytes
- PINS I/O =24 con 3 interrupciones externas
- USART
- Timers 8/16bits =1/3, 4 timers en total
- Modos de oscilador:
  - Externo hasta 48MHz
  - Externo auxiliar, para módulo USB
  - Interno con RC

  - Interno programable de 31kHz hasta 8MHz
- Rango de voltaje desde 2V hasta 5.5V.
- USB V2.0: Baja Velocidad (1.5 Mb/s) y Velocidad Completa (12 Mb/s).
- Interrupciones: 3 interrupciones de entradas externas, interrupción de cambio de estado en el portB, interrupciones de los Timers, interrupción de la UART al enviar y al recibir, interrupción de final de conversión A/D, interrupción del USB, etc...

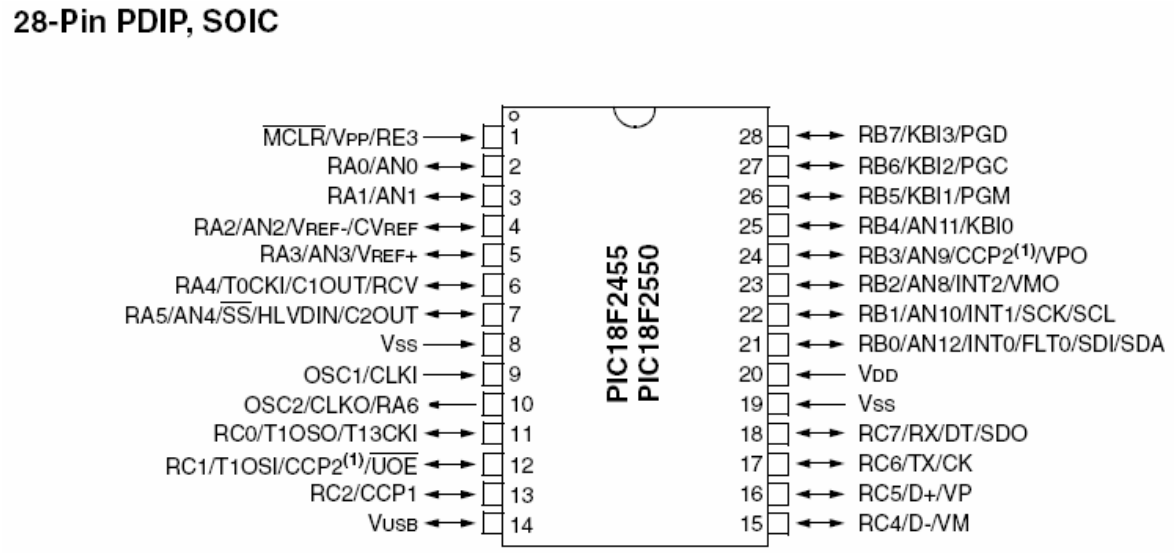


Figura 1.3 Pins del microcontrolador PIC 18F2550.

Además de las funcionalidades de que dispone este PIC, una ventaja sobre la serie 16F es el juego de instrucciones y el sistema de memoria lineal.

La familia 18F dispone de 77 instrucciones sobre las 35 de la familia 16F. El hecho que más puede facilitar la programación es la estructura de la memoria interna, tanto la memoria de datos como la de programa. Los PIC de las familias inferiores a la 18F sólo pueden acceder a direcciones de 8 bits.

Los PIC 18F disponen de memoria lineal y pueden direccionarla toda, esto resulta cómodo para manejar buffers lineales. En realidad esto no nos perjudica ni nos beneficia, puesto que programaremos el firmware del PIC en lenguaje C.

1.3.2 Módem GSM.

El módem GSM es la parte que se comunicará con el mundo exterior. En la elección del módem, pese a ser lo más importante, no ha hecho falta tener en cuenta muchos parámetros, ya que todos los módems GSM cumplen los mismos requisitos mínimos necesarios.

El módem GSM elegido es del fabricante Siemens, modelo TC35i, dotado de interfaz RS-232, conector de antena tipo jack JME, soporte de tarjeta SIM, LED indicador

de estado y alimentación mediante un adaptador 220V AC-12V DC. Se ha elegido este modelo porque ya disponía de él, conozco su funcionamiento y cumple con los requisitos necesarios para el desarrollo de este proyecto

Figura 1.4 Modem GSM Siemens TC35i.

Las características técnicas del módem son las siguientes:

- Banda Dual 900/1800 MHz GSM.
- Comunicaciones serie a 2.4, 4.8, 9.6 y 14.4 kbps.
- GSM Clase 1 y Clase 2 Grupo 3 Fax
- Short Message Services (SMS)
- Zócalo para tarjeta SIM.
- Compatible con los comandos AT.

Se deberá conectar una antena GSM a la salida del módem. En caso de no hacerlo no se garantizaría conectarse a la red con la suficiente calidad.

Figura 1.5 Antena GSM conectada al Módem.

1.3.3 Entorno PC.

Existen diferentes entornos para la interfaz de usuario, LINUX, UNIX, MS-DOS, WINDOWS, etc. De los distintos entornos, es el de Microsoft WINDOWS con el que estoy más familiarizado, además de ser el más popular.

El entorno MS-DOS es un entorno conocido por su robustez y sencillez, pero tiene el inconveniente de ser un entorno poco amigable para el usuario final, además que no nos permitiría realizar comunicaciones por el puerto USB del PC.

Por lo tanto, el entorno elegido es Microsoft Windows, trabajando con la versión XP Service Pack 2, ya que es un entorno conocido por la inmensa mayoría de usuarios y dispone además de un gran abanico de posibilidades de desarrollo del software.

1.3.4 Lenguaje de Programación.

Existen muchos lenguajes de programación de tipo visual (programación orientada a objetos): Visual Basic, Visual C++, Visual C#, Delphi, Java, etc...

Siendo el lenguaje C el más utilizado a lo largo de la carrera y teniendo nulos conocimientos de cualquier lenguaje de programación visual, se decide adoptar el software de programación Microsoft Visual C# por su gran similitud con el lenguaje C, que ya conozco, y por ser de libre distribución en su versión Express.

1.3.5 Interfaz PC-Dispositivo.

Los ordenadores PC disponen de un gran surtido de conexiones con el exterior, son las siguientes:

- Puertos PS/2: usados para conectar teclado y ratón.
- Puerto paralelo: usado para conectar impresoras y escáneres, aunque cada vez se utiliza menos este puerto debido a la gran versatilidad del puerto USB. Ancho de banda de 8 a 16 Mbps.
- Puertos COM: son puertos serie RS-232. Al igual que el paralelo, cada vez se usan menos. Suele predominar su uso en aplicaciones industriales más que en domésticas. Las comunicaciones RS-232 suelen establecerse a: 9600 bps, 19200 bps, 38400 bps, 57600 bps y 115200 bps.
- Puerto IEEE 1394: conocido como FireWire, es un puerto serie de alta velocidad, usado sobretodo en aplicaciones de edición de video. Su ancho de banda es de entre 400 Mbps y 800 Mbps, dependiendo si es la versión 1 o 2 respectivamente.
- Puerto USB: actualmente se usa para conectar teclados, ratones, impresoras, escáneres, teléfonos móviles, adaptadores IrDA y Bluetooth, memorias flash

externas, etc… En su versión 2.0 se dispone de un ancho de banda de 480 Mbps.

- IrDA: es un tipo de conexión inalámbrica que funciona por infrarrojos; es de muy corto alcance, se suele utilizar para comunicar dispositivos móviles como ordenadores de bolsillo o teléfonos celulares con el PC. Su ancho de banda oscila entre 9600 bps y 4 Mbps.

- Bluetooth: es otro tipo de conexión inalámbrica, éste funciona por RF; tiene una cobertura típica de 10 metros alrededor del emisor. Es utilizado para comunicar dispositivos móviles entre ellos y con el PC, teniendo un ancho de banda de entre 1 y 3 Mbits/s.

La solución adoptada es utilizar el puerto USB, porque es un puerto muy extendido, fácil y cómodo de usar. Solo es necesario conectar el dispositivo y el sistema operativo (Windows XP o superior) lo reconocerá. Esto es una buena idea para nuestro lector, ya que dará más facilidad de utilización. Para nuestra aplicación su gran ancho de banda no es necesario, apenas transferiremos unas centenas de Bytes.

1.3.6 Interfaz PC-Modem.

El MODEM GSM elegido tiene integrada una interfaz para puerto serie RS-232, por lo que se implementará, mediante la UART del PIC, una comunicación serie con el MODEM.

Los parámetros elegidos para esta comunicación serán los siguientes:

- 9600 bps.
- 8 bits.
- Sin paridad.
- 1 bit de ‘stop’.

1.3.7 Dispositivo visualizador.

En nuestro circuito se hace necesaria la visualización del estado del PIC, del MODEM y de la comunicación entre PIC y PC. A través de la versión SUPERVISOR del software, se visualizan todas las tramas enviadas y recibidas por el PIC, el MODEM y el PC, además de un flag que indica en qué posición crítica se encuentra la ejecución del programa interno del PIC.

Una vez el circuito está configurado, puede funcionar independientemente del software, por lo que es aquí cuando nos interesa visualizar su correcto funcionamiento.

De las múltiples opciones que existen, las más apropiadas serían las de instalar un display de cristal líquido (LCD) y la de utilizar diodos electroluminiscentes (LED).

Para simplificar el circuito, debido a su mínimo espacio y a las suficientes posibilidades que nos ofrecen, haremos uso de los LED, en su versión de dos colores. Estos LED se conocen como tricolores porque iluminan en verde, rojo y la suma de los dos genera el ámbar.

Utilizaremos dos LED tricolores, VERDE, ROJO y AMBAR, con un encapsulado de 5mm de diámetro.

Figura 1.6 LED 5mm tricolor.

1.3.8 Alimentación.

Teniendo en cuenta que nuestro MODEM se alimentará desde la red eléctrica y que el circuito deberá conectarse al PC en ocasiones, para transferir los datos necesarios para su funcionamiento, se opta por aprovechar la tensión de 5V y hasta 500mA de corriente que suministra el host USB del PC.

Alternativas posibles serían:

- Conexión a la red eléctrica vía transformador, rectificador, filtros, estabilizador, etc.
- Uso de baterías recargables.
- Uso de pilas.

La solución óptima sería la alimentación del dispositivo desde la red eléctrica, pero para demostrar el funcionamiento del USB se decide alimentar desde éste. Por otro lado, de esta manera se ahorra espacio y medios en la regulación de la tensión y en el uso de pilas o baterías.

1.4 Comunicaciones serie.

La conexión a través de esta interfaz es muy importante debido a su gran flexibilidad. En los ordenadores personales la interfaz serie se utiliza para conectar múltiples dispositivos como plotters, módems, ratones, impresoras...

En la transmisión serie se van transfiriendo los bits de información de uno en uno a través de una línea de datos.

En las transferencias asíncronas, tanto el receptor como el transmisor deben trabajar a la misma velocidad. En este caso se envía también información de sincronización a través de la línea de datos, que se corresponde con un bit de comienzo (*bit de star*t), que

indica el comienzo de una unidad de datos, un bit de fin (*bit de stop*) indicando su finalización y, opcionalmente, un bit de paridad para controlar los posibles errores. El bit de paridad lo generan los controladores serie de forma automática, pudiendo configurarse entre las opciones de: sin paridad, paridad par (*odd*) o paridad impar (*even*).

Las tasas de transferencia de datos se miden en baudios. Los baudios indican el número de veces que puede cambiar una señal en la línea de transmisión por segundo. En una interfaz serie, las señales cambian siempre a la misma frecuencia y se realiza una codificación binaria de la información de forma que cuando se quiere enviar un '1' se pone la línea a nivel alto y cuando se quiere enviar un '0' se pone la línea a nivel bajo. En este caso los baudios coinciden con el número de bits por segundo transferidos si se incluyen también los bits de comienzo, de fin y de paridad.

1.4.1 La interfaz RS-232.

Este estándar lo incorporan todos los ordenadores personales, a excepción de algunos ordenadores portátiles, y está definido por la EIA (*Electronic Industries Association*) aunque en Europa se le conoce como el estándar V.24 definido por la CCITT (*Consultative Committee for Internacional Telephone and Telegraph*). En él se definen todas las características mecánicas, eléctricas y la norma de facto necesarios para conectar un equipo terminal de datos (DTE- *Data Terminal Equipment*) con un equipo transmisor de datos (DCE – *Data Carrier Equipment*). Inicialmente se definió para realizar la comunicación entre un ordenador personal y un módem, aunque actualmente se utiliza con muchos otros propósitos para enviar datos de forma serializada.

El estándar define voltajes que oscilan entre +[3-15]V para el nivel alto y –[3-15]V para el nivel bajo.

Si se utiliza este estándar para conectar otros periféricos diferentes de los módems, éstos se comportan como dispositivos DTE y, por lo tanto, las señales cambian de significado. [10]

1.4.2 El Bus Serie Universal (USB).

El USB es un estándar creado en 1995 que define un bus utilizado para conectar periféricos al ordenador. La principal característica que tiene es que la conexión es muy sencilla, ya que utiliza un único conector para conectar a través de un bus serie todos los dispositivos. En él se definen los conectores y los cables, una topología especial tipo estrella para conectar hasta 127 dispositivos y protocolos que permiten la detección y configuración automática de los dispositivos conectados. USB 1.0 soporta dos tasas de transferencia diferentes, una baja de 1,5 Mbps para la conexión de dispositivos lentos de bajo coste (joysticks, ratones) y otra alta de hasta 12 Mbps para la conexión de dispositivos que requieren un mayor ancho de banda (discos o CDROMS).

La especificación de este estándar ha sido respaldada por las empresas líderes mundiales en el campo de la informática: Intel, IBM, DEC, Microsoft, Compaq, NEC y Northem Telecom, empresas que garantizan su continuidad y utilización.

A mediados del año 2000 aparece la versión 2.0, que fue creada por el conjunto de compañías arriba mencionadas, a las cuales se unieron Hewlett Packard, Lucent y Philips. USB 2.0 multiplica la velocidad del bus por un factor de 30 o 40, llegando a alcanzar una velocidad de 480 Mbps, con una diferencia de coste casi inapreciable. Es compatible con la versión anterior y utiliza los mismos cables y conectores, únicamente se necesitan nuevos hubs que soporten la versión 2.0. Estos hubs son algo más complejos que los anteriores, ya que tienen que manejar el tráfico de datos de tres velocidades distintas sin ser excluyentes entre ellas.

Cabe también destacar que USB 2.0 nunca llegará a reemplazar completamente a USB 1.0, ya que existen algunos tipos de dispositivos, como los HID (teclados, ratones,…), que no requieren las altas velocidades que alcanza esta nueva versión y que únicamente encarecerían el dispositivo.

Anteriormente los periféricos se conectaban mapeados directamente en direcciones de E/S, se les asignaba una dirección específica y en algunos casos un canal DMA. Esta situación conducía a tener conflictos en la asignación de estos recursos, puesto que siempre han estado bastante limitados en el ordenador. Además cada dispositivo tenía su propio puerto de conexión y utilizaba sus cables específicos, lo que daba lugar a un incremento de los costes. Debido a que a cada dispositivo se le tenían que asignar unos recursos específicos la detección del mismo debía hacerse a la hora de arrancar el sistema y nunca se podía incorporar un nuevo dispositivo cuando el sistema estaba en marcha.

Los dos aspectos fundamentales que motivaron la realización de este estándar fueron la necesidad de configurar de forma sencilla los periféricos conectados al ordenador y la necesidad de aumentar el número de puertos disponibles.

Este estándar define una topología de conexión en estrella, tal como se muestra en la figura 2.1, por medio de la incorporación de varios concentradores conectados en serie. Cada concentrador se conecta por un lado al ordenador, que contiene una o dos interfaces de este tipo en la placa base, o a otro concentrador y, por otro lado, se conecta a varios dispositivos o incluso a otro concentrador. De este modo pueden existir periféricos que vengan ya preparados con nuevos conectores USB para incorporar nuevos dispositivos, hasta un total de 127, todos ellos funcionando simultáneamente.

Los hubs tienen la misión de ampliar el número de dispositivos que se pueden conectar al bus. Son concentradores cableados que permiten la conexión simultánea de múltiples dispositivos y lo más importante es que se pueden concatenar entre sí ampliando la cantidad de puertos disponibles para los periféricos. El concentrador detecta cuándo un periférico es conectado o desconectado a/de uno de sus puertos, notificándolo de inmediato al controlador de USB. También realiza funciones de acoplamiento de las velocidades de los dispositivos más lentos.

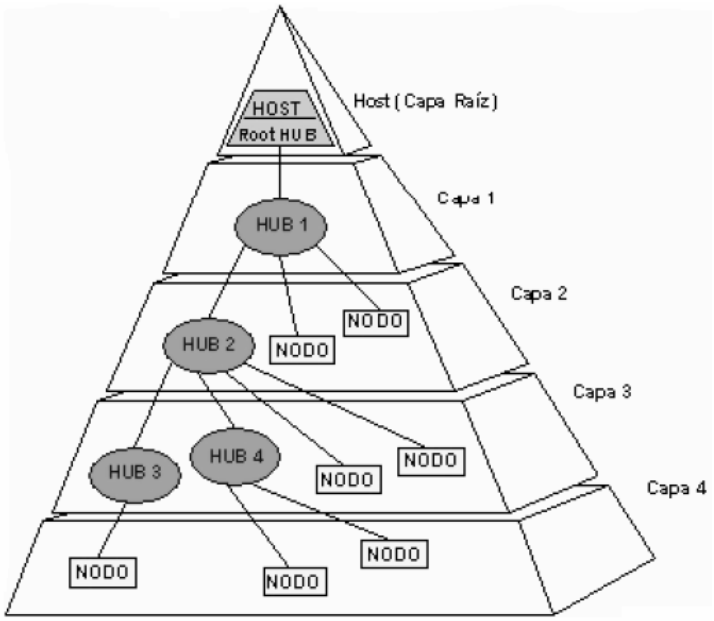


Figura 1.7 Topología de una conexión USB.

Existe una gran variedad de dispositivos USB que se conectan todos al mismo bus. La característica más importante es que todos ellos utilizan el mismo tipo de cable y de conector y se conectan de la misma forma. El host decide qué dispositivo puede acceder al bus, utilizando un protocolo parecido al de paso de testigo. Este protocolo se caracteriza porque entre los diferentes dispositivos se va pasando un identificador a lo largo del tiempo que permite la utilización del bus.

El host USB tiene las funciones de:

- Detectar la conexión/desconexión de dispositivos y configurarlos.
- Controlar las transferencias de datos y de control que tienen lugar en el bus.
- Realización de auditorias sobre la actividad del sistema.
- Servir como fuente de alimentación a los dispositivos.

El USB define dos líneas para transmitir datos y otras dos para transmitir potencia (véase la figura 1.8). Los datos de transmiten de forma balanceada a velocidades entre 1,5 Mbps y 12 Mbps. La señal se transmite codificada en un código autoreloj de no retorno a cero invertido (NRZI) para poder incluir junto con los datos información de sincronización. Las líneas de alimentación (Vbus y GND) evitan la necesidad de utilizar fuentes de alimentación externas. Tiene una tensión de 5 V y la corriente se limita a un máximo de 500 mA, siendo el consumo y la configuración eléctrica totalmente transparente al usuario. La distancia entre dos periféricos conectados al mismo cable no debe ser superior a 5 metros para evitar problemas de caídas de tensión.

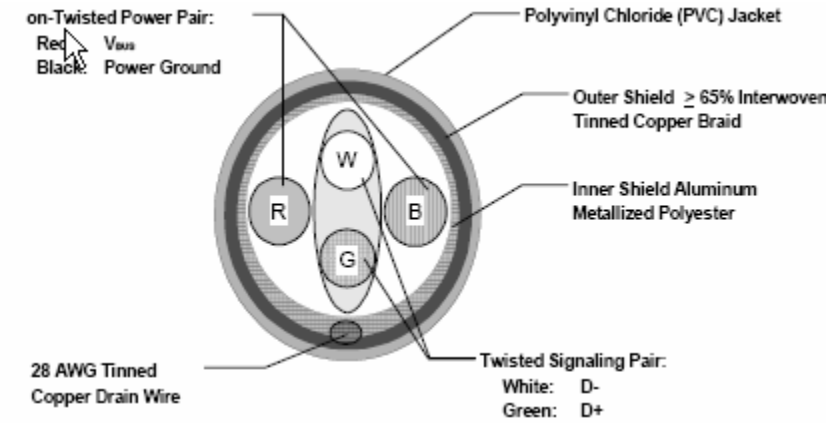


Figura 1.8 Sección del cable USB.

El computador identifica automáticamente el dispositivo que se conecta mientras opera y lo configura sin tener que instalar drivers específicos del fabricante. Al comienzo se detectan los dispositivos conectados midiendo los niveles de voltaje de las líneas. Si un dispositivo está conectado, entonces el dispositivo envía información sobre el tipo o la clase a la que pertenece, qué modo de transferencia utilizará y cuáles son sus necesidades de ancho de banda. El host reconocerá el dispositivo buscando en la lista de drivers del sistema operativo y teniendo en cuenta los demás dispositivos conectados le asignará un ancho de banda determinado. De la misma forma también se pueden desconectar los dispositivos del sistema.

El controlador USB del host asigna un número diferente de dispositivo a cada uno de los periféricos que se conectan a este bus. Para empezar la transferencia, éste envía un paquete que identifica al dispositivo objeto de la transferencia.

El protocolo soporta cuatro tipos de transferencias:

- **Control.** Son transferencias que se utilizan para leer información de los descriptores en los registros de los dispositivos (llamados *endpoints*), interpretarla y poder configurarlos.
- **Interrupción.** Usadas en los periféricos del tipo de los controladores de juegos, teclados y ratones, cuya comunicación es unidireccional y poco frecuente.
- **Masiva.** Son transferencias no periódicas que precisan de todo el ancho de banda disponible. Utilizadas por las impresoras y los scanners.
- **Isócrona.** Dedicadas a las transferencias de telecomunicaciones, como voz o vídeo, que garantiza unas tasas de transferencia constantes. Se caracterizan porque el número de pulsos de reloj que transcurren entre la transmisión de dos caracteres es constante, por lo tanto, se está enviando información constantemente entre el host y el dispositivo. [11, 12]

1.4.2.1 Conceptos básicos.

Para comprender las comunicaciones USB tenemos que conocer los siguientes conceptos básicos:

- *HC Controller (HC)*, que es el encargado de desencadenar todas las comunicaciones.

- *Función* que se define como una utilidad exclusiva de las que dispone un periférico. Así pues, un componente o periférico puede tener una o varias funciones. Por ejemplo un teclado tiene una función para comunicar la tecla pulsada y una diferente para iluminar el led indicando la selección de mayúsculas u otras selecciones.

- *Hub* que es un componente destinado a encadenar puertos de salida para otros componentes que pueden ser más periféricos u otros hubs. Éstos han de tener los componentes necesarios para realizar funciones tales como identificarse, transmitir órdenes o alimentar otros componentes, entre otras. Algunas veces se puede distinguir entre *componente* y *periférico*, así por ejemplo un componente puede estar formado por un periférico y se le pueden incluir funciones de Hub, lo cual a nivel de USB se le otorgarían dos direcciones diferentes.

- *Endpoint* se podría definir como destino. Éste, como dice su propia traducción, es el final y origen de toda comunicación en el bus. Cada dispositivo dispone de un mínimo de un endpoint de control más los que sean necesarios para la propia finalidad de la aplicación.

- Finalmente aclararemos el concepto de *puerto* en este contexto. Hasta ahora relacionábamos el término puerto con una dirección (sentido) específica con su dirección, forma y camino exclusivo para la información. No obstante en el entorno del USB, cada periférico está situado en un puerto exclusivo pero a diferencia del anterior solo tenemos el HC (HC Controller) con una dirección de memoria, lo que implica que todos los periféricos, a pesar de tener su conexión y cables propios, comparten el tiempo y espacio en memoria de comunicación y sólo uno de los periféricos o el HC puede enviar información en cada instante.

1.4.2.2 Proceso de detección y enumeración del dispositivo.

Cuando se conecta un dispositivo al Hub USB, el Host Controller (HC) designa una dirección para el periférico y le reclama información para saber más sobre este sistema.

Todas las transmisiones se realizan en 1 milisegundo, esto quiere decir que en este espacio todos los periféricos tienen un margen asignado por el HC para recibir y enviar información. Durante el proceso de enumeración cada periférico informa al HC del ancho de banda que necesita para comunicaciones con ancho de banda garantizado (como pueden ser equipos de música que necesiten recibir información de forma continuada y de forma

ininterrumpida). Si el ancho de banda que requiere el periférico no está disponible para la saturación del sistema se deniega al periférico el uso del bus. Es entonces cuando el driver puede pedir un ancho de banda inferior o esperar que éste esté disponible para poder funcionar.

Otra responsabilidad del HC consiste en controlar la información para poder determinar si han habido errores en la comunicación. De esta manera puede añadir información para que ésta sea enviada de forma redundante o bien enviar bits para controlar errores. En caso que existan errores, se procede al reenvío de la información. De todas maneras existe la posibilidad de enviar información evitando que ésta se vea sometida a controles de paridad o de errores de forma que se garantice una cantidad de datos transmitidos cuando no es necesario o no se puede perder tiempo en mirar si existen errores.

También es trabajo del HC detectar cuando un periférico tiene problemas en la recepción o transmisión de información, en este caso avisa al driver que a la vez avisa a la aplicación que tendrá que tomar las medidas correspondientes para solucionar el problema.

Finalmente destacaremos como una última función del HC alimentar los periféricos siempre que éstos lo pidan y que cumplan unas condiciones (consumo máximo limitado, posibilidad de entrar en estado de *Suspensión* reduciendo al mínimo el consumo pero con atención al canal de comunicaciones para despertar cuando sea necesario).

Un periférico nunca puede comenzar una comunicación de forma espontánea con el HC, de esta manera ha de esperar y responder a peticiones por parte de éste, ya sea por polling o por respuesta a peticiones.

El controlador USB o interfaz que incorpora cada periférico controla en gran parte las comunicaciones con el HC, de todas maneras dependiendo de este controlador veremos hasta qué punto tenemos que programar en el dispositivo esclavo de la interfaz parte de las respuestas.

Cada periférico controla en todo momento las comunicaciones que se realizan en el sistema, y éste tiene que estar preparado para responder cuando la dirección a la que está dirigida la comunicación sea la que tiene asignada como propia, que ha determinado el HC con anterioridad.

Al igual que el HC envía bits de paridad, el periférico tiene que tomar las medidas pertinentes cuando detecta un error que en su caso equivale a validar la información o ignorarla cuando detecte un error.

Si el periférico está alimentado por el propio bus, ha de cumplir unas condiciones necesarias para respetar las limitaciones de éste. Estas limitaciones son consumir no más de 500 mA en funcionamiento estándar, no consumir más de 100 mA durante el proceso de enumeración o consumir un máximo de entre 0.5 i 2.5 mA dependiendo de la configuración en estado de *StandBy*.

Si consideramos un driver como un programa de software específico que utilizamos para comunicarnos con un dispositivo de hardware, lo primero que tenemos que hacer es especificar cual es el driver que nuestro periférico utilizará para relacionarse con el sistema y en concreto con nuestra aplicación.

Lo que el sistema operativo debe hacer es detectar e instalar el Hardware DD o Device Driver. Éste podría ser creado específicamente para un periférico en cuestión como en nuestro caso, que se utilizarán los drivers realizados por Microchip para los microcontroladores PIC con USB 2.0.

Los pasos a seguir son los siguientes: en primer lugar conectaremos el dispositivo, Windows en este punto comienza automáticamente el proceso de enumeración a partir del cual, y una vez obtenidas las tablas descriptoras del dispositivo, las compara con los archivos de descripciones *.INF de donde obtiene los drivers necesarios que pueda necesitar cada uno de los dispositivos que componen la base de datos de históricos con dispositivos conectados con anterioridad.

Gráficamente los estados por los que pasa el periférico son representados en el gráfico que vemos a continuación. Destacaremos que por definición del estándar, cualquier dispositivo que, configurado o no, no detecte actividad en el bus durante un periodo de 3 ms ha de pasar a la posición de suspensión.

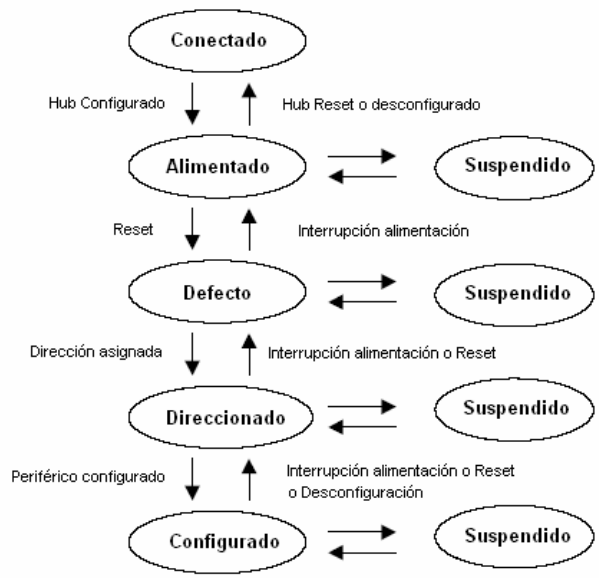


Figura 1.9 Diagrama de estados del periférico.

Queda reflejado en el gráfico anterior las situaciones que pueden hacer que un dispositivo pase de un estado a otro.

1. El primer paso consiste en conectar el dispositivo al puerto USB, al encontrarse conectado y por el hecho de tener el Hub configurado, el dispositivo pasa directamente al estado de alimentado.

2. En este momento el Hub al cual se haya conectado el periférico detecta la presencia de un periférico y su velocidad de comunicación (alta velocidad 12 Mbs o baja velocidad 1.5 Mbs). Físicamente el Hub dispone de dos resistencias entre las dos líneas de datos y masa mientras que el dispositivo tiene una entre la línea de alimentación y una de las líneas de datos.

Si ésta es la D+ se trata de un dispositivo de alta velocidad mientras que si ésta está conectada a la línea de datos D- se trata de un dispositivo de baja velocidad. El Hub es responsable de hacer llegar la novedad de que un periférico se acaba de conectar al HC. En este momento el periférico aún no está en condiciones de recibir información.

3. A partir de este momento el HC sabe que alguna cosa pasa en uno de los Hubs y envía el comando Get_Port_Status para determinar qué ha pasado en el puerto en concreto, de esta manera ahora es el HC que ya tiene conocimiento de la anexión de un nuevo periférico.

4. Lo que hace después el HC es ordenar al Hub que haga un reset al periférico con la instrucción propia de los Hubs, *Set_Port_Feature*, exclusiva para uno de los puertos en concreto de los que disponga el Hub. De esta manera otros periféricos que puedan estar conectados en el mismo Hub no tienen conocimiento de lo que pasa en el otro puerto.

5. Al finalizar este reset nuestro periférico se encuentra en el estado *Defecto* o *Default State*. En este momento el periférico no puede consumir más de 100 mA, es el único de los periféricos que en este momento puede responder a las instrucciones que el HC envía a la dirección por defecto que es la 0H y tiene establecido un canal o *pipe* de comunicación con el Endpoint 0 (nombramos Endpoint0 al conjunto formado por el Endpoint 0 In y el Endpoint 0 Out). A partir de este momento el dispositivo está en condiciones de responder a los comandos de control.

6. En alguno de estos momentos, antes o después del Reset, siempre dependiendo de cada Hub, se detecta la velocidad del periférico, pero no es hasta este momento que el HC recibe esta información a través del comando *Get_Port_Status*.

7. A continuación el HC se pone en contacto con el periférico con la instrucción de control *Get_Descriptor*. Esta instrucción se dirige a la dirección 0 Endpoint 0.

   Sólo un periférico puede responder ya que solo un periférico se puede numerar cada vez (el HC es responsable que así sea). El periférico responde con una de las tablas de que dispone en la memoria del dispositivo, el cual describe entre otras cosas la capacidad máxima del Endpoint 0.

8. El siguiente paso que realiza el HC es asignar una dirección que será única para nuestro periférico mediante el comando de control *Set_Address*. Esta dirección puede variar cada vez que el periférico se conecta, pero una vez conectado y numerado ésta será inalterable. Tenemos en este punto el periférico Direccionado.

9. El HC a partir de este momento envía otra vez el comando de control *Get_Descriptor* a la nueva dirección asignada al periférico con la finalidad de comprobar que éste está donde debería estar y compara la respuesta obtenida con la obtenida en el paso 7 para verificar que es el mismo dispositivo. Una vez verificado el correcto funcionamiento hasta este punto se solicita el resto de información de los descriptores o tablas que almacena el dispositivo. Esto incluye información sobre Endpoints, posibles configuraciones, consumo, fabricante y también la clase a la que pertenece el dispositivo.

10. A partir de la información obtenida de los descriptores del periférico y comparando el fabricante, producto y clase a la que pertenece éste, el sistema compara la información con la almacenada en los archivos de sistema .INF comentados anteriormente y selecciona en este momento el driver a cargar para las comunicaciones. Finalizando en este momento el proceso de numeración y dejando el dispositivo listo para ser utilizado por las aplicaciones que se puedan ejecutar en el ordenador.

Son finalmente estos drivers los que seleccionan alguna de las configuraciones posibles, si es que más de una está disponible basándose en disponibilidad del sistema, elección del propio usuario u otros. A partir de este momento el proceso finaliza y ya tenemos el periférico configurado y listo para funcionar y responder a las peticiones para enviar y recibir información que hagan las aplicaciones del usuario.

1.5 Comunicaciones móviles GSM.

1.5.1 Antecedentes.

Actualmente en España existe un índice de penetración de la telefonía móvil de un 100%. Este índice no indica un valor real de usuarios, puesto que se contabilizan las líneas de prepago y las de uso ocasional, pero sí nos indica que realmente el uso del teléfono móvil está muy extendido. Además, el 98% del territorio poblado español cuenta con cobertura GSM. Teniendo en cuenta el fácil manejo, el bajo coste y el gran número de usuarios de teléfonos móviles, se hace idóneo para un proyecto de control y/o supervisión a distancia. [13, 14]

1.5.2 Red GSM.

El sistema GSM es el sistema de comunicación de móviles digital de 2ª generación basado en células de radio. Apareció para dar respuestas a los problemas de los sistemas analógicos.

Fue diseñado para la transmisión de voz por lo que se basa en la conmutación de circuitos, aspecto en el que se diferencia del sistema GPRS. Al realizar la transmisión mediante conmutación de circuitos los recursos quedan ocupados durante toda la comunicación y la tarificación es por tiempo.

1.5.2.1 Arquitectura de una red GSM.

Todas las redes GSM se pueden dividir en cuatro partes fundamentales y bien diferenciadas:

1. *La Estación Móvil o Mobile Station (MS):* Consta a su vez de dos elementos básicos que debemos conocer, por un lado el terminal o equipo móvil y por otro lado la tarjeta SIM. La diferencia que existe entre unos terminales y otros radica fundamentalmente en la potencia que tienen que va desde los 20 W (generalmente instalados en vehículos) hasta los 2 W de los teléfonos comunes.

   Una tarjeta SIM (acrónimo de Subscriber Identity Module, 'Módulo de Identificación del Suscriptor') es una tarjeta inteligente desmontable usada en teléfonos móviles que almacena de forma segura la clave de servicio del suscriptor usada para identificarse ante la red, de forma que sea posible cambiar la línea de un terminal a otro simplemente cambiando la tarjeta.

2. *El Subsistema de Conmutación y Red o Network and Switching Subsystem (NSS):* Este sistema se encarga de administrar las comunicaciones que se realizan entre los diferentes usuarios de la red.

3. *La Estación Base o Base Station Subsystem (BSS):* Sirve para conectar a las estaciones móviles con los NSS, además de ser los encargados de la transmisión y recepción. Como los MS también constan de dos elementos diferenciados: La Base Transceiver Station (BTS) o Base Station y la Base Station Controller (BSC). La BTS consta de transceivers y antenas usadas en cada célula de la red y que suelen estar situadas en el centro de la célula, generalmente su potencia de transmisión determinan el tamaño de la célula.

4. *Los Subsistemas de soporte y Operación o Operation and Support Subsystem (OSS):* Los OSS se conectan a diferentes NSS y BSC para controlar y monitorizar toda la red GSM. La tendencia actual en estos sistemas es que, dado que el número de BSS se está incrementando se pretende delegar funciones que actualmente se encarga de hacerlas el subsistema OSS en los BTS de modo que se reduzcan los costes de mantenimiento del sistema.

1.5.2.2 Transmisión de datos por GSM.

Las redes GSM tienen ciertas limitaciones para la transmisión de datos:

- Velocidad de transferencia de 9,6 Kbps.
- Tiempo de establecimiento de conexión, de 15 a 30 segundos.
- Pago por tiempo de conexión.
- Problemas para mantener la conectividad en itinerancia (Roaming).

Las tradicionales redes GSM no se adaptan adecuadamente a las necesidades de transmisión de datos con terminales móviles. Por ello surge una nueva tecnología portadora denominada GPRS (General Packet Radio Service) que unifica el protocolo de

internet IP con la telefonía móvil, creándose toda una red paralela a la red GSM y orientada exclusivamente a la transmisión de datos.

Al sistema GPRS se le conoce también como GSM-IP ya que usa la tecnología IP (Internet Protocol) para acceder directamente a los proveedores de contenidos de Internet.

Debido al pequeño volumen de información que debemos enviar, que será inferior a 100 caracteres, y a que los envíos se producirán en ocasiones puntuales, cuando el usuario presione alguno de los pulsadores del dispositivo, se utilizará la tecnología GSM mediante los mensajes cortos SMS.

En el caso de necesitar enviar un volumen más grande y continuo, la solución adecuada sería usar la tecnología GPRS, facturando por datos enviados y no por mensaje enviado.

1.5.3 Servicio de Mensajes Cortos SMS.

1.5.3.1 Introducción al SMS.

El servicio de mensajes cortos SMS (Short Message Service) permite enviar o recibir mensajes breves de texto (máximo de 160 caracteres), desde un teléfono GSM o Centro de Información a otro teléfono GSM. Los mensajes serán visualizados directamente en la pantalla del teléfono o del dispositivo que se encuentre conectado al módem GSM. Al recibir el mensaje, además del texto, se obtiene una serie de datos como es el remitente, la hora y la fecha de recepción.

Para el envío de un SMS, es necesario que el dispositivo GSM tenga configurado el número telefónico del Centro de Mensajes propio del operador. Por defecto el módem utiliza el número que viene almacenado en la propia tarjeta SIM. Se deberá redactar el texto a enviar y finalmente se deberá introducir el número telefónico del destinatario. Una vez enviado, el SMS es transmitido desde la red hacia el teléfono destino. Si éste permanece apagado o fuera de cobertura, la red reintentará enviar el mensaje durante 48 horas, superadas las cuales éste ya no se entregará. En ocasiones el mensaje enviado no es recibido por el destinatario debido a que su buzón de entrada se encuentra saturado, por eso es importante controlar que no se llene el buzón del teléfono.

1.5.3.2 Funcionamiento del sistema SMS.

En el standard GSM hay especificados dos tipos diferentes de SMS:

- SMS Point to Point (SMS/PP)
- SMS Cell Broadcast (SMS/CB)

El primer tipo permite enviar un mensaje de texto de un teléfono GSM a otro, mientras que el tipo *SMS/CB* permite enviar uno o más mensajes simultáneamente (broadcast) a todos los teléfonos que estén dentro de una determinada zona de cobertura de uno o más emisores de señal de radio.

El SMS usa un protocolo sin conexión, ya que cuando se transmite un mensaje no se produce ninguna conexión directa entre el terminal que envía y el que recibe, como sucede, por ejemplo, en el caso de las llamadas de voz, datos o fax.

El envío de un *SMS/PP* desde un teléfono GSM a otro, tiene que ser considerado como la concatenación de dos operaciones diferentes: la transmisión del mensaje desde el teléfono móvil a una entidad especial de la red, llamada *SMSC* (Short Message Service Center), y luego desde el SMSC hasta el teléfono receptor. La primera operación se denomina *SMS-MO (SMS Mobile Originated)*, mientras que la segunda se conoce como *SMS-MT (SMS Mobile Terminated)*.

Los mensajes de texto SMS pueden ser de dos formatos diferentes, uno es el modo texto y el otro es como datos binarios a 8 bits (modo PDU).

Utilizando la transmisión binaria, el teléfono ya no visualizará ningún mensaje en la pantalla y se harán necesarios un hardware y una aplicación software capaces de saber leer las informaciones binarias. De esta forma es posible introducir fácilmente una compresión de datos para aumentar la capacidad de cada uno de los mensajes breves o aprovechar este tipo de transmisión para otras aplicaciones. La capacidad máxima con los datos a 8 bit es de 140 bytes (1120 bit), pero utilizando los mensajes normales de texto (codificación a 7 bit), la capacidad máxima posible, es de 160 caracteres (7 x 160 = 1120).

Dado que el *SMS* es un protocolo sin conexión, porque el terminal transmisor y el receptor no están conectados directamente como sucede durante una llamada, el tiempo que transcurre entre la transmisión del mensaje desde un terminal móvil y la recepción del mismo por parte de otro terminal no es estándar.

El envío de un mensaje entre un teléfono y el SMSC, sin importar en qué dirección se realiza, emplea un tiempo de 3 a 5 segundos. Dado que el envío de un mensaje de un teléfono a otro es la concatenación de dos operaciones de transmisión, el retraso total alcanza de los 6 a los 10 segundos. En la práctica, sin embargo, el retraso puede ser notablemente superior. De hecho, los mensajes SMS son transmitidos a través de canales de control, y la velocidad de transferencia de los mensajes se ve considerablemente influida por el tráfico generado por todas las demás señales que comparten el mismo canal de control, como, por ejemplo, informaciones relativas a la radioconexión, control de la potencia, actualización de la posición, etc.

1.6 Comandos AT.

1.6.1 Introducción.

El control del módem GSM a través del microcontrolador se basa en el uso de comandos denominados Hayes o AT, de los que exponemos brevemente algunas referencias.

Hayes es el nombre de una empresa que en los orígenes de la comunicación por módem definió una serie de comandos u órdenes para que el software de comunicaciones pudiera comunicarse con el módem, pudiendo controlar este último.

Estos comandos tuvieron tanto éxito que se convirtieron en el virtual estándar de comunicaciones, y los módems que los comprenden (la mayoría de los módems modernos) se denominan compatibles Hayes.

Todos los comandos de este protocolo comienzan por AT y acaban por el carácter <CR> (ascii 13) (aunque este último carácter es configurable).

Las respuestas del módem tienen el siguiente formato:

<CR><LF>(texto respuesta)<CR><LF>. En adelante solo mostraremos el texto de la respuesta, obviando los caracteres <CR> y <LF>.

1.6.2 Tipos de Comandos AT.

Existen dos tipos principales de comandos:

- Comandos que ejecutan acciones inmediatas (ATD marcación, ATA contestación o ATH desconexión)

- Comandos que cambian algún parámetro del módem (ATS2=43 configuración del carácter de escape).

1.6.3 Listado de comandos utilizados.

A continuación presentamos un listado de comandos AT utilizados en este proyecto así como las respuestas que retorna el módem al recibir dichos comandos.

El listado completo de comandos AT soportados por el MODEM Siemens TC35i lo podremos encontrar en el "TC35i_AT_Command_Set", disponible en la web del fabricante.

1.6.3.1 Configuración comunicación serie.

Para comprobar que el MODEM responde y además configurarlo a 9600 baudios, se utiliza el comando AT+IPR.

Se envía **AT+IPR=9600<cr>** y se recibe un **OK** .

1.6.3.2 Códigos de acceso.

Para consultar o introducir el código de acceso, se utiliza el comando AT+CPIN.

Primero debe consultarse el estado del código de acceso, enviamos **AT+CPIN?** y podemos recibir las siguientes respuestas:

- **+CPIN: SIM PIN**: se requiere el código PIN.
- **+CPIN: READY**: el PIN ya se ha introducido con anterioridad.

- **+CPIN: SIM PUK**: se requiere el código de desbloqueo PUK.

En todos los casos, se recibe un **OK** a continuación.

Se envia **AT+CPIN=1234** y se recibe un **OK** o un **ERROR** dependiendo si es correcto o no.

1.6.3.3 Lectura del Número del Centro de Mensajes.

El comando **AT+CSCA?** solicita al módem el número de teléfono del Centro de Mensajes. Responde con la siguiente trama:

+CSCA: "+34656000311",145 seguido de un **OK**.

Este parámetro no lo necesitamos para el funcionamiento del dispositivo, simplemente lo leemos para almacenarlo en la memoria de datos del PIC y mostrarlo en la casilla correspondiente del software.

1.6.3.4 Configuración tipo SMS.

El comando AT+CMGF configura el tipo de SMS que se va a manejar. Hay dos opciones, modo TEXT y modo PDU. El modo PDU codifica el mensaje recibido, creando una trama ilegible directamente. Por ello, para facilitar la tarea, lo configuramos en Text Mode.

Se envía **AT+CMGF=1** y se recibe un **OK**.

1.6.3.5 Configuración aviso SMS.

Para que el módem envíe un mensaje cuando reciba un nuevo SMS, debemos configurarlo con el comando AT+CNMI.

Se envía **AT+CNMI=3,1,0,0** y se recibe un **OK**.

Donde '**3**' es el valor que permite la recepción de un aviso sin solicitarlo, el '**1**' solicita que se envíe además la posición en memoria donde se ha almacenado el mensaje recibido y los dos '0' corresponden a funciones que no vamos a manejar.

Al recibir un nuevo SMS, la trama que nos envía el módem es la siguiente:

+CMTI: "SM",1, donde 'SM' indica que el mensaje se ha almacenado en la memoria de la tarjeta SIM y el '1' indica que en la primera posición.

1.6.3.6 Borrado de SMS.

Cada vez que se recibe y se trata un SMS, éste es borrado con el fin de asegurar que cualquier SMS recibido tenga sitio en la memoria para ser almacenado.

El comando utilizado es AT+CGMD, y se utiliza así.

Se envía **AT+CMGD=1** para borrar el mensaje 1 de la memoria y se recibe un **OK**.

1.6.3.7 Envío de SMS.

Para el envío de mensajes SMS el procedimiento a seguir es algo más complejo. El comando a usar es AT+CMGS, a continuación explicamos su funcionamiento con un ejemplo práctico:

Enviamos **AT+CMGS=”+34666777888”,145**, donde ‘+34666777888’ es el número del teléfono móvil destinatario y ‘145’ indica que el número de teléfono lo insertamos en formato internacional, con prefijo.

Una vez se ha enviado el comando, el módem responde con un carácter ‘>’ que nos indica que ya podemos introducir el cuerpo del mensaje. Una vez se haya introducido el texto que se quiere enviar, se debe finalizar el texto con Ctrl-Z, que es el carácter ascii 26.

El módem responderá con la posición donde se ha almacenado el mensaje enviado **+CMGS: 1** seguido de un **OK**.

Todo el proceso quedaría de la siguiente forma:

→ **AT+CMGS=”+34666777888”,145**

← >

→ **mensaje de prueba<ctrl-z>**

← **+CMGS: 1**

← **OK**

1.6.3.8 Lectura de SMS recibido.

Como se ha comentado anteriormente, cuando el módem recibe un nuevo mensaje SMS nos envía un aviso como este:

+CMTI: "SM",1, donde ‘SM’ indica que el mensaje se ha almacenado en la memoria de la tarjeta SIM y el ‘1’ indica que en la primera posición.

Una vez sabemos la posición en la que se ha almacenado el SMS, en el caso del ejemplo la 1, procedemos a enviar el comando AT de lectura AT+CMGR.

Se envía AT+CMGR=1 y se recibe la siguiente cadena:

+CMGR: "REC UNREAD","+34666777888",,"15/05/08,09:04:44+08"
Texto del mensaje
OK

De la respuesta obtenida, sólo haremos uso del cuerpo del mensaje, no utilizaremos ni el teléfono del remitente ni la fecha ni la hora del envío. Solo almacenaremos el texto del mensaje para su posterior tratamiento.

1.7 Pseudocomandos de comunicación entre PC y microcontrolador.

Para enviar los parámetros necesarios al dispositivo, al igual que para recibirlos para visualizarlos y modificarlos si es necesario, debemos crear unos comandos que entiendan el software del PC y el firmware del microcontrolador PIC.

1.7.1 Envío de datos.

Los pseudocomandos para el envío de datos tienen el siguiente formato:

[valor función][dígitos parámetro a enviar][parámetro a enviar]

Donde:

[valor función] puede ser:

- 1 = Envío de código PIN.
- 2 = Envío de número de teléfono del destinatario 1.
- 3 = Envío de número de teléfono del destinatario 2.

[dígitos parámetro a enviar] puede ser:

- 4, para el código PIN.
- 9, para cada uno de los números de teléfono destinatarios, (se manejan móviles nacionales)

Por ejemplo, para enviar el código PIN “1234”, el pseudocomando necesario es: 141234

1.7.2 Solicitud de datos.

Los pseudocomandos para la petición de datos al dispositivo, para mostrarlos en la ventana del software tienen el siguiente formato:

[valor función]

Donde:

[valor función] puede ser:

- 5 = Solicitud de código PIN.

- 6 = Solicitud del número de teléfono del Centro de Mensajes.
- 7 = Solicitud del número de teléfono del destinatario 1.
- 8 = Solicitud del número de teléfono del destinatario 2.
- 91 = Solicitud del texto del SMS 1.
- 92 = Solicitud del texto del SMS 2.
- 93 = Solicitud del texto del SMS Consigna.

Cuando el PIC recibe como primer dato un 9, entiende que se está solicitando un dato relativo a los SMS y vuelve a leer el siguiente byte para determinar qué mensaje se está solicitando.

El carácter 4 como valor de función no se utiliza, se reserva para futuras aplicaciones.

1.8 Mapa de direcciones de los datos almacenados en EEPROM.

En la memoria EEPROM interna del PIC es donde se almacenan los datos que necesitamos para configurar y manejar el módem GSM.

La capacidad de la memoria es de 256 bytes, y en todas las posiciones de ésta se lee por defecto un 0xFF.

La organización de la memoria es, en bytes, la siguiente:

Bytes (inicio-fin)	Valor
0-3	Código PIN
4-12	Número de teléfono del Centro de Mensajes
13-21	Número de teléfono del Destinatario 1
22-30	Número de teléfono del Destinatario 2
31-132	Texto del SMS 1
133-232	Texto del SMS 1
233-237	Texto del SMS Consigna

Tabla 1.1 Direcciones de los valores almacenados.

Se observa que las últimas posiciones de la memoria (hasta la 255) están libres, se reservan por si fuera necesario ampliar el contenido del SMS Consigna.

1.9 Descripción General del Equipo.

1.9.1 Esquema global.

El presente proyecto consiste en el diseño y montaje de un circuito que controle un MODEM GSM mediante un microcontrolador para que se envíen mensajes cortos SMS a unos destinatarios fijados por el usuario y, por otro lado, responda mediante unas acciones preprogramadas al recibir un SMS determinado.

El usuario realiza una función imprescindible para el funcionamiento del circuito, ya que se deben introducir una serie de datos a la memoria EEPROM de datos del PIC a través del software del PC. Una vez han sido introducidos los datos, el circuito funciona de manera independiente, llevando a cabo sus tareas sin más apoyo del PC que la alimentación que recibe de éste a través del puerto USB. El circuito se comunica con el MODEM mediante RS-232, vía por la cual se transmiten los comandos AT, explicados más adelante.

Para un uso supervisor, se ha diseñado una conexión RS-232 entre el circuito y el PC, a través de la que se verán reflejadas las comunicaciones serie que existan entre circuito y MODEM.

A continuación se muestra el esquema global de conexionado, mostrando en un tono más claro, la conexión opcional de supervisión.

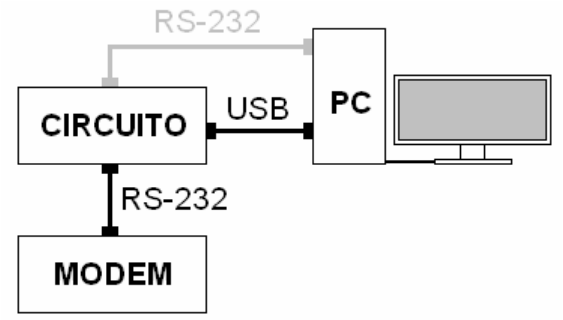


Figura 1.10 Esquema global del conexionado.

Esta conexión de supervisión sirve para monitorizar las comunicaciones entre el microcontrolador y el módem. Ha sido imprescindible para la programación del firmware y es una opción interesante para visualizar el funcionamiento del sistema desde el software en su versión SUPERVISOR.

1.9.2 Composición del sistema.

1.9.2.1 Circuito del microcontrolador PIC.

A continuación se muestra el esquema electrónico del circuito.

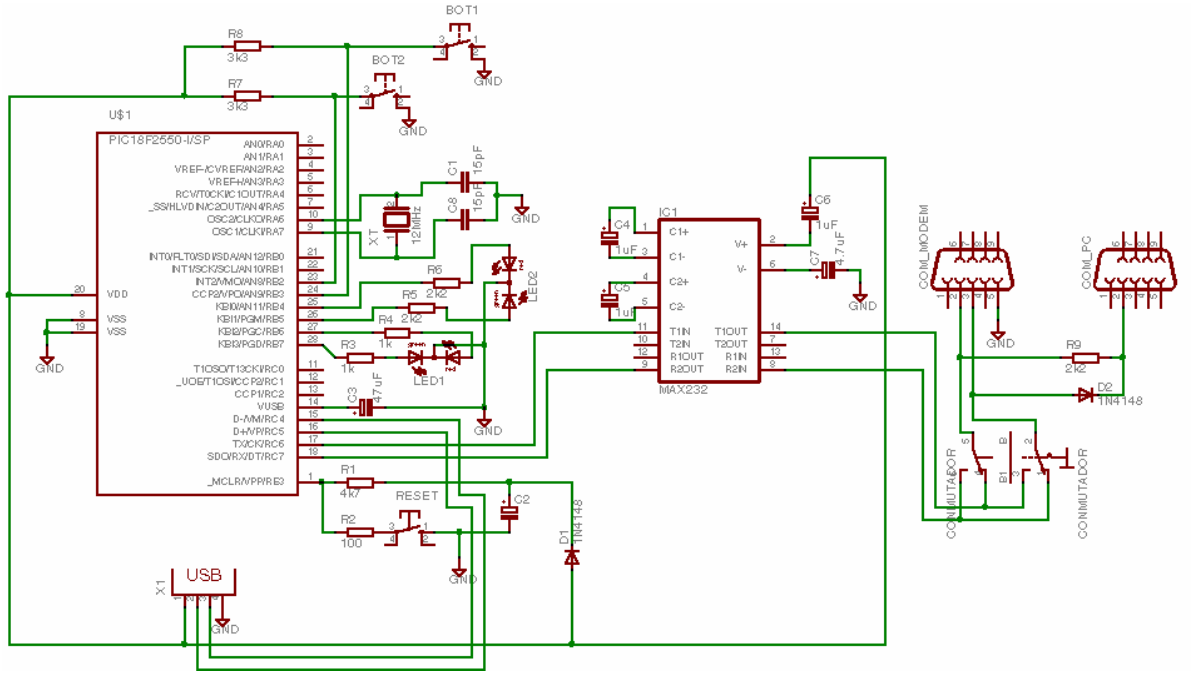


Figura 1.11 Esquema electrónico del circuito.

1.9.2.2 Comunicación entre circuito y PC.

Los datos que el PIC necesita para la configuración y el manejo del módem GSM los ha de introducir el usuario a través del software diseñado a tal efecto. Desde el momento en que sean introducidos los datos, aunque se desconecte el circuito de la alimentación, al almacenarse los datos en la memoria EEPROM del PIC, el circuito funcionará correctamente en posteriores encendidos. Además, el PC mostrará los datos que están almacenados en la memoria EEPROM del PIC (si es el caso). Para transferir estos datos lo haremos por el bus serie USB, aprovechando que el circuito está conectado a este puerto del PC para su alimentación.

Pese a transferir datos a muy baja velocidad, se ha implementado un *handshaking* para asegurar la correcta lectura del dato desde el receptor. Este control de errores consiste en devolver cada byte que se recibe, de tal manera que el emisor puede comprobar que el dato se ha leído correctamente y puede proceder al envío del siguiente.

1.9.2.3 Interfaz en el PC.

El software utilizado para interactuar con el usuario está creado con Visual C# de Microsoft Visual Studio 2005. Visual C# es un lenguaje de programación orientado a objetos, con una sintaxis que se asemeja a la del C. Un factor importante es que se pueden usar las funciones de las APIs suministradas por MICROCHIP.

El aspecto visual del programa pretende ser simple e intuitivo, para que se pueda utilizar sin apenas haber leído el manual de usuario.

1.10 Especificaciones técnicas.

1.10.1 Entorno PC.

El entorno en el cual se va a programar la aplicación y donde posteriormente se ejecutará, tiene las siguientes características:

- El sistema operativo debe ser Windows con motor NT, se recomienda Microsoft Windows XP.

- Se debe disponer de las librerías estándar utilizadas por Visual C#, además de la versión 2.0.50727 de Microsoft .NET Framework, puesto que las funciones de comunicación serie vienen definidas en esta versión y no en anteriores.

- La versión de los puertos USB debe ser la 2.0. En ordenadores antiguos que solo manejen USB 1.0, las comunicaciones entre software y dispositivo no funcionarán correctamente.

1.10.2 Alimentación del circuito.

El circuito se alimentará del puerto USB del PC. Éste suministra una tensión de 5V y una corriente de hasta 500 mA, obteniendo una potencia máxima de 2.5W. Tal y como se comprueba en la memoria de cálculo, el circuito consume un pequeño porcentaje de la intensidad máxima que ofrece el host USB.

1.10.3 Microcontrolador.

El microcontrolador elegido para este proyecto es el 18F2550 de Microchip, un dispositivo de bajo consumo y de altas prestaciones que cuenta con conectividad USB.

- Tensión de alimentación: 5 V.
- Corriente máxima de entrada: 250 mA.
- Corriente máxima de salida por todos los pines: 200 mA.
- Corriente máxima de entrada en un pin I/O: 25 mA.
- Corriente máxima de salida en un pin I/O: 25 mA.
- Temperatura ambiente “Under bias”: -40 ºC < T < +85 ºC
- Temperatura de almacenamiento: -65 ºC < T < +150 ºC
- Potencia disipada total: 1.0 W
- Memoria FLASH: 32 kbytes
- Memoria SRAM: 2048 bytes
- Memoria EEPROM: 256 bytes

- Dimensiones: 10,34 x 17,87 x 2,50 mm
- Tipo de encapsulado: PDIP

1.11 Entorno de trabajo.

Una vez determinados los elementos que componen el sistema, se debe realizar la programación del firmware del microcontrolador y la grabación del programa en la memoria del PIC. También se deberá realiza la programación del software del PC.

1.11.1 Programa del microcontrolador.

1.11.1.1 Compilador.

Los lenguajes más extendidos son el ensamblador y el C. El lenguaje de programación elegido para este proyecto ha sido el C por su facilidad para la realización de tareas complejas, simplificándolas notablemente. El inconveniente de este tipo de programación es el mayor número de instrucciones que genera en comparación con el ensamblador, aunque cada vez los compiladores de C están mejor diseñados y generan códigos similares en tamaño a los desarrollados puramente en ASM.

Una vez determinado el lenguaje de programación con el que se va a trabajar, en este caso el C, el siguiente paso es seleccionar el compilador. Existen varios programas, aunque los más comunes son el C18 de Microchip y el PCWH de CCS. El compilador elegido es el PCWH de CCS, por su facilidad de uso y el gran número de microcontroladores que soporta.

A continuación se detalla el procedimiento seguido para crear y compilar el código.

- Al ejecutar el software PCW C Compiler, en la pantalla principal, se crea un nuevo archivo llamado Picmodem_07.C:

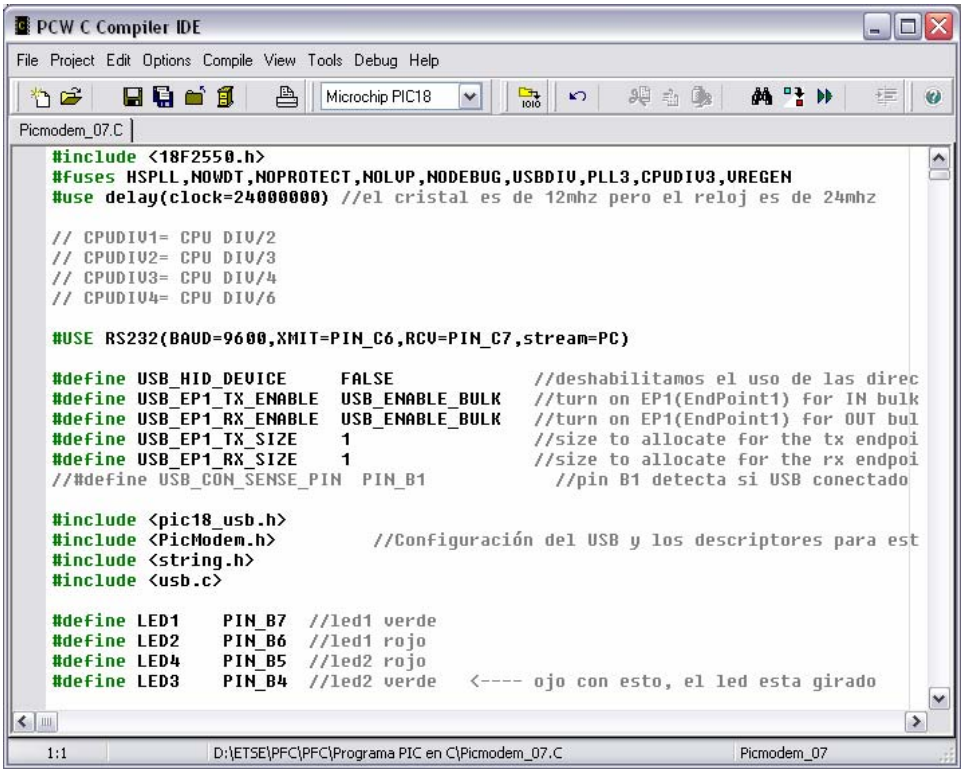


Figura 1.12 Vista del editor de código.

- Una vez se terminó de escribir el código, se procedió a la compilación:

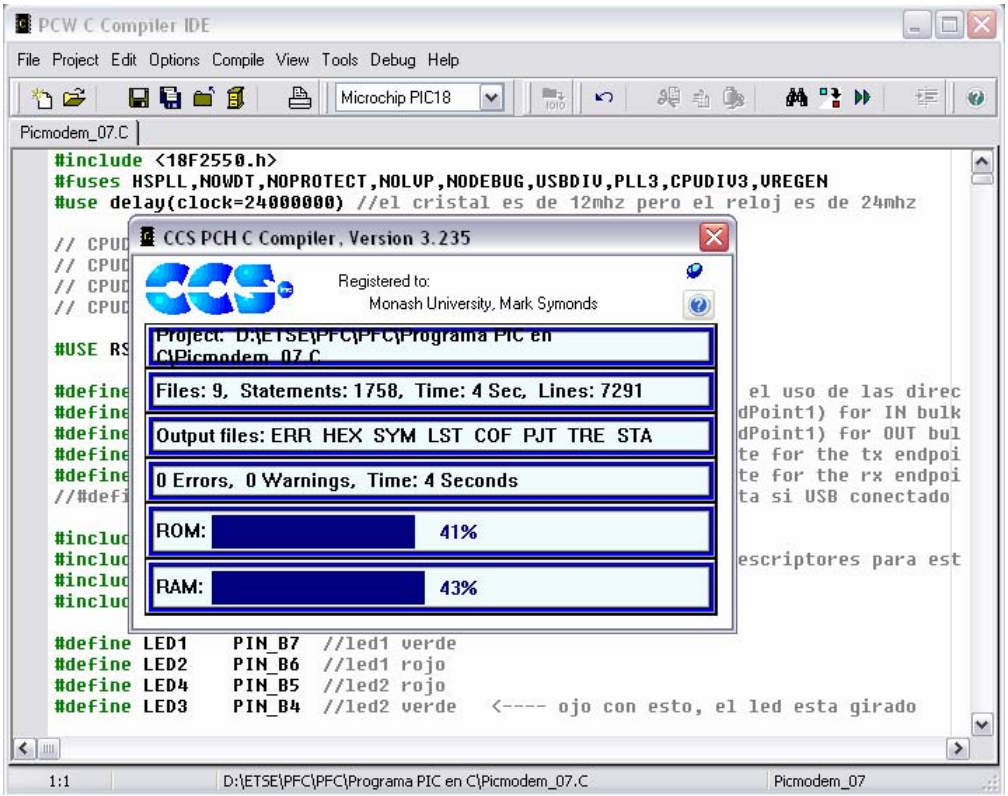


Figura 1.13 Vista del resultado de la compilación.

Durante la programación del código se fueron haciendo sucesivas compilaciones para comprobar que no se estaban cometiendo errores, o para corregirlos. El compilador PCWH indica la línea donde se ha cometido un error facilitando así su modificación.

1.11.1.2 Software de grabación.

Una vez se ha compilado correctamente el código creado con el software PCWH C Compiler, se obtiene un archivo hexadecimal que es el que hay que grabar en el microcontrolador.

El software de grabación usado es el WinPIC800, debido a su gran difusión y su libre distribución a través de internet. Este software admite gran variedad de programadores por puerto serie, paralelo e incluso por puerto USB.

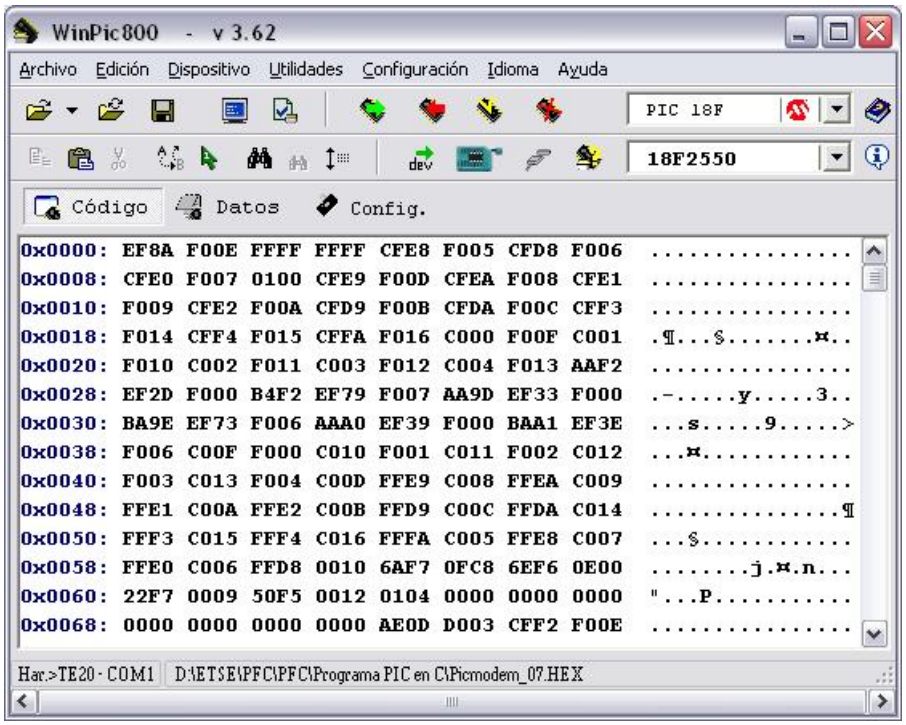


Figura 1.14 Software de grabación WinPIC800.

En la anterior figura se observa la pantalla principal del software de grabación con el fichero .HEX cargado.

Al grabar el programa en el PIC, aparece una pantalla mostrando el progreso de la programación:

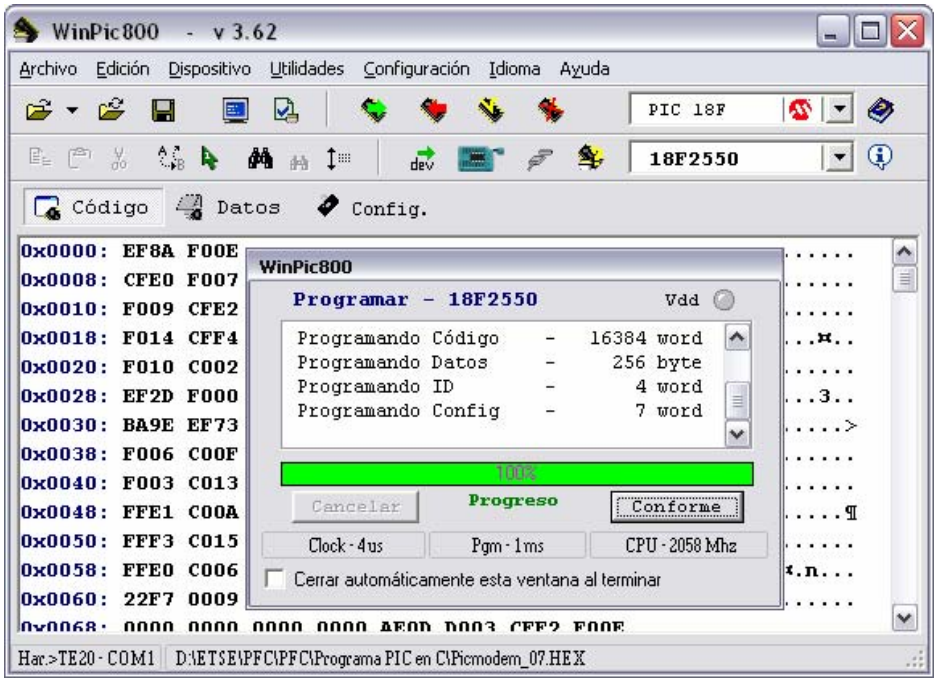


Figura 1.15 Grabación finalizada.

1.11.1.3 Hardware de grabación.

Para llevar a cabo la programación del PIC, es necesario un hardware que sea reconocido por el software de grabación. En este caso usamos un programador clásico tipo JDM modelo TE-20, que usa el puerto serie del PC.

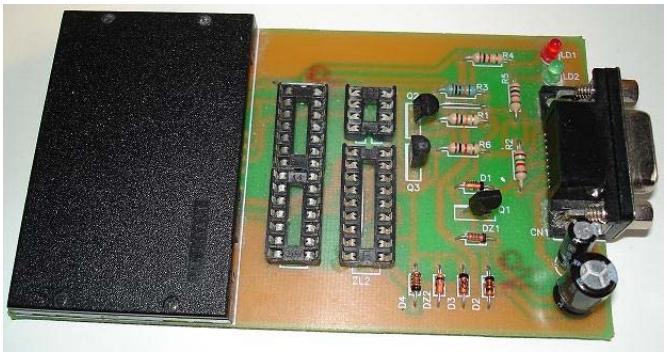


Figura 1.16 Programador TE-20.

1.11.2 Programa del PC.

El lenguaje elegido es Visual C# 2008 Express Edition de Microsoft, por su facilidad de uso y el gran número de herramientas de ayuda que existen.

A continuación se detalla el procedimiento seguido para crear y compilar el código.

Al ejecutar el software Visual C# 2008 Express Edition, en la pantalla principal, se crea un nuevo proyecto donde se irán agrupando todos los ficheros.

Se puede ejecutar el asistente para que nos facilite la tarea de crear el proyecto.

Figura 1.17 Vista del asistente para crear un nuevo proyecto.

Una vez se ha escrito el código del programa y se ha diseñado la interfaz gráfica, se procederá a la compilación del proyecto. Si no ha habido ningún error, aparecerá la interfaz en pantalla y ya estará funcionando el software. En caso de haber cometido algún error, Visual C# nos indicará en qué fichero y en qué líneas del código está el fallo.

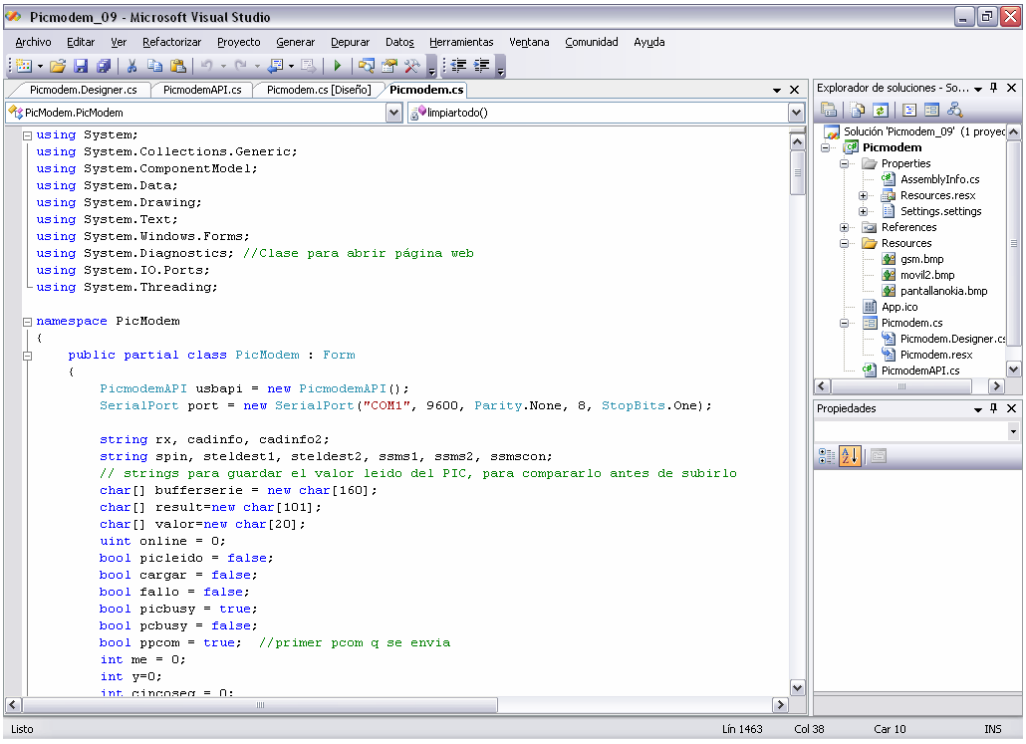


Figura 1.18 Vista del proyecto Picmodem de Visual C#.

2 Memoria de Cálculo.

2.1 Alimentación requerida.

El circuito está compuesto por un microcontrolador, un circuito integrado MAX232, dos LEDs y algunos componentes pasivos.

El circuito integrado MAX232 necesita una tensión dentro del rango de 4.5V y 5.5V. El microcontrolador abarca un mayor rango, desde 2V hasta 5.5V. La tensión suministrada por el puerto serie es de 5V, valor ideal para nuestro diseño.

El módem GSM se alimenta con su propio adaptador 220V AC – 12V DC.

2.2 Corriente en los LED.

El circuito dispone de dos LED bicolor, cada led tiene tres patas, una para cada color y una masa común. Por lo tanto cada LED estará conectado a dos salidas del PIC.

La corriente que puede suministrar el microcontrolador a cada pin de entrada/salida es de 25mA, por lo que se deberá calcular una resistencia que proporcione un valor por debajo de éste máximo.

Corriente LED 1: $I_{LED1} = \frac{5V}{1k\Omega} = 5mA$ (1)

Corriente LED 2: $I_{LED2} = \frac{5V}{2k2\Omega} = 2.27mA$ (2)

Para obtener la misma luminosidad en ambos LEDs, se calcula una resistencia mayor en el LED 2.

La corriente total a suministrar a los LED será:

$$2 * I_{LED1} + 2 * I_{LED2} = 2 * 5mA + 2 * 2.27mA = 14.54mA \quad (3)$$

2.3 Corriente del dispositivo.

La corriente necesaria para el suministro del dispositivo será la que consuma el microcontrolador, el integrado MAX232 y los LED.

El microcontrolador tiene un consumo, en funcionamiento normal y a 5V, de entre 25 y 50 mA. Se deduce pues que el consumo máximo del microcontrolador será el equivalente a la suma de su consumo basal con el consumo de los LEDs, esto es un consumo de 65mA aproximadamente.

El otro componente relevante en el consumo total del circuito es el integrado MAX232. Su consumo normal es de 22mA y el máximo es de 30mA.

Como se puede observar, en el caso más desfavorable, el consumo que deberá suministrar el puerto USB del PC a nuestro dispositivo será de 95mA.

La corriente que puede suministrar el host USB es de 500mA, por lo que le estamos requiriendo un 20% de su capacidad.

2.4 Inicialización del PIC.

Para la programación del microcontrolador se utiliza el compilador de C CSS, concretamente la versión 3.235 que incorpora bibliotecas para el uso de USB. Antes de empezar a escribir nuestro programa hay que modificar algunas características de los archivos que proporciona el compilador y que utilizaremos en el proyecto. Estos archivos contienen la definición de los registros y funciones USB.

Estos ficheros son:

18F2550.h: Fichero de cabecera estándar para dispositivos PIC18F2550.
pic18_usb.h: Capa hardware de los dispositivos PIC18Fxx5x.
USB.C: Manejador de peticiones USB estándar.
USB.H: Prototipos de funciones, definiciones y variables globales USB.

Al no ser el periférico HID, habrá que hacer un cambio en USB.H:

```
#ifndef USB_HID_DEVICE
        #define USB_HID_DEVICE FALSE
#endif
```

Una vez hechas estas modificaciones, empezamos a desarrollar el programa que ejecutará el microcontrolador. Lo primero será establecer la palabra de configuración seleccionando las opciones adecuadas: oscilador en modo HSPLL con los valores de pre y post-escala adecuados para obtener una frecuencia de 48 MHz en el módulo USB, habilitación del regulador interno del módulo USB y deshabilitación del watchdog timer que no vamos a utilizar en el proyecto. Por último, para terminar con las definiciones, se habilita el punto final 1 para que soporte transferencias de interrupción de entrada.

Para configurar el microcontrolador desde el compilador, nos bastará con hacer uso de los *flags* disponibles. A continuación se muestran los *flags* utilizados y sus posibles configuraciones:

- NOPROTECT: Código no protegido frente escritura. PROTECT habilitaría esta protección.

- NOWDT: No Watch Dog Timer.
  Configuraciones posibles:
  - WDT1 Watch Dog Timer uses 1:1 Postscale

- WDT2 Watch Dog Timer uses 1:2 Postscale
- WDT4 Watch Dog Timer uses 1:4 Postscale
- WDT8 Watch Dog Timer uses 1:8 Postscale
- WDT16 Watch Dog Timer uses 1:16 Postscale
- WDT32 Watch Dog Timer uses 1:32 Postscale
- WDT64 Watch Dog Timer uses 1:64 Postscale
- WDT128 Watch Dog Timer uses 1:128 Postscale
- WDT Watch Dog Timer

- NODEBUG: No modo Debug para ICD, DEBUG habilitaría el uso con ICD.

- NOLVP: No Low Voltage Programming, LVP habilitaría la programación a 5V.

- HSPLL: High Speed Crystal/Resonator with PLL enabled

Configuraciones posibles:

- INTXT Internal Oscillator, XT used by USB
- INTHS Internal Oscillator, HS used by USB
- HS High speed Osc (> 4mhz)

- PLL3: Divide por 3 (entrada oscilador 12MHz)

Configuraciones posibles:

- PLL1 No PLL PreScaler
- PLL2 Divide por 2 (entrada oscilador 8MHz)
- PLL4 Divide por 4 (entrada oscilador 16MHz)
- PLL5 Divide por 5 (entrada oscilador 20MHz)
- PLL6 Divide por 6 (entrada oscilador 24MHz)
- PLL10 Divide por 10 (entrada oscilador 40MHz)
- PLL12 Divide por 12 (entrada oscilador 48MHz)

- CPUDIV3: System Clock por 4

Configuraciones posibles:

- CPUDIV1 System Clock por 2
- CPUDIV2 System Clock por 3
- CPUDIV4 System Clock por 6

- USBDIV USB Clock proveniente del PLL dividido por 2, la otra opción: NOUSBDIV: USB Clock proveniente del oscilador primario.

- VREGEN USB regulador de tensión habilitado, NOVREGEN deshabilita la regulación.

Se ha creado también un fichero de cabecera (picUSB.h) que contiene los descriptores del dispositivo USB siguiendo el esquema de la figura 28. Además de estos se han incluido dos descriptores de cadena que indican el fabricante del producto y su descripción.

2.4.1 Configuración del oscilador.

A continuación se muestra el esquema del módulo del oscilador indicando la fuente utilizada (oscilador primario) y los valores que toman los distintos bits de estos registros para conseguir, a partir de un cristal de 12 MHz, una velocidad en el módulo USB de 48 MHz (USB 2.0).

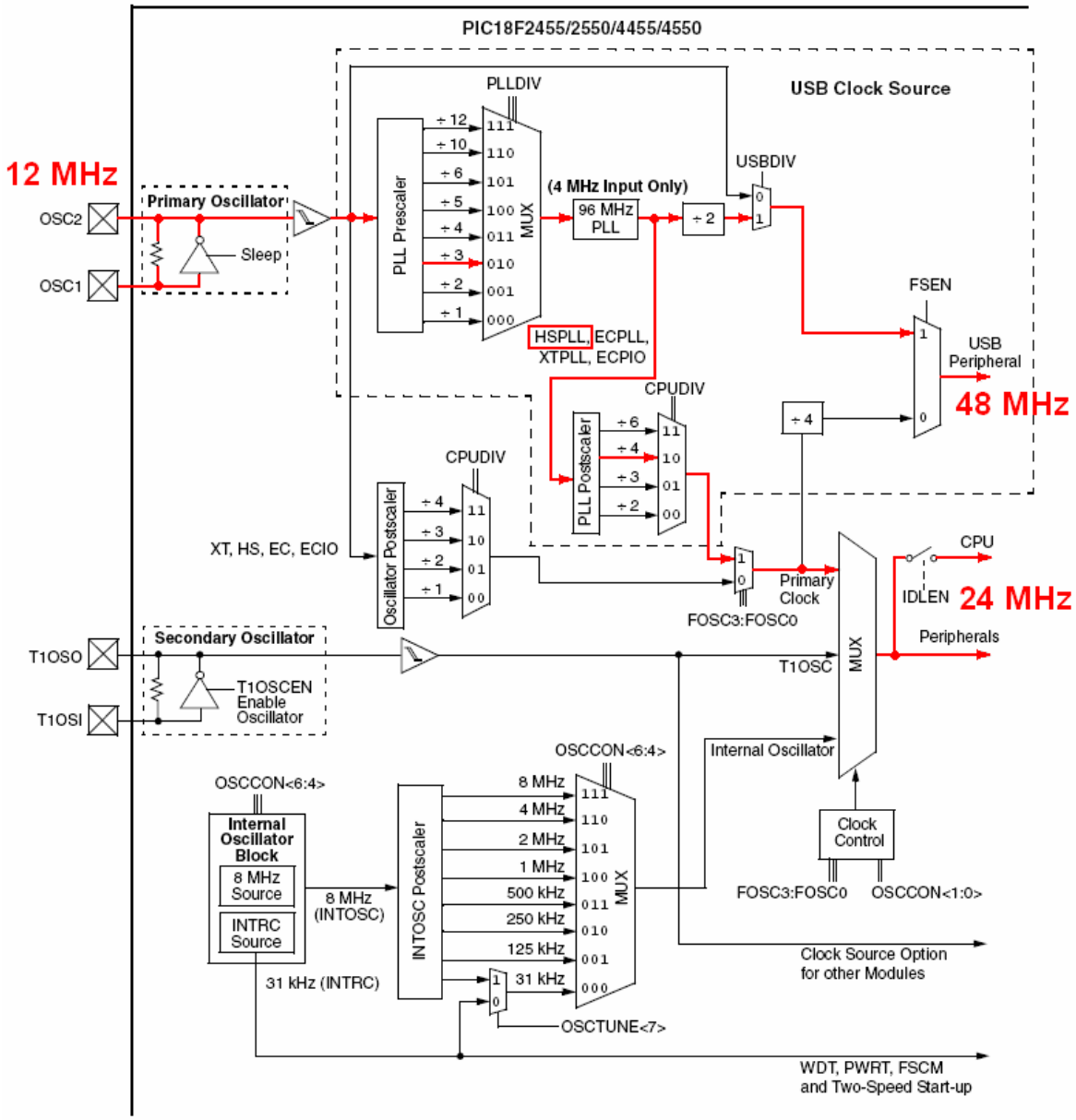


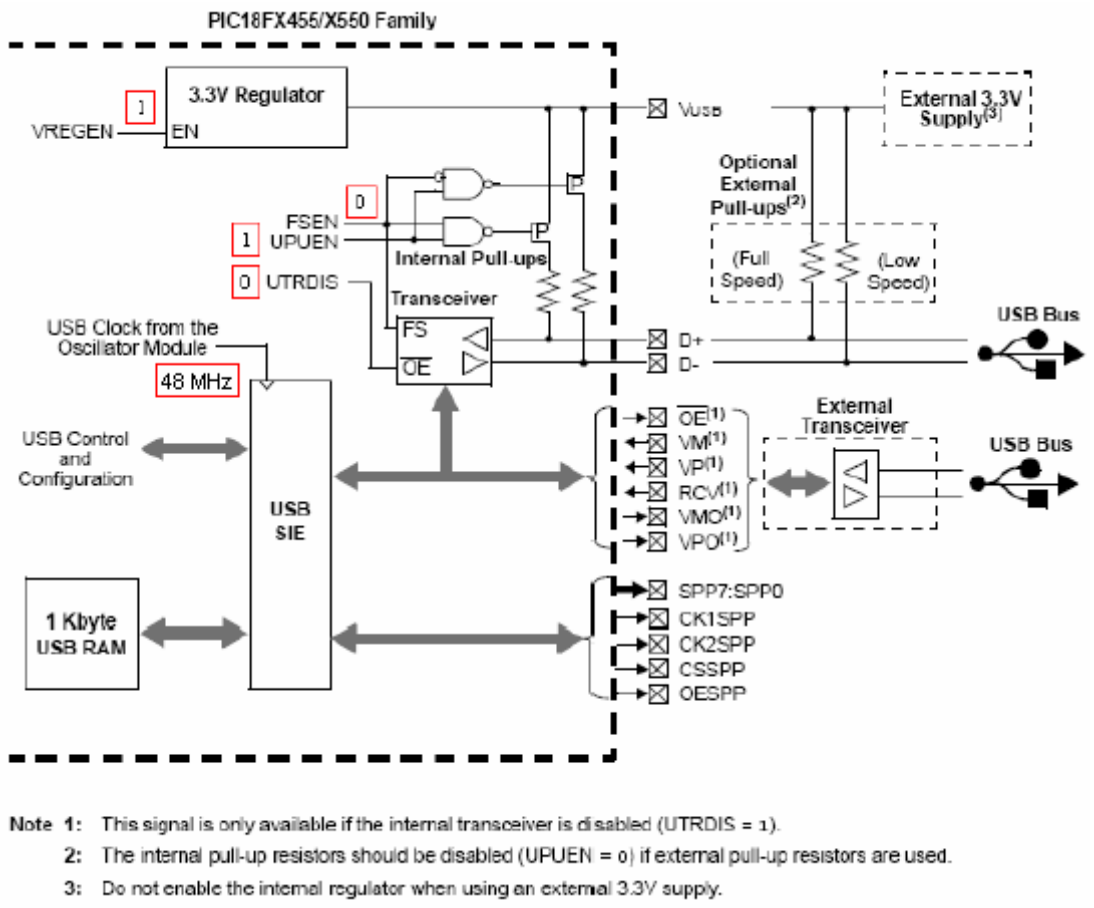
Figura 2.1 Módulo del oscilador.

En la figura anterior observamos los valores necesarios para conseguir las frecuencias deseadas para el reloj del sistema y para el reloj del USB.

2.4.2 Configuración del módulo USB.

El módulo USB dispone de un regulador interno de 3,3 V para suministrar tensión al transceiver interno. Esto se debe a que USB utiliza 3,3 V para las comunicaciones, por lo tanto, mientras el resto del micro puede utilizar una alimentación de 5 V, el transceiver debe alimentarse de una fuente distinta (Vusb).

El regulador se controla a través del flag VREGEN. Cuando está habilitado el voltaje es visible en el pin Vusb. Si el regulador está deshabilitado, una tensión de 3,3 V debe ser suministrada externamente a través de Vusb. Hay que tener en cuenta que las resistencias de pullup internas únicamente pueden utilizarse si el transceiver interno está activo.



Note 1: This signal is only available if the internal transceiver is disabled (UTRDIS = 1).
2: The internal pull-up resistors should be disabled (UPUEN = 0) if external pull-up resistors are used.
3: Do not enable the internal regulator when using an external 3.3V supply.

Figura 2.2 Regulador de tensión del módulo USB.

2.5 Inicialización del módem GSM.

El módem GSM, para comenzar a funcionar normalmente, necesita ser configurado con una serie de parámetros.

Por defecto, el módem GSM está configurado de la siguiente manera:

- Autobaud

- 8 bits de datos
- 1 bit de stop
- Sin paridad

En nuestro caso solo hará falta modificar el parámetro *autobaud* fijándolo a 9600 bps.

Se necesitará introducir el código de acceso PIN antes de poder realizar alguna operación con el módem.

En el apartado 1.6.3 se detallan los comandos usados por el programa del PIC para la inicialización y normal funcionamiento del circuito.

Los comandos AT propios de la inicialización son:

- AT+IPR=9600
- AT+CPIN=<pin>
- AT+CMGF=1
- AT+CNMI=3,1,0,0
- AT+CMGD=1

3 Diseño del Hardware.

3.1 Alimentación del sistema.

Como se ha comentado anteriormente, la alimentación del dispositivo será a través del bus USB, que suministra una tensión regulada de 5 V. En la siguiente figura se muestra el conector USB tipo B utilizado. A continuación, en la tabla, aparece la correspondencia de los pines del conector.

Figura 3.1 Conector USB tipo B.

PIN	Nombre Señal	Color Cable
1	Vcc	Rojo
2	Data -	Blanco
3	Data +	Verde
4	GND	Negro

Tabla 3.1 Correspondencia pins conector USB.

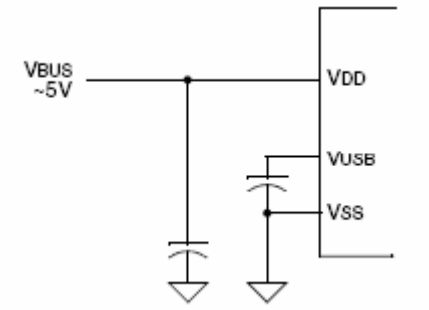


Figura 3.2 Conexionado alimentación USB al PIC.

3.2 Oscilador.

Como se ha comentado anteriormente, los microcontroladores de la familia 18F utilizan dos fuentes de reloj. Para el módulo USB se utiliza el oscilador primario, sin embargo, el resto del micro puede utilizar cualquier otra fuente.

Dependiendo de si utilizamos USB de baja velocidad o de velocidad completa, la señal de reloj del módulo USB deberá ser de 6 MHz o de 48 MHz, respectivamente.

Pese a la poca velocidad necesaria, nuestro dispositivo trabajará a velocidad completa, así que el reloj del módulo USB deberá ser de 48 MHz.

Se ha decidido usar la velocidad completa para tratar al circuito como un dispositivo actual. De esta manera, si fyera necesario manejar un gran volumen de datos a gran velocidad, solo sería necesario modificar el código de los programas firmware y software.

En la figura siguiente se muestran los modos posibles de funcionamiento del oscilador del microcontrolador PIC:

1.	XT	Crystal/Resonator
2.	XTPLL	Crystal/Resonator with PLL enabled
3.	HS	High-Speed Crystal/Resonator
4.	HSPLL	High-Speed Crystal/Resonator with PLL enabled
5.	EC	External Clock with Fosc/4 output
6.	ECIO	External Clock with I/O on RA6
7.	ECPLL	External Clock with PLL enabled and Fosc/4 output on RA6
8.	ECPIO	External Clock with PLL enabled, I/O on RA6
9.	INTHS	Internal Oscillator used as microcontroller clock source, HS Oscillator used as USB clock source
10.	INTXT	Internal Oscillator used as microcontroller clock source, XT Oscillator used as USB clock source
11.	INTIO	Internal Oscillator used as microcontroller clock source, EC Oscillator used as USB clock source, digital I/O on RA6
12.	INTCKO	Internal Oscillator used as microcontroller clock source, EC Oscillator used as USB clock source, Fosc/4 output on RA6

Figura 3.3 Modos de funcionamiento del oscilador.

De los doce modos de funcionamiento del oscilador, sólo los ocho primeros pueden utilizarse para USB y de estos se ha optado por HSPLL, con un cristal de 12 MHz. Necesitaremos el PLL para obtener, de los 12MHz del cristal oscilador, los 24MHz del reloj del sistema. Al utilizar esta frecuencia el módulo USB tendrá un reloj de 48 MHz, utilizando únicamente el oscilador primario.

En el apartado 2.4.1 se ha detallado como se han de configurar los flags para la obtención de las frecuencias de 24 y 48 MHz.

Según el datasheet del microcontrolador, figura siguiente, los valores de los condensadores que acompañan al cristal deben ser de entre 15 y 22 pF. Se instalan unos condensadores de 15 pF.

Osc Type	Crystal Freq	Typical Capacitor Values Tested:	
		C1	C2
XT	4 MHz	27 pF	27 pF
HS	4 MHz	27 pF	27 pF
	8 MHz	22 pF	22 pF
	20 MHz	15 pF	15 pF

Tabla 3.2 Capacidades para el oscilador.

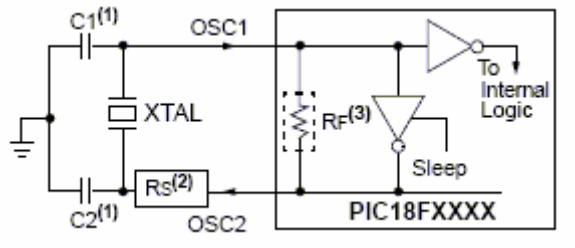


Figura 3.4 Conexión cristal oscilador.

3.3 Pulsadores.

El circuito utiliza tres pulsadores, uno pequeño para generar un RESET en el sistema y dos pulsadores más grandes para realizar el envío de mensajes SMS, uno para cada uno de los dos destinatarios configurables posibles.

Figura 3.5 Pulsador de envío de SMS.

El esquema de conexionado del pulsador de RESET es el típico para cualquier microcontrolador PIC:

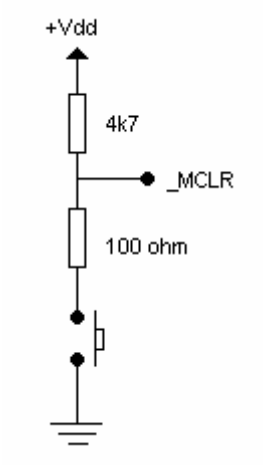


Figura 3.6 Conexión pulsador RESET.

Los pulsadores para el envío de mensajes SMS se han configurado de tal manera que se detecte la pulsación con un nivel bajo a la entrada de las patas RB3 y RB2.

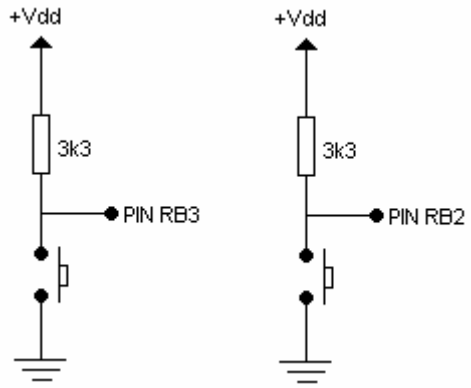


Figura 3.7 Conexión pulsadores envío de SMS.

3.4 Diodos electroluminiscentes, LEDs.

La conexión de los dos LEDs de que dispone el circuito, se ha realizado a las patas RB7, RB6, RB5 y RB4.

El ánodo correspondiente al color verde del LED 1 se ha conectado a la pata RB7 y el correspondiente al color rojo a la pata RB6.

En el LED 2 se ha conectado el ánodo correspondiente al color verde a la pata RB4 y el correspondiente al color rojo a la pata RB5.

El LED 1 nos indicará el estado de la comunicación con el MODEM, mientras que el LED 2 nos indicará el estado de la conexión USB con el PC.

4 Diseño del Firmware.

4.1 Diagramas del sistema.

Los diagramas de flujo son representaciones del flujo que siguen las instrucciones de un programa y nos resultarán útiles para seguir la secuencia lógica de procesos complejos, intercambiar entre estructuras de control, redireccionar el flujo, cambiar un programa de un lenguaje a otro y esbozar fácilmente un procedimiento complejo.

4.1.1 Diagrama general.

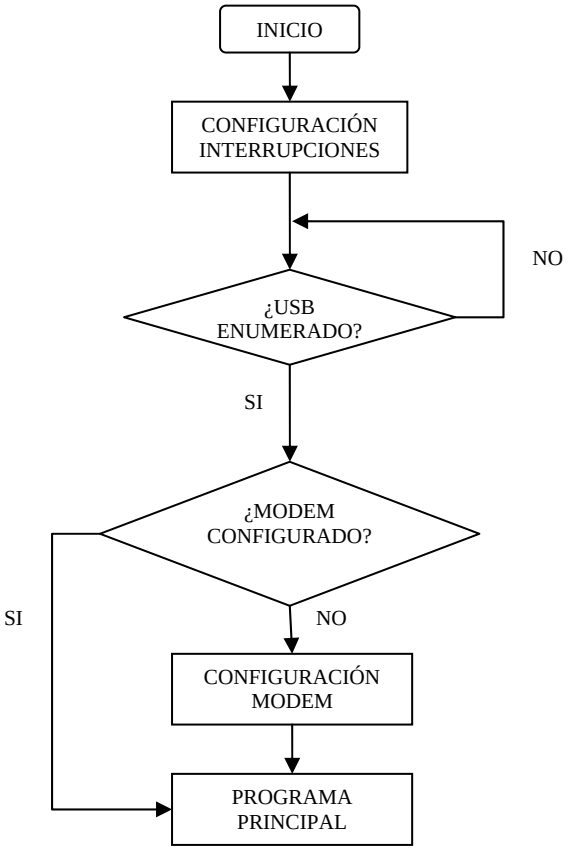


Figura 4.1 Diagrama general de funcionamiento.

4.1.2 Diagramas de manejo de datos

4.1.2.1 Diagrama de lectura de datos en memoria del PIC.

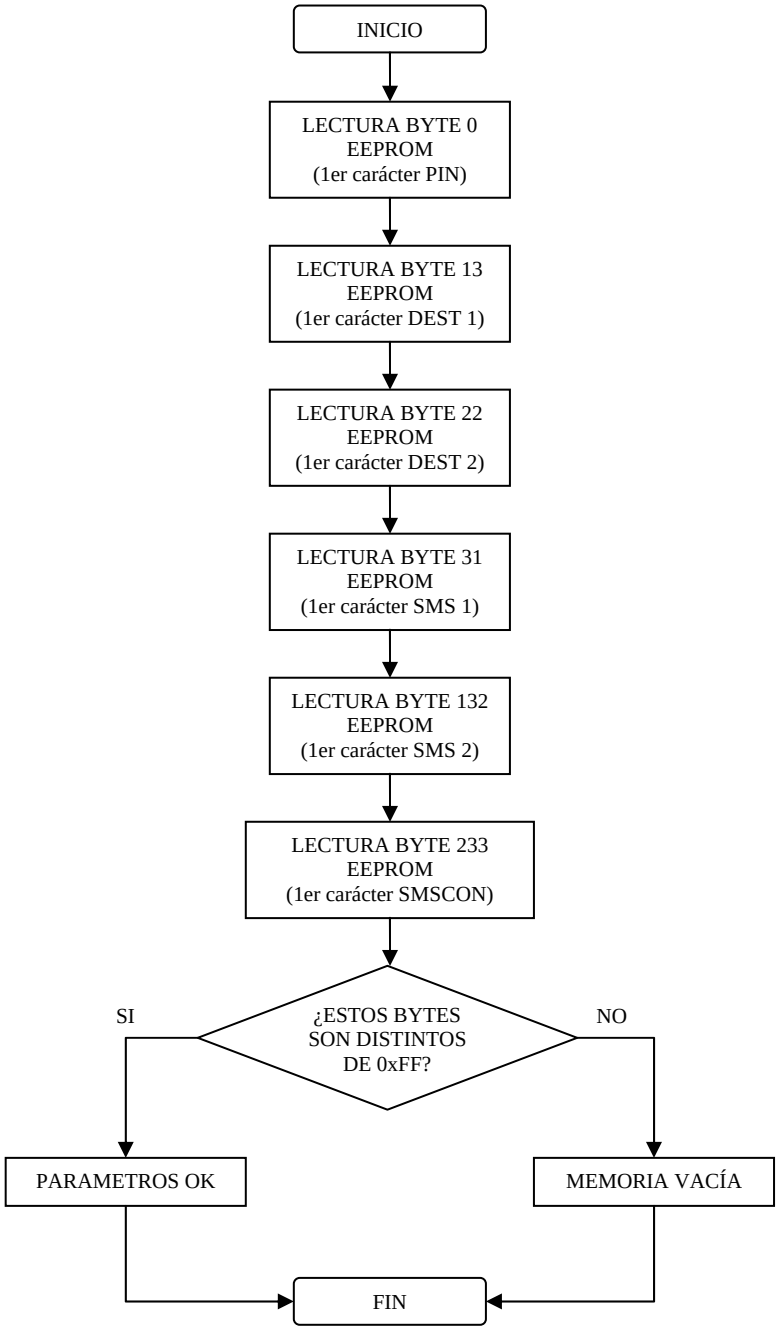


Figura 4.2 Diagrama lectura EEPROM.

4.1.2.2 Diagrama de lectura de datos enviados desde el PC.

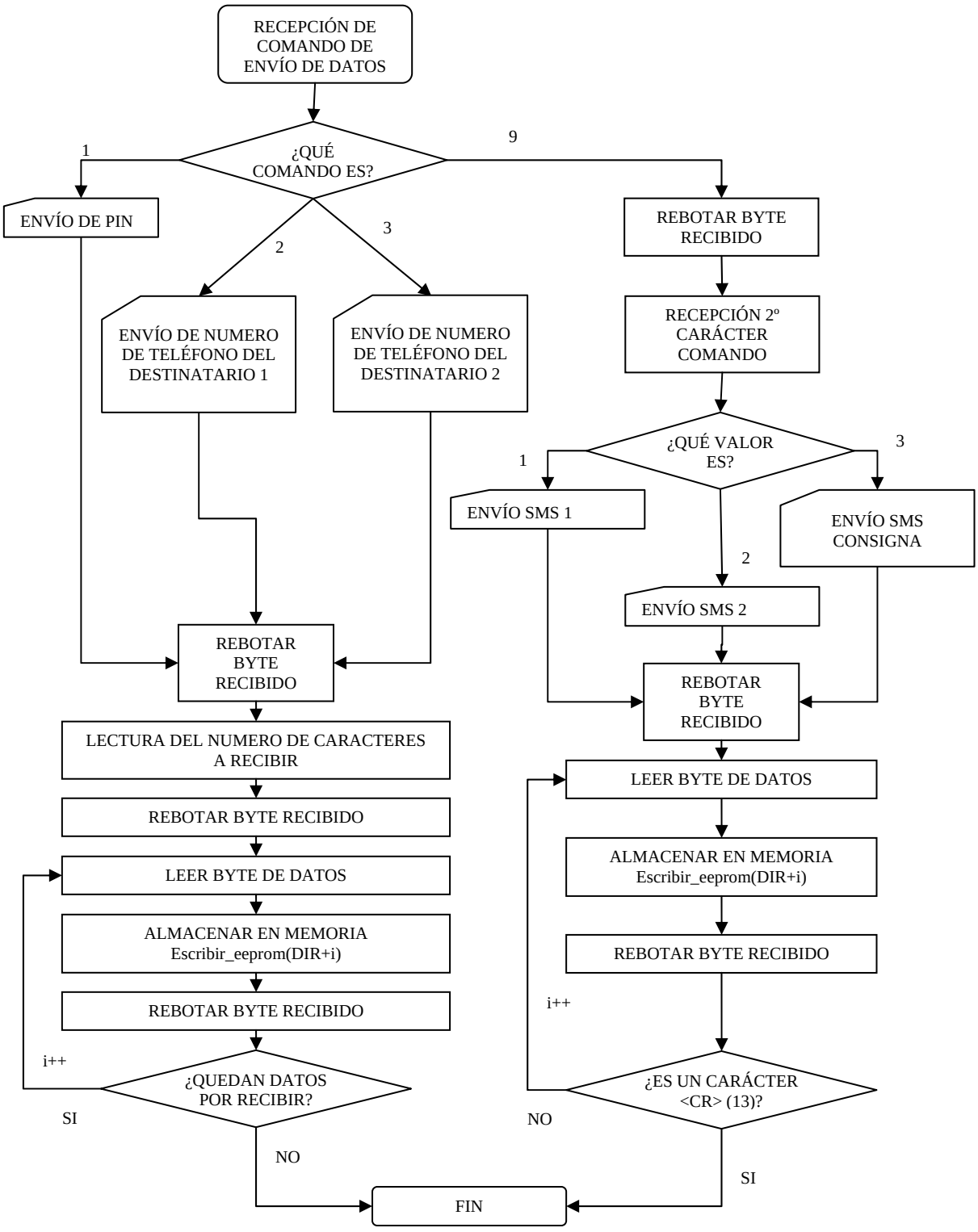


Figura 4.3 Diagrama de comando de envío de datos desde el PC.

4.1.2.3 Diagrama de envío de datos hacia el PC.

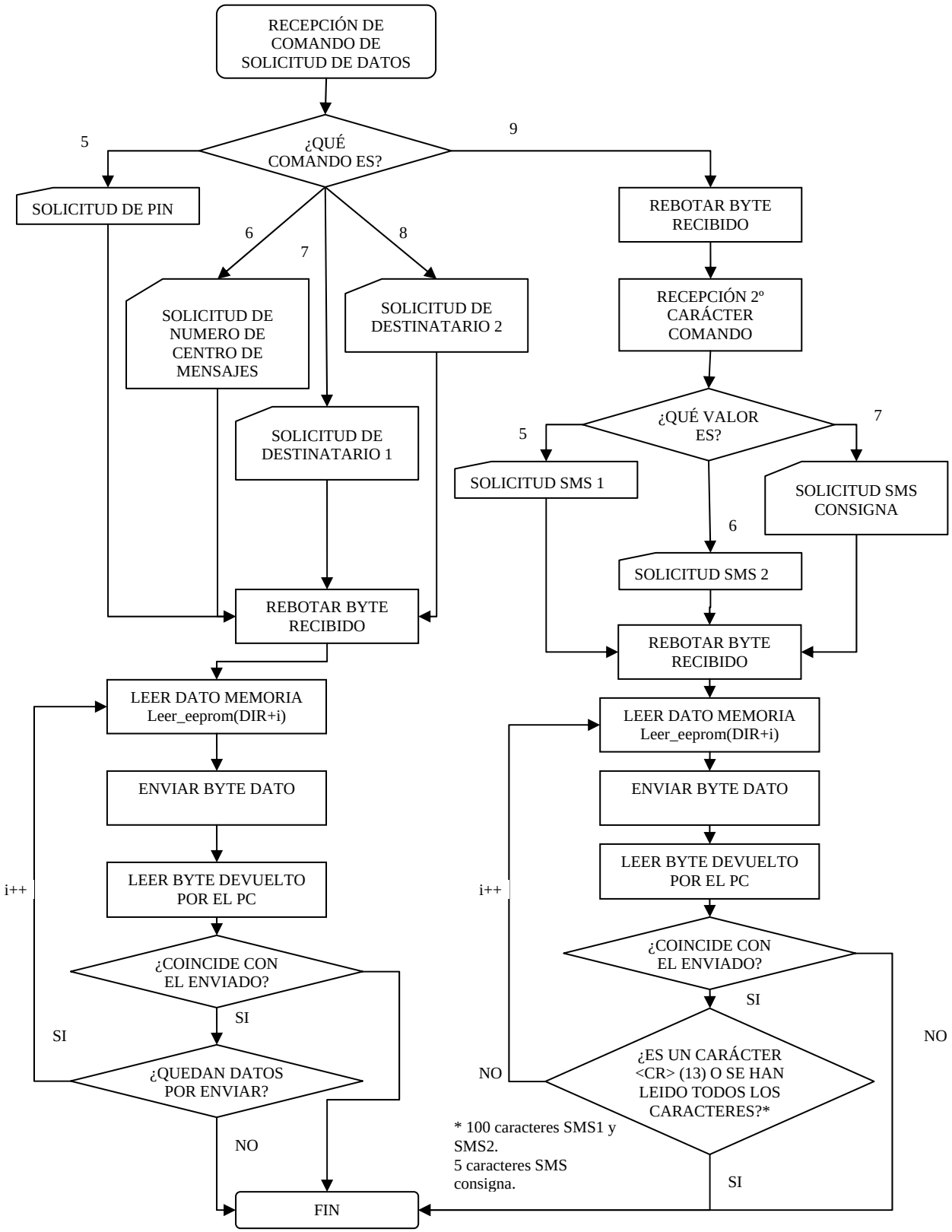


Figura 4.4 Diagrama de comando de petición de datos.

4.1.3 Diagramas de comunicación con módem.

4.1.3.1 Diagrama de configuración del módem.

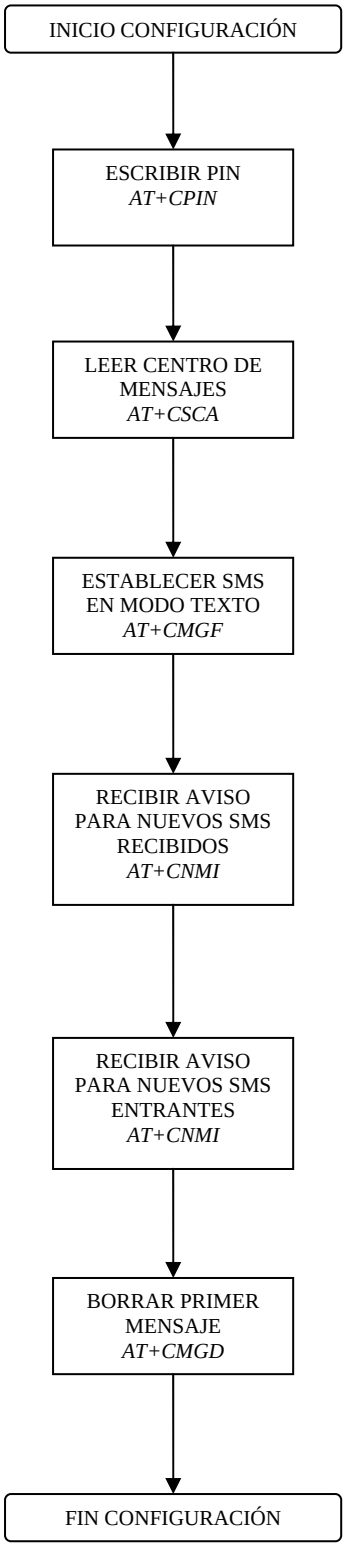


Figura 4.5 Diagrama de configuración del módem.

4.1.3.1.1 Diagrama de comando AT+IPR.

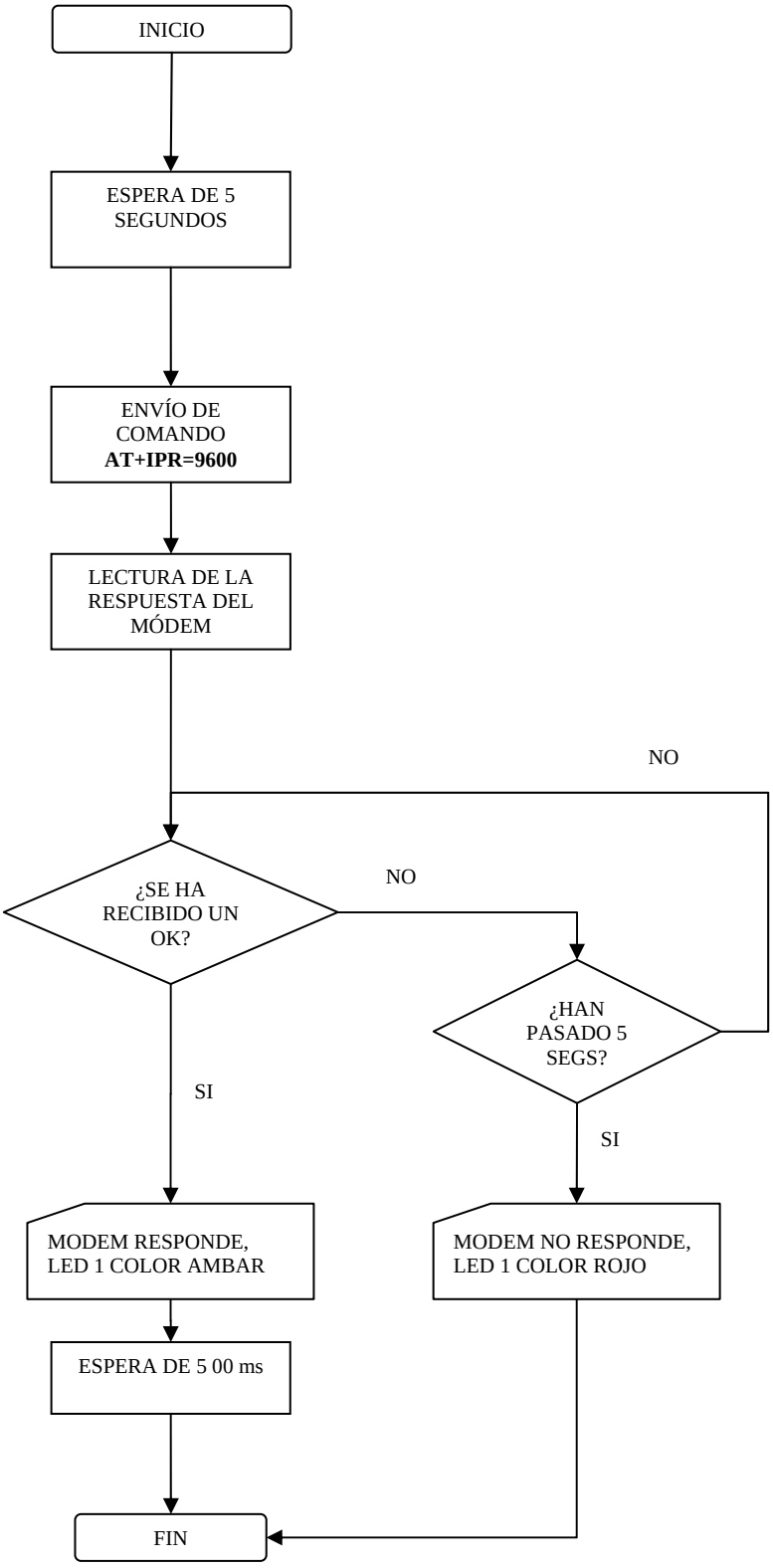


Figura 4.6 Diagrama configuración baudios.

4.1.3.1.2 Diagrama de comando AT+CPIN.

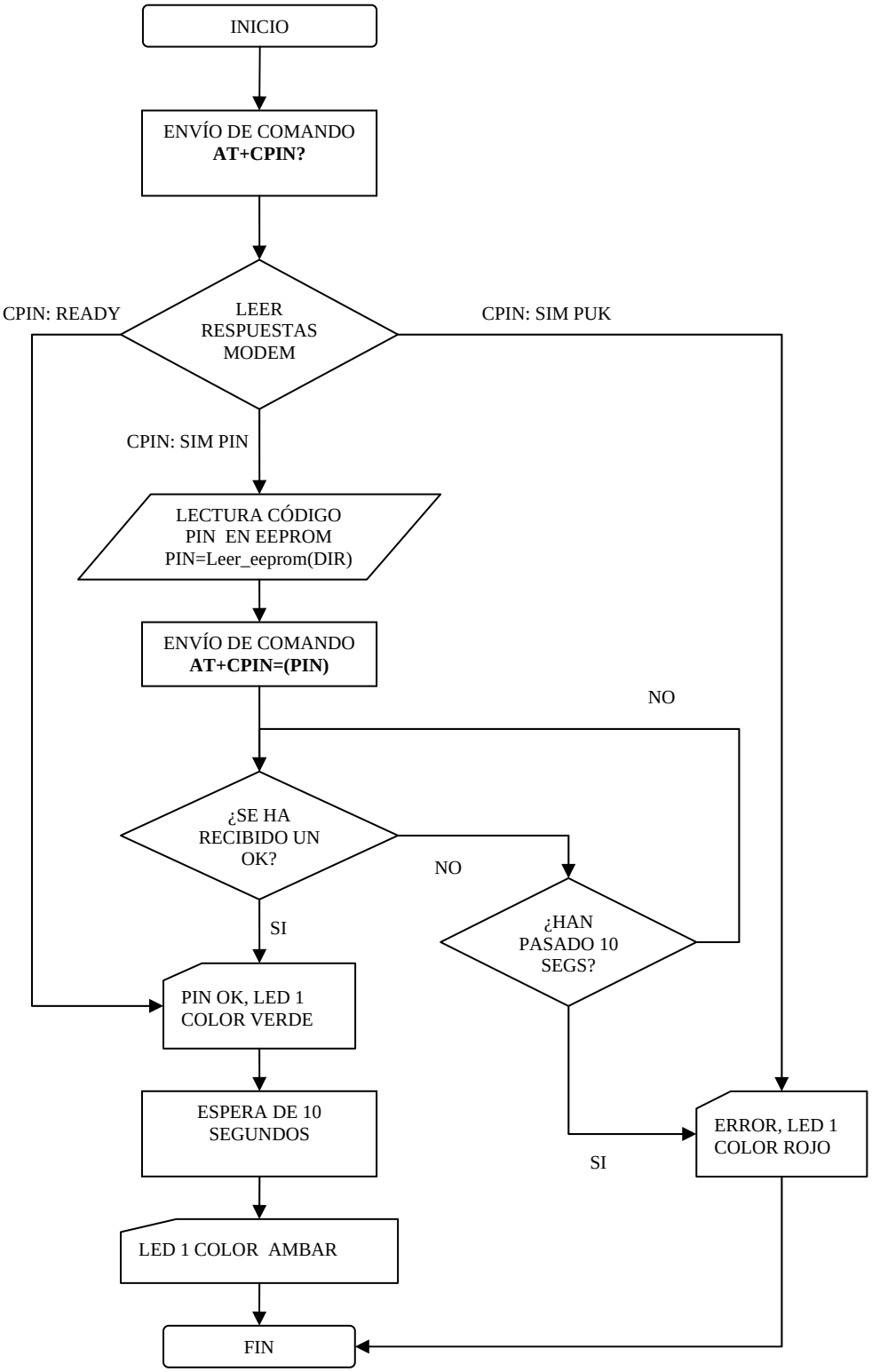


Figura 4.7 Diagrama de introducción del código PIN.

4.1.3.1.3 Diagrama de comando AT+CSCA.

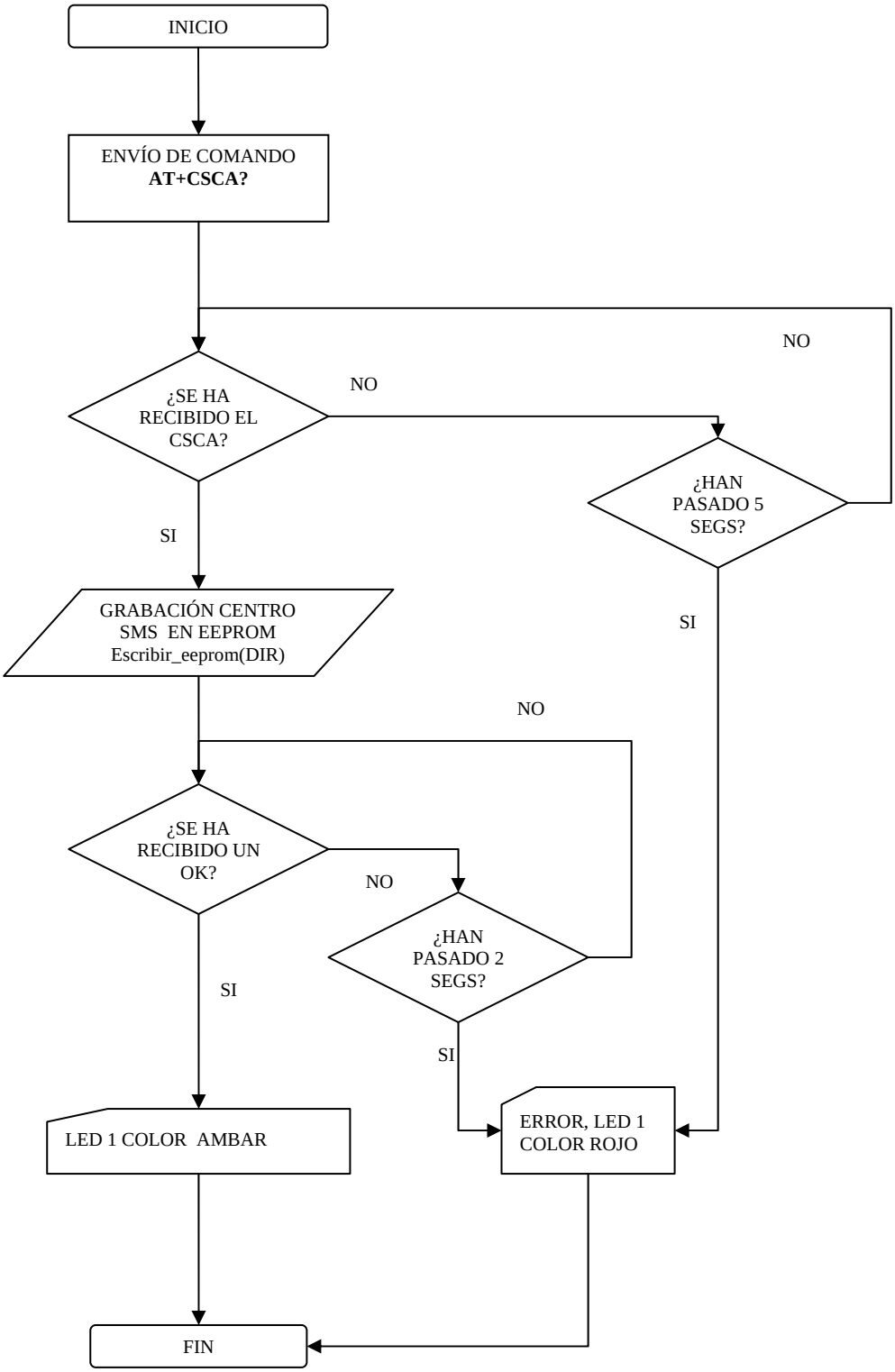


Figura 4.8 Diagrama de lectura del Centro de Mensajes.

4.1.3.1.4 Diagrama de comando AT+CMGF.

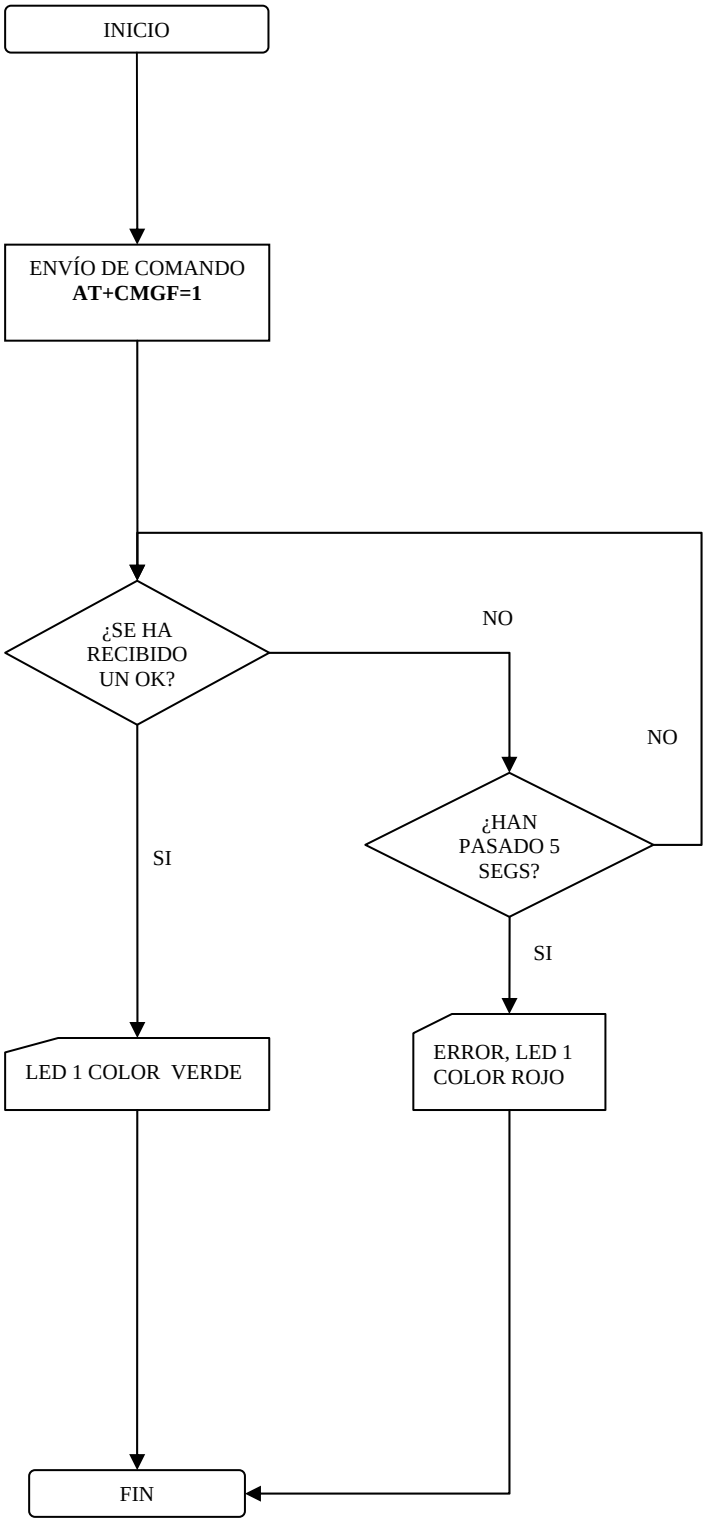


Figura 4.9 Diagrama de configuración SMS en modo texto.

4.1.3.1.5 Diagrama de comando AT+CNMI.

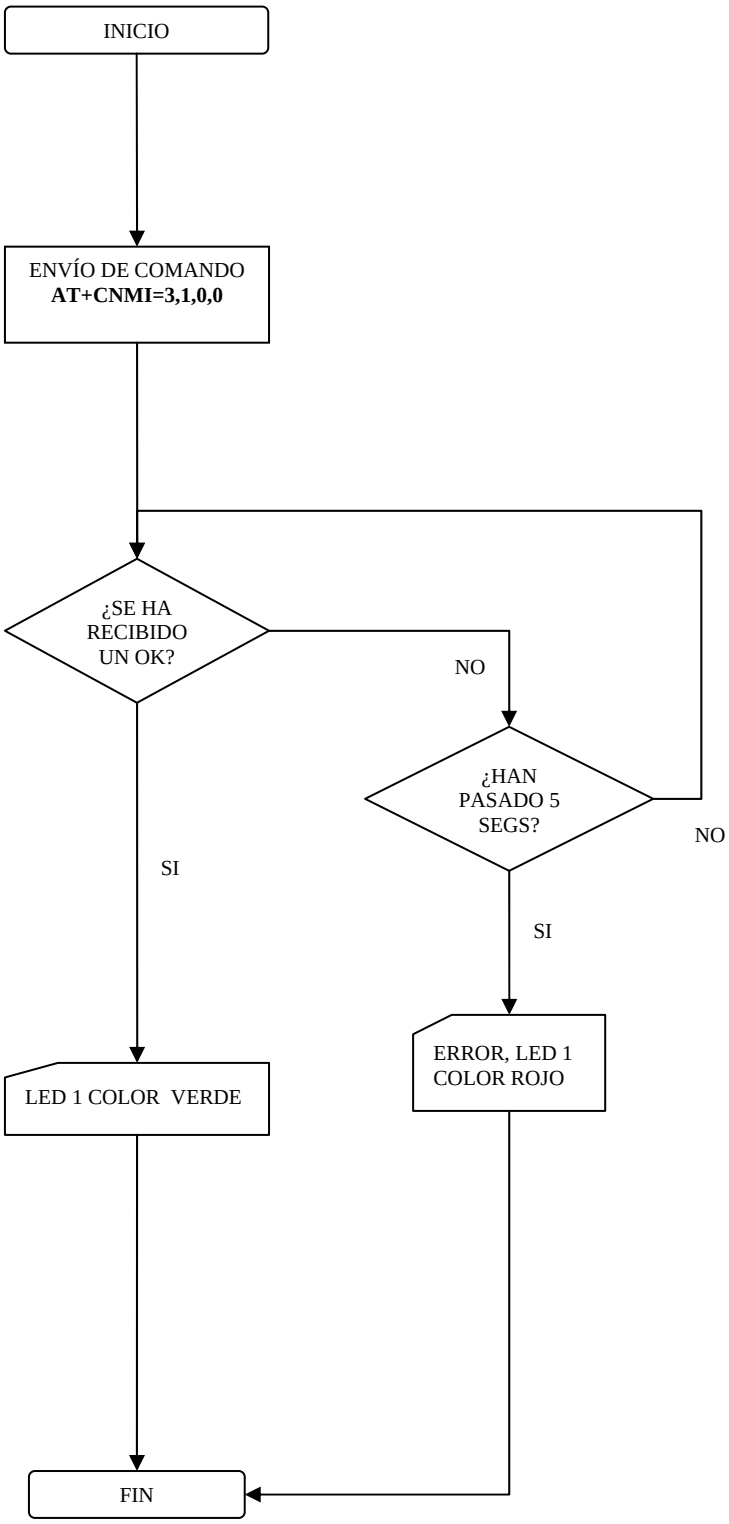


Figura 4.10 Diagrama de configuración aviso para SMS entrantes.

4.1.3.1.6 Diagrama de comando AT+CMGD.

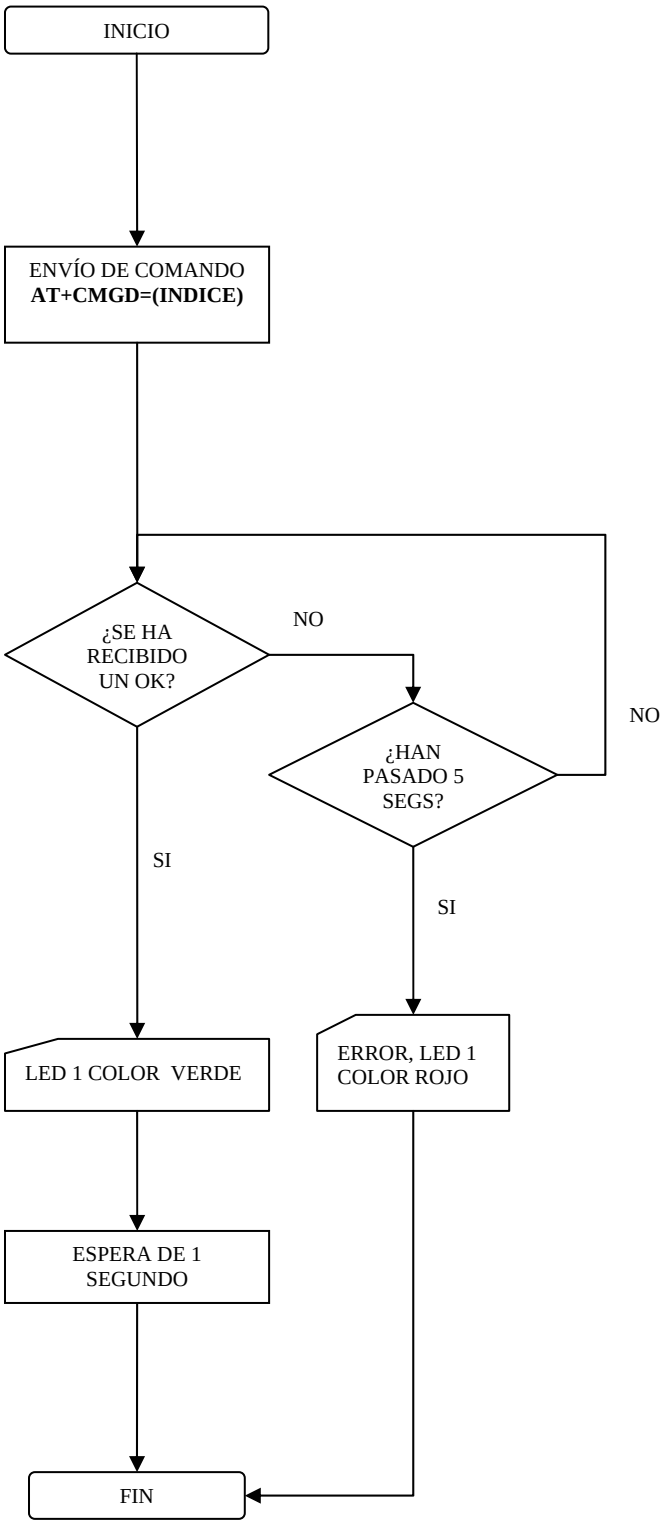


Figura 4.11 Diagrama de borrado de mensajes en memoria.

4.1.3.2 Diagrama de envío de SMS, comando AT+CMGS.

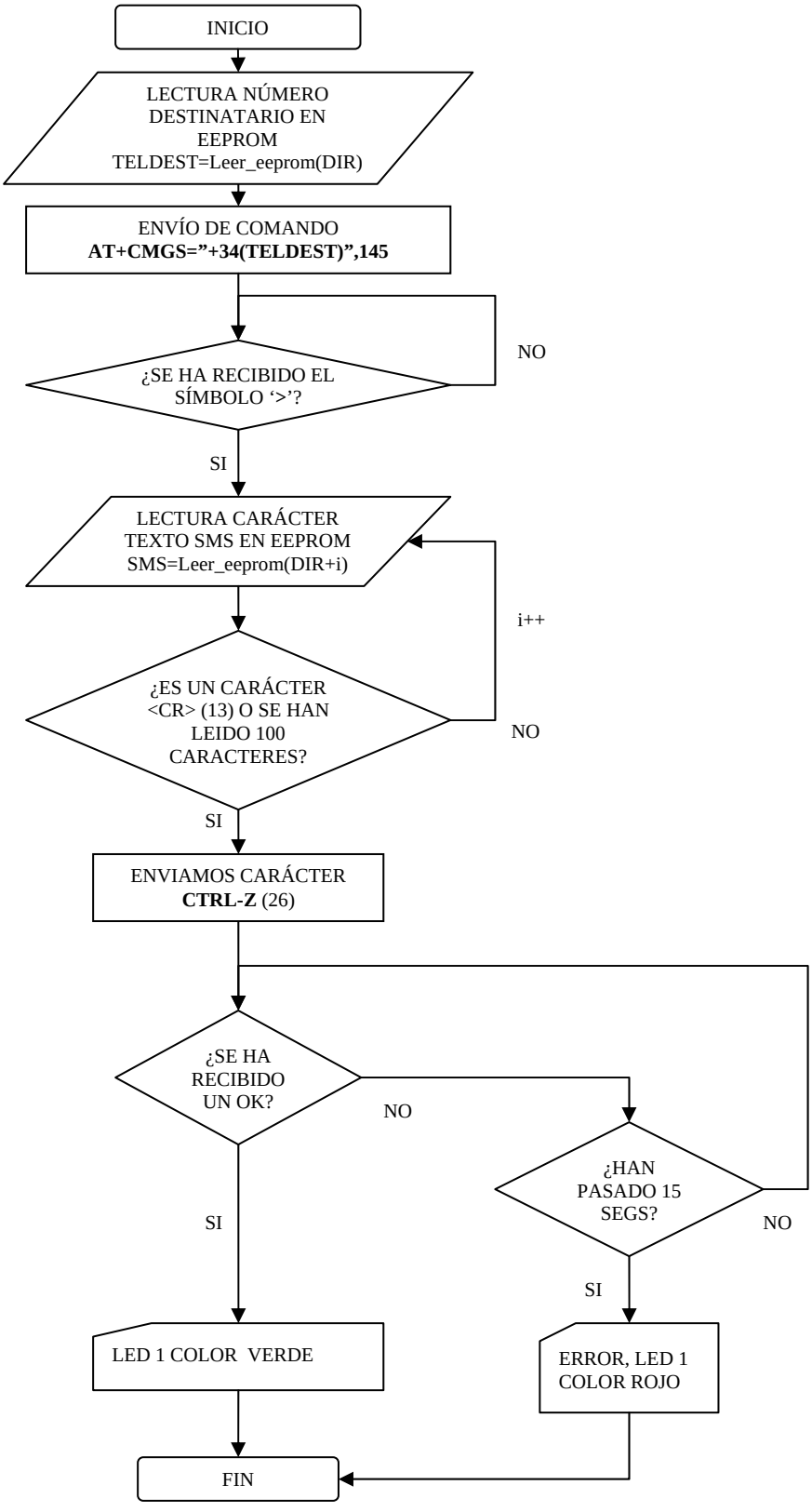


Figura 4.12 Diagrama de envío de mensajes SMS.

4.1.3.3 Diagrama de lectura de SMS, comando AT+CMGR.

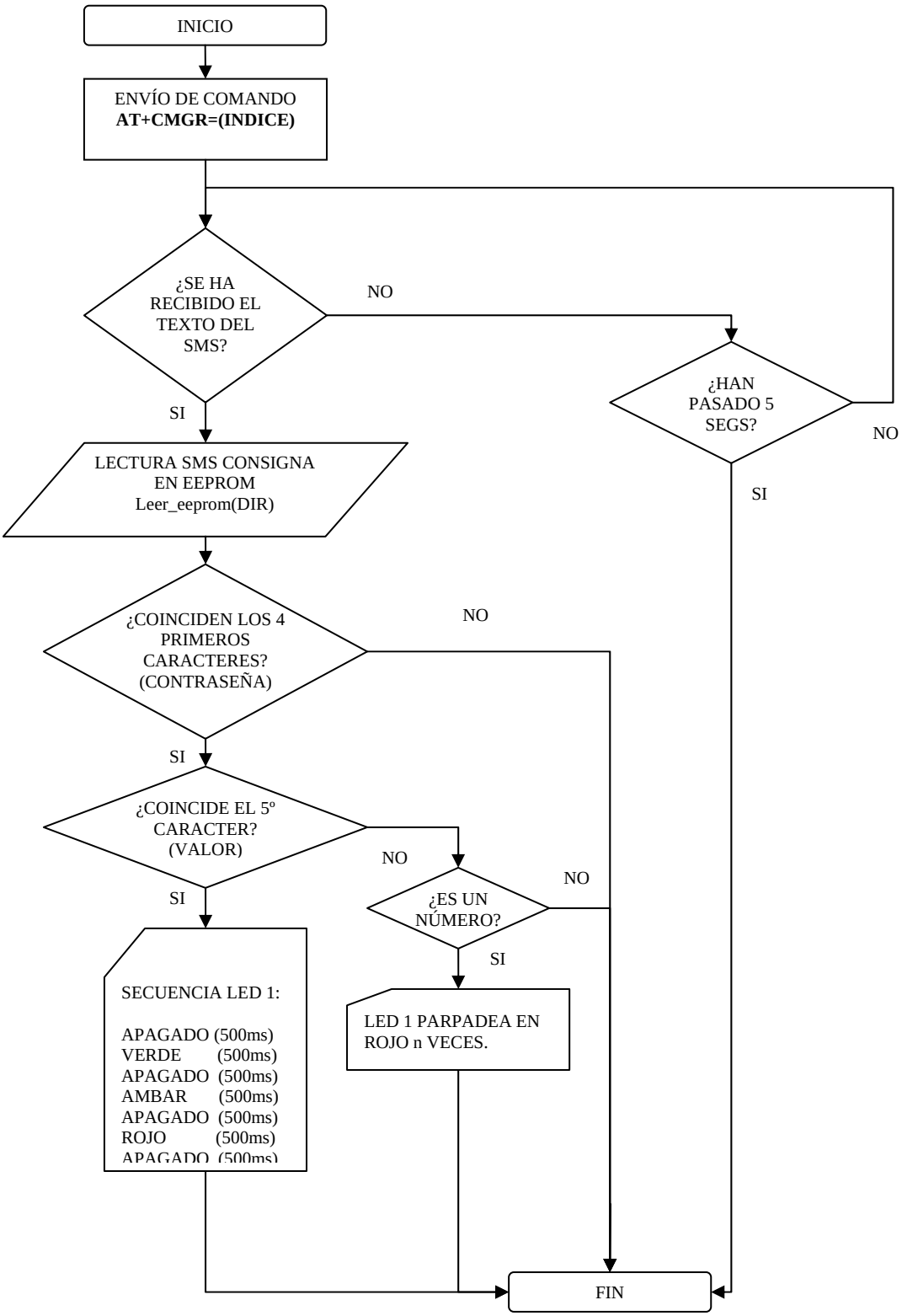


Figura 4.13 Diagrama de lectura de mensajes SMS.

4.2 Programación de la aplicación del PIC.

4.2.1 Compilador y librerías.

El compilador utilizado para crear el programa en el PIC es el PCWH de la compañía CCS.

En este apartado nos centramos en las librerías utilizadas para la correcta creación del programa.

Las librerías del programa del PIC son las siguientes:

- ***18F2550.h*** : Librería con las direcciones de los pines del PIC 18F2550 y la definición de constantes esenciales para programar este microcontrolador.
- ***pic18_usb.h*** : Librería que incluye las funciones y constantes necesarias para la comunicación USB con los microcontroladores de la familia 18Fxx5x.
- ***PicModem.h*** : Librería con la configuración del dispositivo USB y sus descriptores que dan identidad personal al producto. Es el responsable de que salga el nombre del producto por pantalla una vez conectado el lector al puerto USB.
- ***usb.c*** : Otra librería con más funciones generales sobre la comunicación USB, esta librería además contiene la librería usb.h.

4.2.2 Funciones de la comunicación USB.

Ahora detallamos las funciones necesarias para la comunicación USB entre el PIC y el PC.

Cada dispositivo USB está compuesto por unos puntos finales independientes y una dirección única asignada por el sistema en tiempo de conexión, de forma dinámica. A su vez, cada punto final dispone de un identificador único dentro del dispositivo (número de endpoint) que viene asignado de fábrica, además de una determinada orientación del flujo de datos. Cada punto final es por si solo una conexión simple, que soporta un flujo de datos de entrada o de salida.

Un canal USB es una conexión lógica entre un punto final del periférico y el software del host, que permite intercambiar datos entre ellos. El canal que esta formado por el punto final 0 se denomina canal de control por defecto. Este canal está siempre disponible una vez se ha conectado el dispositivo y ha recibido un reset del bus. El resto de canales aparecen después de que se configure el dispositivo. El canal de control por defecto es utilizado por el software USB del sistema para obtener la identificación y para configurar al periférico.

De los diferentes endpoints que hay para establecer la comunicación, se utiliza siempre el endpoint0 (EP0) porque es el de control. Y para la transferencia de datos entre el dispositivo y el host usaremos el endpoint 1, que es bidireccional y tiene tres modos de comunicación, el “bulk”, por interrupción o isócrona. En nuestro caso, el modo de comunicación más conveniente es el “bulk”, que nos permite hacer transferencias de paquetes de datos de diferentes tamaños de una manera rápida.

```
#define USB_HID_DEVICE FALSE //deshabilitamos directivas HID

#define USB_EP1_TX_ENABLE USB_ENABLE_BULK
//activa el EP1(EndPoint1) para transferencias de entrada bulk/interrupt

#define USB_EP1_RX_ENABLE USB_ENABLE_BULK
//activa el EP1(EndPoint1) para transferencias de salida bulk/interrupt

#define USB_EP1_TX_SIZE 1
//tamaño de información para reservar en el buffer de envío de datos del
endpoint 1(1 byte)

#define USB_EP1_RX_SIZE 1
//tamaño de información para reservar en el buffer de recibir datos del
endpoint 1 (1 byte)
```

Las funciones para la inicialización del dispositivo USB con el PC son las siguientes:

- **`usb_init();`** Inicializa el stack USB, la periferia del USB y conecta la unidad al bus USB. También habilita las interrupciones, en general, inicializa el USB.
- **`usb_wait_for_enumeration();`** Se mantiene en un bucle infinito hasta que el dispositivo es enumerado.
- **`usb_enumerated();`** Esta función devuelve TRUE si el dispositivo ha sido enumerado (configurado) por el HOST, y FALSE si no ha sido así. Se recomienda no hacer uso de las funciones del USB hasta que no se haya enumerado y configurado el dispositivo.
- **`usb_kbhit(1);`** Devuelve TRUE si la salida (OUT) del endpoint contiene datos del HOST, el 1 significa que se refiere al endpoint 1 (EP1).

Las funciones para el envío y recepción de datos en el PC son las siguientes:

- **`usb_get_packet(1, buffer, 1);`** Se coge el paquete de tamaño 1 byte del EP1 y se almacena en buffer.
- **`usb_put_packet(1, buffer, 1, USB_DTS_TOGGLE);`** Se envía el paquete de tamaño 1 byte del EP1 almacenado en la variable buffer.

5 Diseño del Software.

5.1 Programación del software para el PC.

5.1.1 Compilador y librerías.

En este caso, el compilador utilizado para crear el programa en el PC es el Visual C# 2008 Express Edition, como se ha comentado con anterioridad.

Las librerías utilizadas han sido las propias del compilador, aunque para programar la comunicación USB, se han hecho servir funciones proporcionadas por MICROCHIP que vienen en la libreria *"mpusbapi.dll"*. Las funciones utilizadas en el programa del PC se pueden ver en el siguiente apartado, y el resto de las funciones y contenido de este archivo se puede ver en los anexos.

5.1.2 Funciones comunicación USB.

En el fichero del API llamado *"PicmodemAPI.cs"*, hay que definir el VID (Vendor Identification) y el PID (Product Identification):

```
#region Definición de los Strings: EndPoint y VID_PID
string vid_pid_norm = "vid_04d8&pid_0008";     //pid_0008 por '08
```

Existen una serie de números para uso personal, pero para realizar una aplicación real, los implementadores del USB te proporcionan unos valores únicos para tu producto, después de pasar una serie de pruebas para garantizar el correcto funcionamiento del dispositivo USB, y así tu poder llevar el logo USB que garantice al consumidor que el producto ha sido probado por "USB-IF Inc" ("USB Implementers Forum Inc.").

Figura 5.1 Logotipos USB.

El periférico que se va a desarrollar no pertenece a ninguna clase genérica, así que habrá que indicar el driver que se le quiere asignar. El driver que se utilizará será un driver USB de propósito general para Windows suministrado por Microchip (*mchpusb.sys*). Por otro lado, Microchip proporciona un identificador de vendedor (0x04D8) que puede ser utilizado en productos desarrollados con microcontroladores PIC. El identificador de producto se ha definido como 0x0008, pero podría haberse elegido otro valor.

Se ha de definir el *endpoint* por el que van a pasar los datos, que en este caso será el EP1:

```
string out_pipe = "\\MCHP_EP1"; //de salida
string in_pipe = "\\MCHP_EP1"; //de entrada
#endregion
```

Una vez definida la ruta de paso de datos e identificado el producto, habrá que importar las funciones que necesitaremos para el proyecto. Estas funciones se importan de la librería *mpusbapi.dll* y son las siguientes:

```
#region
[DllImport("mpusbapi.dll")]

private static extern DWORD _MPUSBGetDLLVersion();
[DllImport("mpusbapi.dll")]

private static extern DWORD _MPUSBGetDeviceCount(string pVID_PID);
[DllImport("mpusbapi.dll")]

private static extern void* _MPUSBOpen(DWORD instance, string
pVID_PID, string pEP, DWORD dwDir, DWORD dwReserved);
[DllImport("mpusbapi.dll")]

private static extern DWORD _MPUSBRead(void* handle, void* pData,
DWORD dwLen, DWORD* pLength, DWORD dwMilliseconds);
[DllImport("mpusbapi.dll")]

private static extern DWORD _MPUSBWrite(void* handle, void* pData,
DWORD dwLen, DWORD* pLength, DWORD dwMilliseconds);
[DllImport("mpusbapi.dll")]

private static extern DWORD _MPUSBReadInt(void* handle, DWORD*
pData, DWORD dwLen, DWORD* pLength, DWORD dwMilliseconds);
[DllImport("mpusbapi.dll")]

private static extern bool _MPUSBClose(void* handle);
#endregion
```

Estas funciones son las que se usarán luego en el programa para crear la comunicación USB entre el PC y el microcontrolador. Estas funciones sirven para abrir o cerrar los canales de entrada o salida y para enviar o recibir paquetes de datos.

Son las siguientes:

```
void* myOutPipe;
void* myInPipe;

public void OpenPipes()
{
      DWORD selection = 0;
      myOutPipe = _MPUSBOpen(selection, vid_pid_norm, out_pipe, 0, 0);
      myInPipe = _MPUSBOpen(selection, vid_pid_norm, in_pipe, 1, 0);
}
```

```
public void ClosePipes()
{
      _MPUSBClose(myOutPipe);
      _MPUSBClose(myInPipe);
}

private void SendPacket(byte* SendData, DWORD SendLength)
{
      uint SendDelay = 20;
      DWORD SentDataLength;
      OpenPipes();
      _MPUSBWrite(myOutPipe, (void*)SendData, SendLength,
      &SentDataLength, SendDelay);
      ClosePipes();
}

private void ReceivePacket(byte* ReceiveData, DWORD* ReceiveLength)
{
      uint ReceiveDelay = 20;
      DWORD ExpectedReceiveLength = *ReceiveLength;
      OpenPipes();
      _MPUSBRead(myInPipe, (void*)ReceiveData, ExpectedReceiveLength,
      ReceiveLength, ReceiveDelay);
      ClosePipes();
}
```

Existe otra función "`_MPUSBGetDeviceCount(vid_04d8&pid_0008)`" que nos indica si el dispositivo con ese VID y PID está conectado al host USB.

Para aplicar las funciones anteriores se crea una función para el envío y otra para la recepción de datos.

La función de enviar es la siguiente:

```
public void enviaUSB(uint valor)
{
      byte* send_buf = stackalloc byte[1];
      send_buf[1] = (byte)valor;
      SendPacket(send_buf, 1);
}
```

Para leer el puerto USB, utilizamos la siguiente función:

```
public uint recibeUSB()
{
      uint rx=0;
      byte* receive_buf = stackalloc byte[1];

      DWORD RecvLength = 1;

      ReceivePacket(receive_buf, &RecvLength);
      rx = (uint)receive_buf[0];
      return rx;
}
```

5.1.3 Instalación software y drivers.

Cuando el sistema operativo detecta un nuevo periférico USB, intenta determinar qué driver debe utilizar y cargarlo. Esto lo realiza el Administrador de Dispositivos, que utiliza instaladores de clase y dispositivo y archivos INF para encontrar el driver adecuado para cada dispositivo. El Administrador de Dispositivos además de ser el responsable de instalar, configurar y desinstalar dispositivos, también se encarga de añadir información sobre cada dispositivo en el registro del sistema, el cual contiene información sobre el hardware y el software instalados.

Los instaladores de clase y dispositivo son DLL's. Windows tiene una serie de instaladores por defecto que el Administrador de Dispositivos utiliza para localizar y cargar los drivers para dispositivos pertenecientes a clases soportadas por el sistema operativo. El archivo INF es un fichero de texto que contiene información que ayuda a Windows a identificar el dispositivo. Este archivo indica al sistema operativo qué drivers debe utilizar y qué información debe almacenar en el registro.

El dispositivo desarrollado no pertenece a una de las clases USB por defecto que soporta el sistema operativo así que tanto el instalador de clase como el archivo INF habrá que crearlos.

Cuando se conecta el dispositivo al ordenador, Windows muestra el mensaje *Nuevo Hardware Encontrado*. Si el periférico nunca ha sido enumerado en el sistema, el sistema operativo necesita localizar un driver para él. Al ser la primera vez que se conecta el dispositivo y no ser de una clase genérica, Windows no encontrará un archivo INF que pueda utilizar para asignarle un driver y lanzará el *Asistente para Agregar Nuevo Hardware* donde se puede indicar la localización del archivo INF necesario para la instalación del periférico. Una vez localizado, se carga el driver indicado, se añade el periférico en el Administrador de Dispositivos y se indica al usuario que la instalación del dispositivo ha finalizado de forma correcta. En el Administrador de Dispositivos se puede ver también la descripción del dispositivo y el fabricante, obtenidos del archivo INF.

El ejecutable, además de instalar el software, copia los drivers en sus correspondientes directorios. Se copia el fichero INF (*picmodem.inf*) en el directorio *\windows\inf*, el driver (*mchpusb.sys*) en *\windows\system32\drivers* y el instalador de clase (*mpusbci.dll*) en *\windows\system32*. De esta manera, la próxima vez que se conecte el dispositivo, el ordenador lo reconocerá sin necesidad de indicar ningún dato.

La información en un archivo INF está dividida en secciones. Cada sección puede ser identificada porque su nombre está entre corchetes. Las principales secciones son:

- **Versión:** Especifica el sistema operativo para el que está destinado el fichero INF. Indica el nombre de la clase de los dispositivos instalados con este fichero así como un identificador de clase que es utilizado por el registro del sistema para identificar a los dispositivos pertenecientes a esta clase.

- **ClassInstall**: En esta sección se indica si se ha de instalar una nueva clase. El Administrador de Dispositivos únicamente procesa la sección si una clase de dispositivo aún no está instalada en el sistema. Aquí será donde indicaremos el icono, el nombre y la descripción de la clase así como el

nombre del archivo DLL (*picmodemci.dll*) que instalará la nueva clase en el Administrador de Dispositivos.

- **Manufacturer**: Contiene el identificador de vendedor y producto (*USB\VID_04D8&PID_0008*). Cuando el Administrador de Dispositivo encuentra una coincidencia entre estos valores y los identificadores obtenidos del dispositivo durante la enumeración, sabe que ha encontrado el archivo INF correcto.

- **DestinationDirs**: Indica el directorio o directorios donde se copiarán los ficheros.

- **Strings**: Define las cadenas referenciadas en las otras secciones. El contenido del fichero INF creado (*picmodem.inf*) puede verse en el anexo.

5.1.4 Software versión SUPERVISOR.

El software creado inicialmente muestra por pantalla todos los datos que se transmiten por el puerto RS-232 y por el USB, tanto los enviados como los recibidos. Esto ha servido para la depuración del firmware y puede ser útil para observar lo que el módem GSM nos está comunicando en todo momento.

El programa consta de una sola ventana donde están todos los controles necesarios para la comunicación entre el dispositivo y el PC. Se muestra en la barra inferior si el dispositivo se encuentra conectado “ONLINE” o desconectado “OFFLINE”.

En la región superior izquierda hay tres cuadros de texto en los que se reflejan los bytes enviados por USB desde el PC al PIC, los bytes recibidos desde el PIC y un cuadro que muestra mensajes informativos sobre el proceso. Hay además un cuadro de texto en el que se pueden escribir comandos para enviarlos al PIC o al módem directamente.

En la región superior derecha hay un cuadro de texto en el que se pueden leer los comandos AT enviados desde el PIC como las respuestas enviadas desde el módem, a través del puerto RS-232.

En la región inferior aparecen todos los cuadros de texto de parámetros, tales como código PIN, número de teléfono del Centro de Mensajes y operador, número de teléfono del destinatario 1, número de teléfono del destinatario 2, texto del mensaje SMS 1, texto del mensaje SMS 2 y texto del mensaje de consigna compuesto por una contraseña y un valor determinado.

En el manual de usuario incluido en el anexo se detalla su funcionamiento.

5.1.5 Software versión simple.

Para un uso sencillo y cómodo, se ha simplificado el software eliminando las casillas relativas a comunicaciones. En esta versión únicamente aparecen los cuadros de texto de los parámetros, para visualizar los existentes en el PIC o introducirlos desde el PC.

Esta versión también dispone de la barra de estado inferior.

En el manual de usuario incluido en el anexo se detalla su funcionamiento.

5.2 Creación del archivo instalable.

Mediante el software MEP Installer 2 se genera un fichero *.exe* que contiene los drivers y las dos versiones del software desarrollado, así como la actualización de .NET Framework necesaria para su funcionamiento.

Los drivers los copia en las carpetas correspondientes y el software lo instala por defecto en el directorio *\Archivos de Programa\PicModem* aunque el usuario lo puede instalar donde desee.

Figura 5.2 Pantalla principal instalable.

El proceso de instalación, pese a ser en inglés el software instalador, es muy intuitivo y fácil de completar.

6 Presupuesto.

6.1 Coste de recursos Hardware.

El hardware de desarrollo del sistema consta de un ordenador personal de sobremesa de las siguientes características:

- Procesador AMD Athlon 64 3200+ de doble núcleo.
- Memoria RAM de 2GB.
- Disco duro de 300GB.
- Monitor Samsung LCD de 19”.

Además del PC, se ha hecho uso de un voltímetro digital, el programador TE-20, una estación soldadora y un cable de dos metros USB tipo B para la conexión del circuito al ordenador.

Para calcular el importe proporcional al uso dado, teniendo en cuenta el periodo de amortización de los recursos hardware utilizados, se establece un periodo medio de amortización de 60 meses para cada uno de los ítems.

CONCEPTO	PRECIO	€ / MES	MESES	IMPORTE
PC AMD Athlon 3200+	1500 €	25	6	150 €
Programador TE-20	25 €	0,42	6	2,5 €
Estación de soldar	30 €	0,5	6	3 €
Multímetro digital	30 €	0,5	6	3 €
			TOTAL	158,5 €

Tabla 6.1 Coste recursos hardware.

6.2 Coste de recursos Software.

El software utilizado en el desarrollo del proyecto consta de:

- Sistema operativo: Microsoft Windows XP Pro SP2.
- Compilador de C: PCWH 3.235 de CCS.
- Grabador PIC: WinPic800.
- Software desarrollo aplicaciones: Visual C# 2008 Express Edition.
- Software diseño circuitos: Eagle Professional 4.16
- Software creación ejecutable: MEP Installer 2.1.3

Como sistema operativo se ha utilizado Microsoft Windows XP Professional SP2. La actualización Service Pack 2 es imprescindible para la programación del software.

El software Visual C# 2008 Express Edition, el WinPic800 y el MEP Installer 2.1.3 son gratuitos, o bien ofrecen un periodo de demostración, por lo que su uso no repercute en el presupuesto.

Para calcular el importe proporcional de los componentes software, también se establece un periodo de amortización de 60 meses.

CONCEPTO	PRECIO	€ / MES	MESES	IMPORTE
Microsoft Windows XP Professional SP2	125 €	2,08	6	12,5 €
Eagle Professional 4.16	180 €	3	6	18 €
Compilador PCWH 3.235	335 €	5,58	6	33,5 €
			TOTAL	64,0 €

Tabla 6.2 Coste recursos software.

6.3 Coste de componentes.

A continuación se detalla el precio, en euros, y la cantidad de cada componente utilizado:

CONCEPTO	PRECIO UNIDAD	UDS.	PRECIO TOTAL
Resistencia 0,25W 1kΩ	0,03	2	0,06
Resistencia 0,25W 100Ω	0,03	1	0,03
Resistencia 0,25W 4k7Ω	0,04	1	0,04
Resistencia 0,25W 2k2Ω	0,04	3	0,12
Resistencia 0,25W 3k3Ω	0,04	2	0,08
Condensador electrolítico 47uF	0,18	1	0,18
Condensador electrolítico 4,7uF	0,18	1	0,18
Condensador electrolítico 1uF	0,18	3	0,54
Condensador poliester 100nF	0,11	1	0,11
Condensador cerámico 15pF	0,03	2	0,06
Diodo 1N4148	0,05	2	0,10
Pulsador 6x6mm	0,26	2	0,52

CONCEPTO	PRECIO UNIDAD	UDS.	PRECIO TOTAL
Conector USB tipo B hembra	1,25	1	1,25
Conector RS232 DB9	2,45	2	4,90
CI MAX232	2,50	1	2,50
Diodo Led 5mm tricolor	0,37	2	0,74
Cristal oscilador de cuarzo 12MHz	0,91	1	0,91
Conmutador 2 circuitos	1,35	1	1,35
Zócalo de 28 pines	0,27	1	0,27
Zócalo de 16 pines	0,27	1	0,27
Placa de prototipos	2,85	1	2,85
Microcontrolador PIC 18F2550	5,30	1	5,30
Cable USB A macho-B macho	2,95	1	2,95
Módem GSM Siemens TC35i	82,50	1	82,50
		TOTAL	**107,87 €**

Tabla 6.3 Coste componentes electrónicos.

6.4 Coste de recursos humanos.

Las tareas realizadas son el análisis de requisitos, diseño, implementación y experimentación.

A continuación se muestra el tiempo dedicado a cada una de las tareas y el importe resultante aplicando un precio de 30€/hora:

CONCEPTO	UDS.	IMPORTE
Análisis de requisitos	40	1.200 €
Diseño	150	4.500 €
Implementación	230	6.900 €
Experimentación	60	1.800 €
	TOTAL	**14.400 €**

Tabla 6.4 Coste de recursos humanos.

6.5 Coste total del proyecto.

Los precios mostrados en todo el presupuesto incluyen el 16% de IVA. A continuación se muestra el coste total del proyecto:

CONCEPTO	IMPORTE
Recursos Hardware	158,5 €
Recursos Software	64 €
Componentes	107,87 €
Recursos Humanos	14.400 €
TOTAL	**14.730,37**

Tabla 6.5 Coste total del proyecto.

El coste total del proyecto asciende a CATORCE MIL SETECIENTOS TREINTA euros con TREINTA Y SIETE céntimos.

6.6 Coste de explotación del dispositivo.

Para una posible explotación comercial del producto, se ha realizado un cálculo del precio final para una cantidad hipotética de 1000 unidades. El coste de recursos humanos para el desarrollo de este proyecto de 14.400 €. Para una venta de 1000 unidades, se obtiene un importe de amortización de 14,4€.

El precio correspondiente al montaje realizado por un técnico, es de 8 € / hora. El tiempo que se utilizará en el montaje de cada unidad será de ½ hora. Se realiza un descuento de un 20% en los costes de componentes, por compra en grandes cantidades.

CONCEPTO	IMPORTE
Mano de obra, montaje.	4 €
Componentes	86,30 €
Amortización proyecto	14,40 €
TOTAL PRODUCTO	**104,70**

Tabla 6.6 Coste de explotación del dispositivo.

El precio de venta de cada producto sería de 104,70€, IVA incluido.

7 Bibliografía.

1. Peter Wright, *Beginning Visual C# 2005 Express Edition,* Apress, 1ª edición, 2006.

2. Andrés Cánovas López, *Manual de usuario del compilador PCW de CCS.* 2006.

3. José María Angulo Usategui, Eugenio Martín Cuenca, Ignacio Angulo Martínez, *Microcontroladores PIC, la solución en un chip.* Ed. Thomson, 2000.

4. Adela M. Rodriguez Zaragozá, *PFC – Desarrollo de una estación meteorológica USB.* Universitat de València, ETSE, 2006

5. Página oficial de MICROCHIP, *http://www.microchip.com*, 2008

6. Página oficial de Siemens, especificaciones módem TC35i. *https://pia.khe.siemens.com/index.aspx?nr=14278*, 2008

7. Foro sobre microcontroladores PIC, *http://www.todopic.net/foros*, 2008

8. Página personal de RedRaven, *http://picmania.garcia-cuervo.com*, 2008

9. Página oficial del creador de WinPic800, *http://www.winpic800.com*, 2008

10. Consulta RS-232 en la enciclopedia online Wikipedia, *http://es.wikipedia.org/wiki/RS232*, 2008

11. Consulta USB en la enciclopedia online Wikipedia, *http://es.wikipedia.org/wiki/USB*, 2008

12. Especificaciones de USB 2.0, *http://www.usb.org/developers/docs/*, 2008

13. Publicación electrónica, *http://www.canalpda.com/node/794*, 2008

14. Consulta indice de cobertura GSM en España en la enciclopedia online Wikipedia, *http://es.wikipedia.org/wiki/ESPAÑA*, 2008

8 Anexos.

8.1 Manual de usuario.

8.1.1 Instalación.

8.1.1.1 Instalación de la aplicación y de los drivers.

Para instalar la aplicación hay que ejecutar el programa *instalar.exe*, este ejecutable instalará los drivers en las carpetas de sistema correspondientes y las versiones normal y SUPERVISOR. Por defecto, la instalación se realiza en *\Archivos de Programa\PicModem*, aunque se puede cambiar la ubicación haciendo clic en *Examinar*, seleccionando la nueva carpeta y pinchando en *Aceptar*.

Por último, se pregunta si se desea crear un icono de acceso directo en el escritorio y, a continuación, empieza la instalación del programa. Una vez se han terminado de copiar los archivos, el asistente da la opción de instalar .NET Framework 2.0. Si ya se tiene instalado, no es necesario reinstalarlo, pero si no se tiene, es imprescindible instalarlo para poder ejecutar la aplicación.

8.1.1.2 Instalación del dispositivo.

Al conectar el dispositivo por primera vez, aparecerá el asistente para la instalación de nuevo hardware, que lo primero que hace es preguntar si se quiere conectar a Windows Update. Se marca la opción *No* y se pincha sobre *Siguiente*. En ese momento, comenzarán a copiarse los archivos necesarios en el ordenador. Una vez se termine de copiar, aparecerá una pantalla indicando que la instalación del driver está terminada. El LED 2 del dispositivo se encenderá en color verde, lo que indica que está listo para ser usado.

Si todo ha ido bien, en el Administrador de Dispositivos debe aparecer algo así:

Figura 8.1 Panel Administrador de Dispositivos.

Si abrimos las propiedades del dispositivo, veremos la siguiente ventana:

Figura 8.2 Propiedades del dispositivo.

El resto de veces que se conecte el dispositivo al ordenador, será reconocido de forma automática.

8.1.2 Uso de la aplicación en su versión SUPERVISOR.

Una vez se ejecute el software, si el dispositivo no se encuentra conectado al puerto USB del PC, o está conectado pero no ha sido correctamente enumerado, el software mostrará una ventana indicando que el dispositivo se encuentra OFFLINE.

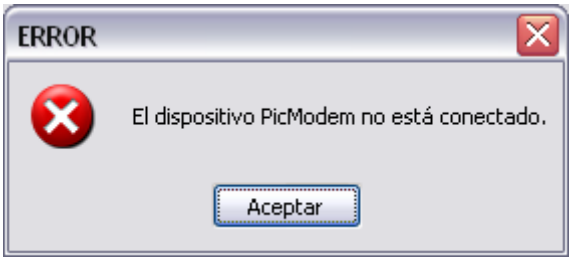


Figura 8.3 Mensaje de error de dispositivo no conectado.

Si el dispositivo no se conecta, el software no permite ninguna comunicación con éste. Para detectar que se ha conectado el dispositivo, el software realiza una consulta cada 500 ms.

En caso de desconectarse el dispositivo una vez conectado con aterioridad, se mostrará la siguiente ventana:

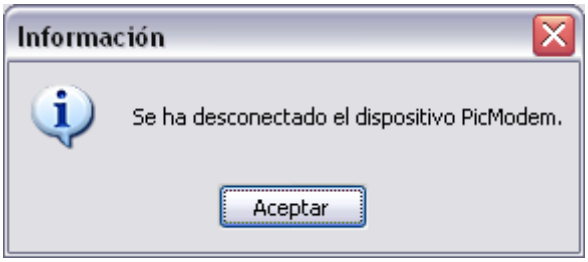


Figura 8.4 Mensaje de error de dispositivo desconectado.

Si el dispositivo se conecta y se enumera correctamente, en la barra de estado inferior se mostrará:

PFC - Control de módulo GSM desde microcontrolador. <<ONLINE>>

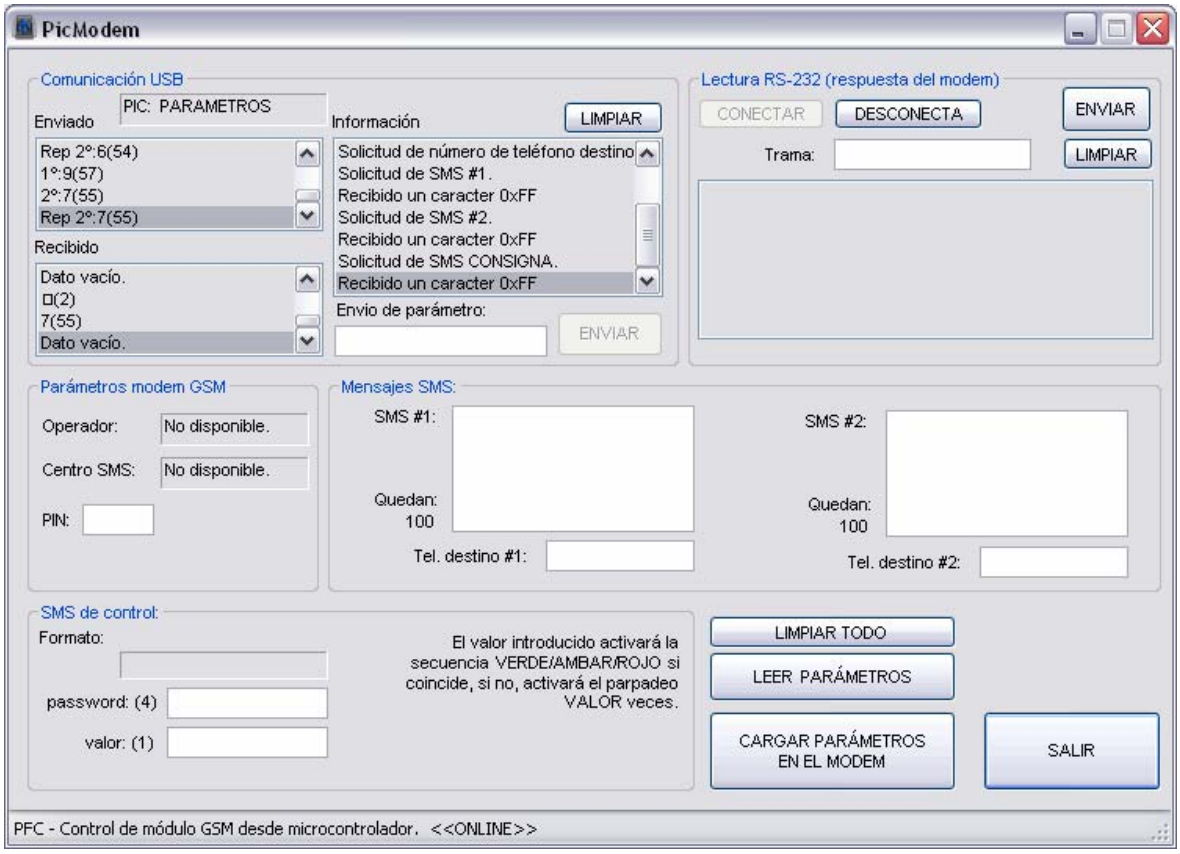


Figura 8.5 Vista del software con el dispositivo ONLINE.

A partir del momento en que se conecte el dispositivo, transcurrirán 5 segundos hasta que se ejecuten los comandos de lectura de datos correspondientes. Después de leer los datos, en la pantalla aparecerá un mensaje indicando qué valores se encuentran en la memoria del PIC y cuáles no.

Figura 8.6 Mensaje información parámetros.

Por lo general, tal y como se ha diseñado el software, el PIC se encontrará con todos o, por el contrario, con ningún dato almacenado, puesto que desde el software no se permite el envío de parámetros si no se han rellenado correctamente todos los campos.

Si existen datos almacenados en el dispositivo, éstos se mostrarán en los cuadros de texto correspondientes, de no ser así, los cuadros permanecerán vacíos.

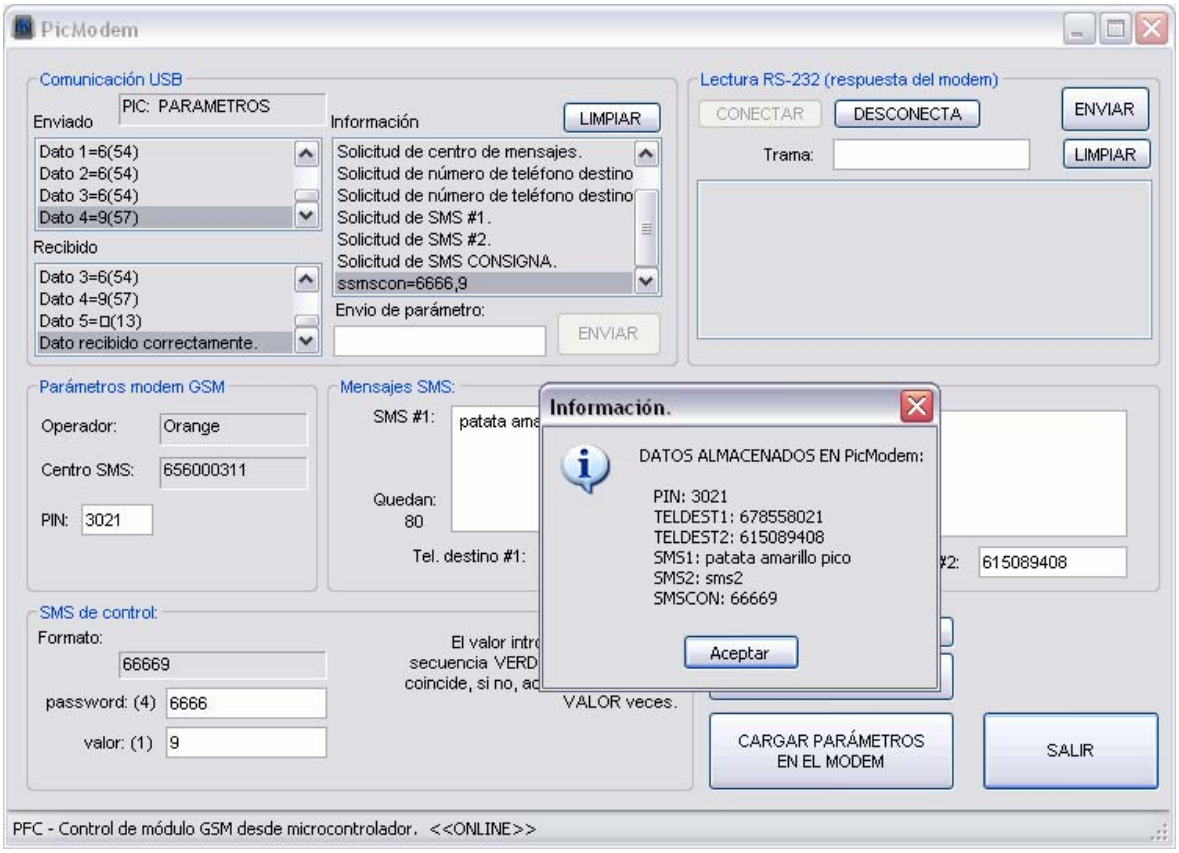


Figura 8.7 Vista del software después de leer los datos del PIC.

En ambos casos, tanto si existen datos almacenados en el dispositivo como si se encuentra vacío, rellenando todos los campos y pulsando el botón CARGAR PARÁMETROS EN EL MÓDEM, el software procederá al envío, uno por uno, de todos los datos escritos en los cuadros de texto.

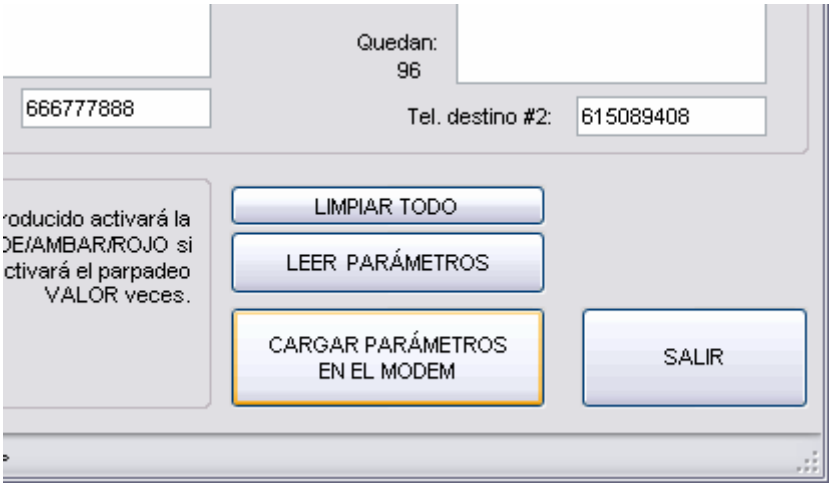


Figura 8.8 Vista del botón CARGAR PARÁMETROS EN EL MODEM.

Los campos se han de rellenar de la siguiente manera:

- **PIN:** Se han de introducir los 4 dígitos correspondientes a la tarjeta SIM que se ha introducido en el módem GSM.
- **SMS #1 y SMS #2:** Se ha de escribir el contenido del mensaje SMS asociado a los destinatarios 1 y 2 respectivamente. El límite es de 100 caracteres.
- **Tel. destino #1 y Tel. destino #2:** Se ha de escribir el número de teléfono de los destinatarios 1 y 2 respectivamente. El límite es de 9 dígitos, y al tratarse de teléfonos móviles, han de comenzar por 6.
- **SMS Consigna:** Para componer el mensaje de consigna, se introducirá una contraseña y un valor determinados.
    - **Password:** La contraseña será de 4 dígitos.
    - **Valor:** Será de 1 dígito, preferiblemente numérico.

Para cerrar la aplicación basta con pulsar ESC, sobre el aspa situado en la parte superior derecha o sobre el botón CERRAR de la ventana.

Existen dos botones LIMPIAR para limpiar las cajas de txto del cuadro donde están insertados y un LIMPIAR TODO que vacía todas las cajas de texto de la ventana.

En la parte superior derecha se encuentra el cuadro de las comunicaciones RS232. En este cuadro tenemos cuatro botones: dos botones para CONECTAR y DESCONECTAR el puerto COM, un tercer botón para ENVIAR la cadena introducida en el cuadro de texto y el cuarto botón de LIMPIAR, comentado en el párrafo anterior.

Para poder transmitir los datos escritos desde el PC hacia el exterior mediante el puerto serie, debemos conectar el cable serie del PC al conector MODEM/PC y se deberá mover el conmutador a la posición PC.

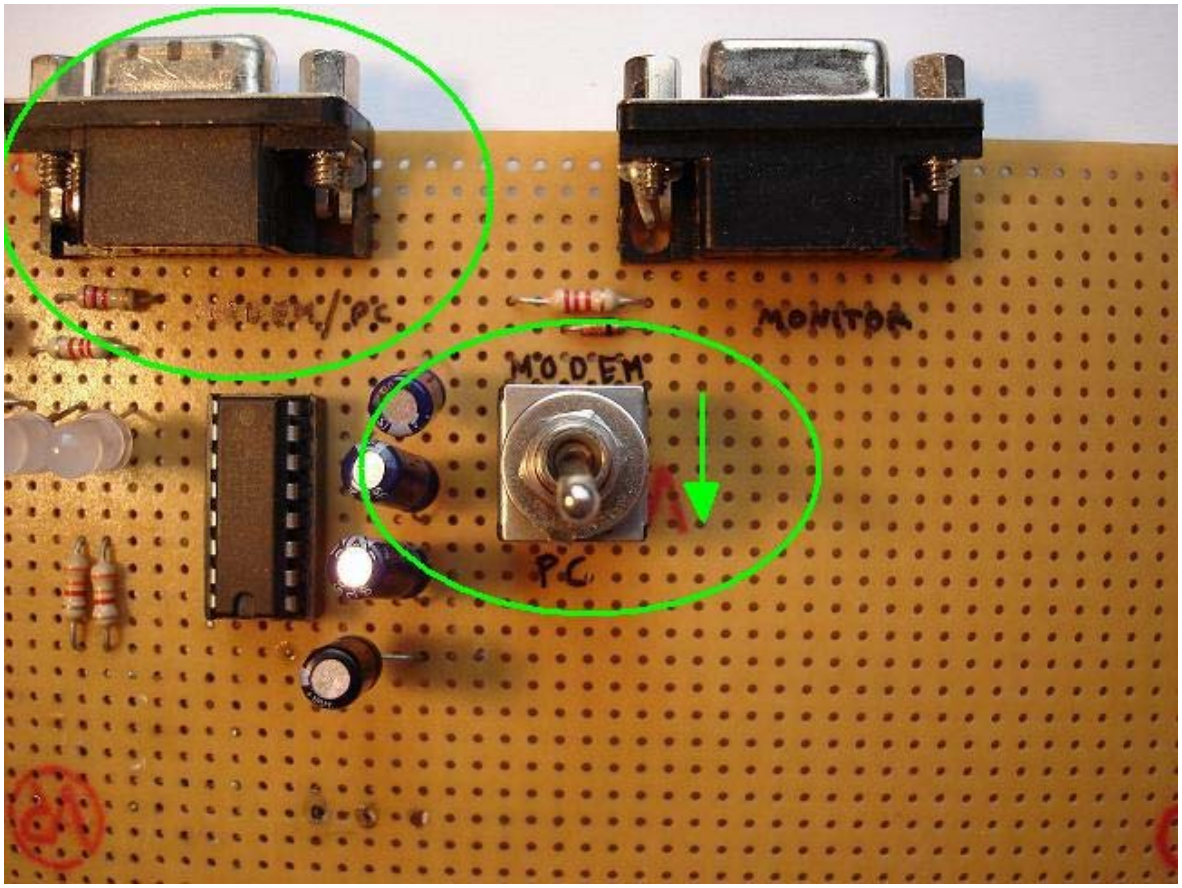


Figura 8.9 Vista del conmutador y del conector para comunicaciones serie desde el PC.

8.1.3 Uso de la aplicación en su versión básica.

El funcionamiento es el mismo que en la versión SUPERVISOR, léase el apartado anterior.

En esta versión básica existe una limitación, no se pueden visualizar las transferencias realizadas y no es posible escribir comandos para enviarlos al dispositivo.

Al no visualizar las comunicaciones entre PIC, módem y PC, el usuario puede tener la sensación que el software no responde. Para que se pueda visualizar el proceso de lectura de los datos desde el PIC, en la barra inferior de estado se ha instalado una barra de progreso.

Figura 8.10 Detalle de la barra de progreso.

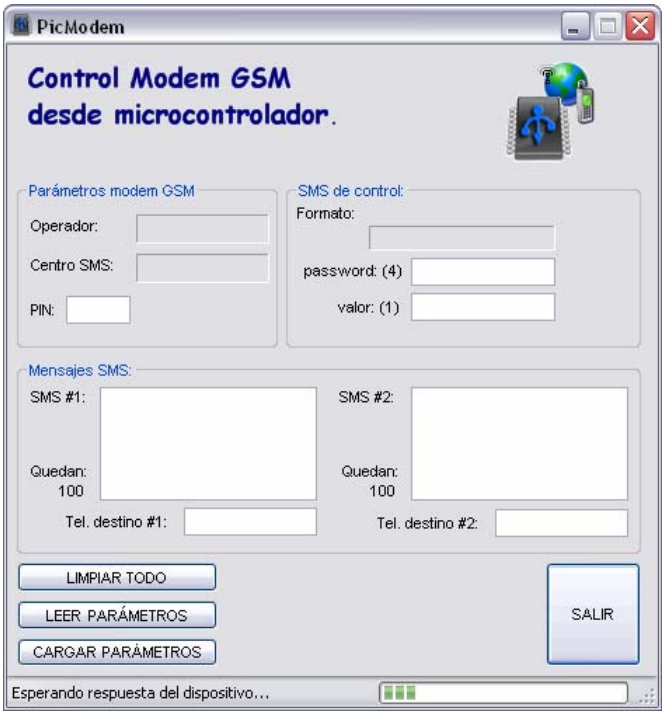


Figura 8.11 Vista del software en proceso de obtención de datos desde el PIC.

Una vez se ha completado la operación, la ventana muestra los datos de los que dispone el microcontrolador. Es ahora cuando el usuario puede cambiar los datos, introduciendo los nuevos valores deseados si fuera el caso.

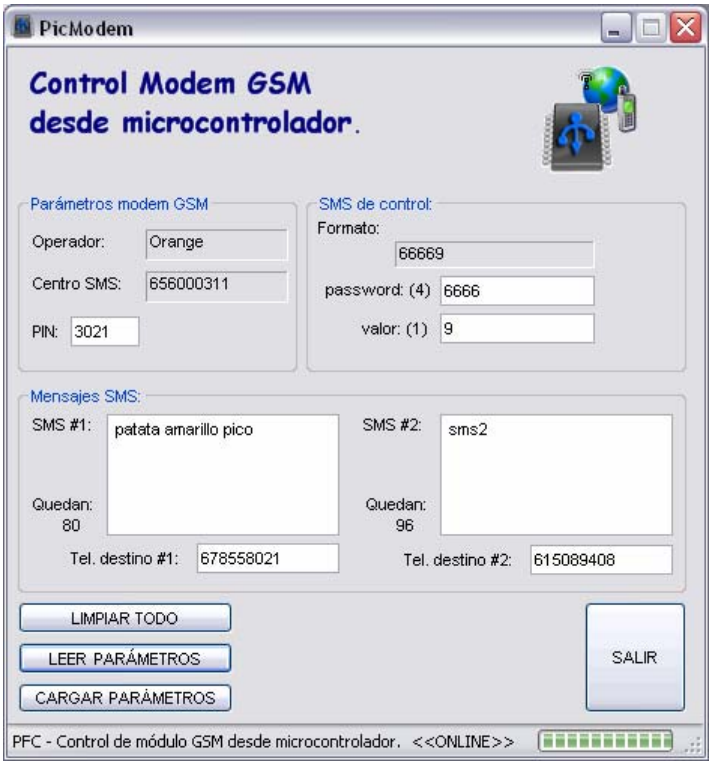


Figura 8.12 Vista del software después de recibir los datos.

8.1.4 Uso del dispositivo.

El dispositivo, una vez conectado y enumerado, encenderá el LED 2 en verde. El LED 1 se encenderá en ámbar mientras el PIC esté ocupado.

Transcurridos cinco segundos desde la conexión del dispositivo, el PIC realizará la lectura de datos de la memoria EEPROM interna. Si no existen los datos, el LED 1 parpadeará en ámbar hasta que no se detecte la existencia de éstos. Para comprobar la existencia de datos, el PIC realiza la lectura de la EEPROM cada 15 segundos.

Si se han introducido los datos, el PIC procederá a configurar el módem GSM con los valores introducidos. Durante este proceso, el LED 1 permanecerá en ámbar, siempre y cuando no se haya introducido un código PIN erróneo. En caso de obtener un error en la introducción del código PIN, el LED 1 cambiará a color rojo y el dispositivo cancelará cualquier nueva comunicación con el módem.

Si todo el proceso de configuración del módem se ha realizado con éxito, el LED 1 permanecerá iluminado en color verde y el dispositivo se mantendrá a la espera de que se reciba o se solicite el envío de algún mensaje SMS.

Durante este estado de espera, el usuario podrá modificar los parámetros que desee y el dispositivo estará preparado con estos nuevos parámetros inmediatamente después de su recepción.

El mensaje SMS consigna está pensado para que el dispositivo pueda recibir un mensaje desde cualquier teléfono móvil, leerlo, comparar el texto del SMS recibido con el SMS consigna y proceder como se indica a continuación:

Si los 4 primeros caracteres del texto del SMS recibido coinciden con los 4 dígitos del “password” almacenado en memoria, pueden suceder dos situaciones:

1. Si, además, coincide el quinto carácter del texto SMS con el “valor” almacenado en memoria, en el LED 1 se visualizará la siguiente secuencia:

    1. APAGADO (500 ms)
    2. VERDE (500 ms)
    3. APAGADO (500 ms)
    4. AMBAR (500 ms)
    5. APAGADO (500 ms)
    6. ROJO (500 ms)
    7. APAGADO (500 ms)

2. Si por el contrario, el valor recibido no coincide con “valor”, si es un valor numérico, el LED 1 realizará tantos parpadeos en ROJO como sea el valor recibido. En caso de ser un carácter, el dispositivo no realizará ninguna acción.

Si no coinciden los caracteres con la contraseña, el dispositivo no realizará ninguna función.

Para el envío de mensajes, basta con pulsar sobre el pulsador deseado 1 o 2, dependiendo de si se quiere enviar al destinatario 1 o 2 respectivamente. Durante el proceso de envío del mensaje, que puede tardar algunos segundos, el LED 2 se iluminará en rojo.

Si se diera la difícil situación en la que se pulsan los dos botones a la vez, por defecto la función que se activaría sería el envío a destinatario 1.

Si, por algún motivo, el dispositivo quedara bloqueado, se puede reiniciar pulsando el botón RESET.

En la figura siguiente se nombran los elementos mencionados en los párrafos anteriores:

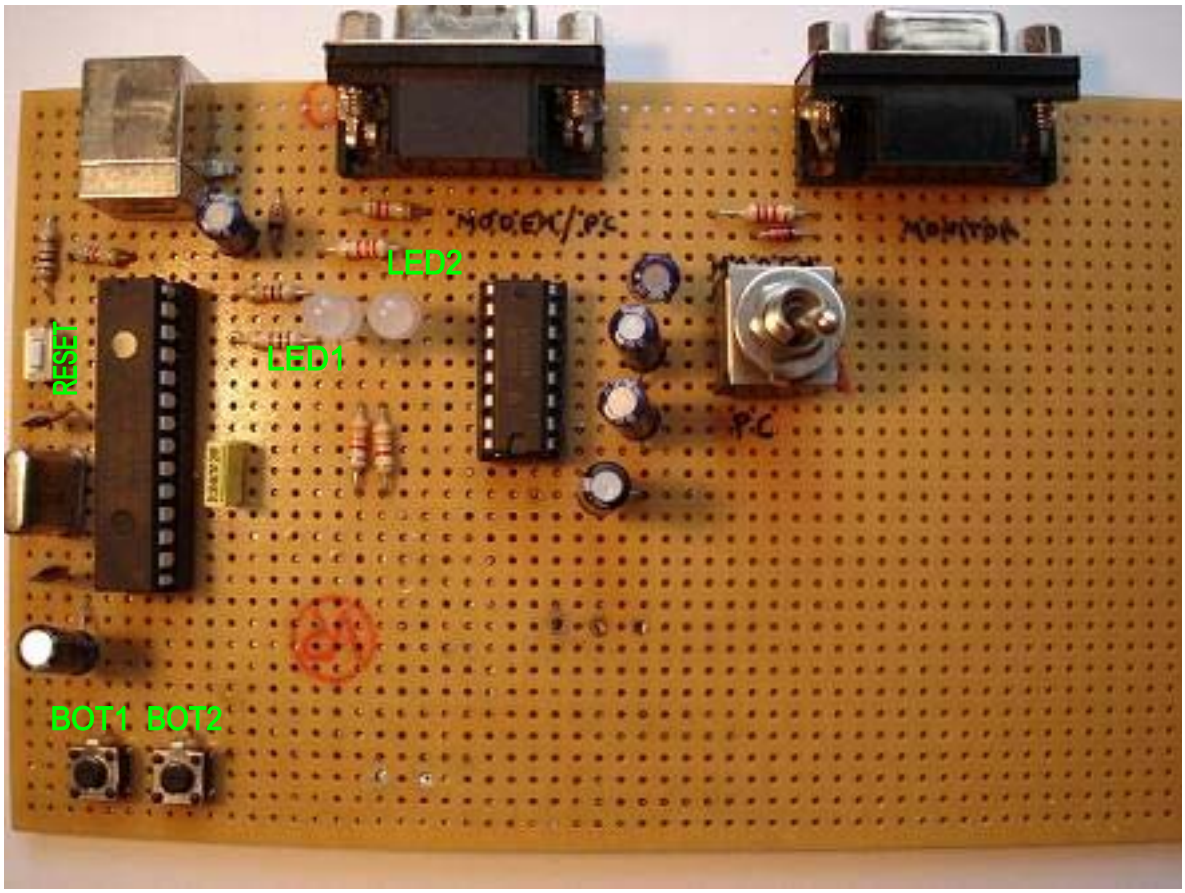


Figura 8.13 Detalle de los elementos nombrados del circuito.

8.2 Código fuente del programa en PIC.

8.2.1 picmodem.c

```
#include <18F2550.h>
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL3,CPUDIV3,VREGEN
#use delay(clock=24000000) //el cristal es de 12mhz pero el reloj es de
24mhz

// CPUDIV1= CPU DIV/2
// CPUDIV2= CPU DIV/3
// CPUDIV3= CPU DIV/4
// CPUDIV4= CPU DIV/6

#USE RS232(BAUD=9600,XMIT=PIN_C6,RCV=PIN_C7,stream=PC)

#define USB_HID_DEVICE     FALSE             //deshabilitamos el uso de
las directivas HID
#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK   //turn on EP1(EndPoint1) for
IN bulk/interrupt transfers
#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK   //turn on EP1(EndPoint1) for
OUT bulk/interrupt transfers
#define USB_EP1_TX_SIZE    1                 //size to allocate for the
tx endpoint 1 buffer (1 byte)
#define USB_EP1_RX_SIZE    1                 //size to allocate for the
rx endpoint 1 buffer (1 byte)

#include <pic18_usb.h>
#include <PicModem.h>        //Configuración del USB y los descriptores
para este dispositivo
#include <string.h>
#include <usb.c>

#define LED1    PIN_B7  //led1 verde
#define LED2    PIN_B6  //led1 rojo
#define LED4    PIN_B5  //led2 rojo
#define LED3    PIN_B4  //led2 verde   <---- ojo con esto, el led esta
girado
#define BOT1    PIN_B3
#define BOT2    PIN_B2
#define LED_ON  output_high
#define LED_OFF output_low

int const MAXLENBUFF=101;      // maxima longitud del buffer

typedef struct {
   int numcars;
   int bytesrecibidos;
   int pcom;
   int cat;
   int id;
   int id2;
   int busy;
   char car;
   char car2;
   char buffer[101];
} transmision;
```

```
typedef struct {
   int parsok;          // todos los parametros ok
   int busy;
   int ok;
   int error;
   int resp;
   int estesms;         // esperando texto sms
   int pin;             // 0=nada, 1=SIM PIN, 2=READY, 3=SIM PUK
   int csca;            // 0=nada, 1=csca ok
   int cmgf;
   int cmti;            // 0=nada, 1=sms recibido
   int cmgs;
   int cmgr;
   int smsr;            // flag que indica si sms entrante almacenado en
sms[]
   int preparado;
   char respuesta[MAXLENBUFF];
   char car;
} at;

transmision usb;
at modem;

int posbuffer=0;
int m=0;             // contador buffer auxiliar
int n;
int tics=0;
int tics2=0;
int tics232=0;
int ticusb=0;
int ticat=0;   // ticat, tics desde ultimo envio USB,para no solapar
int ticsreset=0;
int txusb=0;
int segs=0;
int segs2=0;
int segsreset=0;
int reset=0;
int enbucle=0;
int flag232=0;
char cab;

BYTE i, address, value, emer;             //variables tipo byte para el
manejo de la EEPROM
int j,d0,d1,d2,d3,lc,error=0;
char cad1[10];
char sms[5];
int8 aux;
char c,d,e;
int usbcon=1;
int usb_enum=0;
int flag=0;
int flagsms=0;

char buffaux[160];    //buffer auxiliar para ver los caracteres que se
reciben
int p=0;
char nsca[9];
```

```
#int_RDA
void serial_isr()     // =========== interrupcion puerto serie =========
{
   while(kbhit(PC))
   {
   cab=getc(PC);
   modem.busy=1;
   switch(cab)
   {                  // (13)(10)OK(13)(10) -->  CR=13 y LF=10!!!!!!!
      case 13:
         // nada, para que el CR(13) no entre en default
         break;
      case 10:                //si llega un LF(10)
         if(modem.resp==0)      //si llega el 1er LF(10)
         {
             modem.resp=1;
             posbuffer=0;
         }
         else                   //si llega el 2º LF(10) --> fin respuesta
         {
             modem.resp=0;
             if(modem.cmgr==1)
             {
               sms[0]=modem.respuesta[0];
               sms[1]=modem.respuesta[1];
               sms[2]=modem.respuesta[2];
               sms[3]=modem.respuesta[3];
               sms[4]=modem.respuesta[4];
               modem.smsr=1;
               modem.cmgr=0;
             }
             else
             {
                // --- OK ---
                if((modem.respuesta[0]=='O')&&(modem.respuesta[1]=='K'))
                {
                    modem.ok=1;
                    //flag=15;
                }
                else
                {

if((modem.respuesta[0]=='E')&&(modem.respuesta[1]=='R'))   //en el primer
if solo habran seguro 2 bytes
                  {
                     // --- ERROR ---
                     modem.error=1;
                     //flag=16;
                  }
                  else  // --- +CPIN / +CSCA / +CMGS / +CMTI / +CMGR ---
                  {

if((modem.respuesta[0]=='+')&&(modem.respuesta[1]=='C'))
                        {
                           // --- +CPIN ---

if((modem.respuesta[2]=='P')&&(modem.respuesta[7]=='S')&&(modem.respuesta
[8]=='I'))
                           {
                              if(modem.respuesta[12]=='I')
                              {
```

```
                                    modem.pin=2;      //+CPIN: SIM PIN
                                    //flag=17;
                                 }
                                 else
                                 {
                                    modem.pin=4;   //+CPIN: SIM PUK
                                 }
                              }
                              else
                              {

if((modem.respuesta[2]=='P')&&(modem.respuesta[7]=='R')&&(modem.respuesta
[8]=='E'))
                                 {
                                    modem.pin=3;   //+CPIN: READY
                                    //flag=18;
                                 }
                                 else
                                 {
                                 // --- +CSCA ---

if((modem.respuesta[2]=='S')&&(modem.respuesta[3]=='C')&&(modem.respuesta
[4]=='A'))
                                    {
                                       nsca[0]=modem.respuesta[11];
                                       nsca[1]=modem.respuesta[12];
                                       nsca[2]=modem.respuesta[13];
                                       nsca[3]=modem.respuesta[14];
                                       nsca[4]=modem.respuesta[15];
                                       nsca[5]=modem.respuesta[16];
                                       nsca[6]=modem.respuesta[17];
                                       nsca[7]=modem.respuesta[18];
                                       nsca[8]=modem.respuesta[19];

                                       modem.csca=1;
                                       //flag=19;
                                    }
                                    else
                                    {
                                       // --- +CMGS ---

if((modem.respuesta[2]=='M')&&(modem.respuesta[3]=='G')&&(modem.respuesta
[4]=='S'))
                                       {
                                          modem.cmgs=modem.respuesta[7]-48;
//guardamos la posicion del mensaje
                                          if(posbuffer==8)  //si el indice
de sms tiene 2 digitos
                                          {
                                             modem.cmgs=modem.cmgs*10;
                                             modem.cmgs+=modem.respuesta[8]-
48;
                                          }
                                       }
                                       else
                                       {
                                          // --- +CMTI ---

if((modem.respuesta[2]=='M')&&(modem.respuesta[3]=='T')&&(modem.respuesta
[4]=='I'))
                                          {
```

```
                                                flagsms=22;
                                                modem.cmti=modem.respuesta[12]-
48;
                                                if(posbuffer==14)
                                                {
                                                   modem.cmti=modem.cmti*10;

modem.cmti+=modem.respuesta[13]-48;
                                                }
                                             }
                                             else
                                             {
                                                // --- +CMGR ---

if((modem.respuesta[2]=='M')&&(modem.respuesta[3]=='G')&&(modem.respuesta
[4]=='R'))
                                                {
                                                   modem.cmgr=1;
                                                }
                                             }
                                          }
                                       }
                                    }
                                 } //else
                              } //if
                           } //else
                        }//else
                  }
               }//else
               posbuffer=0;

               // ------- vaciado del buffer --------
               //for(n=0;n<posbuffer;n++)
               //{ n
                  modem.respuesta[0]='\0';
               //}
               modem.busy=0;
               //flag232=0;
               break;   //fin case 10 LF

            default:
               if(modem.cmgr==1)
               {
                  modem.resp=1;
               }
               if(modem.resp==1) //solo leera caracteres despues del primer LF
               {
                  modem.respuesta[posbuffer]=cab;  // añado caracter recibido
al buffer
                  posbuffer++;
                  if(cab==62)      // si es '>' (3E)=62
                  {
                     modem.estesms=1;
                     modem.resp=0;
                  }
               }
            }//switch
         }
      }

      //=====================================================================
```

```
#int_RTCC               // CICLO RELOJ: 41.66ps / CICLO INSTR.: 0.16us
RTCC_isr()              // RTCC: 10,92ms(68250 instr.) / 1bit: 104us(650
instr.)
{
   tics++;    // ========== interrupcion timer0 cada 10,92ms ============
   tics2++;
   tics232++;
   ticsreset++;
   ticat++;
//------ timer 1 ------
   if(tics>91)
   {
      segs++;
      tics=0;
   }
//------ timer 2 ------
   if(tics2>91)     // tics2 y segs2 para el intervalo de quince segundos
   {
      segs2++;
      tics2=0;
   }
//------ timer 232 ------
   if(tics232>=254)      // tics232 para la espera de medio segundo antes
de enviar el buffer auxiliar
   {
      tics232=0;
   }
//------ timer envio USB ------
   if(ticat>=254)      // ticat para no solapar envios USB
   {
      ticat=0;
   }
//------ timer reset ------
   if(ticsreset>91)      // ticat para no solapar envios USB
   {
      segsreset++;
      ticsreset=0;
      if(segsreset>10)
      {
         reset=1;
      }
   }
//------ timer tic USB ------
   if(usb_enum==1)
   {
      ticusb++;
   }
   if(ticusb>91)      // ticusb, cada segundo da un toque por usb
   {
      ticusb=0;
      txusb=1;
   }
}

//=====================================================================

void leerparametros(void)
{
   int par[6];
   int ii;
```

```
   modem.busy=1;
   segs2=0;
   flag=2;
   // 0=pin,4=centromen(no se
lee),13=td1,22=td2,31=sms1,132=sms2,233=smscon
   par[0]=read_eeprom(0);
   delay_ms(4);
   par[1]=read_eeprom(13);
   delay_ms(4);
   par[2]=read_eeprom(22);
   delay_ms(4);
   par[3]=read_eeprom(31);
   delay_ms(4);
   par[4]=read_eeprom(132);
   delay_ms(4);
   par[5]=read_eeprom(233);
   delay_ms(4);

   if(par[0]==255)
   {
      modem.parsok=0;
   }
   else
   {
      modem.parsok=1;
   }

   for(ii=1;ii<6;ii++)
   {
      if((par[ii]!=255)&&(modem.parsok==1))
      {
         modem.parsok=1;
      }
      else
      {
         modem.parsok=0;
      }
   }
   modem.busy=0;
}
//---------------------------------------------------------------------
void escribirat(void)
{
   modem.busy=1;
   tics=0;
   segs=0;
   while(segs<5)  //retardo de 5 segundos antes de 1er comando at
   {
      //modem.busy=1;
   }
   modem.ok=0;
   printf("AT+IPR=9600\r");         // enviamos AT(13) para detectar que
el modem esté preparado
   tics=0;                          // y de paso se configura a 9600
baudios
   segs=0;
   while((modem.ok==0)&&(segs<5))
   {
      modem.busy=1;
   }
   if(modem.ok==0)
```

```
   {
      modem.error=1;
      LED_OFF(LED1);
      LED_ON(LED2);     //error, color rojo
   }
   else
   {
      LED_ON(LED1);
      LED_ON(LED2);    // ambar, el modem aun no esta listo
      tics=0;
      while(tics<50)       // aprox 1/2 segundo
      {
         modem.busy=1;
      }
   }
   modem.busy=0;
}
//---------------------------------------------------------------------
void escribirpin(void)
{
   char cpin[4];
   modem.busy=1;
   modem.pin=1;      //pin=0:no leido / 1:leido / 2:sim pin
   modem.ok=0;       //pin=3:ready / 4:sim puk / 5:error
   flag=3;
   printf("AT+CPIN?\r");
   tics=0;
   segs=0;
   LED_ON(LED1);
   LED_ON(LED2);  // se pone ambar durante la espera
   while((modem.pin==1)&&(segs<5))
   {
      //modem.busy=1;
   }
   if(modem.pin==1)
   {
      modem.pin=0;
      LED_OFF(LED1);
      LED_ON(LED2);     //error, color rojo
   }
   if((modem.pin==2)&&(modem.error==0))
   {
      LED_ON(LED1);
      LED_OFF(LED2);    // verde
      for(i=0;i<4;i++)
      {
         cpin[i]=read_eeprom(i);
         delay_ms(4);
      }
      modem.ok=0;
      printf("AT+CPIN=\"%c%c%c%c\"\r",cpin[0],cpin[1],cpin[2],cpin[3]);
      tics=0;
      segs=0;
      LED_ON(LED1);
      LED_ON(LED2);  // se pone ambar durante la espera
      while((modem.ok==0)&&(segs<10)&&(modem.error==0))
      {
         modem.busy=1;
      }
      if(modem.error==1)
      {
```

```
            modem.pin=5;   //intento fallido
            LED_OFF(LED1);
            LED_OFF(LED2);  //apagado
         }
         if(modem.ok==1)
         {
            modem.pin=3;
         }
         else
         {
            //modem.pin=0;
            LED_OFF(LED1);
            LED_ON(LED2);     //rojo
         }
      }

      if(modem.pin==3)
      {
         LED_ON(LED1);
         LED_OFF(LED2);    // verde durante 10 segundos
         tics=0;
         segs=0;
         while(segs<10)
         {
            modem.busy=1;
         }
      }
      modem.busy=0;
   }
   //---------------------------------------------------------------------------
   void leercsca(void)
   {
      flag=5;
      modem.csca=0;
      modem.ok=0;
      modem.busy=1;
      printf("AT+CSCA?\r");
      tics=0;
      segs=0;
      while((modem.csca==0)&&(segs<5))
      {
         //modem.busy=1;
      }
      tics=0;
      segs=0;
      while((modem.ok==0)&&(segs<2))
      {
         //modem.busy=1;
      }

      if(modem.csca==0)
      {
         modem.error=1;
         LED_OFF(LED1);
         LED_ON(LED2);     //error, color rojo
      }
      else
      {
         LED_ON(LED1);
         LED_ON(LED2);    // ambar
         modem.busy=0;
```

```
      if(modem.preparado!=4)
      {
         modem.preparado=1;   // csca ok
      }

      tics=0;
      segs=0;
   }
   modem.busy=0;
}
//---------------------------------------------------------------------------
void modotexto(void)
{
   flag=6;
   modem.busy=1;
   modem.ok=0;
   printf("AT+CMGF=1\r");
   tics=0;
   segs=0;
   while((modem.ok==0)&&(segs<5))
   {
      modem.busy=1;
   }
   if(modem.ok==0)
   {
      modem.error=1;
      LED_OFF(LED1);
      LED_ON(LED2);     //error, color rojo
   }
   else
   {
      if(modem.preparado!=4)
      {
         modem.preparado=2;   // cmgf ok
      }
      modem.cmgf=1;
      LED_ON(LED1);
      LED_OFF(LED2);    // verde
      tics=0;
      segs=0;
//      while(segs<1)
//      {
//         //modem.busy=1;
//      }
   }
   modem.busy=0;
}
//---------------------------------------------------------------------------
void avisosms(void)
{
   flag=7;
   modem.busy=1;
   modem.ok=0;
   printf("AT+CNMI=3,1,0,0\r");
   tics=0;
   segs=0;
   while((modem.ok==0)&&(segs<5))
   {
      //espera de OK
   }
   if(modem.ok==0)
```

```
   {
      modem.error=1;
      LED_OFF(LED1);
      LED_ON(LED2);     //error, color rojo
   }
   else
   {
      if(modem.preparado!=4)
      {
         modem.preparado=3;   // cnmi ok
      }

      LED_ON(LED1);
      LED_OFF(LED2);    // verde
      tics=0;
      segs=0;
      while(segs<1)
      {
         //modem.busy=1;
      }
   }
   modem.busy=0;
}
//---------------------------------------------------------------------------
void borrarsms(int ind)
{
   int iters;
   flag=8;
   modem.busy=1;
   for(iters=ind;iters>=1;iters--)  // decreciente 3,2,1
   {
      modem.ok=0;
      printf("AT+CMGD=%d\r",iters);  //borra el sms de la primera
posicion
      tics=0;
      segs=0;
      while((modem.ok==0)&&(segs<5))
      {
         //espera de OK
      }
      if(modem.ok==0)
      {
         modem.error=1;
         LED_OFF(LED1);
         LED_ON(LED2);     //error, color rojo
      }
      else
      {
         LED_ON(LED1);
         LED_OFF(LED2);    // verde
         if(modem.preparado==3)
         {
            modem.preparado=4;   // cmgd ok
         }
         tics=0;
         segs=0;
         while(segs<1)
         {
            //modem.busy=1;
            delay_ms(50);
         }
```

```
      }
   }
   modem.busy=0;
}
//---------------------------------------------------------------------
void enviarsms(int a)
{
   char cad1[9];
   char cad2[101];
   int i,j,k;
   modem.busy=1;
   modem.ok=0;
   flag=9;
   if(modem.parsok==1)
   {
      if(a==1)
      {
         j=13;    // posicion 0 td1 eeprom
         k=31;    // posicion 0 sms1 eeprom
      }
      if(a==2)
      {
         j=22;    // posicion 0 td2 eeprom
         k=132;   // posicion 0 sms2 eeprom
      }
      for(i=0;i<9;i++)
      {
         cad1[i]=read_eeprom(i+j);
         delay_ms(4);
      }

printf("AT+CMGS=\"+34%c%c%c%c%c%c%c%c%c\",145\r",cad1[0],cad1[1],cad1[2],
cad1[3],cad1[4],cad1[5],cad1[6],cad1[7],cad1[8]);
      while((modem.estesms==0)&&(modem.error==0))
      {
         LED_ON(LED1);
         LED_ON(LED2);  // se pone ambar durante la espera
      }
      if(modem.estesms==1)
      {
         modem.estesms=0;
         // aqui pongo el cuerpo del mensaje sms
         for(i=0;i<100;i++)
         {
            cad2[i]=read_eeprom(i+k);
            delay_ms(4);
            if(cad2[i]==13)
            {
               cad2[i]=26;
               modem.ok=0;              // ctrl-z (26)(0x1A)
               printf("%c",cad2[i]);
               i=100;
            }
            else
            {
               printf("%c",cad2[i]);
               if(i==99)      //si el sms tiene 100 cars escribimos el
ctrl-z en el 101
               {
                  cad2[i+1]=26;        // ctrl-z (26)(0x1A)
                  modem.ok=0;
```

```
                  printf("%c",cad2[i]);
               }
            }
         }
      }
      segs=0;
      while((modem.ok==0)&&(modem.error==0)&&(segs<15))
      {
         delay_ms(100);
         LED_ON(LED1);
         LED_ON(LED2);     // se pone ambar durante la espera
      }
      if(modem.ok==1)
      {
         LED_ON(LED1);
         LED_OFF(LED2);    // verde
      }
      else
      {
         LED_OFF(LED1);
         LED_ON(LED1);     //rojo
      }
   }
   modem.busy=0;
}
//-----------------------------------------------------------------------
void leersms(void)
{
   int cons=0;    //flag coincidencia con consigna
   int iters=0;
   int valor=0;
   char csmsc[5];    // caracter sms consigna
   char sa;          // sms aux
   modem.busy=1;
   modem.ok=0;
   modem.smsr=0;
   printf("AT+CMGR=%d\r",modem.cmti);  //lee el sms de la posicion
almacenada
   tics=0;
   segs=0;
   while((modem.smsr==0)&&(segs<5))
   {
      //modem.busy=1;
      LED_ON(LED1);
      LED_ON(LED2);
   }
   if(modem.smsr==1)
   {
      modem.smsr=0;
      for(i=0;i<4;i++)
      {
         csmsc[i]=read_eeprom(i+233);
         delay_ms(4);
         sa=(char)csmsc[i];
         if(sms[i]==sa)
         {
            cons=1;
         }
         else
         {
            cons=0;
```

```
   }
}
if(cons==1)
{
   i=4;
   csmsc[4]=read_eeprom(i+233);
   delay_ms(4);
   sa=(char)csmsc[i];
   if(sms[4]==sa)
   {
      flagsms=21;

      LED_OFF(LED1);
      LED_OFF(LED2); // apagado
      tics=0;
      while(tics<45)
      {}

      LED_ON(LED1);
      LED_OFF(LED2); // verde
      tics=0;
      while(tics<45)
      {}

      LED_OFF(LED1);
      LED_OFF(LED2); // apagado
      tics=0;
      while(tics<45)
      {}

      LED_ON(LED1);
      LED_ON(LED2);  // ambar
      tics=0;
      while(tics<45)
      {}

      LED_OFF(LED1);
      LED_OFF(LED2); // apagado
      tics=0;
      while(tics<45)
      {}

      LED_OFF(LED1);
      LED_ON(LED2);  // rojo
      tics=0;
      while(tics<45)
      {}
      LED_OFF(LED1);
      LED_OFF(LED2); // apagado
      tics=0;
      while(tics<45)
      {}
   }
   else
   {
      flagsms=20;
      valor=sms[4]-48;
      if((valor>0)&&(valor<=9))   //descartamos el 0
      {
         for(iters=0;iters<valor;iters++)
         {
```

```
                    LED_OFF(LED1);
                    LED_OFF(LED2); // apagados
                    tics=0;
                    while(tics<45) //0,5 segs
                    {}
                    LED_ON(LED2); // rojo
                    tics=0;
                    while(tics<45) //0,5 segs
                    {}
                 }
                 LED_OFF(LED1);
                 LED_OFF(LED2); // apagados
                 tics=0;
                 while(tics<45) //0,5 segs
                 {}
                 LED_ON(LED1); // verde
                 LED_OFF(LED2);
              }
           }
        }
     }
     else
     {
        modem.error=1;
        LED_OFF(LED1);
        LED_ON(LED2);     //error, color rojo
     }
     tics=0;
     segs=0;
     while((modem.ok==0)&&(segs<5))   //espera del OK
     {
        //modem.busy=1;
        LED_ON(LED1);
        LED_ON(LED2);
     }

     printf("SMS:%c%c%c%c%c",sms[0],sms[1],sms[2],sms[3],sms[4]);
     tics=0;
     segs=0;
     while(segs<1)
     {}
     modem.busy=0;
}
//---------------------------------------------------------------------


// ===================================================================
// ============================== M A I N ============================
// ===================================================================

void main(void)
{
   int it;
   usb.numcars=0;
   usb.bytesrecibidos=0;
   usb.pcom=0;
   usb.id=0;
   usb.id2=0;
   usb.buffer[0]='\0';
   usb.busy=0;
   enbucle=1;
```

```
   modem.parsok=0;
   modem.busy=0;
   modem.ok=0;
   modem.error=0;
   modem.resp=0;
   modem.estesms=0;
   modem.pin=0;
   modem.csca=0;
   modem.cmti=0;
   modem.cmgf=0;
   modem.cmgs=0;
   modem.cmgr=0;
   modem.preparado=0;    //indica que el modem se encuentra listo para
funcionamiento normal

   set_tris_b(0xF0);    //configuro RB7,RB6,RB5 y RB4 como salidas

   set_timer0(0);
   setup_counters( RTCC_INTERNAL, RTCC_DIV_256 | RTCC_8_BIT);
   enable_interrupts(INT_RTCC);
   enable_interrupts(INT_RDA);
   enable_interrupts(GLOBAL);

   LED_ON(LED1);           //ambar para indicar que esta en proceso
   LED_ON(LED2);
   LED_OFF(LED3);
   LED_OFF(LED4);

   enbucle=0;
   usb_init();                          // inicializamos el USB
   usb_wait_for_enumeration();          // esperamos hasta que el pic sea
configurado por el host
   usb_enum=usb_enumerated();
   LED_ON(LED3);
   LED_OFF(LED4);    //verde

   while (TRUE)
   {
      enbucle=1;
      ticsreset=0;

      if(usb_kbhit(1))          //si se reciben datos del host
      {
         usb.busy=1;
         ticat=0;
         LED_ON(LED3);  //ambar
         LED_ON(LED4);
         enbucle=0;
         usb_get_packet(1, &usb.car, 1);     //usb_get_packet(endpoint,
*ptr, max)
         usb.bytesrecibidos++;

         if(usb.car==4)    // si recibimos un error
         {
            error=1;
            usb_put_packet(1, &usb.car, 1, USB_DTS_TOGGLE); // enviamos
byte 04
         }
         else
         {
            // discriminacion de caracteres invalidos
```

```
            if(usb.bytesrecibidos==1)  // es primer byte?
            {
               if((usb.car<49)||((usb.car>57)&&(usb.car!=65))) // 65='A'
               {
                  error=1;
               }
               else
               {
                  if((error==0)&&(usb.pcom==0)&&(usb.cat==0))
                  {
                     usb_put_packet(1, &usb.car, 1, USB_DTS_TOGGLE); //
enviamos byte a byte
                     if((usb.car>48)&&(usb.car<=57))     // es
pseudocomando?
                     {
                        usb.id = usb.car-48;             // guardamos el
identificador de la trama
                        usb.cat = 0;                     // desactivamos
flag COMANDO AT
                        if((usb.id>4)&&(usb.id<=8))      // entre 5 y 8
                        {
                           usb.pcom = 2;                 // pseudocomando
de peticion de datos (5,6,7,8)
                           //usb.bytesrecibidos = 2;     // asi entramos
en el siguiente if
                           j=0;
                           switch(usb.id)
                           {
                              case 5:
                              usb.numcars = 4;
                              break;
                              default:
                              usb.numcars = 9;
                           }
                        }
                        if(usb.id<4)
                        {
                           usb.pcom = 1;                 // pseudocomando
de envio de datos (1,2,3)
                        }
                        if(usb.id==9)
                        {
                           usb.pcom=3;
                        }
                     }
                     else  // si no es pseudocomando, es comando AT o
cualquier cosa rara
                     {
                        if(usb.car==65)      //si es 'A'
                        {
                           usb.cat=1;
                           usb.pcom=0;
                           usb.buffer[usb.bytesrecibidos-1]=usb.car;
                        }
                        else
                        {                                    // si es
cualquier otra cosa
                           usb.cat=0;
                           usb.pcom=0;
                           usb.bytesrecibidos=0;
```

```
                        }
                     }
                  }
               }
            }
            else
            {
               if(error==0)                    // si es 2º byte o
posterior
               {
                  if(usb.pcom==1)      // si pseudocomando de envio datos
(desde PC)
                  {
                     if(usb.bytesrecibidos==2)     // si es segundo byte
                     {
                        usb_put_packet(1, &usb.car, 1, USB_DTS_TOGGLE);
// enviamos byte a byte
                        usb.numcars = usb.car-48;     // guardamos el
numero de caracteres
                     }
                     else
                     {     // si es tercer byte o posterior
                        if(usb.bytesrecibidos-3<usb.numcars) // si no ha
finalizado
                        {
                           usb_put_packet(1, &usb.car, 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                           usb.buffer[usb.bytesrecibidos-3] = usb.car;
// almacenamos en buffer el caracter recibido
                        }
                        else
                        {        //ha finalizado,aqui recibimos el <CR>
                           //identificar el pseudocomando y enviar los
comandos AT correspondientes al modem
                           switch(usb.id)
                           {
                              case 1:  // envio codigo PIN
                              for(j=0;j<4;j++)
                              {
                                 write_eeprom(j,usb.buffer[j]);  // graba
pin en 00 -> 03
                                 delay_ms(7);  // un ciclo de
borrado/escritura son unos 4ms
                              }

                              break;
                              case 2:  // envio numero destinatario 1
                              for(j=0;j<9;j++)
                              {
                                 write_eeprom(j+13,usb.buffer[j]);  //
graba dest en 13 -> 21
                                 delay_ms(7);  // un ciclo de
borrado/escritura son unos 4ms
                              }

                              break;
                              case 3:  // envio numero destinatario 2
                              for(j=0;j<9;j++)
                              {
                                 write_eeprom(j+22,usb.buffer[j]);  //
graba dest en 22 -> 30
```

```
                                delay_ms(7);  // un ciclo de
borrado/escritura son unos 4ms
                             }

                             break;
                          }
                          usb.buffer[0]='\0';
                          usb.bytesrecibidos=0;
                          usb.numcars=0;
                          usb.pcom=0;
                          usb.id=0;
                          usb.busy=0;
                       }  //fin else  (si ya ha finalizado)
                    }  // fin else (tercer byte o posterior)
                 }     // fin else if(pcom==1)

                 if(usb.pcom==2)      // si pseudocomando de ID=5,6,7 u
8
                 {
                    switch(usb.id)
                    {
                       case 5:  //solicitud de pin
                       j=usb.bytesrecibidos-2;
                       if(j==0)                   // si es el primer
dato
                       {
                          cad1[j]=read_eeprom(j);
                          delay_ms(4);
                          usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                          delay_ms(1);
                       }
                       if((j>0)&&(j<4))  // si es 2º o posterior dato
                       {
                          if(usb.car!=cad1[j-1])  // se compara el
recibido del PC con el enviado de antes del PIC
                          {
                             j=4;
                             error=1;
                             //usb.buffer[0]=0;
                             usb.buffer[0]=4;
                             usb_put_packet(1, &usb.buffer[0], 1,
USB_DTS_TOGGLE); // enviamos byte 04
                             usb.busy=0;
                          }
                          else  // si es correcto
                          {
                             cad1[j]=read_eeprom(j);
                             delay_ms(4);
                             usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                             delay_ms(1);
                          }
                       }
                       if(j==4)    // si es el ultimo byte
                       {
                          if(usb.car==cad1[j-1])  // si es correcto
                          {
                             cad1[j]=13;
                             usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos caracter 13 <CR>
```

```
                           }
                           else
                           {
                              error=1;
                              usb.buffer[0]=4;
                              usb_put_packet(1, &usb.buffer[0], 1,
USB_DTS_TOGGLE); // enviamos byte 04
                           }
                           lc=strlen(usb.buffer);
                           for(j=0;j<lc;j++)
                           {
                              usb.buffer[j]='\0';
                           }
                           usb.pcom=0;
                           usb.bytesrecibidos=0;
                           usb.id=0;
                           usb.busy=0;
                        }
                        break;

                        case 6:  //solicitud de centro de mensajes
                        j=usb.bytesrecibidos-2;
                        if(j==0)                    // si es el primer
dato
                        {
                           cad1[j]=read_eeprom(j+4);
                           delay_ms(4);
                           usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                           delay_ms(1);
                        }
                        if((j>0)&&(j<9))  // si es 2º o posterior dato
                        {
                           if(usb.car!=cad1[j-1])  // se compara el
recibido del PC con el enviado de antes del PIC
                           {
                              j=9;
                              error=1;
                              //usb.buffer[0]=0;
                              usb.buffer[0]=4;
                              usb_put_packet(1, &usb.buffer[0], 1,
USB_DTS_TOGGLE); // enviamos byte 04
                              usb.busy=0;
                           }
                           else  // si es correcto
                           {
                              cad1[j]=read_eeprom(j+4);
                              delay_ms(4);
                              usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                              delay_ms(1);
                           }
                        }
                        if(j==9)    // si es el ultimo byte
                        {
                           if(usb.car==cad1[j-1])  // si es correcto
                           {
                              cad1[j]=13;
                              usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                           }
```

```
                           else
                           {
                              error=1;
                              usb.buffer[0]=4;
                              usb_put_packet(1, &usb.buffer[0], 1,
USB_DTS_TOGGLE); // enviamos byte 04
                           }
                           lc=strlen(usb.buffer);
                           for(j=0;j<lc;j++)
                           {
                              usb.buffer[j]='\0';
                           }
                           usb.pcom=0;
                           usb.bytesrecibidos=0;
                           usb.id=0;
                           usb.busy=0;
                        }
                        break;
                        case 7:  //solicitud de telefono destino #1
                        j=usb.bytesrecibidos-2;
                        if(j==0)                  // si es el primer
dato
                        {
                           cad1[j]=read_eeprom(j+13);
                           delay_ms(4);
                           usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                           delay_ms(1);
                        }
                        if((j>0)&&(j<9))  // si es 2º o posterior dato
                        {
                           if(usb.car!=cad1[j-1])  // se compara el
recibido del PC con el enviado de antes del PIC
                           {
                              j=9;
                              error=1;
                              usb.buffer[0]=4;
                              usb_put_packet(1, &usb.buffer[0], 1,
USB_DTS_TOGGLE); // enviamos byte 04
                              usb.busy=0;
                           }
                           else  // si es correcto
                           {
                              cad1[j]=read_eeprom(j+13);
                              delay_ms(4);
                              usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                              delay_ms(1);
                           }
                        }
                        if(j==9)    // si es el ultimo byte
                        {
                           if(usb.car==cad1[j-1])  // si es correcto
                           {
                              cad1[j]=13;
                              usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                           }
                           else
                           {
                              error=1;
```

```
                              usb.buffer[0]=4;
                              usb_put_packet(1, &usb.buffer[0], 1,
USB_DTS_TOGGLE); // enviamos byte 04
                           }
                           lc=strlen(usb.buffer);
                           for(j=0;j<lc;j++)
                           {
                              usb.buffer[j]='\0';
                           }
                           usb.pcom=0;
                           usb.bytesrecibidos=0;
                           usb.id=0;
                           usb.busy=0;
                        }
                        break;
                        case 8:  //solicitud de telefono destino #2
                        j=usb.bytesrecibidos-2;
                        if(j==0)                  // si es el primer
dato
                        {
                           cad1[j]=read_eeprom(j+22);
                           delay_ms(4);
                           usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                           delay_ms(1);
                        }
                        if((j>0)&&(j<9))  // si es 2º o posterior dato
                        {
                           if(usb.car!=cad1[j-1])  // se compara el
recibido del PC con el enviado de antes del PIC
                           {
                              j=9;
                              error=1;
                              usb.buffer[0]=4;
                              usb_put_packet(1, &usb.buffer[0], 1,
USB_DTS_TOGGLE); // enviamos byte 04
                              usb.busy=0;
                           }
                           else  // si es correcto
                           {
                              cad1[j]=read_eeprom(j+22);
                              delay_ms(4);
                              usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                              delay_ms(1);
                           }
                        }
                        if(j==9)    // si es el ultimo byte
                        {
                           if(usb.car==cad1[j-1])  // si es correcto
                           {
                              cad1[j]=13;
                              usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                           }
                           else
                           {
                              error=1;
                              usb.buffer[0]=4;
                              usb_put_packet(1, &usb.buffer[0], 1,
USB_DTS_TOGGLE); // enviamos byte 04
```

```
                              }
                              lc=strlen(usb.buffer);
                              for(j=0;j<lc;j++)
                              {
                                 usb.buffer[j]='\0';
                              }
                              usb.pcom=0;
                              usb.bytesrecibidos=0;
                              usb.id=0;
                              usb.busy=0;
                           }
                           break;
                        }     //fin switch
                     }        //fin if(pcom==2)

                     if(usb.pcom==3)
                     {
                        usb_put_packet(1, &usb.car, 1, USB_DTS_TOGGLE);

                        if(usb.bytesrecibidos==2)  //si es el segundo dígito
del orden (orden2)
                        {
                           usb.id2=usb.car-48;     // id2 =1,2,3,5,6 o 7
                           if(usb.id2<4)
                           {
                              usb.pcom=3;    // envio de datos desde PC
                           }
                           else
                           {
                              usb.pcom=4;    // peticion de datos desde PC
                           }
                        }
                        else  //si es caracter de SMS
                        {
                           usb.buffer[usb.bytesrecibidos-3] = usb.car;  //
almacenamos en buffer el caracter recibido

                           if(usb.car==13) // 13=<CR>   si ha finalizado
                           {
                              switch(usb.id2)
                              {
                                 case 1:  // envio SMS 1
                                 for(j=0;j<=usb.bytesrecibidos-3;j++)
                                 {
                                    write_eeprom(j+31,usb.buffer[j]);  //
graba SMS 1 en 31 -> 131 (100+<cr>)
                                    delay_ms(7);  // un ciclo de
borrado/escritura son unos 4ms
                                 }
                                 //printf("SMS #1: %S",usb.buffer);
                                 break;
                                 case 2:  // envio SMS 2
                                 for(j=0;j<=usb.bytesrecibidos-3;j++)
                                 {
                                    write_eeprom(j+132,usb.buffer[j]);  //
graba SMS 2 en 132 -> 232
                                    delay_ms(7);  // un ciclo de
borrado/escritura son unos 4ms
                                 }
                                 //printf("SMS #2: %S",usb.buffer);
                                 break;
```

```
                              case 3:  // envio SMS CONSIGNA
                              for(j=0;j<=usb.bytesrecibidos-3;j++)
                              {
                                 write_eeprom(j+233,usb.buffer[j]);  //
graba SMSCON en 233 -> 238
                                 delay_ms(7);  // un ciclo de
borrado/escritura son unos 4ms
                              }
                              //printf("SMS CON: %S",usb.buffer);
                              break;
                           }
                           lc=strlen(usb.buffer);
                           for(j=0;j<lc;j++)
                           {
                              usb.buffer[j]='\0';
                           }
                           usb.pcom=0;
                           usb.bytesrecibidos=0;
                           usb.id=0;
                           usb.id2=0;
                           usb.busy=0;
                        }
                     }
                  }

                  if((usb.pcom==4)&&(usb.bytesrecibidos>2))
                  {
                     switch(usb.id2)
                     {
                        case 5:  //solicitud de SMS 1
                        j=usb.bytesrecibidos-3;
                        if(j==0)                  // si es el primer
dato
                        {
                           cad1[j]=read_eeprom(j+31);
                           delay_ms(4);
                           usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                           delay_ms(1);
                           if(cad1[0]==255)   // 63=?  255=(vacio)
                           {
                              usb.buffer[0]='\0';
                              usb.pcom=0;
                              usb.bytesrecibidos=0;
                              usb.id=0;
                              usb.id2=0;
                              usb.busy=0;
                              j=0;
                           }
                        }
                        if(j>0)  // si es 2º o posterior dato
                        {
                           if(usb.car!=cad1[j-1])  // se compara el
recibido del PC con el enviado de antes del PIC
                           {
                              j=4;
                              error=1;
                              usb.buffer[0]=4;
                              usb_put_packet(1, usb.buffer, 1,
USB_DTS_TOGGLE); // enviamos byte 04
                              usb.busy=0;
```

```
                        }
                        else  // si es correcto
                        {
                           if(j<100) //si no es el ultimo byte
                           {
                              cad1[j]=read_eeprom(j+31);
                              delay_ms(4);
                              usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                              delay_ms(1);
                           }
                           if((j==100)||(cad1[j]==13))   //se acaba en
el byte 99 o en el caracter <CR>
                           {
                              if(j==100)
                              {
                                 cad1[j]=13;
                                 usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos <CR>
                                 delay_ms(1);
                              }
                              lc=strlen(usb.buffer);
                              for(j=0;j<lc;j++)
                              {
                                 usb.buffer[j]='\0';
                              }
                              usb.pcom=0;
                              usb.bytesrecibidos=0;
                              usb.id=0;
                              usb.busy=0;
                           }
                        }
                     }
                     break;
                     case 6:  //solicitud de SMS 2
                     j=usb.bytesrecibidos-3;
                     if(j==0)                  // si es el primer
dato
                     {
                        cad1[j]=read_eeprom(j+132);
                        delay_ms(4);
                        usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                        delay_ms(1);
                        if(cad1[0]==255)   // 63=?  255=(vacio)
                        {
                           usb.buffer[0]='\0';
                           usb.pcom=0;
                           usb.bytesrecibidos=0;
                           usb.id=0;
                           usb.id2=0;
                           usb.busy=0;
                           j=0;
                        }
                     }
                     if(j>0)  // si es 2º o posterior dato
                     {
                        if(usb.car!=cad1[j-1])  // se compara el
recibido del PC con el enviado de antes del PIC
                        {
                           j=4;
```

```
                              error=1;
                              usb.buffer[0]=4;
                              usb_put_packet(1, &usb.buffer[0], 1,
USB_DTS_TOGGLE); // enviamos byte 04
                              usb.busy=0;
                           }
                           else  // si es correcto
                           {
                              if(j<100) //si no es el ultimo byte
                              {
                                 cad1[j]=read_eeprom(j+132);
                                 delay_ms(4);
                                 usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                                 delay_ms(1);
                              }
                              if((j==100)||(cad1[j]==13))   //se acaba en
el byte 99 o en el caracter <CR>
                              {
                                 if(j==100)
                                 {
                                    cad1[j]=13;
                                    usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos <CR>
                                    delay_ms(1);
                                 }
                                 lc=strlen(usb.buffer);
                                 for(j=0;j<lc;j++)
                                 {
                                    usb.buffer[j]='\0';
                                 }
                                 usb.pcom=0;
                                 usb.bytesrecibidos=0;
                                 usb.id=0;
                                 usb.busy=0;
                              }
                           }
                        }
                        break;
                        case 7:  //solicitud de SMS CONSIGNA
                        j=usb.bytesrecibidos-3;
                        if(j==0)                  // si es el primer
dato
                        {
                           cad1[j]=read_eeprom(j+233);
                           delay_ms(4);
                           usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                           delay_ms(1);
                           if(cad1[0]==255)   // 63=?  255=(vacio)
                           {
                              usb.buffer[0]='\0';
                              usb.pcom=0;
                              usb.bytesrecibidos=0;
                              usb.id=0;
                              usb.id2=0;
                              usb.busy=0;
                              j=0;
                           }
                        }
                        if(j>0)  // si es 2º o posterior dato
```

```
                        {
                           if(usb.car!=cad1[j-1])  // se compara el
recibido del PC con el enviado de antes del PIC
                           {
                              j=4;
                              error=1;
                              usb.buffer[0]=4;
                              usb_put_packet(1, &usb.buffer[0], 1,
USB_DTS_TOGGLE); // enviamos byte 04
                              usb.busy=0;
                           }
                           else  // si es correcto
                           {
                              if(j<5)
                              {
                                 cad1[j]=read_eeprom(j+233);
                                 delay_ms(4);
                                 usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos byte a byte
                                 delay_ms(1);
                              }
                              if(j==5) //si es el ultimo
                              {
                                 cad1[j]=13;
                                 usb_put_packet(1, &cad1[j], 1,
USB_DTS_TOGGLE); // enviamos <CR>
                                 delay_ms(1);

                                 lc=strlen(usb.buffer);
                                 for(j=0;j<lc;j++)
                                 {
                                    usb.buffer[j]='\0';
                                 }
                                 usb.pcom=0;
                                 usb.bytesrecibidos=0;
                                 usb.id=0;
                                 usb.busy=0;
                              }
                           }
                        }
                        break;
                     }
                  }

// >>>>>>>>>>>>>>>>>>>>>>>>> SI ES UN COMANDO AT
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

                  if(usb.cat==1)
                  {
                     usb_put_packet(1, &usb.car, 1, USB_DTS_TOGGLE);
                     usb.buffer[usb.bytesrecibidos-1] = usb.car;     //
almacenamos en buffer el caracter recibido

                     if(usb.car==13)   //si ha finalizado el comando AT
                     {
                        lc=strlen(usb.buffer);
                        // toca enviarlo
                        printf("%S",usb.buffer);
                        // comando AT finalizado
                        modem.busy=0;
                        usb.buffer[0]='\0';
```

```
                        usb.bytesrecibidos=0;
                        usb.numcars=0;
                        usb.pcom=0;
                        usb.cat=0;
                        usb.id=0;
                        usb.busy=0;
                     }
                  }
               }  //fin 2º byte o posterior
            }
         }//else car==4
         if(error==1)
         {
            usb.pcom=0;
            usb.cat=0;
            usb.bytesrecibidos=0;
            usb.id=0;
            usb.numcars=0;
            usb.busy=0;
         }
         ticat=0;
      }  //fin usb_kbhit

      // ------------------------------ FIN USB_KBHIT ------------------
-------------


if((usb.busy==0)&&(txusb==1)&&(modem.busy==0)&&(usb_enum==1)&&(ticat>18))
      {
         //ticat>18 = 200 ms aprox   <-- antes 45
         txusb=0;
         usb.busy=1;
         if(flagsms!=0)
         {
            usb.car2=flagsms;  // (char)
            flagsms=0;
         }
         else
         {
            usb.car2=flag; // (char)
         }
         usb_put_packet(1, &usb.car2, 1, USB_DTS_TOGGLE);  //envia FLAG
si dentro bucle principal
         usb.busy=0;
         ticat=0;
      }

      if((usb.busy==0)&&(usb_enum==1))
      {
         LED_ON(LED3);           // led verde si USB conectado y
disponible
         LED_OFF(LED4);
      }
      else
      {
         if(usb.busy==0)
         {
            LED_OFF(LED3);          // si no está conectado al USB, led 2
en rojo
            LED_ON(LED4);
         }
```

```
         else
         {
            LED_ON(LED3);
            LED_ON(LED4);           // ambar si USB ocupado
         }
      }

      // ------ resetear segs, segs2 --------
      if(segs>=254)     // para que no se desborde (es un int, 0-255)
      {
         segs=0;
      }
      if(segs2>=254)     // para que no se desborde (es un int, 0-255)
      {
         segs2=0;
      }

      //------------------------------ lectura pulsadores ---------------
-------------

      if((!input(BOT1))&&(usb.busy==0))     //si se pulsa bot1
      {
         LED_OFF(LED3);     //led2 en rojo
         LED_ON(LED4);
         if((modem.busy==0)&&(modem.preparado==4))
         {
            enbucle=0;
            enviarsms(1);
            borrarsms(modem.cmgs);
         }
      }
      else  //al ponerlo anidado, no se pueden leer los 2 pulsados a la
vez
      {
         if((input(BOT2))&&(input(BOT1)))
         {
            if(usb.busy==0)
            {
               LED_ON(LED3);
               LED_OFF(LED4); //led2 verde
            }
            else
            {
               LED_ON(LED3);
               LED_ON(LED4);  //led2 ambar
            }
         }

         if((!input(BOT2))&&(usb.busy==0))
         {
            LED_OFF(LED3);     //pulsador 2 = rojo
            LED_ON(LED4);
            if((modem.busy==0)&&(modem.preparado==4))
            {
               enbucle=0;
               enviarsms(2);
               borrarsms(modem.cmgs);
            }
         }
      }
```

```
      //------------------------------ Comandos AT----------------------
-------------

      if(modem.preparado<=3)     // procedimientos para iniciar el modem
      {
         //========================== LEER PARAMETROS
============================

if((modem.ok==0)&&(modem.parsok==0)&&(modem.error==0)&&(modem.pin==0)&&(m
odem.preparado==0)&&(modem.busy==0))
         {
            //si es el primer acceso
            enbucle=0;
            leerparametros();    // se leen y se inicializa a 0 el
contador de 15 segundos
            escribirat();        // escribimos AT para comprobar que el
modem responde
         }
         // si no se tienen los parametros
         if((modem.parsok==0)&&(modem.busy==0)&&(modem.pin==0))
         {
            //si esta o estuvo conectado
            if((segs2>=15)&&(usb.busy==0))
            {
               enbucle=0;
               leerparametros();
            }
         }
         //==================== PARPADEO LED1 FALTA DE PARAMETROS
=================
         if(modem.parsok==0)        //parpadeo en ambar de led1
         {
            if(tics>45)
            {
               LED_ON(LED1);
               LED_ON(LED2);         // se enciende ambar
            }
            else
            {
               LED_OFF(LED1);
               LED_OFF(LED2);        // se apaga ambar
            }
         }
         //========================== ESCRIBIR PIN
============================

if((modem.parsok==1)&&(modem.ok==1)&&(modem.pin==0)&&(usb.busy==0)&&(mode
m.busy==0))  //si parametros ok y no intento fallido pin
         {
            enbucle=0;
            escribirpin();
         }
         //============================ LEER CSCA
============================

if((modem.pin==3)&&(modem.preparado==0)&&(usb.busy==0)&&(modem.busy==0))
         {
            //modem.pin=3:
            enbucle=0;
            leercsca();
         }
```

```
         //====================== CONFIGURAR MODO TEXTO
=========================
         if((modem.preparado==1)&&(usb.busy==0)&&(modem.busy==0))  //si
tenemos hasta CSCA, configurar CMGF
         {
            enbucle=0;
            modotexto();
         }
         //=================== CONFIGURAR AVISO SMS RECIBIDO
===================
         if((modem.preparado==2)&&(usb.busy==0)&&(modem.busy==0))  //si
tenemos hasta CMGF, configurar CNMI
         {
            enbucle=0;
            avisosms();
         }
         //====================== BORRAR SMS EN POSICION 1
====================
         if((modem.preparado==3)&&(usb.busy==0)&&(modem.busy==0))  //si
tenemos hasta CNMI, borrar mensaje 1
         {
            enbucle=0;
            borrarsms(1);
         }
      }

      if(modem.preparado==4)     //si modem preparado
      {
         flag=1;
         if((modem.cmti!=0)&&(usb.busy==0)&&(modem.busy==0))
         {
            enbucle=0;
            leersms();
            borrarsms(modem.cmti);
            modem.cmti=0;
         }
      }

      if(modem.csca==1)
      {
         for(i=0;i<9;i++)
         {
            write_eeprom(i+4,nsca[i]);
            delay_ms(7);
            //+4 pq empieza en la posicion 4 de la eeprom
            //+11 porque solo guardamos desde la posicion 11 del arreglo
de la trama recibida
            //+CSCA: "+34656000311",145
         }
         modem.csca=0;
      }
   }  //fin while indefinido
}  //fin main
```

8.3 Código fuente del software en PC, versión SUPERVISOR.

8.3.1 Picmodem.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics; //Clase para abrir página web
using System.IO.Ports;
using System.Threading;

namespace PicModem
{
    public partial class PicModem : Form
    {
        PicmodemAPI usbapi = new PicmodemAPI();
        SerialPort port = new SerialPort("COM1", 9600, Parity.None, 8,
StopBits.One);

        string rx, cadinfo, cadinfo2;
        string spin, steldest1, steldest2, ssms1, ssms2, ssmscon;
        // strings para guardar el valor leido del PIC, para compararlo
antes de subirlo
        char[] bufferserie = new char[160];
        char[] result=new char[101];
        char[] valor=new char[20];
        uint online = 0;
        bool picleido = false;
        bool cargar = false;
        bool leer = false;
        bool fallo = false;
        bool picbusy = true;
        bool pcbusy = false;
        bool envusb = false;    //orden de envio USB de trama escrita
        int leerflag = 0;       //para que lea el flag cada segundo
        int me = 0;
        int y = 0;  // iteraciones para cargar
        int x = 0;  // iteraciones para leer
        int cincoseg = 0;
        int tics = 0;
        int ready3 = 0;     //se tienen que contar 3 tics con pic: ready,
para evitar falsos readys
        bool[] datok = new bool[6]; // new bool -> los inicializa como
'false'
        bool[] dattx = new bool[6];
        // datos: 0=pin, 1=teldest1, 2=teldest2
        // 3=sms1, 4=sms2, 5=smscon
        // datok=datos leidos del pic
        // dattx=datos a enviar al pic,porque no se han leido o porque se
han tecleado


        public PicModem()
        {
            //inicializa los componentes del form
            InitializeComponent();
```

```
            this.port.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.Recepcion);
            inicializaciones();
        }

        private void inicializaciones()
        {
            System.Windows.Forms.Control.CheckForIllegalCrossThreadCalls
= false;   // daba un error
            //operadora.SelectedItem = 0;
            //centromen.Text = "656000311";

            Conectar();     //abre el puerto serie

            bot_conectar.Enabled = false;
            bot_desconectar.Enabled = true;
            bot_enviar.Enabled = false;
            bot_leerpic.Enabled = false;
            bot_cargardatos.Enabled = true;
            pinbox.Enabled = true;
            sms1.Enabled = true;
            sms2.Enabled = true;
            comando.Enabled = true;
            teldest1.Enabled = true;
            teldest2.Enabled = true;
            pass.Enabled = true;
            val.Enabled = true;



            comando.Text = "";

            online = usbapi.Conexion();
            if (online == 1) //esta conectado?
            {
                statuslabel.Text = "PFC - Control de módulo GSM desde
microcontrolador.  <<ONLINE>>";
                cincoseg = 1;
                this.info.Items.Add("Espera de 5s antes de 1er Comando
AT.");
                this.info.SelectedIndex = this.info.Items.Count - 1;
            }
            else
            {
                statuslabel.Text = "PFC - Control de módulo GSM desde
microcontrolador.  <<OFFLINE>>";
            }
        }
 /*       private void leerpic()
        {
            pcbusy = true;
            if (!picbusy)  //llerflag=1 para que se envie entre flag y
flag y no se solapen
            {
                comando.Text = "5";     // lectura de pin
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "6";     // lectura del centro de mensajes
                enviarusb();
```

```
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "7";     // lectura del destinatario 1
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "8";     // lectura del destinatario 2
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "95";    // lectura del sms1
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "96";    // lectura del sms2
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "97";    // lectura del sms de control
                enviarusb();
            }
            picleido = true;


        }
*/
        private void enviarusb()
        {
            // Se enviarán pseudocomandos o comandos AT

            // se enviará un primer byte con el tipo de parametro a
enviar
            // y un segundo byte con el numero de caracteres que componen
el parametro
            // y por ultimo se recibiran los caracteres que componen el
valor
            // Ejemplo: 141234 <- tipo: 1 (pin), caracteres: 4 ,valor:
1234
            // Tipos:   1=pin, 2=tel destinatario1, 3=tel destinatario2
            // 5=solicitud pin, 6=solicitud centromen, 7=solicitud
teldest1, 8=solicitud teldest2
            // 91=sms1, 92=sms2, 93=sms control, 95=sol. sms1, 96=sol.
sms2, 97=sol. sms control

            int b, l, i;
            char cad1;              //caracter de handshaking
            string cad2;
            uint b1 = 0, orden = 0, nocars = 0, c, orden2 = 0;
            bool cat = false;       // es comando at?
            int pcom = 0;
            int delay = 100;
            pcbusy = true;
            tramarec.Text = "";
            l = comando.Text.Length;
            bot_enviar.Enabled = false;
```

```
            cad2 = comando.Text.ToUpper();

            orden = (uint)cad2[0];

            if ((orden > 48) && (orden <= 57)) //  si es un pseudocomando
            {
                cat = false;
                if (orden < 52) // <4
                    pcom = 1;
                else
                {
                    if (orden == 52)
                        pcom = 0;
                    else
                        pcom = 2;
                }
                if (l > 1)
                {
                    nocars = (uint)cad2[1];
                    nocars -= 48;
                }
                else
                {
                    nocars = 4;
                }

                switch (orden)
                {
                    case 49:    // 1
                        if ((nocars == 4) && (nocars == l - 2))
                        {
                            this.info.Items.Add("Envio de código PIN.");
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
                            pinbox.Text = "";
                            pinbox.Text += cad2[2];
                            pinbox.Text += cad2[3];
                            pinbox.Text += cad2[4];
                            pinbox.Text += cad2[5];
                        }
                        else
                        {
                            this.info.Items.Add("ERROR: el código PIN
requiere 4 dígitos.");
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
                            fallo = true;
                        }
                        break;

                    case 50:    // 2
                        if ((nocars == 9) && (nocars == l - 2))
                        {
                            this.info.Items.Add("Envio de número de
teléfono destino #1.");
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
                        }
                        else
                        {
```

```
                            this.info.Items.Add("ERROR: el número de
teléfono requiere 9 dígitos.");
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
                            fallo = true;
                        }
                        break;

                    case 51:    // 3
                        if ((nocars == 9) && (nocars == l - 2))
                        {
                            this.info.Items.Add("Envio de número de
teléfono destino #2.");
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
                        }
                        else
                        {
                            this.info.Items.Add("ERROR: el número de
teléfono requiere 9 dígitos.");
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
                            fallo = true;
                        }
                        break;

                    case 53:    // 5
                        this.info.Items.Add("Solicitud de código PIN.");
                        this.info.SelectedIndex = this.info.Items.Count -
1;
                        l = 6;
                        nocars = 4;     // con l=6 y nocars=4 se obtiene
"fallo=false"
                        break;

                    case 54:    // 6
                        this.info.Items.Add("Solicitud de centro de
mensajes.");
                        this.info.SelectedIndex = this.info.Items.Count -
1;
                        l = 11;
                        nocars = 9;     // con l=11 y nocars=9 se obtiene
"fallo=false"
                        break;

                    case 55:    // 7
                        this.info.Items.Add("Solicitud de número de
teléfono destino #1.");
                        this.info.SelectedIndex = this.info.Items.Count -
1;
                        l = 11;
                        nocars = 9;     // con l=11 y nocars=9 se obtiene
"fallo=false"
                        break;

                    case 56:    // 8
                        this.info.Items.Add("Solicitud de número de
teléfono destino #2.");
                        this.info.SelectedIndex = this.info.Items.Count -
1;
                        l = 11;
```

```
                        nocars = 9;     // con l=11 y nocars=9 se obtiene
"fallo=false"
                        break;

                    case 57:    // 9
                        if (l > 1)
                            c = (uint)cad2[1];
                        else
                            c = 0;

                        switch (c)
                        {
                            case 49:    // si es 91

                                l = sms1.Text.Length;
                                nocars = (uint)l;
                                l = l + 2;          //ha de cumplir:
l=nocars+2
                                orden2 = 49;
                                pcom = 3;
                                cad2 = sms1.Text;
                                if (nocars > 0)
                                {
                                    this.info.Items.Add("Envio de SMS
#1.");
                                    this.info.SelectedIndex =
this.info.Items.Count - 1;
                                }
                                else
                                {
                                    fallo = true;
                                    this.info.Items.Add("ERROR: SMS #1
está vacío.");
                                    this.info.SelectedIndex =
this.info.Items.Count - 1;
                                }
                                break;
                            case 50:    // si es 92
                                l = sms2.Text.Length;
                                nocars = (uint)l;
                                l = l + 2;          //ha de cumplir:
l=nocars+2
                                orden2 = 50;
                                pcom = 3;
                                cad2 = sms2.Text;
                                if (nocars > 0)
                                {
                                    this.info.Items.Add("Envio de SMS
#2.");
                                    this.info.SelectedIndex =
this.info.Items.Count - 1;
                                }
                                else
                                {
                                    fallo = true;
                                    this.info.Items.Add("ERROR: SMS #2
está vacío.");
                                    this.info.SelectedIndex =
this.info.Items.Count - 1;
                                }
                                break;
```

```
                            case 51:    // si es 93
                                nocars = 5;
                                l = 7;          //ha de cumplir:
l=nocars+2
                                orden2 = 51;
                                pcom = 3;
                                cad2 = pass.Text + val.Text;
                                this.info.Items.Add("Envio de SMS
CONSIGNA.");
                                this.info.SelectedIndex =
this.info.Items.Count - 1;
                                break;
                            case 53:    // si es 95
                                this.info.Items.Add("Solicitud de SMS
#1.");
                                this.info.SelectedIndex =
this.info.Items.Count - 1;
                                l = 102;
                                nocars = 100;
                                orden2 = 53;
                                pcom = 4;
                                break;
                            case 54:    // si es 96
                                this.info.Items.Add("Solicitud de SMS
#2.");
                                this.info.SelectedIndex =
this.info.Items.Count - 1;
                                l = 102;
                                nocars = 100;
                                orden2 = 54;
                                pcom = 4;
                                break;
                            case 55:    // si es 97
                                this.info.Items.Add("Solicitud de SMS
CONSIGNA.");
                                this.info.SelectedIndex =
this.info.Items.Count - 1;
                                l = 7;
                                nocars = 5; // son 5 caracteres
                                orden2 = 55;
                                pcom = 4;
                                break;
                            case 0:     // si solo se ha escrito un
digito
                                this.info.Items.Add("Necesario un 2º
dígito.");
                                this.info.SelectedIndex =
this.info.Items.Count - 1;
                                fallo = true;
                                break;
                            default:
                                this.info.Items.Add("DESCONOCIDO.");
                                this.info.SelectedIndex =
this.info.Items.Count - 1;
                                fallo = true;
                                break;
                        }
                        break;
                    default:
                        this.info.Items.Add("DESCONOCIDO.");
```

```
                        this.info.SelectedIndex = this.info.Items.Count -
1;
                        fallo = true;
                        break;
                }

                if (l != nocars + 2)
                {
                    fallo = true;
                }
                if ((l == nocars + 2) && (fallo == false))
                    fallo = false;

                if (!fallo)
                {
                    usbapi.enviaUSB(orden);     //envio primer byte
                    this.enviadousb.Items.Add("1º:" + (char)orden + "(" +
orden + ")");
                    this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;

                    Thread.Sleep(delay);   // delay de 100ms
                    cad1 = (char)usbapi.recibeUSB();  // recibimos de 1
en 1 byte!!!

                    this.recibidousb.Items.Add("(" + (int)cad1 + ")" +
cad1);
                    this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                    if (orden != cad1)  // si es diferente
                    {
                        if ((uint)cad1 != 4)
                        {
                            switch ((uint)cad1)
                            {
                                case 1:
                                    picbusy = false;
                                    estadopic.Text = "PIC: READY";
                                    break;
                                case 2:
                                    picbusy = false;
                                    estadopic.Text = "PIC: PARAMETROS";
                                    break;
                                case 3:
                                    picbusy = false;
                                    estadopic.Text = "PIC: PIN";
                                    break;
                                case 5:
                                    picbusy = false;
                                    estadopic.Text = "PIC: CSCA";
                                    break;
                                case 6:
                                    picbusy = false;
                                    estadopic.Text = "PIC: CMGF";
                                    break;
                                case 7:
                                    picbusy = false;
                                    estadopic.Text = "PIC: CNMI";
                                    break;
                                case 8:
```

```
                                    picbusy = false;
                                    estadopic.Text = "PIC: BORR.SMS";
                                    break;
                                case 9:
                                    picbusy = false;
                                    estadopic.Text = "PIC: ENV.SMS";
                                    break;
                                case 15:
                                    picbusy = false;
                                    estadopic.Text = "RS232: OK";
                                    break;
                                case 16:
                                    picbusy = false;
                                    estadopic.Text = "RS232: ERROR";
                                    break;
                                case 17:
                                    picbusy = false;
                                    estadopic.Text = "RS232: SIM PIN";
                                    break;
                                case 18:
                                    picbusy = false;
                                    estadopic.Text = "RS232: SIM READY";
                                    break;
                                case 19:
                                    picbusy = false;
                                    estadopic.Text = "RS232: CSCA";
                                    break;
                                case 0:
                                    picbusy = true;
                                    //estadopic.Text = "PIC: BUSY";
                                    break;
                            }
/*
                            Thread.Sleep(delay);   // delay de 200ms
                            cad1 = (char)usbapi.recibeUSB();  //
recibimos de 1 en 1 byte!!!
                            this.recibidousb.Items.Add("(" + (int)cad1 +
")" + cad1);
                            this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;
 */
                        }
                        else
                        {
                            orden = 4;
                            fallo = true;
                            usbapi.enviaUSB(orden); // se envia byte 04
para indicar error
                            this.enviadousb.Items.Add("Error->4");
                            this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                            Thread.Sleep(delay);   // delay de 100ms
                            cad1 = (char)usbapi.recibeUSB();  //
recibimos retorno de fallo
                            this.recibidousb.Items.Add("Error->4 (" +
(uint)cad1 + ")");
                            this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;
                            MessageBox.Show("Ha ocurrido un error crítico
en la comunicación PC <--> PIC\nDesconecta el dispositivo.", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
                        Application.Exit();
                    }
                }
                if ((pcom == 2)&&!fallo) // comandos de solicitud de
datos 5,6,7 u 8
                {
                    usbapi.enviaUSB(cad1);     //enviamos por 2ª vez
el primer byte
                    this.enviadousb.Items.Add("Rep 1º:" + (char)orden
+ "(" + orden + ")");
                    this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                    Thread.Sleep(delay);   // delay de 100ms
                    cad1 = (char)usbapi.recibeUSB();  // recibimos
1er dato
                    if (cad1 == 4)
                    {
                        fallo = true;
                    }
                    else
                    {
                        valor[0] = cad1;
                        this.recibidousb.Items.Add("Dato 0=" + "(" +
(uint)cad1 + ")" + cad1);
                        this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                        c = (uint)cad1;
                        usbapi.enviaUSB(c);  // rebotamos d0
                        this.enviadousb.Items.Add("Dato 0=" + "(" +
(uint)cad1 + ")" + cad1);
                        this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                    }
                }
                if ((pcom == 4) && !fallo) // comandos de solicitud
de datos 95,96 o 97
                {
                    usbapi.enviaUSB(orden2);     //enviamos segundo
byte
                    this.enviadousb.Items.Add("2º:" + (char)orden2 +
"(" + orden2 + ")");
                    this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                    Thread.Sleep(delay);   // delay de 100ms
                    cad1 = (char)usbapi.recibeUSB();
                    this.recibidousb.Items.Add(cad1 + "(" + (int)cad1
+ ")");
                    this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                    if (cad1 == 4)
                    {
                        fallo = true;
                    }
                    else
                    {
                        usbapi.enviaUSB(cad1);     //enviamos por 2ª
vez el segundo byte
                        this.enviadousb.Items.Add("Rep 2º:" +
(char)orden2 + "(" + orden2 + ")");
```

```
                            this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                            Thread.Sleep(delay);   // delay de 100ms
                            cad1 = (char)usbapi.recibeUSB();  //
recibimos 1er dato
                            if (cad1 == 4)
                            {
                                fallo = true;
                                MessageBox.Show("Ha ocurrido un error
crítico en la comunicación PC <--> PIC\nDesconecta el dispositivo.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                Application.Exit();
                            }
                            else
                            {
                                if (cad1 == 255)     //si recibimos un
0xFF como 1er caracter
                                {                    //quiere decir que el
sms está vacío
                                    fallo = true;
                                    pcom = 0;
                                    this.info.Items.Add("Recibido un
caracter 0xFF");
                                    this.info.SelectedIndex =
this.info.Items.Count - 1;
                                    this.recibidousb.Items.Add("Dato
vacío.");
                                    this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;
                                    pcbusy = false;
                                }
                                else
                                {
                                    result[0] = cad1;
                                    this.recibidousb.Items.Add("Dato 0="
+ cad1 + "(" + (uint)cad1 + ")");
                                    this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                                    c = (uint)cad1;
                                    usbapi.enviaUSB(c);  // rebotamos d0
                                    this.enviadousb.Items.Add("Dato 0=" +
(char)c + "(" + c + ")");
                                    this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                                }
                            }
                        }
                    }
                    if (!fallo)    //si se ha enviado ok, se sigue
enviando
                    {
                        if (orden < 52) // si orden es 1,2 o 3
                        {
                            for (b = 1; b < nocars + 2; b++)
                            {
                                b1 = (uint)cad2[b]; // cad2=cadena a
enviar en mayusculas
                                // b1=valor ascii del caracter a enviar
                                usbapi.enviaUSB(b1);
```

```
                                this.enviadousb.Items.Add((char)b1 + "("
+ b1 + ")");
                                this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                                Thread.Sleep(delay);   // delay de 100ms
                                cad1 = (char)usbapi.recibeUSB();
// recibimos de 1 en 1 byte!!!
                                this.recibidousb.Items.Add(cad1 + "(" +
(int)cad1 + ")");
                                this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                                result[b] = cad1;

                                if ((b1 != (uint)cad1)||((int)cad1==4))
                                {
                                    b = (int)nocars+2;
                                    fallo = true;
                                    b1 = 4;
                                    usbapi.enviaUSB(b1); // se envia byte
04 para indicar error
                                }
                            }
                            if (!fallo)
                            {
                                b1 = 13;            // <CR>
                                usbapi.enviaUSB(b1);
                                this.enviadousb.Items.Add((char)b1 + "("
+ b1 + ")");
                                this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                            }
                            pcbusy = false;
                        }
                        else
                        {
                            if (orden < 57)
                            {
                                // si es orden 5,6,7 u 8!!!!!!!
                                // arriba hemos enviado el 1er byte,hemos
recibido y
                                // confirmado el byte de 'dato 0'
                                // aqui leeremos el 'dato 1' y
posteriores,iremos rebotando
                                // los caracteres que vayamos recibiendo
                                for (b = 1; b < nocars + 1; b++)
                                {
                                    Thread.Sleep(delay);   // delay de
100ms
                                    cad1 = (char)usbapi.recibeUSB();
// <-- ojo!!!
                                    this.recibidousb.Items.Add("Dato " +
b + "=" + "(" + (uint)cad1 + ")" + cad1);
                                    this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                                    c = (uint)cad1;
                                    valor[b] = cad1;
                                    if ((c != 4) && (c != 13)&& (c != 0))
//si no hay error, no <CR> y no 0
                                    {
```

```
                                        usbapi.enviaUSB(c);
                                        this.enviadousb.Items.Add("Dato "
+ b + "=" + "(" + c + ")" + (char)c);
                                        this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                                    }
                                    else
                                    {
                                        if (c == 13)    //si <CR>, es que
OK
                                        {

this.recibidousb.Items.Add("Dato recibido correctamente.");

this.recibidousb.SelectedIndex = this.recibidousb.Items.Count - 1;
                                            switch (orden)
                                            {
                                                case 53:    // 5 = PIN
                                                    if (valor[0] == 255)
                                                        datok[0] = false;
                                                    else
                                                    {
                                                        datok[0] = true;
                                                        pinbox.Text = "";
                                                        for (i = 0; i <
4; i++)
                                                        {
                                                            pinbox.Text
+= valor[i];
                                                        }
                                                        spin =
pinbox.Text;
                                                    }
                                                    break;
                                                case 54:    // 6 =
CENTROMEN
                                                    centromen.Text = "";
                                                    for (i = 0; i < 9;
i++)
                                                    {
                                                        if (valor[i] !=
255)
                                                        {

centromen.Text += valor[i];


                                                        }
                                                        else
                                                        {

centromen.Text += (char)(valor[i]-192);

                                                        }
                                                    }
                                                    break;
                                                case 55:    //7 =
TELDEST1
                                                    if (valor[0] == 255)
                                                        datok[1] = false;
                                                    else
                                                    {
                                                        datok[1] = true;
```

```
                                                    teldest1.Text =
"";
                                                    for (i = 0; i <
9; i++)
                                                    {
                                                        teldest1.Text
+= valor[i];
                                                    }
                                                    steldest1 =
teldest1.Text;
                                                }
                                                break;
                                            case 56:    //8 =
TELDEST2
                                                if (valor[0] == 255)
                                                    datok[2] = false;
                                                else
                                                {
                                                    datok[2] = true;
                                                    teldest2.Text =
"";
                                                    for (i = 0; i <
9; i++)
                                                    {
                                                        teldest2.Text
+= valor[i];
                                                    }
                                                    steldest2 =
teldest2.Text;
                                                }
                                                break;
                                        }
                                        pcbusy = false;
                                    }
                                    else
                                    {
                                        b = (int)nocars + 1;    //
para que no vuelva a entrar en el FOR
this.recibidousb.Items.Add("Error en recepción de dato.");
this.recibidousb.SelectedIndex = this.recibidousb.Items.Count - 1;
                                        fallo = true;
                                        pcbusy = false;
                                        MessageBox.Show("Ha ocurrido
un error crítico en la comunicación PC <--> PIC\nDesconecta el
dispositivo.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                        Application.Exit();
                                    }
                                }
                            }
                            else        //si es orden 9
                            {
                                if (pcom == 3) //envio datos    91, 92 o
93
                                {
                                    usbapi.enviaUSB(orden2);     //
enviamos orden2
```

```
this.enviadousb.Items.Add((char)(orden2) + "(" + (orden2) + ")");
                                        this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                                        Thread.Sleep(delay);   // delay de
100ms
                                        cad1 = (char)usbapi.recibeUSB();
// recibimos de 1 en 1 byte!!!
                                        this.recibidousb.Items.Add(cad1 + "("
+ (int)cad1 + ")");
                                        this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                                        for (b = 0; b < nocars; b++)
                                        {
                                            b1 = (uint)cad2[b]; //
cad2=cadena a enviar
                                            // b1=valor ascii del caracter a
enviar
                                            usbapi.enviaUSB(b1);

this.enviadousb.Items.Add((char)b1 + "(" + b1 + ")");
                                            this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                                            Thread.Sleep(delay);   // delay
de 100ms
                                            cad1 = (char)usbapi.recibeUSB();
// recibimos de 1 en 1 byte!!!
                                            this.recibidousb.Items.Add("Dato
" + b + "=" + cad1 + "(" + (uint)cad1 + ")");
                                            this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                                            result[b] = cad1;

                                            if ((b1 !=
(uint)cad1)||((int)cad1==4))
                                            {
                                                b = (int)nocars;
                                                b1 = 4;
                                                usbapi.enviaUSB(b1); // se
envia byte 04 para indicar error
                                                pcbusy = false;
                                            }
                                        }
                                        b1 = 13;                // se envia
<CR>
                                        usbapi.enviaUSB(b1);
                                        this.enviadousb.Items.Add("Ultimo
dato=" + (char)b1 + "(" + b1 + ")");
                                        this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                                        Thread.Sleep(delay);   // delay de
100ms
                                        cad1 = (char)usbapi.recibeUSB();
                                        this.recibidousb.Items.Add("Ultimo
dato=" + cad1 + "(" + (int)cad1 + ")");
                                        this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;
                                        pcbusy = false;
                                    }
```

```
                                else    //si pcom==4 , peticion de datos
                                {
                                    // si es orden 9!!!!!!!
                                    // arriba hemos enviado el 1er y 2º
byte,y repetido 2º byte.
                                    // hemos recibido y confirmado el
byte de 'dato 0'
                                    // aqui leeremos el 'dato 1' y
posteriores,iremos rebotando
                                    // los caracteres que vayamos
recibiendo

                                    for (b = 1; b < nocars+1; b++)
//nocars=100 (0 a 99) smscon: nocars=5
                                    {
                                        Thread.Sleep(delay);   // delay
de 100ms
                                        cad1 = (char)usbapi.recibeUSB();
// <-- ojo!!!
                                        this.recibidousb.Items.Add("Dato
" + b + "=" + cad1 + "(" + (uint)cad1 + ")");
                                        this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                                        c = (uint)cad1;
                                        result[b] = cad1;
                                        if ((c != 4) && (c != 13)) //si
no hay error y no <CR>
                                        {
                                            usbapi.enviaUSB(c);

this.enviadousb.Items.Add("Dato " + b + "=" + (char)c + "(" + c + ")");
                                            this.enviadousb.SelectedIndex
= this.enviadousb.Items.Count - 1;
                                        }
                                        if (c == 13)    //si <CR>
                                        {

this.recibidousb.Items.Add("Dato recibido correctamente.");

this.recibidousb.SelectedIndex = this.recibidousb.Items.Count - 1;
                                            switch (orden2)
                                            {
                                                case 53:    // 5 = SMS1
                                                    if (result[0] == 255)
                                                    {
                                                        datok[3] = false;
                                                        sms1.Text =
"(vacío)";
                                                    }
                                                    else
                                                    {
                                                        datok[3] = true;
                                                        sms1.Text = "";
                                                        for (i = 0; i <
b; i++)
                                                        {
                                                            sms1.Text +=
result[i];
                                                        }
```

```
                                                    ssms1 =
sms1.Text;
                                                }
                                                break;
                                            case 54:    // 6 = SMS2
                                                if (result[0] == 255)
                                                {
                                                    datok[4] = false;
                                                    sms2.Text =
"(vacío)";
                                                }
                                                else
                                                {
                                                    datok[4] = true;
                                                    sms2.Text = "";
                                                    for (i = 0; i <
b; i++)
                                                    {
                                                        sms2.Text +=
result[i];
                                                    }
                                                    ssms2 =
sms2.Text;
                                                }
                                                break;
                                            case 55:    // 7 = SMSCON
                                                if (result[0] == 255)
                                                {
                                                    datok[5] = false;
                                                    smscon.Text =
"(vacío)";
                                                    pass.Text = "";
                                                    val.Text = "";
                                                }
                                                else
                                                {
                                                    datok[5] = true;
                                                    smscon.Text = "";
                                                    pass.Text = "";
                                                    ssmscon = "";
                                                    for (i = 0; i <
4; i++)
                                                    {
                                                        smscon.Text
+= result[i];
                                                        pass.Text +=
result[i];
                                                        ssmscon +=
result[i];
                                                    }
                                                    smscon.Text +=
',' + result[4];
                                                    ssmscon += "," +
result[4];
                                                    val.Text = "";
                                                    val.Text +=
result[4];

this.info.Items.Add("ssmscon=" + ssmscon);
```

```
this.info.SelectedIndex = this.info.Items.Count - 1;
                                                    }
                                                    break;
                                                }
                                                pcbusy = false;
                                                b = (int)nocars;    // para
que no vuelva a entrar en el FOR
                                            }
                                            if(c==4)
                                            {


this.recibidousb.Items.Add("Error en recepción de dato.");

this.recibidousb.SelectedIndex = this.recibidousb.Items.Count - 1;
                                                b = (int)nocars;    // para
que no vuelva a entrar en el FOR
                                                pcbusy = false;
                                                fallo = true;
                                            }

                                        }
                                        //leer confirmacion
                                        pcom = 0;
                                    }
                                }
                            }
                        }
                        comando.Text = null;
                        //fallo = false;
                    }

                    else  // >>>>>>>>>>>>>>>>>> comando AT <<<<<<<<<<<<<<<<<<<<
                    {
                        if (orden == 65)    //si es 'A'
                        {
                            b1 = 0;
                            cat = true;
                            for (b = 0; b < l; b++)
                            {
                                b1 = (uint)cad2[b]; // cad2=cadena a enviar en
mayusculas
                                // b1=valor ascii del caracter a enviar
                                usbapi.enviaUSB(b1);
                                this.enviadousb.Items.Add((char)b1 + "(" + b1 +
")");
                                this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                                Thread.Sleep(delay);   // delay de 100ms
                                cad1 = (char)usbapi.recibeUSB();        //
recibimos de 1 en 1 byte!!!
                                this.recibidousb.Items.Add(cad1 + "(" +
(uint)cad1 + ")");
                                this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                                if (b1 != (uint)cad1)
                                {
                                    Thread.Sleep(delay);   // delay de 100ms
```

```
                            cad1 = (char)usbapi.recibeUSB();        //
recibimos de 1 en 1 byte!!!
                            this.recibidousb.Items.Add(cad1 + "(" +
(uint)cad1 + ")");
                            this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;

                            if ((uint)cad1 == 4)
                            {
                                cat = false;
                                b = l;
                                b1 = 4;
                                Thread.Sleep(delay);   // delay de 100ms
                                usbapi.enviaUSB(b1);        // se envia
byte 04 para indicar error
                                this.enviadousb.Items.Add("Error->4");
                                this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                                pcbusy = false;
                            }
                        }
                    }
                    // una vez enviados todos los caracteres del comando
AT
                    if (cat == true)
                    {
                        //enviar <CR> final de trama
                        b1 = 13;
                        Thread.Sleep(delay);   // delay de 100ms
                        usbapi.enviaUSB(b1);
                        this.enviadousb.Items.Add((char)b1 + "(" + b1 +
")");
                        this.enviadousb.SelectedIndex =
this.enviadousb.Items.Count - 1;
                        Thread.Sleep(delay);   // delay de 100ms
                        cad1 = (char)usbapi.recibeUSB();        //
recibimos de 1 en 1 byte!!!
                        this.recibidousb.Items.Add(cad1 + "(" +
(uint)cad1 + ")");
                        this.recibidousb.SelectedIndex =
this.recibidousb.Items.Count - 1;
                    }

                    // trama enviada ok
                    comando.Text = null;
                    pcbusy = false;
                }
            }
            cat = false;
            pcom = 0;
            //fallo = false;
        }

        private void Recepcion(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            char[] rec=new char[160];
            //string rec2;
            // por el puerto serie recibiremos los comandos AT que ordene
el PIC o
```

```
            // sea que recibiremos una cadena de cierta longitud que irá
variando
            // POR LO TANTO: no hay que hacer nada con esta trama, solo
mostrarla.
            // formato: AT+CPIN=1234<CR>
            //          AT+CSCA=+34656000311<CR>
            //int lrx = 0;
            //int it;
            Thread.Sleep(80);   // para que dé tiempo de leer todos los
bytes
            rx = this.port.ReadExisting();  // rx es un string
            //lrx = rx.Length;
            tramarec.Text = rx;
            this.tramasrec.Items.Add(rx);
            this.tramasrec.SelectedIndex = this.tramasrec.Items.Count -
1;
            /*
            rec2 = "(";
            for (it = 0; it < lrx; it++)
            {
                rec[it] = rx[it];
                if (rec[it] < 33)   //si no es un caracter
                {
                    rec2 += (uint)rec[it];
                }
                else
                    rec2 += rec[it];
            }
            rec2 += ")";
            this.tramasrec.Items.Add(rec2);
            this.tramasrec.SelectedIndex = this.tramasrec.Items.Count -
1;
            rec2 = null;
            rec[0]='\0';
            */
            rx = null;
        }
        private void salir_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        private void bot_conectar_Click(object sender, EventArgs e)
        {
            Conectar();
            bot_conectar.Enabled = false;
            bot_desconectar.Enabled = true;
            bot_enviar.Enabled = false;
        }
        private void Conectar()
        {
            port.Open();
        }
        private void Desconectar()
        {
            port.Close();
            bot_conectar.Enabled = true;
            bot_desconectar.Enabled = false;
            bot_enviar.Enabled = false;
            comando.Text = "";
        }
        private void bot_desconectar_Click(object sender, EventArgs e)
```

```
        {
            Desconectar();
        }
        private void botlimpiarusb_Click(object sender, EventArgs e)
        {
            enviadousb.Items.Clear();
            recibidousb.Items.Clear();
            info.Items.Clear();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            char tim;
            uint itim;
            leerflag++;
            online = usbapi.Conexion();     //ajustada una temporizacion
de 500ms

            if (online == 1)                //esta conectado?
            {
                tics++;
                statuslabel.Text = "PFC - Control de módulo GSM desde
microcontrolador.  <<ONLINE>>";
                bot_leerpic.Enabled = true;
                bot_cargardatos.Enabled = true;
                comando.Enabled = true;
                if ((cincoseg == 0)&&(me == 0))
                {
                    this.info.Items.Add("Espera de 5s antes de 1er
Comando AT.");
                    this.info.SelectedIndex = this.info.Items.Count - 1;
                    cincoseg = 1;
                }
                me = 2;
                if ((comando.Text.Length > 0)&&(!pcbusy))
                    bot_enviar.Enabled = true;
                if ((!picleido)&&(!picbusy)&&(tics>10)&&(ready3>=3))
//tics>10: 5 seg
                {
                    tics = 0;
                    ready3=0;
                    leer = true;
                }
                if (me == 1)
                    me = 2;

            }
            else
            {
                bot_leerpic.Enabled = false;
                bot_cargardatos.Enabled = false;
                tics = 0;
                //bot_enviar.Enabled = false;       se gestiona en
comando_textchanged

                estadopic.Text = "";
                statuslabel.Text = "PFC - Control de módulo GSM desde
microcontrolador.  <<OFFLINE>>";

                if (me == 1)
                {
```

```
                    picleido = false;
                    cargar = false;
                    fallo = false;
                    picbusy = true;
                    pcbusy = false;
                    y = 0;
                    cincoseg = 0;

                    datok[0] = false;
                    datok[1] = false;
                    datok[2] = false;
                    datok[3] = false;
                    datok[4] = false;
                    datok[5] = false;

                    dattx[0] = false;
                    dattx[1] = false;
                    dattx[2] = false;
                    dattx[3] = false;
                    dattx[4] = false;
                    dattx[5] = false;
                }

                if (me==0)
                {
                    me = 1;
                    MessageBox.Show("El dispositivo PicModem no está
conectado.", "Advertencia", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                }
                if (me == 2)
                {
                    me = 1;
                    MessageBox.Show("Se ha desconectado el dispositivo
PicModem.", "Información", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                }
            }
            if ((online == 1) && (!pcbusy) && (leerflag >= 2))
            {
                leerflag = 0;   //asi entra cada segundo
                pcbusy = true;
                tim = (char)usbapi.recibeUSB();
                itim = (uint)tim;
                //this.info.Items.Add("Leido: " + itim);
                //this.info.SelectedIndex = this.info.Items.Count - 1;
                if (itim != 4)
                {
                    switch (itim)
                    {
                        case 1:
                            picbusy = false;
                            estadopic.Text = "PIC: READY";
                            ready3++;
                            if (operadora.Text == "No disponible.")
                            {
                                if ((!fallo) && (!picbusy))
                                {
                                    operadora.Text = "Leyendo...";
                                    centromen.Text = "Leyendo...";
```

```
                                    comando.Text = "6";     // lectura
del centro de mensajes
                                    enviarusb();
                                }
                            }
                            break;
                        case 2:
                            picbusy = false;
                            ready3++;
                            estadopic.Text = "PIC: PARAMETROS";
                            break;
                        case 3:
                            picbusy = false;
                            estadopic.Text = "PIC: PIN";
                            break;
                        case 5:
                            picbusy = false;
                            estadopic.Text = "PIC: CSCA";
                            break;
                        case 6:
                            picbusy = false;
                            estadopic.Text = "PIC: CMGF";
                            break;
                        case 7:
                            picbusy = false;
                            estadopic.Text = "PIC: CNMI";
                            break;
                        case 8:
                            picbusy = false;
                            estadopic.Text = "PIC: BORR.SMS";
                            break;
                        case 9:
                            picbusy = false;
                            estadopic.Text = "PIC: ENV.SMS";
                            break;
                        case 15:
                            picbusy = false;
                            estadopic.Text = "RS232: OK";
                            break;
                        case 16:
                            picbusy = false;
                            estadopic.Text = "RS232: ERROR";
                            break;
                        case 17:
                            picbusy = false;
                            estadopic.Text = "RS232: SIM PIN";
                            break;
                        case 18:
                            picbusy = false;
                            estadopic.Text = "RS232: SIM READY";
                            break;
                        case 19:
                            picbusy = false;
                            estadopic.Text = "RS232: CSCA";
                            break;
                        case 20:
                            picbusy = false;
                            this.info.Items.Add("SMS recibido: coincide
PASS.");
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
```

```
                            break;
                        case 21:
                            picbusy = false;
                            this.info.Items.Add("SMS recibido: coincide
PASS+VALOR.");
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
                            break;
                        case 22:
                            picbusy = false;
                            this.info.Items.Add("SMS recibido.");
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
                            break;

                        case 0:
                            picbusy = true;
                            estadopic.Text = "PIC: BUSY";
                            ready3 = 0;
                            break;
                    }
                }
                else
                    fallo = true;

                pcbusy = false;
            }

// ------------------ CARGAR ------------------
            if ((cargar)&&(!fallo)&&(!picbusy))
            {
                y++;
                pcbusy = true;

                switch (y)
                {
                    case 1:
                        if (spin != pinbox.Text)
                        {
                            comando.Text = "14" + pinbox.Text;       //
envio de pin
                            enviarusb();
                        }
                        else
                        {
                            this.info.Items.Add("El PIN es el mismo.");
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
                        }
                        break;
                    case 2:
                        if (steldest1 != teldest1.Text)
                        {
                            comando.Text = "29" + teldest1.Text;     //
envio de teldest1
                            enviarusb();
                        }
                        else
                        {
                            this.info.Items.Add("El TD1 es el mismo.");
```

```
                            this.info.SelectedIndex =
this.info.Items.Count - 1;
                            }
                            break;
                        case 3:
                            if (steldest2 != teldest2.Text)
                            {
                                comando.Text = "39" + teldest2.Text;     //
envio de teldest2
                                enviarusb();
                            }
                            else
                            {
                                this.info.Items.Add("El TD2 es el mismo.");
                                this.info.SelectedIndex =
this.info.Items.Count - 1;
                            }
                            break;
                        case 4:
                            if (ssms1 != sms1.Text)
                            {
                                comando.Text = "91";     // envio de sms1
                                enviarusb();
                            }
                            else
                            {
                                this.info.Items.Add("El SMS1 es el mismo.");
                                this.info.SelectedIndex =
this.info.Items.Count - 1;
                            }
                            break;
                        case 5:
                            if (ssms2 != sms2.Text)
                            {
                                comando.Text = "92";     // envio de sms2
                                enviarusb();
                            }
                            else
                            {
                                this.info.Items.Add("El SMS2 es el mismo.");
                                this.info.SelectedIndex =
this.info.Items.Count - 1;
                            }
                            break;
                        case 6:
                            if (ssmscon != smscon.Text)
                            {
                                comando.Text = "93";     // envio de smscon
                                enviarusb();
                            }
                            else
                            {
                                this.info.Items.Add("El SMSCON es el
mismo.");
                                this.info.SelectedIndex =
this.info.Items.Count - 1;
                            }
                            cargar = false;
                            y = 0;
                            break;
                    }
```

```
            }

// -------------------- LEER -------------------
            if ((leer) && (!fallo) && (!picbusy))
            {
                x++;
                pcbusy = true;

                switch (x)
                {
                    case 1:
                        comando.Text = "5";     // lectura de pin
                        enviarusb();
                        break;
                    case 2:
                        comando.Text = "6";     // lectura del centro de
mensajes
                        enviarusb();
                        break;
                    case 3:
                        comando.Text = "7";     // lectura del
destinatario 1
                        enviarusb();
                        break;
                    case 4:
                        comando.Text = "8";     // lectura del
destinatario 2
                        enviarusb();
                        break;
                    case 5:
                        comando.Text = "95";    // lectura del sms1
                        enviarusb();
                        break;
                    case 6:
                        comando.Text = "96";    // lectura del sms2
                        enviarusb();
                        break;
                    case 7:
                        comando.Text = "97";    // lectura del sms de
control
                        enviarusb();
                        leer = false;
                        x = 0;
                        picleido = true;

                        cadinfo = "DATOS ALMACENADOS EN PicModem:\n";
                        cadinfo2 = "\n\nDATOS NECESARIOS:\n";
                        if (datok[0])
                            cadinfo += "\n   PIN: " + pinbox.Text;
                        else
                            cadinfo2 += "\n   PIN: contraseña de 4
dígitos";
                        if (datok[1])
                            cadinfo += "\n   TELDEST1: " + teldest1.Text;
                        else
                            cadinfo2 += "\n   TELDEST1: telefono
destinatario 1 de 9 dígitos";
                        if (datok[2])
                            cadinfo += "\n   TELDEST2: " + teldest2.Text;
                        else
```

```
                            cadinfo2 += "\n   TELDEST2: telefono
destinatario 2 de 9 dígitos";
                        if (datok[3])
                            cadinfo += "\n   SMS1: " + sms1.Text;
                        else
                            cadinfo2 += "\n   SMS1: mensaje 1 de hasta
100 caracteres";
                        if (datok[4])
                            cadinfo += "\n   SMS2: " + sms2.Text;
                        else
                            cadinfo2 += "\n   SMS2: mensaje 2 de hasta
100 caracteres";
                        if (datok[5])
                            cadinfo += "\n   SMSCON: " + smscon.Text;
                        else
                            cadinfo2 += "\n   SMSCON: mensaje consigna de
4+1 caracteres";

                        if (!datok[0] && !datok[1] && !datok[2] &&
!datok[3] && !datok[4] && !datok[5])
                        {
                            cadinfo += "\n   (VACÍO)";
                        }
                        if (cadinfo2 != "\n\nDATOS NECESARIOS:\n")
                            cadinfo += cadinfo2;

                        MessageBox.Show(cadinfo, "Información.",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                        teldest1.Enabled = true;
                        teldest2.Enabled = true;

                        pcbusy = false;
                        break;
                    }
                }

                if ((envusb)&&(!picbusy))
                {
                    envusb = false;
                    enviarusb();
                }
                if (fallo)
                {
                    cargar = false;
                    y = 0;
                    fallo = false;
                }
            }

            private void comando_TextChanged_1(object sender, EventArgs e)
            {
                if (comando.Text != "")
                {
                    if (comando.Text[0] == '0')
                    {
                        this.info.Items.Add("No se puede comenzar una trama
con el caracter 0.");
                        this.info.SelectedIndex = this.info.Items.Count - 1;
                        comando.Text = "";
                    }
                }
```

```
            string aux;
            aux = comando.Text;
            if ((online == 1) && (aux.Length > 0))
            {
                bot_enviar.Enabled = true;
            }
            else
            {
                bot_enviar.Enabled = false;
            }
        }
        private void bot_enviar_Click_1(object sender, EventArgs e)
        {
            bot_enviar.Enabled = false;
            envusb = true;
        }
        private void teldest_TextChanged(object sender, EventArgs e)
        {
            teldest1.Enabled = true;
            if (teldest1.Text != "")
            {
                if ((teldest1.Text[0] != '6')&&(teldest1.Text[0]!=255))
                {
                    this.info.Items.Add("El número de teléfono ha de
comenzar por 6.");
                    this.info.SelectedIndex = this.info.Items.Count - 1;
                    teldest1.Text = "";
                }
            }
        }
        private void botlimpiarserie_Click(object sender, EventArgs e)
        {
            tramarec.Text = "";
            tramasrec.Items.Clear();
        }
        private void sms1_TextChanged(object sender, EventArgs e)
        {
            int i;
            sms1.Enabled = true;
            i = 100 - sms1.Text.Length;
            contsms1.Text = Convert.ToString(i);
            if (sms1.Text.Length > 0)
            {
                if (sms1.Text[0] == '?')
                {
                    sms1.Text = "";
                    this.info.Items.Add("El primer caracter no puede ser
'?'.");
                    this.info.SelectedIndex = this.info.Items.Count - 1;
                }
            }
        }
        private void centromen_TextChanged(object sender, EventArgs e)
        {
            switch (centromen.Text)
            {
                case "656000311":
                    operadora.Text = "Orange";
                    break;
                case "607003110":
                    operadora.Text = "Vodafone";
```

```
                    break;
                case "609090909":
                    operadora.Text = "Movistar";
                    break;
                case "622996111":
                    operadora.Text = "Yoigo";
                    break;
                case "?????????":
                    operadora.Text = "No disponible.";
                    centromen.Text = "No disponible.";
                    break;
                default:
                    if(operadora.Text!="No disponible.")
                        operadora.Text = "(desconocido)";
                    break;
            }
        }
        private void pinbox_TextChanged(object sender, EventArgs e)
        {
            pinbox.Enabled = true;
        }
        private void pass_TextChanged(object sender, EventArgs e)
        {
            smscon.Text = "";
            if ((pass.Text.Length == 4)&&(val.Text.Length==1))
            {
                smscon.Text = pass.Text + val.Text;
            }
        }
        private void val_TextChanged(object sender, EventArgs e)
        {
            smscon.Text = "";
            if ((pass.Text.Length == 4) && (val.Text.Length == 1))
            {
                smscon.Text = pass.Text + val.Text;
            }
        }
        private void bot_leerpic_Click(object sender, EventArgs e)
        {
            limpiartodo();
            leer = true;
        }
        private void teldest2_TextChanged(object sender, EventArgs e)
        {
            teldest2.Enabled = true;
            if (teldest2.Text != "")
            {
                if ((teldest2.Text[0] != '6')&&(teldest2.Text[0]!=255))
                {
                    this.info.Items.Add("El número de teléfono ha de
comenzar por 6.");
                    this.info.SelectedIndex = this.info.Items.Count - 1;
                    teldest2.Text = "";
                }
            }
        }
        private void sms2_TextChanged(object sender, EventArgs e)
        {
            int i;
            sms2.Enabled = true;
            i = 100 - sms2.Text.Length;
```

```
            contsms2.Text = Convert.ToString(i);
            if (sms2.Text.Length > 0)
            {
                if (sms2.Text[0] == '?')
                {
                    sms2.Text = "";
                    this.info.Items.Add("El primer caracter no puede ser
'?'.");
                    this.info.SelectedIndex = this.info.Items.Count - 1;
                }
            }
        }

        private void bot_cargardatos_Click(object sender, EventArgs e)
        {
            cadinfo = "Existen campos sin rellenar.\n\n";

            if (pinbox.Text.Length == 4)
                dattx[0] = true;
            else
                cadinfo += "PIN.\n";

            if (teldest1.Text.Length == 9)
                dattx[1] = true;
            else
                cadinfo += "Teléfono destinatario 1.\n";

            if (teldest2.Text.Length == 9)
                dattx[2] = true;
            else
                cadinfo += "Teléfono destinatario 2.\n";

            if (sms1.Text.Length >0)
                dattx[3] = true;
            else
                cadinfo += "SMS 1.\n";

            if (sms2.Text.Length > 0)
                dattx[4] = true;
            else
                cadinfo += "SMS 2.\n";

            if (smscon.Text != "")
                dattx[5] = true;
            else
                cadinfo += "SMS DE CONTROL.\n";

            if (dattx[0] && dattx[1] && dattx[2] && dattx[3] && dattx[4]
&& dattx[5])
            {
                cargar = true;
                //Thread.Sleep(20);   // delay de 20ms
            }
            else
            {
                MessageBox.Show(cadinfo, "¡ERROR!", MessageBoxButtons.OK,
MessageBoxIcon.Error );
            }
        }

        private void bot_limpiartodo_Click(object sender, EventArgs e)
```

```
        {
            int i;
            limpiartodo();
            for (i = 0; i <= 5; i++)
            {
                dattx[i] = false;
            }
        }
        private void limpiartodo()
        {
            enviadousb.Items.Clear();
            recibidousb.Items.Clear();
            info.Items.Clear();
            tramarec.Text = "";
            tramasrec.Items.Clear();
            sms1.Text = "";
            sms2.Text = "";
            pinbox.Text = "";
            pass.Text = "";
            val.Text = "";
            teldest1.Text = "";
            teldest2.Text = "";
        }
        private void Enviar()
        {
            // Write a character
            tramarec.Text+=(char)(13);
            port.Write(tramarec.Text);
            this.tramasrec.Items.Add(tramarec.Text);
            this.tramasrec.SelectedIndex = this.tramasrec.Items.Count -
1;
            // Write a set of bytes
            //port.Write(new byte[] { 0x0A, 0xE2, 0xFF }, 0, 3);
        }

        private void bot_envserie_Click(object sender, EventArgs e)
        {
            Enviar();
        }
    }
}
```

8.3.2 Picmodem.Designer.cs

```
namespace PicModem
{
    partial class PicModem
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources =
new System.ComponentModel.ComponentResourceManager(typeof(PicModem));
            this.botlimpiarusb = new System.Windows.Forms.Button();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.bot_envserie = new System.Windows.Forms.Button();
            this.botlimpiarserie = new System.Windows.Forms.Button();
            this.tramarec = new System.Windows.Forms.TextBox();
            this.tramasrec = new System.Windows.Forms.ListBox();
            this.bot_desconectar = new System.Windows.Forms.Button();
            this.label12 = new System.Windows.Forms.Label();
            this.bot_conectar = new System.Windows.Forms.Button();
            this.salir = new System.Windows.Forms.Button();
            this.statusStrip1 = new System.Windows.Forms.StatusStrip();
            this.statuslabel = new
System.Windows.Forms.ToolStripStatusLabel();
            this.timer1 = new
System.Windows.Forms.Timer(this.components);
            this.label4 = new System.Windows.Forms.Label();
            this.recibidousb = new System.Windows.Forms.ListBox();
            this.label3 = new System.Windows.Forms.Label();
            this.enviadousb = new System.Windows.Forms.ListBox();
            this.groupBox3 = new System.Windows.Forms.GroupBox();
            this.estadopic = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.info = new System.Windows.Forms.ListBox();
            this.label1 = new System.Windows.Forms.Label();
            this.bot_enviar = new System.Windows.Forms.Button();
```

```
            this.comando = new System.Windows.Forms.TextBox();
            this.groupBox5 = new System.Windows.Forms.GroupBox();
            this.operadora = new System.Windows.Forms.Label();
            this.centromen = new System.Windows.Forms.Label();
            this.label10 = new System.Windows.Forms.Label();
            this.pinbox = new System.Windows.Forms.TextBox();
            this.label9 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.bot_leerpic = new System.Windows.Forms.Button();
            this.bot_cargardatos = new System.Windows.Forms.Button();
            this.teldest1 = new System.Windows.Forms.TextBox();
            this.label8 = new System.Windows.Forms.Label();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.label17 = new System.Windows.Forms.Label();
            this.contsms2 = new System.Windows.Forms.Label();
            this.label18 = new System.Windows.Forms.Label();
            this.contsms1 = new System.Windows.Forms.Label();
            this.label19 = new System.Windows.Forms.Label();
            this.teldest2 = new System.Windows.Forms.TextBox();
            this.sms2 = new System.Windows.Forms.TextBox();
            this.label6 = new System.Windows.Forms.Label();
            this.sms1 = new System.Windows.Forms.TextBox();
            this.label13 = new System.Windows.Forms.Label();
            this.groupBox4 = new System.Windows.Forms.GroupBox();
            this.label16 = new System.Windows.Forms.Label();
            this.label15 = new System.Windows.Forms.Label();
            this.val = new System.Windows.Forms.TextBox();
            this.label14 = new System.Windows.Forms.Label();
            this.pass = new System.Windows.Forms.TextBox();
            this.label11 = new System.Windows.Forms.Label();
            this.smscon = new System.Windows.Forms.Label();
            this.bot_limpiartodo = new System.Windows.Forms.Button();
            this.groupBox2.SuspendLayout();
            this.statusStrip1.SuspendLayout();
            this.groupBox3.SuspendLayout();
            this.groupBox5.SuspendLayout();
            this.groupBox1.SuspendLayout();
            this.groupBox4.SuspendLayout();
            this.SuspendLayout();
            //
            // botlimpiarusb
            //
            this.botlimpiarusb.Location = new System.Drawing.Point(342,
21);
            this.botlimpiarusb.Name = "botlimpiarusb";
            this.botlimpiarusb.Size = new System.Drawing.Size(64, 21);
            this.botlimpiarusb.TabIndex = 21;
            this.botlimpiarusb.Text = "LIMPIAR";
            this.botlimpiarusb.UseVisualStyleBackColor = true;
            this.botlimpiarusb.Click += new
System.EventHandler(this.botlimpiarusb_Click);
            //
            // groupBox2
            //
            this.groupBox2.Controls.Add(this.bot_envserie);
            this.groupBox2.Controls.Add(this.botlimpiarserie);
            this.groupBox2.Controls.Add(this.tramarec);
            this.groupBox2.Controls.Add(this.tramasrec);
            this.groupBox2.Controls.Add(this.bot_desconectar);
            this.groupBox2.Controls.Add(this.label12);
            this.groupBox2.Controls.Add(this.bot_conectar);
```

```
            this.groupBox2.Location = new System.Drawing.Point(434, 12);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(302, 190);
            this.groupBox2.TabIndex = 6;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Lectura RS-232 (respuesta del modem)";
            //
            // bot_envserie
            //
            this.bot_envserie.Location = new System.Drawing.Point(238,
11);
            this.bot_envserie.Name = "bot_envserie";
            this.bot_envserie.Size = new System.Drawing.Size(58, 30);
            this.bot_envserie.TabIndex = 32;
            this.bot_envserie.Text = "ENVIAR";
            this.bot_envserie.UseVisualStyleBackColor = true;
            this.bot_envserie.Click += new
System.EventHandler(this.bot_envserie_Click);
            //
            // botlimpiarserie
            //
            this.botlimpiarserie.Location = new System.Drawing.Point(239,
44);
            this.botlimpiarserie.Name = "botlimpiarserie";
            this.botlimpiarserie.Size = new System.Drawing.Size(58, 21);
            this.botlimpiarserie.TabIndex = 31;
            this.botlimpiarserie.Text = "LIMPIAR";
            this.botlimpiarserie.UseVisualStyleBackColor = true;
            this.botlimpiarserie.Click += new
System.EventHandler(this.botlimpiarserie_Click);
            //
            // tramarec
            //
            this.tramarec.Location = new System.Drawing.Point(93, 45);
            this.tramarec.MaxLength = 30;
            this.tramarec.Name = "tramarec";
            this.tramarec.Size = new System.Drawing.Size(126, 20);
            this.tramarec.TabIndex = 31;
            //
            // tramasrec
            //
            this.tramasrec.BackColor =
System.Drawing.SystemColors.Control;
            this.tramasrec.FormattingEnabled = true;
            this.tramasrec.HorizontalScrollbar = true;
            this.tramasrec.ItemHeight = 14;
            this.tramasrec.Location = new System.Drawing.Point(6, 71);
            this.tramasrec.Name = "tramasrec";
            this.tramasrec.Size = new System.Drawing.Size(291, 102);
            this.tramasrec.TabIndex = 29;
            //
            // bot_desconectar
            //
            this.bot_desconectar.Location = new System.Drawing.Point(93,
19);
            this.bot_desconectar.Name = "bot_desconectar";
            this.bot_desconectar.Size = new System.Drawing.Size(95, 20);
            this.bot_desconectar.TabIndex = 7;
            this.bot_desconectar.Text = "DESCONECTAR";
            this.bot_desconectar.UseVisualStyleBackColor = true;
```

```
            this.bot_desconectar.Click += new
System.EventHandler(this.bot_desconectar_Click);
            //
            // label12
            //
            this.label12.AutoSize = true;
            this.label12.Location = new System.Drawing.Point(47, 48);
            this.label12.Name = "label12";
            this.label12.Size = new System.Drawing.Size(40, 14);
            this.label12.TabIndex = 22;
            this.label12.Text = "Trama:";
            //
            // bot_conectar
            //
            this.bot_conectar.Location = new System.Drawing.Point(6, 19);
            this.bot_conectar.Name = "bot_conectar";
            this.bot_conectar.Size = new System.Drawing.Size(81, 20);
            this.bot_conectar.TabIndex = 6;
            this.bot_conectar.Text = "CONECTAR";
            this.bot_conectar.UseVisualStyleBackColor = true;
            this.bot_conectar.Click += new
System.EventHandler(this.bot_conectar_Click);
            //
            // salir
            //
            this.salir.DialogResult =
System.Windows.Forms.DialogResult.Cancel;
            this.salir.Location = new System.Drawing.Point(622, 422);
            this.salir.Name = "salir";
            this.salir.Size = new System.Drawing.Size(114, 51);
            this.salir.TabIndex = 5;
            this.salir.Text = "SALIR";
            this.salir.UseVisualStyleBackColor = true;
            this.salir.Click += new
System.EventHandler(this.salir_Click);
            //
            // statusStrip1
            //
            this.statusStrip1.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.statuslabel});
            this.statusStrip1.Location = new System.Drawing.Point(0,
485);
            this.statusStrip1.Name = "statusStrip1";
            this.statusStrip1.Size = new System.Drawing.Size(743, 22);
            this.statusStrip1.TabIndex = 7;
            this.statusStrip1.Text = "PFC - Control Modem GSM via PIC.";
            //
            // statuslabel
            //
            this.statuslabel.Name = "statuslabel";
            this.statuslabel.Size = new System.Drawing.Size(253, 17);
            this.statuslabel.Text = "PFC - Control Modem GSM via PIC.
<<OFFLINE>>";
            //
            // timer1
            //
            this.timer1.Enabled = true;
            this.timer1.Interval = 500;
            this.timer1.Tick += new
System.EventHandler(this.timer1_Tick);
```

```
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(3, 107);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(48, 14);
            this.label4.TabIndex = 27;
            this.label4.Text = "Recibido";
            //
            // recibidousb
            //
            this.recibidousb.BackColor =
System.Drawing.SystemColors.Control;
            this.recibidousb.FormattingEnabled = true;
            this.recibidousb.ItemHeight = 14;
            this.recibidousb.Location = new System.Drawing.Point(5, 124);
            this.recibidousb.Name = "recibidousb";
            this.recibidousb.Size = new System.Drawing.Size(184, 60);
            this.recibidousb.TabIndex = 26;
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(3, 27);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(45, 14);
            this.label3.TabIndex = 24;
            this.label3.Text = "Enviado";
            //
            // enviadousb
            //
            this.enviadousb.BackColor =
System.Drawing.SystemColors.Control;
            this.enviadousb.FormattingEnabled = true;
            this.enviadousb.ItemHeight = 14;
            this.enviadousb.Location = new System.Drawing.Point(5, 44);
            this.enviadousb.Name = "enviadousb";
            this.enviadousb.Size = new System.Drawing.Size(184, 60);
            this.enviadousb.TabIndex = 25;
            //
            // groupBox3
            //
            this.groupBox3.Controls.Add(this.estadopic);
            this.groupBox3.Controls.Add(this.label2);
            this.groupBox3.Controls.Add(this.info);
            this.groupBox3.Controls.Add(this.label1);
            this.groupBox3.Controls.Add(this.bot_enviar);
            this.groupBox3.Controls.Add(this.comando);
            this.groupBox3.Controls.Add(this.label3);
            this.groupBox3.Controls.Add(this.botlimpiarusb);
            this.groupBox3.Controls.Add(this.enviadousb);
            this.groupBox3.Controls.Add(this.recibidousb);
            this.groupBox3.Controls.Add(this.label4);
            this.groupBox3.Location = new System.Drawing.Point(12, 12);
            this.groupBox3.Name = "groupBox3";
            this.groupBox3.Size = new System.Drawing.Size(416, 190);
            this.groupBox3.TabIndex = 6;
            this.groupBox3.TabStop = false;
            this.groupBox3.Text = "Comunicación USB";
            //
```

```
            // estadopic
            //
            this.estadopic.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.estadopic.Location = new System.Drawing.Point(59, 16);
            this.estadopic.Name = "estadopic";
            this.estadopic.Size = new System.Drawing.Size(133, 21);
            this.estadopic.TabIndex = 35;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(193, 27);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(63, 14);
            this.label2.TabIndex = 30;
            this.label2.Text = "Información";
            //
            // info
            //
            this.info.BackColor = System.Drawing.SystemColors.Control;
            this.info.FormattingEnabled = true;
            this.info.ItemHeight = 14;
            this.info.Location = new System.Drawing.Point(195, 44);
            this.info.Name = "info";
            this.info.Size = new System.Drawing.Size(211, 102);
            this.info.TabIndex = 29;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(196, 147);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(103, 14);
            this.label1.TabIndex = 28;
            this.label1.Text = "Envio de parámetro:";
            //
            // bot_enviar
            //
            this.bot_enviar.Location = new System.Drawing.Point(338,
155);
            this.bot_enviar.Name = "bot_enviar";
            this.bot_enviar.Size = new System.Drawing.Size(68, 29);
            this.bot_enviar.TabIndex = 1;
            this.bot_enviar.Text = "ENVIAR";
            this.bot_enviar.UseVisualStyleBackColor = true;
            this.bot_enviar.Click += new
System.EventHandler(this.bot_enviar_Click_1);
            //
            // comando
            //
            this.comando.Location = new System.Drawing.Point(196, 164);
            this.comando.MaxLength = 30;
            this.comando.Name = "comando";
            this.comando.Size = new System.Drawing.Size(136, 20);
            this.comando.TabIndex = 23;
            this.comando.TextChanged += new
System.EventHandler(this.comando_TextChanged_1);
            //
            // groupBox5
            //
```

```
            this.groupBox5.Controls.Add(this.operadora);
            this.groupBox5.Controls.Add(this.centromen);
            this.groupBox5.Controls.Add(this.label10);
            this.groupBox5.Controls.Add(this.pinbox);
            this.groupBox5.Controls.Add(this.label9);
            this.groupBox5.Controls.Add(this.label7);
            this.groupBox5.Location = new System.Drawing.Point(12, 208);
            this.groupBox5.Name = "groupBox5";
            this.groupBox5.Size = new System.Drawing.Size(186, 138);
            this.groupBox5.TabIndex = 28;
            this.groupBox5.TabStop = false;
            this.groupBox5.Text = "Parámetros modem GSM";
            //
            // operadora
            //
            this.operadora.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.operadora.Location = new System.Drawing.Point(85, 24);
            this.operadora.Name = "operadora";
            this.operadora.Size = new System.Drawing.Size(95, 21);
            this.operadora.TabIndex = 34;
            //
            // centromen
            //
            this.centromen.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.centromen.Location = new System.Drawing.Point(85, 52);
            this.centromen.Name = "centromen";
            this.centromen.Size = new System.Drawing.Size(95, 21);
            this.centromen.TabIndex = 22;
            this.centromen.TextChanged += new
System.EventHandler(this.centromen_TextChanged);
            //
            // label10
            //
            this.label10.AutoSize = true;
            this.label10.Location = new System.Drawing.Point(8, 53);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(67, 14);
            this.label10.TabIndex = 33;
            this.label10.Text = "Centro SMS:";
            //
            // pinbox
            //
            this.pinbox.Enabled = false;
            this.pinbox.Location = new System.Drawing.Point(35, 82);
            this.pinbox.MaxLength = 4;
            this.pinbox.Name = "pinbox";
            this.pinbox.Size = new System.Drawing.Size(46, 20);
            this.pinbox.TabIndex = 1;
            this.pinbox.TextChanged += new
System.EventHandler(this.pinbox_TextChanged);
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(8, 85);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(25, 14);
            this.label9.TabIndex = 30;
            this.label9.Text = "PIN:";
```

```
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Location = new System.Drawing.Point(8, 25);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(56, 14);
            this.label7.TabIndex = 24;
            this.label7.Text = "Operador:";
            //
            // bot_leerpic
            //
            this.bot_leerpic.Location = new System.Drawing.Point(447,
384);
            this.bot_leerpic.Name = "bot_leerpic";
            this.bot_leerpic.Size = new System.Drawing.Size(158, 32);
            this.bot_leerpic.TabIndex = 39;
            this.bot_leerpic.Text = "LEER PARÁMETROS";
            this.bot_leerpic.UseVisualStyleBackColor = true;
            this.bot_leerpic.Click += new
System.EventHandler(this.bot_leerpic_Click);
            //
            // bot_cargardatos
            //
            this.bot_cargardatos.Location = new System.Drawing.Point(447,
422);
            this.bot_cargardatos.Name = "bot_cargardatos";
            this.bot_cargardatos.Size = new System.Drawing.Size(158, 51);
            this.bot_cargardatos.TabIndex = 36;
            this.bot_cargardatos.Text = "CARGAR PARÁMETROS EN EL MODEM";
            this.bot_cargardatos.UseVisualStyleBackColor = true;
            this.bot_cargardatos.Click += new
System.EventHandler(this.bot_cargardatos_Click);
            //
            // teldest1
            //
            this.teldest1.Enabled = false;
            this.teldest1.Location = new System.Drawing.Point(139, 105);
            this.teldest1.MaxLength = 9;
            this.teldest1.Name = "teldest1";
            this.teldest1.Size = new System.Drawing.Size(95, 20);
            this.teldest1.TabIndex = 3;
            this.teldest1.TextChanged += new
System.EventHandler(this.teldest_TextChanged);
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(53, 108);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(80, 14);
            this.label8.TabIndex = 27;
            this.label8.Text = "Tel. destino #1:";
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.label17);
            this.groupBox1.Controls.Add(this.contsms2);
            this.groupBox1.Controls.Add(this.label18);
            this.groupBox1.Controls.Add(this.contsms1);
            this.groupBox1.Controls.Add(this.label19);
```

```
this.groupBox1.Controls.Add(this.teldest2);
this.groupBox1.Controls.Add(this.sms2);
this.groupBox1.Controls.Add(this.label6);
this.groupBox1.Controls.Add(this.sms1);
this.groupBox1.Controls.Add(this.label13);
this.groupBox1.Controls.Add(this.label8);
this.groupBox1.Controls.Add(this.teldest1);
this.groupBox1.Location = new System.Drawing.Point(204, 208);
this.groupBox1.Name = "groupBox1";
this.groupBox1.Size = new System.Drawing.Size(532, 138);
this.groupBox1.TabIndex = 37;
this.groupBox1.TabStop = false;
this.groupBox1.Text = "Mensajes SMS:";
//
// label17
//
this.label17.AutoSize = true;
this.label17.Location = new System.Drawing.Point(305, 75);
this.label17.Name = "label17";
this.label17.Size = new System.Drawing.Size(48, 14);
this.label17.TabIndex = 44;
this.label17.Text = "Quedan:";
//
// contsms2
//
this.contsms2.AutoSize = true;
this.contsms2.Location = new System.Drawing.Point(325, 89);
this.contsms2.Name = "contsms2";
this.contsms2.Size = new System.Drawing.Size(25, 14);
this.contsms2.TabIndex = 43;
this.contsms2.Text = "100";
//
// label18
//
this.label18.AutoSize = true;
this.label18.Location = new System.Drawing.Point(28, 72);
this.label18.Name = "label18";
this.label18.Size = new System.Drawing.Size(48, 14);
this.label18.TabIndex = 42;
this.label18.Text = "Quedan:";
//
// contsms1
//
this.contsms1.AutoSize = true;
this.contsms1.Location = new System.Drawing.Point(48, 86);
this.contsms1.Name = "contsms1";
this.contsms1.Size = new System.Drawing.Size(25, 14);
this.contsms1.TabIndex = 41;
this.contsms1.Text = "100";
//
// label19
//
this.label19.AutoSize = true;
this.label19.Location = new System.Drawing.Point(330, 112);
this.label19.Name = "label19";
this.label19.Size = new System.Drawing.Size(80, 14);
this.label19.TabIndex = 39;
this.label19.Text = "Tel. destino #2:";
//
// teldest2
//
```

```
            this.teldest2.Enabled = false;
            this.teldest2.Location = new System.Drawing.Point(416, 109);
            this.teldest2.MaxLength = 9;
            this.teldest2.Name = "teldest2";
            this.teldest2.Size = new System.Drawing.Size(95, 20);
            this.teldest2.TabIndex = 5;
            this.teldest2.TextChanged += new
System.EventHandler(this.teldest2_TextChanged);
            //
            // sms2
            //
            this.sms2.Location = new System.Drawing.Point(356, 22);
            this.sms2.MaxLength = 100;
            this.sms2.Multiline = true;
            this.sms2.Name = "sms2";
            this.sms2.Size = new System.Drawing.Size(155, 81);
            this.sms2.TabIndex = 4;
            this.sms2.TextChanged += new
System.EventHandler(this.sms2_TextChanged);
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(303, 22);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(47, 14);
            this.label6.TabIndex = 33;
            this.label6.Text = "SMS #2:";
            //
            // sms1
            //
            this.sms1.Location = new System.Drawing.Point(79, 19);
            this.sms1.MaxLength = 100;
            this.sms1.Multiline = true;
            this.sms1.Name = "sms1";
            this.sms1.Size = new System.Drawing.Size(155, 81);
            this.sms1.TabIndex = 2;
            this.sms1.TextChanged += new
System.EventHandler(this.sms1_TextChanged);
            //
            // label13
            //
            this.label13.AutoSize = true;
            this.label13.Location = new System.Drawing.Point(28, 19);
            this.label13.Name = "label13";
            this.label13.Size = new System.Drawing.Size(47, 14);
            this.label13.TabIndex = 24;
            this.label13.Text = "SMS #1:";
            //
            // groupBox4
            //
            this.groupBox4.Controls.Add(this.label16);
            this.groupBox4.Controls.Add(this.label15);
            this.groupBox4.Controls.Add(this.val);
            this.groupBox4.Controls.Add(this.label14);
            this.groupBox4.Controls.Add(this.pass);
            this.groupBox4.Controls.Add(this.label11);
            this.groupBox4.Controls.Add(this.smscon);
            this.groupBox4.Location = new System.Drawing.Point(12, 352);
            this.groupBox4.Name = "groupBox4";
            this.groupBox4.Size = new System.Drawing.Size(426, 121);
```

```
            this.groupBox4.TabIndex = 39;
            this.groupBox4.TabStop = false;
            this.groupBox4.Text = "SMS de control:";
            //
            // label16
            //
            this.label16.Location = new System.Drawing.Point(210, 16);
            this.label16.Name = "label16";
            this.label16.Size = new System.Drawing.Size(208, 59);
            this.label16.TabIndex = 31;
            this.label16.Text = "El valor introducido activará la
secuencia VERDE/AMBAR/ROJO si coincide, si no, a" +
                "ctivará el parpadeo VALOR veces.";
            this.label16.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
            //
            // label15
            //
            this.label15.Location = new System.Drawing.Point(6, 83);
            this.label15.Name = "label15";
            this.label15.Size = new System.Drawing.Size(77, 14);
            this.label15.TabIndex = 41;
            this.label15.Text = "valor: (1)";
            this.label15.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
            //
            // val
            //
            this.val.Location = new System.Drawing.Point(89, 80);
            this.val.MaxLength = 1;
            this.val.Name = "val";
            this.val.Size = new System.Drawing.Size(103, 20);
            this.val.TabIndex = 7;
            this.val.TextChanged += new
System.EventHandler(this.val_TextChanged);
            //
            // label14
            //
            this.label14.Location = new System.Drawing.Point(6, 57);
            this.label14.Name = "label14";
            this.label14.Size = new System.Drawing.Size(77, 14);
            this.label14.TabIndex = 39;
            this.label14.Text = "password: (4)";
            this.label14.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
            //
            // pass
            //
            this.pass.Location = new System.Drawing.Point(89, 55);
            this.pass.MaxLength = 4;
            this.pass.Name = "pass";
            this.pass.Size = new System.Drawing.Size(103, 20);
            this.pass.TabIndex = 6;
            this.pass.TextChanged += new
System.EventHandler(this.pass_TextChanged);
            //
            // label11
            //
            this.label11.AutoSize = true;
            this.label11.Location = new System.Drawing.Point(6, 16);
            this.label11.Name = "label11";
```

```
            this.label11.Size = new System.Drawing.Size(49, 14);
            this.label11.TabIndex = 30;
            this.label11.Text = "Formato:";
            //
            // smscon
            //
            this.smscon.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.smscon.Location = new System.Drawing.Point(59, 32);
            this.smscon.Name = "smscon";
            this.smscon.Size = new System.Drawing.Size(133, 18);
            this.smscon.TabIndex = 30;
            //
            // bot_limpiartodo
            //
            this.bot_limpiartodo.Location = new System.Drawing.Point(447,
361);
            this.bot_limpiartodo.Name = "bot_limpiartodo";
            this.bot_limpiartodo.Size = new System.Drawing.Size(158, 21);
            this.bot_limpiartodo.TabIndex = 32;
            this.bot_limpiartodo.Text = "LIMPIAR TODO";
            this.bot_limpiartodo.UseVisualStyleBackColor = true;
            this.bot_limpiartodo.Click += new
System.EventHandler(this.bot_limpiartodo_Click);
            //
            // PicModem
            //
            this.AcceptButton = this.bot_enviar;
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.None;
            this.CancelButton = this.salir;
            this.ClientSize = new System.Drawing.Size(743, 507);
            this.Controls.Add(this.pordefecto);
            this.Controls.Add(this.bot_limpiartodo);
            this.Controls.Add(this.bot_leerpic);
            this.Controls.Add(this.groupBox4);
            this.Controls.Add(this.groupBox1);
            this.Controls.Add(this.bot_cargardatos);
            this.Controls.Add(this.groupBox5);
            this.Controls.Add(this.groupBox3);
            this.Controls.Add(this.statusStrip1);
            this.Controls.Add(this.salir);
            this.Controls.Add(this.groupBox2);
            this.Font = new System.Drawing.Font("Arial", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.Margin = new System.Windows.Forms.Padding(2, 3, 2, 3);
            this.MaximizeBox = false;
            this.Name = "PicModem";
            this.Text = "PicModem";
            this.groupBox2.ResumeLayout(false);
            this.groupBox2.PerformLayout();
            this.statusStrip1.ResumeLayout(false);
            this.statusStrip1.PerformLayout();
            this.groupBox3.ResumeLayout(false);
            this.groupBox3.PerformLayout();
            this.groupBox5.ResumeLayout(false);
            this.groupBox5.PerformLayout();
```

```
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.groupBox4.ResumeLayout(false);
            this.groupBox4.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.Button bot_conectar;
        private System.Windows.Forms.Button salir;
        private System.Windows.Forms.Button bot_desconectar;
        private System.Windows.Forms.Button botlimpiarusb;
        private System.Windows.Forms.StatusStrip statusStrip1;
        private System.Windows.Forms.ToolStripStatusLabel statuslabel;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.ListBox recibidousb;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.ListBox enviadousb;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.TextBox pinbox;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.TextBox teldest1;
        public System.Windows.Forms.GroupBox groupBox5;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.Button bot_cargardatos;
        private System.Windows.Forms.Label centromen;
        private System.Windows.Forms.Label label12;
        private System.Windows.Forms.Button bot_enviar;
        private System.Windows.Forms.TextBox comando;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.ListBox tramasrec;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.ListBox info;
        public System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.TextBox sms1;
        private System.Windows.Forms.Label label13;
        private System.Windows.Forms.TextBox sms2;
        public System.Windows.Forms.GroupBox groupBox4;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.Label smscon;
        private System.Windows.Forms.Label label15;
        private System.Windows.Forms.TextBox val;
        private System.Windows.Forms.Label label14;
        private System.Windows.Forms.TextBox pass;
        private System.Windows.Forms.Label label16;
        private System.Windows.Forms.Button bot_leerpic;
        private System.Windows.Forms.TextBox tramarec;
        private System.Windows.Forms.Button botlimpiarserie;
        private System.Windows.Forms.Label label19;
        private System.Windows.Forms.TextBox teldest2;
        private System.Windows.Forms.Label label17;
        private System.Windows.Forms.Label contsms2;
        private System.Windows.Forms.Label label18;
```

```
            private System.Windows.Forms.Label contsms1;
            private System.Windows.Forms.Label operadora;
            private System.Windows.Forms.Button bot_limpiartodo;
            private System.Windows.Forms.Button bot_envserie;
            private System.Windows.Forms.Label estadopic;
            private System.Windows.Forms.Button pordefecto;
            //private AxMSCommLib.AxMSComm pserie;

        }
    }
```

8.4 Código fuente del software en PC, versión BÁSICA.

8.4.1 Picmodem.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics; //Clase para abrir página web
using System.IO.Ports;
using System.Threading;

namespace PicModem
{
    public partial class PicModem : Form
    {
        PicmodemAPI usbapi = new PicmodemAPI();
        SerialPort port = new SerialPort("COM1", 9600, Parity.None, 8,
StopBits.One);

        string cadinfo, cadinfo2;
        string spin, steldest1, steldest2, ssms1, ssms2, ssmscon;
        // strings para guardar el valor leido del PIC, para compararlo
antes de subirlo
        char[] bufferserie = new char[160];
        char[] result = new char[101];
        char[] valor = new char[20];
        uint online = 0;
        bool picleido = false;
        bool cargar = false;
        bool leer = false;
        bool fallo = false;
        bool picbusy = true;
        bool pcbusy = false;
        int leerflag = 0;       //para que lea el flag cada segundo
        int me = 0;
        int y = 0;  // iteraciones en la carga de datos
        int x = 0;
        int cincoseg = 0;
        int tics = 0;
        int ready3 = 0;     //se tienen que contar 3 tics con pic: ready,
para evitar falsos readys
        int esp5 = 0;   // variable de espera de 5 segundos
        bool[] datok = new bool[6]; // new bool -> los inicializa como
'false'
        bool[] dattx = new bool[6];
        // datos: 0=pin, 1=teldest1, 2=teldest2
        // 3=sms1, 4=sms2, 5=smscon
        // datok=datos leidos del pic
        // dattx=datos a enviar al pic,porque no se han leido o porque se
han tecleado


        public PicModem()
        {
            //inicializa los componentes del form
            InitializeComponent();
```

```
            inicializaciones();
        }

        private void inicializaciones()
        {
            System.Windows.Forms.Control.CheckForIllegalCrossThreadCalls
= false;   // daba un error
            //operadora.SelectedItem = 0;
            //centromen.Text = "656000311";

            bot_leerpic.Enabled = false;
            bot_cargardatos.Enabled = true;
            pinbox.Enabled = true;
            sms1.Enabled = true;
            sms2.Enabled = true;
            comando.Enabled = true;
            teldest1.Enabled = true;
            teldest2.Enabled = true;
            pass.Enabled = true;
            val.Enabled = true;
            progressBar1.Enabled = true;
            progressBar1.Step = 1;
            progressBar1.Visible = false;
            progressBar1.Maximum = 47;


            comando.Text = "";

            online = usbapi.Conexion();
            if (online == 1) //esta conectado?
            {
                statuslabel.Text = "Dispositivo conectado  <<ONLINE>>";
                cincoseg = 1;
            }
            else
            {
                statuslabel.Text = "Dispositivo no conectado
<<OFFLINE>>";
            }
        }
/*        private void leerpic()
        {
            pcbusy = true;
            progressBar1.Enabled = true;
            progressBar1.Visible = true;
            picleido = true;
            statuslabel.Text = "Leyendo datos desde el dispositivo...";

            if ((!fallo)&&(!picbusy))
            {
                comando.Text = "5";             // lectura de pin
                progressBar1.PerformStep();
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "6";     // lectura del centro de mensajes
                progressBar1.PerformStep();
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
```

```
            {
                comando.Text = "7";     // lectura del destinatario 1
                progressBar1.PerformStep();
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "8";     // lectura del destinatario 2
                progressBar1.PerformStep();
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "95";    // lectura del sms1
                progressBar1.PerformStep();
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "96";    // lectura del sms2
                progressBar1.PerformStep();
                enviarusb();
            }
            if ((!fallo) && (!picbusy))
            {
                comando.Text = "97";    // lectura del sms de control
                progressBar1.PerformStep();
                enviarusb();
            }
            progressBar1.PerformStep();


            if ((!fallo) && (!picbusy))
            {
                cadinfo = "DATOS ALMACENADOS EN PicModem:\n";
                cadinfo2 = "\n\nDATOS NECESARIOS:\n";
                if (datok[0])
                    cadinfo += "\n   PIN: " + pinbox.Text;
                else
                    cadinfo2 += "\n   PIN: contraseña de 4 dígitos";
                if (datok[1])
                    cadinfo += "\n   TELDEST1: " + teldest1.Text;
                else
                    cadinfo2 += "\n   TELDEST1: telefono destinatario 1
de 9 dígitos";
                if (datok[2])
                    cadinfo += "\n   TELDEST2: " + teldest2.Text;
                else
                    cadinfo2 += "\n   TELDEST2: telefono destinatario 2
de 9 dígitos";
                if (datok[3])
                    cadinfo += "\n   SMS1: " + sms1.Text;
                else
                    cadinfo2 += "\n   SMS1: mensaje 1 de hasta 100
caracteres";
                if (datok[4])
                    cadinfo += "\n   SMS2: " + sms2.Text;
                else
                    cadinfo2 += "\n   SMS2: mensaje 2 de hasta 100
caracteres";
                if (datok[5])
```

```
                    cadinfo += "\n   SMSCON: " + smscon.Text;
                else
                    cadinfo2 += "\n   SMSCON: mensaje consigna de 4+1
caracteres";

                if (!datok[0] && !datok[1] && !datok[2] && !datok[3] &&
!datok[4] && !datok[5])
                {
                    cadinfo += "\n   (VACÍO)";
                }
                if (cadinfo2 != "\n\nDATOS NECESARIOS:\n")
                    cadinfo += cadinfo2;

                progressBar1.Visible = false;
                progressBar1.Enabled = false;
                MessageBox.Show(cadinfo, "Información.",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                teldest1.Enabled = true;
                teldest2.Enabled = true;
            }
            pcbusy = false;

            statuslabel.Text = "Dispositivo conectado  <<ONLINE>>";
        }
*/
        private void enviarusb()
        {
            // Se enviarán pseudocomandos o comandos AT

            // se enviará un primer byte con el tipo de parametro a
enviar
            // y un segundo byte con el numero de caracteres que componen
el parametro
            // y por ultimo se recibiran los caracteres que componen el
valor
            // Ejemplo: 141234 <- tipo: 1 (pin), caracteres: 4 ,valor:
1234
            // Tipos:   1=pin, 2=tel destinatario1, 3=tel destinatario2
            // 5=solicitud pin, 6=solicitud centromen, 7=solicitud
teldest1, 8=solicitud teldest2
            // 91=sms1, 92=sms2, 93=sms control, 95=sol. sms1, 96=sol.
sms2, 97=sol. sms control

            int b, l, i;
            char cad1;              //caracter de handshaking
            string cad2;
            uint b1 = 0, orden = 0, nocars = 0, c, orden2 = 0;
            bool cat = false;       // es comando at?
            int pcom = 0;
            int delay = 100;
            pcbusy = true;
            l = comando.Text.Length;

            cad2 = comando.Text.ToUpper();

            orden = (uint)cad2[0];

            if ((orden > 48) && (orden <= 57)) //  si es un pseudocomando
            {
                cat = false;
                if (orden < 52) // <4
```

```
                pcom = 1;
            else
            {
                if (orden == 52)
                    pcom = 0;
                else
                    pcom = 2;
            }
            if (l > 1)
            {
                nocars = (uint)cad2[1];
                nocars -= 48;
            }
            else
            {
                nocars = 4;
            }

            switch (orden)
            {
                case 49:    // 1
                    if ((nocars == 4) && (nocars == l - 2))
                    {
                        pinbox.Text = "";
                        pinbox.Text += cad2[2];
                        pinbox.Text += cad2[3];
                        pinbox.Text += cad2[4];
                        pinbox.Text += cad2[5];
                    }
                    else
                    {
                        fallo = true;
                    }
                    break;

                case 50:    // 2
                    if ((nocars == 9) && (nocars == l - 2))
                    {

                    }
                    else
                    {
                        fallo = true;
                    }
                    break;

                case 51:    // 3
                    if ((nocars == 9) && (nocars == l - 2))
                    {
                    }
                    else
                    {
                        fallo = true;
                    }
                    break;

                case 53:    // 5
                    l = 6;
                    nocars = 4;     // con l=6 y nocars=4 se obtiene
"fallo=false"
                    break;
```

```
                    case 54:    // 6
                        l = 11;
                        nocars = 9;     // con l=11 y nocars=9 se obtiene
"fallo=false"
                        break;

                    case 55:    // 7
                        l = 11;
                        nocars = 9;     // con l=11 y nocars=9 se obtiene
"fallo=false"
                        break;

                    case 56:    // 8
                        l = 11;
                        nocars = 9;     // con l=11 y nocars=9 se obtiene
"fallo=false"
                        break;

                    case 57:    // 9
                        if (l > 1)
                            c = (uint)cad2[1];
                        else
                            c = 0;

                        switch (c)
                        {
                            case 49:    // si es 91

                                l = sms1.Text.Length;
                                nocars = (uint)l;
                                l = l + 2;          //ha de cumplir:
l=nocars+2

                                orden2 = 49;
                                pcom = 3;
                                cad2 = sms1.Text;
                                if (nocars == 0)
                                {
                                    fallo = true;
                                }
                                break;
                            case 50:    // si es 92
                                l = sms2.Text.Length;
                                nocars = (uint)l;
                                l = l + 2;          //ha de cumplir:
l=nocars+2

                                orden2 = 50;
                                pcom = 3;
                                cad2 = sms2.Text;
                                if (nocars == 0)
                                {
                                    fallo = true;
                                }
                                break;
                            case 51:    // si es 93
                                nocars = 5;
                                l = 7;          //ha de cumplir:
l=nocars+2

                                orden2 = 51;
                                pcom = 3;
                                cad2 = pass.Text + val.Text;
```

```
                                break;
                            case 53:    // si es 95
                                l = 102;
                                nocars = 100;
                                orden2 = 53;
                                pcom = 4;
                                break;
                            case 54:    // si es 96
                                l = 102;
                                nocars = 100;
                                orden2 = 54;
                                pcom = 4;
                                break;
                            case 55:    // si es 97
                                l = 7;
                                nocars = 5; // son 5 caracteres
                                orden2 = 55;
                                pcom = 4;
                                break;
                            case 0:     // si solo se ha escrito un
digito
                                fallo = true;
                                break;
                            default:
                                fallo = true;
                                break;
                        }
                        break;
                    default:
                        fallo = true;
                        break;
                }

                if (l != nocars + 2)
                {
                    fallo = true;
                }
                if ((l == nocars + 2) && (fallo == false))
                    fallo = false;

                if (!fallo)
                {
                    usbapi.enviaUSB(orden);     //envio primer byte

                    Thread.Sleep(delay);   // delay de 100ms
                    cad1 = (char)usbapi.recibeUSB();  // recibimos de 1
en 1 byte!!!

                    if (orden != cad1)  // si es diferente
                    {
                        if ((uint)cad1 != 4)
                        {
                            switch ((uint)cad1)
                            {
                                case 1:
                                    picbusy = false;
                                    break;
                                case 2:
                                    picbusy = false;
                                    break;
                                case 3:
```

```
                                    picbusy = false;
                                    break;
                                case 5:
                                    picbusy = false;
                                    break;
                                case 6:
                                    picbusy = false;
                                    break;
                                case 7:
                                    picbusy = false;
                                    break;
                                case 8:
                                    picbusy = false;
                                    break;
                                case 9:
                                    picbusy = false;
                                    break;
                                case 15:
                                    picbusy = false;
                                    break;
                                case 16:
                                    picbusy = false;
                                    break;
                                case 17:
                                    picbusy = false;
                                    break;
                                case 18:
                                    picbusy = false;
                                    break;
                                case 19:
                                    picbusy = false;
                                    break;
                                case 0:
                                    picbusy = true;
                                    break;
                            }
                        }
                        else
                        {
                            orden = 4;
                            fallo = true;
                            usbapi.enviaUSB(orden); // se envia byte 04
para indicar error
                            Thread.Sleep(delay);   // delay de 100ms
                            cad1 = (char)usbapi.recibeUSB();  //
recibimos retorno de fallo
                            MessageBox.Show("Ha ocurrido un error crítico
en la comunicación PC <--> PIC\nDesconecta el dispositivo.", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                            Application.Exit();
                        }
                    }
                    if ((pcom == 2) && !fallo) // comandos de solicitud
de datos 5,6,7 u 8
                    {
                        usbapi.enviaUSB(cad1);     //enviamos por 2ª vez
el primer byte
                        Thread.Sleep(delay);   // delay de 100ms
                        cad1 = (char)usbapi.recibeUSB();  // recibimos
1er dato
                        if (cad1 == 4)
```

```
                        {
                            fallo = true;
                        }
                        else
                        {
                            valor[0] = cad1;

                            c = (uint)cad1;
                            usbapi.enviaUSB(c);  // rebotamos d0
                        }
                    }
                    if ((pcom == 4) && !fallo) // comandos de solicitud
de datos 95,96 o 97
                    {
                        usbapi.enviaUSB(orden2);     //enviamos segundo
byte

                        Thread.Sleep(delay);   // delay de 100ms
                        cad1 = (char)usbapi.recibeUSB();

                        if (cad1 == 4)
                        {
                            fallo = true;
                        }
                        else
                        {
                            usbapi.enviaUSB(cad1);     //enviamos por 2ª
vez el segundo byte
                            Thread.Sleep(delay);   // delay de 100ms
                            cad1 = (char)usbapi.recibeUSB();  //
recibimos 1er dato
                            if (cad1 == 4)
                            {
                                fallo = true;
                                MessageBox.Show("Ha ocurrido un error
crítico en la comunicación PC <--> PIC\nDesconecta el dispositivo.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                Application.Exit();
                            }
                            else
                            {
                                if (cad1 == 255)     //si recibimos un
0xFF como 1er caracter
                                {                    //quiere decir que el
sms está vacío
                                    fallo = true;
                                    pcom = 0;
                                    pcbusy = false;
                                }
                                else
                                {
                                    result[0] = cad1;

                                    c = (uint)cad1;
                                    usbapi.enviaUSB(c);  // rebotamos d0
                                }
                            }
                        }
                    }
                    if (!fallo)    //si se ha enviado ok, se sigue
enviando
```

```
                    {
                        if (orden < 52) // si orden es 1,2 o 3
                        {
                            for (b = 1; b < nocars + 2; b++)
                            {
                                b1 = (uint)cad2[b]; // cad2=cadena a
enviar en mayusculas
                                // b1=valor ascii del caracter a enviar
                                usbapi.enviaUSB(b1);
                                Thread.Sleep(delay);   // delay de 100ms
                                cad1 = (char)usbapi.recibeUSB();
// recibimos de 1 en 1 byte!!!

                                result[b] = cad1;

                                if ((b1 != (uint)cad1) || ((int)cad1 ==
4))
                                {
                                    b = (int)nocars + 2;
                                    fallo = true;
                                    b1 = 4;
                                    usbapi.enviaUSB(b1); // se envia byte
04 para indicar error
                                }
                            }
                            if (!fallo)
                            {
                                b1 = 13;            // <CR>
                                usbapi.enviaUSB(b1);
                            }
                            pcbusy = false;
                        }
                        else
                        {
                            if (orden < 57)
                            {
                                // si es orden 5,6,7 u 8!!!!!!!
                                // arriba hemos enviado el 1er byte,hemos
recibido y
                                // confirmado el byte de 'dato 0'
                                // aqui leeremos el 'dato 1' y
posteriores,iremos rebotando
                                // los caracteres que vayamos recibiendo
                                for (b = 1; b < nocars + 1; b++)
                                {
                                    Thread.Sleep(delay);   // delay de
100ms
                                    cad1 = (char)usbapi.recibeUSB();
// <-- ojo!!!

                                    c = (uint)cad1;
                                    valor[b] = cad1;
                                    if ((c != 4) && (c != 13) && (c !=
0)) //si no hay error, no <CR> y no 0
                                    {
                                        usbapi.enviaUSB(c);
                                    }
                                    else
                                    {
                                        if (c == 13)    //si <CR>, es que
OK
```

```
                                                {
                                                    switch (orden)
                                                    {
                                                        case 53:    // 5 = PIN
                                                            if (valor[0] == 255)
                                                                datok[0] = false;
                                                            else
                                                            {
                                                                datok[0] = true;
                                                                pinbox.Text = "";
                                                                for (i = 0; i <
4; i++)
                                                                {
                                                                    pinbox.Text
+= valor[i];
                                                                }
                                                                spin =
pinbox.Text;
                                                            }
                                                            break;
                                                        case 54:    // 6 =
CENTROMEN
                                                            centromen.Text = "";
                                                            for (i = 0; i < 9;
i++)
                                                            {
                                                                if (valor[i] !=
255)
                                                                {
centromen.Text += valor[i];

                                                                }
                                                                else
                                                                {
centromen.Text += (char)(valor[i] - 192);

                                                                }
                                                            }
                                                            break;
                                                        case 55:    //7 =
TELDEST1
                                                            if (valor[0] == 255)
                                                                datok[1] = false;
                                                            else
                                                            {
                                                                datok[1] = true;
                                                                teldest1.Text =
""";
                                                                for (i = 0; i <
9; i++)
                                                                {
                                                                    teldest1.Text
+= valor[i];
                                                                }
                                                                steldest1 =
teldest1.Text;
                                                            }
                                                            break;
                                                        case 56:    //8 =
TELDEST2
                                                            if (valor[0] == 255)
```

```
                                                datok[2] = false;
                                            else
                                            {
                                                datok[2] = true;
                                                teldest2.Text =
"";
                                                for (i = 0; i <
9; i++)
                                                {
                                                    teldest2.Text
+= valor[i];
                                                }
                                                steldest2 =
teldest2.Text;
                                            }
                                            break;
                                        }
                                        pcbusy = false;
                                    }
                                    else
                                    {
                                        b = (int)nocars + 1;    //
para que no vuelva a entrar en el FOR
                                        fallo = true;
                                        pcbusy = false;
                                        MessageBox.Show("Ha ocurrido
un error crítico en la comunicación PC <--> PIC\nDesconecta el
dispositivo.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                        Application.Exit();

                                    }
                                }
                            }
                        }
                        else        //si es orden 9
                        {
                            if (pcom == 3) //envio datos    91, 92 o
93
                            {
                                usbapi.enviaUSB(orden2);     //
enviamos orden2
                                Thread.Sleep(delay);   // delay de
100ms
                                cad1 = (char)usbapi.recibeUSB();
// recibimos de 1 en 1 byte!!!

                                for (b = 0; b < nocars; b++)
                                {
                                    b1 = (uint)cad2[b]; //
cad2=cadena a enviar
                                    // b1=valor ascii del caracter a
enviar
                                    usbapi.enviaUSB(b1);
                                    Thread.Sleep(delay);   // delay
de 100ms
                                    cad1 = (char)usbapi.recibeUSB();
// recibimos de 1 en 1 byte!!!

                                    result[b] = cad1;
```

```
                                    if ((b1 != (uint)cad1) ||
((int)cad1 == 4))
                                    {
                                        b = (int)nocars;
                                        b1 = 4;
                                        usbapi.enviaUSB(b1); // se
envia byte 04 para indicar error
                                        pcbusy = false;
                                    }
                                }
                                b1 = 13;                // se envia
<CR>
                                usbapi.enviaUSB(b1);

                                Thread.Sleep(delay);   // delay de
100ms
                                cad1 = (char)usbapi.recibeUSB();

                                pcbusy = false;
                            }

                            else    //si pcom==4 , peticion de datos
                            {
                                // si es orden 9!!!!!!!
                                // arriba hemos enviado el 1er y 2º
byte,y repetido 2º byte.
                                // hemos recibido y confirmado el
byte de 'dato 0'
                                // aqui leeremos el 'dato 1' y
posteriores,iremos rebotando
                                // los caracteres que vayamos
recibiendo

                                for (b = 1; b < nocars + 1; b++)
//nocars=100 (0 a 99) smscon: nocars=5
                                {
                                    Thread.Sleep(delay);   // delay
de 100ms
                                    cad1 = (char)usbapi.recibeUSB();
// <-- ojo!!!

                                    c = (uint)cad1;
                                    result[b] = cad1;
                                    if ((c != 4) && (c != 13)) //si
no hay error y no <CR>
                                    {
                                        usbapi.enviaUSB(c);
                                    }
                                    if (c == 13)    //si <CR>
                                    {
                                        switch (orden2)
                                        {
                                            case 53:    // 5 = SMS1
                                                if (result[0] == 255)
                                                {
                                                    datok[3] = false;
                                                    sms1.Text =
"(vacío)";
                                                }
                                                else
                                                {
```

```
                                                    datok[3] = true;
                                                    sms1.Text = "";
                                                    for (i = 0; i <
b; i++)
                                                    {
                                                        sms1.Text +=
result[i];
                                                    }
                                                    ssms1 =
sms1.Text;
                                                }
                                                break;
                                            case 54:    // 6 = SMS2
                                                if (result[0] == 255)
                                                {
                                                    datok[4] = false;
                                                    sms2.Text =
"(vacío)";
                                                }
                                                else
                                                {
                                                    datok[4] = true;
                                                    sms2.Text = "";
                                                    for (i = 0; i <
b; i++)
                                                    {
                                                        sms2.Text +=
result[i];
                                                    }
                                                    ssms2 =
sms2.Text;
                                                }
                                                break;
                                            case 55:    // 7 = SMSCON
                                                if (result[0] == 255)
                                                {
                                                    datok[5] = false;
                                                    smscon.Text =
"(vacío)";
                                                    pass.Text = "";
                                                    val.Text = "";
                                                }
                                                else
                                                {
                                                    datok[5] = true;
                                                    smscon.Text = "";
                                                    pass.Text = "";
                                                    ssmscon = "";
                                                    for (i = 0; i <
4; i++)
                                                    {
                                                        smscon.Text
+= result[i];
                                                        pass.Text +=
result[i];
                                                        ssmscon +=
result[i];
                                                    }
                                                    smscon.Text +=
',' + result[4];
```

```
                                                    ssmscon += "," +
result[4];
                                                    val.Text = "";
                                                    val.Text +=
result[4];
                                                }
                                                break;
                                            }
                                            pcbusy = false;
                                            b = (int)nocars;    // para
que no vuelva a entrar en el FOR
                                        }
                                        if (c == 4)
                                        {
                                            b = (int)nocars;    // para
que no vuelva a entrar en el FOR
                                            pcbusy = false;
                                            fallo = true;
                                        }

                                    }
                                    //leer confirmacion
                                    pcom = 0;
                                }
                            }
                        }
                    }
                    comando.Text = null;
                    //fallo = false;
                }

                else  // >>>>>>>>>>>>>>>>>> comando AT <<<<<<<<<<<<<<<<<<<<
                {
                    if (orden == 65)    //si es 'A'
                    {
                        b1 = 0;
                        cat = true;
                        for (b = 0; b < l; b++)
                        {
                            b1 = (uint)cad2[b]; // cad2=cadena a enviar en
mayusculas
                            // b1=valor ascii del caracter a enviar
                            usbapi.enviaUSB(b1);
                            Thread.Sleep(delay);   // delay de 100ms
                            cad1 = (char)usbapi.recibeUSB();        //
recibimos de 1 en 1 byte!!!

                            if (b1 != (uint)cad1)
                            {
                                Thread.Sleep(delay);   // delay de 100ms
                                cad1 = (char)usbapi.recibeUSB();        //
recibimos de 1 en 1 byte!!!

                                if ((uint)cad1 == 4)
                                {
                                    cat = false;
                                    b = l;
                                    b1 = 4;
                                    Thread.Sleep(delay);   // delay de 100ms
```

```
                                usbapi.enviaUSB(b1);        // se envia
byte 04 para indicar error
                                pcbusy = false;
                            }
                        }
                    }
                    // una vez enviados todos los caracteres del comando
AT
                    if (cat == true)
                    {
                        //enviar <CR> final de trama
                        b1 = 13;
                        Thread.Sleep(delay);   // delay de 100ms
                        usbapi.enviaUSB(b1);
                        Thread.Sleep(delay);   // delay de 100ms
                        cad1 = (char)usbapi.recibeUSB();        //
recibimos de 1 en 1 byte!!!
                    }

                    // trama enviada ok
                    comando.Text = null;
                    pcbusy = false;
                }
            }
            cat = false;
            pcom = 0;
            //fallo = false;
        }

        private void salir_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            char tim;
            uint itim;
            leerflag++;
            online = usbapi.Conexion();     //ajustada una temporizacion
de 500ms

            if (online == 1)                //esta conectado?
            {
                tics++;
                if (!picleido)
                {
                    progressBar1.Visible = true;
                    progressBar1.PerformStep();
                    if (esp5 == 0)
                    {
                        esp5 = 1;
                        statuslabel.Text = "Esperando respuesta del
dispositivo...";
                    }
                }
/*                else
                {
                    statuslabel.Text = "Dispositivo conectado
<<ONLINE>>";
                }*/
```

```
                bot_leerpic.Enabled = true;
                bot_cargardatos.Enabled = true;
                comando.Enabled = true;
                if ((cincoseg == 0) && (me == 0))
                {
                    cincoseg = 1;
                }
                me = 2;
                if ((!picleido) && (!picbusy) && (tics > 10) && (ready3
>= 3)) //tics>10: 5 seg
                {
                    tics = 0;
                    ready3 = 0;
                    statuslabel.Text = "Leyendo datos desde el
dispositivo...";
                    leer = true;
                }
                if (me == 1)
                    me = 2;

            }
            else
            {
                bot_leerpic.Enabled = false;
                bot_cargardatos.Enabled = false;
                tics = 0;
                //bot_enviar.Enabled = false;       se gestiona en
comando_textchanged

                statuslabel.Text = "Dispositivo no conectado
<<OFFLINE>>";

                if (me == 1)
                {
                    picleido = false;
                    cargar = false;
                    leer = false;
                    fallo = false;
                    picbusy = true;
                    pcbusy = false;
                    y = 0;
                    x = 0;
                    cincoseg = 0;

                    datok[0] = false;
                    datok[1] = false;
                    datok[2] = false;
                    datok[3] = false;
                    datok[4] = false;
                    datok[5] = false;

                    dattx[0] = false;
                    dattx[1] = false;
                    dattx[2] = false;
                    dattx[3] = false;
                    dattx[4] = false;
                    dattx[5] = false;
                }

                if (me == 0)
                {
```

```
                    me = 1;
                    MessageBox.Show("El dispositivo PicModem no está
conectado.", "Advertencia", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                }
                if (me == 2)
                {
                    me = 1;
                    MessageBox.Show("Se ha desconectado el dispositivo
PicModem.", "Información", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                }
            }
            if ((online == 1) && (!pcbusy) && (leerflag >= 2))
            {
                leerflag = 0;   //asi entra cada segundo
                pcbusy = true;
                tim = (char)usbapi.recibeUSB();
                itim = (uint)tim;
                if (itim != 4)
                {
                    switch (itim)
                    {
                        case 1:
                            picbusy = false;
                            ready3++;
                            if (operadora.Text == "No disponible.")
                            {
                                if ((!fallo) && (!picbusy))
                                {
                                    operadora.Text = "Leyendo...";
                                    centromen.Text = "Leyendo...";
                                    comando.Text = "6";     // lectura
del centro de mensajes
                                    enviarusb();
                                }
                            }
                            break;
                        case 2:
                            picbusy = false;
                            ready3++;
                            break;
                        case 3:
                            picbusy = false;
                            break;
                        case 5:
                            picbusy = false;
                            break;
                        case 6:
                            picbusy = false;
                            break;
                        case 7:
                            picbusy = false;
                            break;
                        case 8:
                            picbusy = false;
                            break;
                        case 9:
                            picbusy = false;
                            break;
                        case 15:
```

```
                            picbusy = false;
                            break;
                        case 16:
                            picbusy = false;
                            break;
                        case 17:
                            picbusy = false;
                            break;
                        case 18:
                            picbusy = false;
                            break;
                        case 19:
                            picbusy = false;
                            break;
                        case 20:
                            picbusy = false;
                            break;
                        case 21:
                            picbusy = false;
                            break;
                        case 22:
                            picbusy = false;
                            break;

                        case 0:
                            picbusy = true;
                            ready3 = 0;
                            break;
                    }
                }
                else
                    fallo = true;

                pcbusy = false;
            }
            //------------- CARGAR ----------------
            if ((cargar) && (!fallo) && (!picbusy))
            {
                y++;
                pcbusy = true;

                switch (y)
                {
                    case 1:
                        if (spin != pinbox.Text)
                        {
                            comando.Text = "14" + pinbox.Text;       //
envio de pin
                            enviarusb();
                        }
                        break;
                    case 2:
                        if (steldest1 != teldest1.Text)
                        {
                            comando.Text = "29" + teldest1.Text;     //
envio de teldest1
                            enviarusb();
                        }
                        break;
                    case 3:
                        if (steldest2 != teldest2.Text)
```

```
                        {
                            comando.Text = "39" + teldest2.Text;     //
envio de teldest2
                            enviarusb();
                        }
                        break;
                    case 4:
                        if (ssms1 != sms1.Text)
                        {
                            comando.Text = "91";     // envio de sms1
                            enviarusb();
                        }
                        break;
                    case 5:
                        if (ssms2 != sms2.Text)
                        {
                            comando.Text = "92";     // envio de sms2
                            enviarusb();
                        }
                        break;
                    case 6:
                        if (ssmscon != smscon.Text)
                        {
                            comando.Text = "93";     // envio de smscon
                            enviarusb();
                        }
                        cargar = false;
                        y = 0;
                        break;
                }
            }
            //------------- LEER ----------------
            if ((leer) && (!fallo) && (!picbusy))
            {
                x++;
                pcbusy = true;
                picleido = true;

                switch (x)
                {
                    case 1:
                        comando.Text = "5";     // lectura de pin
                        progressBar1.PerformStep();
                        enviarusb();
                        break;
                    case 2:
                        comando.Text = "6";     // lectura del centro de
mensajes
                        progressBar1.PerformStep();
                        enviarusb();
                        break;
                    case 3:
                        comando.Text = "7";     // lectura del
destinatario 1
                        progressBar1.PerformStep();
                        enviarusb();
                        break;
                    case 4:
                        comando.Text = "8";     // lectura del
destinatario 2
                        progressBar1.PerformStep();
```

```
                        enviarusb();
                        break;
                    case 5:
                        comando.Text = "95";    // lectura del sms1
                        progressBar1.PerformStep();
                        enviarusb();
                        break;
                    case 6:
                        comando.Text = "96";    // lectura del sms2
                        progressBar1.PerformStep();
                        enviarusb();
                        break;
                    case 7:
                        comando.Text = "97";    // lectura del sms de
control
                        progressBar1.PerformStep();
                        enviarusb();
                        leer = false;
                        x = 0;
                        cadinfo = "DATOS ALMACENADOS EN PicModem:\n";
                        cadinfo2 = "\n\nDATOS NECESARIOS:\n";
                        if (datok[0])
                            cadinfo += "\n   PIN: " + pinbox.Text;
                        else
                            cadinfo2 += "\n   PIN: contraseña de 4
dígitos";
                        if (datok[1])
                            cadinfo += "\n   TELDEST1: " + teldest1.Text;
                        else
                            cadinfo2 += "\n   TELDEST1: telefono
destinatario 1 de 9 dígitos";
                        if (datok[2])
                            cadinfo += "\n   TELDEST2: " + teldest2.Text;
                        else
                            cadinfo2 += "\n   TELDEST2: telefono
destinatario 2 de 9 dígitos";
                        if (datok[3])
                            cadinfo += "\n   SMS1: " + sms1.Text;
                        else
                            cadinfo2 += "\n   SMS1: mensaje 1 de hasta
100 caracteres";
                        if (datok[4])
                            cadinfo += "\n   SMS2: " + sms2.Text;
                        else
                            cadinfo2 += "\n   SMS2: mensaje 2 de hasta
100 caracteres";
                        if (datok[5])
                            cadinfo += "\n   SMSCON: " + smscon.Text;
                        else
                            cadinfo2 += "\n   SMSCON: mensaje consigna de
4+1 caracteres";
                        if (!datok[0] && !datok[1] && !datok[2] &&
!datok[3] && !datok[4] && !datok[5])
                        {
                            cadinfo += "\n   (VACÍO)";
                        }
                        if (cadinfo2 != "\n\nDATOS NECESARIOS:\n")
                            cadinfo += cadinfo2;
                        progressBar1.Maximum = 30;
                        progressBar1.Visible = false;
                        progressBar1.Enabled = false;
```

```
                        MessageBox.Show(cadinfo, "Información.",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                        teldest1.Enabled = true;
                        teldest2.Enabled = true;
                        break;
                    }
                    pcbusy = false;
                }

                if (fallo)
                {
                    cargar = false;
                    leer = false;
                    y = 0;
                    x = 0;
                    fallo = false;
                }
            }

            private void comando_TextChanged_1(object sender, EventArgs e)
            {
                if (comando.Text != "")
                {
                    if (comando.Text[0] == '0')
                    {
                        comando.Text = "";
                    }
                }
                string aux;
                aux = comando.Text;
            }
            private void teldest_TextChanged(object sender, EventArgs e)
            {
                teldest1.Enabled = true;
                if (teldest1.Text != "")
                {
                    if ((teldest1.Text[0] != '6') && (teldest1.Text[0] !=
255))
                    {
                        teldest1.Text = "";
                    }
                }
            }
            private void sms1_TextChanged(object sender, EventArgs e)
            {
                int i;
                sms1.Enabled = true;
                i = 100 - sms1.Text.Length;
                contsms1.Text = Convert.ToString(i);
                if (sms1.Text.Length > 0)
                {
                    if (sms1.Text[0] == '?')
                    {
                        sms1.Text = "";
                    }
                }
            }
            private void centromen_TextChanged(object sender, EventArgs e)
            {
                switch (centromen.Text)
                {
```

```
                case "656000311":
                    operadora.Text = "Orange";
                    break;
                case "607003110":
                    operadora.Text = "Vodafone";
                    break;
                case "609090909":
                    operadora.Text = "Movistar";
                    break;
                case "622996111":
                    operadora.Text = "Yoigo";
                    break;
                case "?????????":
                    operadora.Text = "No disponible.";
                    centromen.Text = "No disponible.";
                    break;
                default:
                    if (operadora.Text != "No disponible.")
                        operadora.Text = "(desconocido)";
                    break;
            }
        }
        private void pinbox_TextChanged(object sender, EventArgs e)
        {
            pinbox.Enabled = true;
        }
        private void pass_TextChanged(object sender, EventArgs e)
        {
            smscon.Text = "";
            if ((pass.Text.Length == 4) && (val.Text.Length == 1))
            {
                smscon.Text = pass.Text + val.Text;
            }
        }
        private void val_TextChanged(object sender, EventArgs e)
        {
            smscon.Text = "";
            if ((pass.Text.Length == 4) && (val.Text.Length == 1))
            {
                smscon.Text = pass.Text + val.Text;
            }
        }
        private void bot_leerpic_Click(object sender, EventArgs e)
        {
            limpiartodo();
            progressBar1.Enabled = true;
            progressBar1.Maximum = 8;
            progressBar1.Value = 0;
            statuslabel.Text = "Leyendo datos desde el dispositivo...";
            x = 0;
            picleido = false;
            leer = true;
        }
        private void teldest2_TextChanged(object sender, EventArgs e)
        {
            teldest2.Enabled = true;
            if (teldest2.Text != "")
            {
                if ((teldest2.Text[0] != '6') && (teldest2.Text[0] !=
255))
                {
```

```
                teldest2.Text = "";
            }
        }
    }
    private void sms2_TextChanged(object sender, EventArgs e)
    {
        int i;
        sms2.Enabled = true;
        i = 100 - sms2.Text.Length;
        contsms2.Text = Convert.ToString(i);
        if (sms2.Text.Length > 0)
        {
            if (sms2.Text[0] == '?')
            {
                sms2.Text = "";
            }
        }
    }

    private void bot_cargardatos_Click(object sender, EventArgs e)
    {
        cadinfo = "Existen campos sin rellenar.\n\n";

        if (pinbox.Text.Length == 4)
            dattx[0] = true;
        else
            cadinfo += "PIN.\n";

        if (teldest1.Text.Length == 9)
            dattx[1] = true;
        else
            cadinfo += "Teléfono destinatario 1.\n";

        if (teldest2.Text.Length == 9)
            dattx[2] = true;
        else
            cadinfo += "Teléfono destinatario 2.\n";

        if (sms1.Text.Length > 0)
            dattx[3] = true;
        else
            cadinfo += "SMS 1.\n";

        if (sms2.Text.Length > 0)
            dattx[4] = true;
        else
            cadinfo += "SMS 2.\n";

        if (smscon.Text != "")
            dattx[5] = true;
        else
            cadinfo += "SMS DE CONTROL.\n";

        if (dattx[0] && dattx[1] && dattx[2] && dattx[3] && dattx[4]
&& dattx[5])
        {
            cargar = true;
            //Thread.Sleep(20);   // delay de 20ms
        }
        else
        {
```

```
                MessageBox.Show(cadinfo, "¡ERROR!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void bot_limpiartodo_Click(object sender, EventArgs e)
        {
            limpiartodo();
        }
        private void limpiartodo()
        {
            int i;
            sms1.Text = "";
            sms2.Text = "";
            pinbox.Text = "";
            pass.Text = "";
            val.Text = "";
            teldest1.Text = "";
            teldest2.Text = "";
            centromen.Text = "";
            operadora.Text = "";
            for (i = 0; i <= 5; i++)
            {
                dattx[i] = false;
            }
        }
    }
}
```

8.4.2 Picmodem.Designer.cs

```
namespace PicModem
{
    partial class PicModem
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources =
new System.ComponentModel.ComponentResourceManager(typeof(PicModem));
            this.salir = new System.Windows.Forms.Button();
            this.statusStrip1 = new System.Windows.Forms.StatusStrip();
            this.statuslabel = new
System.Windows.Forms.ToolStripStatusLabel();
            this.timer1 = new
System.Windows.Forms.Timer(this.components);
            this.comando = new System.Windows.Forms.TextBox();
            this.groupBox5 = new System.Windows.Forms.GroupBox();
            this.operadora = new System.Windows.Forms.Label();
            this.centromen = new System.Windows.Forms.Label();
            this.label10 = new System.Windows.Forms.Label();
            this.pinbox = new System.Windows.Forms.TextBox();
            this.label9 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.bot_leerpic = new System.Windows.Forms.Button();
            this.bot_cargardatos = new System.Windows.Forms.Button();
            this.teldest1 = new System.Windows.Forms.TextBox();
            this.label8 = new System.Windows.Forms.Label();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.label17 = new System.Windows.Forms.Label();
            this.contsms2 = new System.Windows.Forms.Label();
            this.label18 = new System.Windows.Forms.Label();
            this.contsms1 = new System.Windows.Forms.Label();
            this.label19 = new System.Windows.Forms.Label();
```

```
            this.teldest2 = new System.Windows.Forms.TextBox();
            this.sms2 = new System.Windows.Forms.TextBox();
            this.label6 = new System.Windows.Forms.Label();
            this.sms1 = new System.Windows.Forms.TextBox();
            this.label13 = new System.Windows.Forms.Label();
            this.groupBox4 = new System.Windows.Forms.GroupBox();
            this.label15 = new System.Windows.Forms.Label();
            this.val = new System.Windows.Forms.TextBox();
            this.label14 = new System.Windows.Forms.Label();
            this.pass = new System.Windows.Forms.TextBox();
            this.label11 = new System.Windows.Forms.Label();
            this.smscon = new System.Windows.Forms.Label();
            this.bot_limpiartodo = new System.Windows.Forms.Button();
            this.pordefecto = new System.Windows.Forms.Button();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.label3 = new System.Windows.Forms.Label();
            this.pictureBox2 = new System.Windows.Forms.PictureBox();
            this.progressBar1 = new System.Windows.Forms.ProgressBar();
            this.statusStrip1.SuspendLayout();
            this.groupBox5.SuspendLayout();
            this.groupBox1.SuspendLayout();
            this.groupBox4.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit(
);

((System.ComponentModel.ISupportInitialize)(this.pictureBox2)).BeginInit(
);
            this.SuspendLayout();
            //
            // salir
            //
            this.salir.DialogResult =
System.Windows.Forms.DialogResult.Cancel;
            this.salir.Location = new System.Drawing.Point(384, 369);
            this.salir.Name = "salir";
            this.salir.Size = new System.Drawing.Size(68, 74);
            this.salir.TabIndex = 5;
            this.salir.Text = "SALIR";
            this.salir.UseVisualStyleBackColor = true;
            this.salir.Click += new
System.EventHandler(this.salir_Click);
            //
            // statusStrip1
            //
            this.statusStrip1.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.statuslabel});
            this.statusStrip1.Location = new System.Drawing.Point(0,
450);
            this.statusStrip1.Name = "statusStrip1";
            this.statusStrip1.Size = new System.Drawing.Size(464, 22);
            this.statusStrip1.TabIndex = 7;
            this.statusStrip1.Text = "PFC - Control Modem GSM via PIC.";
            //
            // statuslabel
            //
            this.statuslabel.Name = "statuslabel";
            this.statuslabel.Size = new System.Drawing.Size(0, 17);
            //
            // timer1
```

```
            //
            this.timer1.Enabled = true;
            this.timer1.Interval = 500;
            this.timer1.Tick += new
System.EventHandler(this.timer1_Tick);
            //
            // comando
            //
            this.comando.Location = new System.Drawing.Point(190, 383);
            this.comando.MaxLength = 30;
            this.comando.Name = "comando";
            this.comando.Size = new System.Drawing.Size(136, 20);
            this.comando.TabIndex = 23;
            this.comando.Visible = false;
            this.comando.TextChanged += new
System.EventHandler(this.comando_TextChanged_1);
            //
            // groupBox5
            //
            this.groupBox5.Controls.Add(this.operadora);
            this.groupBox5.Controls.Add(this.centromen);
            this.groupBox5.Controls.Add(this.label10);
            this.groupBox5.Controls.Add(this.pinbox);
            this.groupBox5.Controls.Add(this.label9);
            this.groupBox5.Controls.Add(this.label7);
            this.groupBox5.Location = new System.Drawing.Point(7, 97);
            this.groupBox5.Name = "groupBox5";
            this.groupBox5.Size = new System.Drawing.Size(186, 119);
            this.groupBox5.TabIndex = 28;
            this.groupBox5.TabStop = false;
            this.groupBox5.Text = "Parámetros modem GSM";
            //
            // operadora
            //
            this.operadora.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.operadora.Location = new System.Drawing.Point(85, 24);
            this.operadora.Name = "operadora";
            this.operadora.Size = new System.Drawing.Size(95, 21);
            this.operadora.TabIndex = 34;
            //
            // centromen
            //
            this.centromen.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.centromen.Location = new System.Drawing.Point(85, 52);
            this.centromen.Name = "centromen";
            this.centromen.Size = new System.Drawing.Size(95, 21);
            this.centromen.TabIndex = 22;
            this.centromen.TextChanged += new
System.EventHandler(this.centromen_TextChanged);
            //
            // label10
            //
            this.label10.AutoSize = true;
            this.label10.Location = new System.Drawing.Point(8, 53);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(67, 14);
            this.label10.TabIndex = 33;
            this.label10.Text = "Centro SMS:";
            //
```

```
            // pinbox
            //
            this.pinbox.Enabled = false;
            this.pinbox.Location = new System.Drawing.Point(35, 82);
            this.pinbox.MaxLength = 4;
            this.pinbox.Name = "pinbox";
            this.pinbox.Size = new System.Drawing.Size(46, 20);
            this.pinbox.TabIndex = 1;
            this.pinbox.TextChanged += new
System.EventHandler(this.pinbox_TextChanged);
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(8, 85);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(25, 14);
            this.label9.TabIndex = 30;
            this.label9.Text = "PIN:";
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Location = new System.Drawing.Point(8, 25);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(56, 14);
            this.label7.TabIndex = 24;
            this.label7.Text = "Operador:";
            //
            // bot_leerpic
            //
            this.bot_leerpic.Location = new System.Drawing.Point(7, 396);
            this.bot_leerpic.Name = "bot_leerpic";
            this.bot_leerpic.Size = new System.Drawing.Size(143, 21);
            this.bot_leerpic.TabIndex = 39;
            this.bot_leerpic.Text = "LEER PARÁMETROS";
            this.bot_leerpic.UseVisualStyleBackColor = true;
            this.bot_leerpic.Click += new
System.EventHandler(this.bot_leerpic_Click);
            //
            // bot_cargardatos
            //
            this.bot_cargardatos.Location = new System.Drawing.Point(7,
423);
            this.bot_cargardatos.Name = "bot_cargardatos";
            this.bot_cargardatos.Size = new System.Drawing.Size(143, 20);
            this.bot_cargardatos.TabIndex = 36;
            this.bot_cargardatos.Text = "CARGAR PARÁMETROS";
            this.bot_cargardatos.UseVisualStyleBackColor = true;
            this.bot_cargardatos.Click += new
System.EventHandler(this.bot_cargardatos_Click);
            //
            // teldest1
            //
            this.teldest1.Enabled = false;
            this.teldest1.Location = new System.Drawing.Point(119, 105);
            this.teldest1.MaxLength = 9;
            this.teldest1.Name = "teldest1";
            this.teldest1.Size = new System.Drawing.Size(95, 20);
            this.teldest1.TabIndex = 3;
```

```
            this.teldest1.TextChanged += new
System.EventHandler(this.teldest_TextChanged);
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(33, 108);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(80, 14);
            this.label8.TabIndex = 27;
            this.label8.Text = "Tel. destino #1:";
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.label17);
            this.groupBox1.Controls.Add(this.contsms2);
            this.groupBox1.Controls.Add(this.label18);
            this.groupBox1.Controls.Add(this.contsms1);
            this.groupBox1.Controls.Add(this.label19);
            this.groupBox1.Controls.Add(this.teldest2);
            this.groupBox1.Controls.Add(this.sms2);
            this.groupBox1.Controls.Add(this.label6);
            this.groupBox1.Controls.Add(this.sms1);
            this.groupBox1.Controls.Add(this.label13);
            this.groupBox1.Controls.Add(this.label8);
            this.groupBox1.Controls.Add(this.teldest1);
            this.groupBox1.Location = new System.Drawing.Point(7, 225);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(445, 138);
            this.groupBox1.TabIndex = 37;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Mensajes SMS:";
            //
            // label17
            //
            this.label17.AutoSize = true;
            this.label17.Location = new System.Drawing.Point(230, 72);
            this.label17.Name = "label17";
            this.label17.Size = new System.Drawing.Size(48, 14);
            this.label17.TabIndex = 44;
            this.label17.Text = "Quedan:";
            //
            // contsms2
            //
            this.contsms2.AutoSize = true;
            this.contsms2.Location = new System.Drawing.Point(250, 86);
            this.contsms2.Name = "contsms2";
            this.contsms2.Size = new System.Drawing.Size(25, 14);
            this.contsms2.TabIndex = 43;
            this.contsms2.Text = "100";
            //
            // label18
            //
            this.label18.AutoSize = true;
            this.label18.Location = new System.Drawing.Point(8, 72);
            this.label18.Name = "label18";
            this.label18.Size = new System.Drawing.Size(48, 14);
            this.label18.TabIndex = 42;
            this.label18.Text = "Quedan:";
            //
            // contsms1
```

```
            //
            this.contsms1.AutoSize = true;
            this.contsms1.Location = new System.Drawing.Point(28, 86);
            this.contsms1.Name = "contsms1";
            this.contsms1.Size = new System.Drawing.Size(25, 14);
            this.contsms1.TabIndex = 41;
            this.contsms1.Text = "100";
            //
            // label19
            //
            this.label19.AutoSize = true;
            this.label19.Location = new System.Drawing.Point(255, 109);
            this.label19.Name = "label19";
            this.label19.Size = new System.Drawing.Size(80, 14);
            this.label19.TabIndex = 39;
            this.label19.Text = "Tel. destino #2:";
            //
            // teldest2
            //
            this.teldest2.Enabled = false;
            this.teldest2.Location = new System.Drawing.Point(341, 106);
            this.teldest2.MaxLength = 9;
            this.teldest2.Name = "teldest2";
            this.teldest2.Size = new System.Drawing.Size(95, 20);
            this.teldest2.TabIndex = 5;
            this.teldest2.TextChanged += new
System.EventHandler(this.teldest2_TextChanged);
            //
            // sms2
            //
            this.sms2.Location = new System.Drawing.Point(281, 19);
            this.sms2.MaxLength = 100;
            this.sms2.Multiline = true;
            this.sms2.Name = "sms2";
            this.sms2.Size = new System.Drawing.Size(155, 81);
            this.sms2.TabIndex = 4;
            this.sms2.TextChanged += new
System.EventHandler(this.sms2_TextChanged);
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(228, 19);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(47, 14);
            this.label6.TabIndex = 33;
            this.label6.Text = "SMS #2:";
            //
            // sms1
            //
            this.sms1.Location = new System.Drawing.Point(59, 19);
            this.sms1.MaxLength = 100;
            this.sms1.Multiline = true;
            this.sms1.Name = "sms1";
            this.sms1.Size = new System.Drawing.Size(155, 81);
            this.sms1.TabIndex = 2;
            this.sms1.TextChanged += new
System.EventHandler(this.sms1_TextChanged);
            //
            // label13
            //
```

```
            this.label13.AutoSize = true;
            this.label13.Location = new System.Drawing.Point(8, 19);
            this.label13.Name = "label13";
            this.label13.Size = new System.Drawing.Size(47, 14);
            this.label13.TabIndex = 24;
            this.label13.Text = "SMS #1:";
            //
            // groupBox4
            //
            this.groupBox4.Controls.Add(this.label15);
            this.groupBox4.Controls.Add(this.val);
            this.groupBox4.Controls.Add(this.label14);
            this.groupBox4.Controls.Add(this.pass);
            this.groupBox4.Controls.Add(this.label11);
            this.groupBox4.Controls.Add(this.smscon);
            this.groupBox4.Location = new System.Drawing.Point(199, 97);
            this.groupBox4.Name = "groupBox4";
            this.groupBox4.Size = new System.Drawing.Size(253, 119);
            this.groupBox4.TabIndex = 39;
            this.groupBox4.TabStop = false;
            this.groupBox4.Text = "SMS de control:";
            //
            // label15
            //
            this.label15.Location = new System.Drawing.Point(6, 83);
            this.label15.Name = "label15";
            this.label15.Size = new System.Drawing.Size(77, 14);
            this.label15.TabIndex = 41;
            this.label15.Text = "valor: (1)";
            this.label15.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
            //
            // val
            //
            this.val.Location = new System.Drawing.Point(89, 80);
            this.val.MaxLength = 1;
            this.val.Name = "val";
            this.val.Size = new System.Drawing.Size(103, 20);
            this.val.TabIndex = 7;
            this.val.TextChanged += new
System.EventHandler(this.val_TextChanged);
            //
            // label14
            //
            this.label14.Location = new System.Drawing.Point(6, 57);
            this.label14.Name = "label14";
            this.label14.Size = new System.Drawing.Size(77, 14);
            this.label14.TabIndex = 39;
            this.label14.Text = "password: (4)";
            this.label14.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
            //
            // pass
            //
            this.pass.Location = new System.Drawing.Point(89, 55);
            this.pass.MaxLength = 4;
            this.pass.Name = "pass";
            this.pass.Size = new System.Drawing.Size(103, 20);
            this.pass.TabIndex = 6;
            this.pass.TextChanged += new
System.EventHandler(this.pass_TextChanged);
```

```
            //
            // label11
            //
            this.label11.AutoSize = true;
            this.label11.Location = new System.Drawing.Point(6, 16);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(49, 14);
            this.label11.TabIndex = 30;
            this.label11.Text = "Formato:";
            //
            // smscon
            //
            this.smscon.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.smscon.Location = new System.Drawing.Point(59, 32);
            this.smscon.Name = "smscon";
            this.smscon.Size = new System.Drawing.Size(133, 18);
            this.smscon.TabIndex = 30;
            //
            // bot_limpiartodo
            //
            this.bot_limpiartodo.Location = new System.Drawing.Point(7,
369);
            this.bot_limpiartodo.Name = "bot_limpiartodo";
            this.bot_limpiartodo.Size = new System.Drawing.Size(143, 21);
            this.bot_limpiartodo.TabIndex = 32;
            this.bot_limpiartodo.Text = "LIMPIAR TODO";
            this.bot_limpiartodo.UseVisualStyleBackColor = true;
            this.bot_limpiartodo.Click += new
System.EventHandler(this.bot_limpiartodo_Click);
            //
            // pordefecto
            //
            this.pordefecto.Location = new System.Drawing.Point(0, 0);
            this.pordefecto.Name = "pordefecto";
            this.pordefecto.Size = new System.Drawing.Size(75, 23);
            this.pordefecto.TabIndex = 0;
            //
            // pictureBox1
            //
            this.pictureBox1.Location = new System.Drawing.Point(0, 0);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(100, 50);
            this.pictureBox1.TabIndex = 0;
            this.pictureBox1.TabStop = false;
            //
            // label3
            //
            this.label3.Font = new System.Drawing.Font("Comic Sans MS",
14.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.label3.ForeColor = System.Drawing.Color.Navy;
            this.label3.Location = new System.Drawing.Point(12, 9);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(255, 64);
            this.label3.TabIndex = 43;
            this.label3.Text = "Control Modem GSM desde
microcontrolador.";
            //
            // pictureBox2
            //
```

```
            this.pictureBox2.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox2.Image")));
            this.pictureBox2.Location = new System.Drawing.Point(353, 9);
            this.pictureBox2.Name = "pictureBox2";
            this.pictureBox2.Size = new System.Drawing.Size(74, 82);
            this.pictureBox2.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.AutoSize;
            this.pictureBox2.TabIndex = 44;
            this.pictureBox2.TabStop = false;
            //
            // progressBar1
            //
            this.progressBar1.Location = new System.Drawing.Point(265,
454);
            this.progressBar1.Maximum = 10;
            this.progressBar1.Name = "progressBar1";
            this.progressBar1.Size = new System.Drawing.Size(178, 16);
            this.progressBar1.TabIndex = 45;
            //
            // PicModem
            //
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.None;
            this.CancelButton = this.salir;
            this.ClientSize = new System.Drawing.Size(464, 472);
            this.Controls.Add(this.progressBar1);
            this.Controls.Add(this.pictureBox2);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.bot_limpiartodo);
            this.Controls.Add(this.bot_leerpic);
            this.Controls.Add(this.groupBox4);
            this.Controls.Add(this.comando);
            this.Controls.Add(this.groupBox1);
            this.Controls.Add(this.bot_cargardatos);
            this.Controls.Add(this.groupBox5);
            this.Controls.Add(this.statusStrip1);
            this.Controls.Add(this.salir);
            this.Font = new System.Drawing.Font("Arial", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.Margin = new System.Windows.Forms.Padding(2, 3, 2, 3);
            this.MaximizeBox = false;
            this.Name = "PicModem";
            this.Text = "PicModem";
            this.statusStrip1.ResumeLayout(false);
            this.statusStrip1.PerformLayout();
            this.groupBox5.ResumeLayout(false);
            this.groupBox5.PerformLayout();
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.groupBox4.ResumeLayout(false);
            this.groupBox4.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox2)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();
```

```
        }

        #endregion

        private System.Windows.Forms.Button salir;
        private System.Windows.Forms.StatusStrip statusStrip1;
        private System.Windows.Forms.ToolStripStatusLabel statuslabel;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.TextBox pinbox;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.TextBox teldest1;
        public System.Windows.Forms.GroupBox groupBox5;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.Button bot_cargardatos;
        private System.Windows.Forms.Label centromen;
        private System.Windows.Forms.TextBox comando;
        public System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.TextBox sms1;
        private System.Windows.Forms.Label label13;
        private System.Windows.Forms.TextBox sms2;
        public System.Windows.Forms.GroupBox groupBox4;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.Label smscon;
        private System.Windows.Forms.Label label15;
        private System.Windows.Forms.TextBox val;
        private System.Windows.Forms.Label label14;
        private System.Windows.Forms.TextBox pass;
        private System.Windows.Forms.Button bot_leerpic;
        private System.Windows.Forms.Label label19;
        private System.Windows.Forms.TextBox teldest2;
        private System.Windows.Forms.Label label17;
        private System.Windows.Forms.Label contsms2;
        private System.Windows.Forms.Label label18;
        private System.Windows.Forms.Label contsms1;
        private System.Windows.Forms.Label operadora;
        private System.Windows.Forms.Button bot_limpiartodo;
        private System.Windows.Forms.Button pordefecto;
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.PictureBox pictureBox2;
        private System.Windows.Forms.ProgressBar progressBar1;
        //private AxMSCommLib.AxMSComm pserie;
    }
}
```

8.5 Código fuente de la API común para las dos versiones.

8.5.1 PicmodemAPI.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Runtime.InteropServices; // Clase para importar DLL

using PVOID = System.IntPtr;
using DWORD = System.UInt32;

namespace PicModem
{
    unsafe public class PicmodemAPI
    {
        #region Definición de los Strings: EndPoint y VID_PID
        string vid_pid_norm = "vid_04d8&pid_0008";
        string out_pipe = "\\MCHP_EP1";
        string in_pipe = "\\MCHP_EP1";
        #endregion
        #region Funciones importadas de la DLL: mpusbapi.dll
        [DllImport("mpusbapi.dll")]
        private static extern DWORD _MPUSBGetDLLVersion();
        [DllImport("mpusbapi.dll")]
        private static extern DWORD _MPUSBGetDeviceCount(string
pVID_PID);
        [DllImport("mpusbapi.dll")]
        private static extern void* _MPUSBOpen(DWORD instance, string
pVID_PID, string pEP, DWORD dwDir, DWORD dwReserved);
        [DllImport("mpusbapi.dll")]
        private static extern DWORD _MPUSBRead(void* handle, void* pData,
DWORD dwLen, DWORD* pLength, DWORD dwMilliseconds);
        [DllImport("mpusbapi.dll")]
        private static extern DWORD _MPUSBWrite(void* handle, void*
pData, DWORD dwLen, DWORD* pLength, DWORD dwMilliseconds);
        [DllImport("mpusbapi.dll")]
        private static extern DWORD _MPUSBReadInt(void* handle, DWORD*
pData, DWORD dwLen, DWORD* pLength, DWORD dwMilliseconds);
        [DllImport("mpusbapi.dll")]
        private static extern bool _MPUSBClose(void* handle);
        #endregion

        void* myOutPipe;
        void* myInPipe;

        static void Main()
        {
            Application.EnableVisualStyles();
            Application.Run(new PicModem());
        }

        public void OpenPipes()
        {
            DWORD selection = 0;

            myOutPipe = _MPUSBOpen(selection, vid_pid_norm, out_pipe, 0,
0);
            myInPipe = _MPUSBOpen(selection, vid_pid_norm, in_pipe, 1,
0);
```

```
        }

        public void ClosePipes()
        {
            _MPUSBClose(myOutPipe);
            _MPUSBClose(myInPipe);
        }

        private void SendPacket(byte* SendData, DWORD SendLength)
        {
            uint SendDelay = 20;

            DWORD SentDataLength;

            OpenPipes();
            _MPUSBWrite(myOutPipe, (void*)SendData, SendLength,
&SentDataLength, SendDelay);
            ClosePipes();
        }

        private void ReceivePacket(byte* ReceiveData, DWORD*
ReceiveLength)
        {
            uint ReceiveDelay = 20;

            DWORD ExpectedReceiveLength = *ReceiveLength;

            OpenPipes();
            _MPUSBRead(myInPipe, (void*)ReceiveData,
ExpectedReceiveLength, ReceiveLength, ReceiveDelay);
            ClosePipes();
        }

        public uint recibeUSB()
        {
            uint rx=0;
            byte* receive_buf = stackalloc byte[1];

            DWORD RecvLength = 1;

            ReceivePacket(receive_buf, &RecvLength);
            rx = (uint)receive_buf[0];
            return rx;
        }

        public void enviaUSB(uint valor)
        {
            byte* send_buf = stackalloc byte[1];
            send_buf[0] = (byte)valor;
            SendPacket(send_buf, 1);    //envia el byte
        }

        public uint Conexion()
        {
            uint usb_connection;
            usb_connection = _MPUSBGetDeviceCount(vid_pid_norm);
            return usb_connection;
        }
    }
}
```

8.6 Código del archivo *picmodem.inf*.

```
[Version]
Signature = "$WINDOWS NT$"
Class = %ClassName%
ClassGuid = {4D36E911-E325-11CE-BFC1-08002BE10318}
Provider = %MFGNAME%
DriverVer = 05/13/2006, 1.0.0.0
CatalogFile = picmodem.cat

[DestinationDirs]
DefaultDestDir = 12
PicModem.ClassCopyFiles = 11

;------------------------------------------------------------------------
; Class Install Sections
;------------------------------------------------------------------------

[ClassInstall32]
AddReg = PicModem.ClassReg
CopyFiles = PicModem.ClassCopyFiles

[PicModem.ClassReg]
HKR,,,0,%ClassName%
HKR,,Class,,%ClassDesc%
HKR,,Icon,,11
HKR,,Installer32,,"picmodemci.dll,PicModemClassInstaller"

[PicModem.ClassCopyFiles]
picmodemci.dll

;------------------------------------------------------------------------
; Device Install Sections
;------------------------------------------------------------------------

[Manufacturer]
%MFGNAME% = Standard

[Standard]
%DESCRIPTION% = DriverInstall, USB\VID_04D8&PID_0008

[SourceDisksNames]
1 = %INSTDISK%,,,""

[SourceDisksFiles]
mchpusb.sys = 1,,
picmodemci.dll = 1,,

;------------------------------------------------------------------------
; Windows 2000/XP Sections
;------------------------------------------------------------------------

[DriverInstall.NT]
CopyFiles = DriverCopyFiles

[DriverCopyFiles]
mchpusb.sys
```