ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Projecto Fin de Carrera

Diseño e implantación de un sistema basado en Lector de Tarjeta Estudiante UPCT para control presencial.

AUTOR: José García Marín
DIRECTOR: Alfonso Aniorte Carbonell
Septiemre/ 2013

Autor	José García Marín
E-mail del Autor	correaljose@hotmail.com
Director(es)	Alfonso Aniorte Carbonell
E-mail del Director	alfonso.aniorte@upct.es
Título del PFC	Diseño e implantación de un sistema basado en lector de Tarjeta Estudiante UPCT para control presencial.

Resumen

La disposición por parte de los alumnos de la Universidad Politécnica de Cartagena de un chip inteligente integrado en su tarjeta de estudiante, permite controlar mediante lectores, dispositivos asociados y aplicaciones, la presencia o ausencia del alumno en un recinto.

Con este planteamiento inicial, se desarrolla en esta obra una aplicación capaz de leer el DNI de la tarjeta de estudiante y guardar en una base de datos la información sobre la fecha y hora del registro.

Así mismo esta obra pretende ser los cimientos para proyectos posteriores, por lo que en ella se exponen las características y el funcionamiento de las tarjetas inteligentes así como la comunicación entre tarjeta, lector y PC.

Titulación	Ingeniero Técnico de Telecomunicaciones especialidad en Telemática
Departamento	Electrónica, Tecnología de Computadoras y Proyectos
Fecha de Presentación	Septiembre – 20103

Índice General

Índice de Figuras.

Índice de tablas.

1 Introducción.

1.1 Objetivos del proyecto.

El prestigioso psicólogo estadounidense Abraham Maslow sostiene en su obra "*A Theory of Human Motivation"* de 1943, que las necesidades de seguridad son inherentes al ser humano, solo la necesidad fisiológica está antes que la necesidad de seguridad en su famosa *Pirámide de Maslow*.

La necesidad de seguridad es para las organizaciones, empresas o instituciones tan importante como lo es para las personas, una organización quizá no necesite seguridad física, moral o familiar, pero, sin duda alguna, necesita seguridad para sus recursos y propiedades.

Los sistemas de control de acceso satisfacen la necesidad de dar o denegar acceso a los recursos de una organización y así mismo saber quien y cuando los usa. Existen de diversos tipos, más y menos tecnológicos, desde un simple vigilante de seguridad hasta un avanzado escáner de retina.

El objetivo principal de esta obra es llevar a cabo el desarrollo de un sistema para el control de acceso a uno o varios recintos por medio de la tarjeta de estudiante de la Universidad Politécnica de Cartagena.

Las principales pesquisas que se llevarán a cabo para la consecución del objetivo de esta obra son:

- Estudio hardware y software sobre smart card, y más concretamente sobre la smart card de estudiante UPCT.
- Estudio de los protocolos de comunicación entre smart card y lector.
- Estudio de los protocolos de comunicación entre lector y PC.
- Estudio del protocolo de comunicaciones RS232 y su uso con microcontroladores PIC.
- Desarrollo del software necesario para la comunicación PIC/PC
- Desarrollo de una aplicación para el control de acceso a recintos mediante smart card estudiante UPCT.
- Desarrollo de una base de datos que contenga información sobre los alumnos y los permisos que estos tienen asociados.
- Desarrollo de una aplicación para la gestión de la base de datos de alumnos y permisos.

1.2. Resumen de la memoria.

La memoria se ha dividido en los siguientes capítulos intentando ofrecer una visión clara de todas las fases del proyecto:

- **Capítulo 1**: Enunciado y objetivos del proyecto y resumen del contenido de la memoria.
- **Capítulo 2**: Estado del arte, es decir, la base teórica sobre la que se basa el proyecto.
- **Capítulo 3**: En este capítulo se exponen aquellos aspectos del proyecto sujetos a distintas alternativas y las características más importantes de cada una.
- **Capítulo 4**: En este capitulo se explica la elección de los componentes con los que se llevará a cavo el proyecto y las características principales de estos.
- **Capítulo 5**: Descripción del sistema y diseño de sus componentes.
- **Capítulo 6**: En este capítulo se explica como se ha llevado a cabo el proceso de implementación del sistema para el control de acceso, objeto de esta obra.
- **Capítulo 7**: En este capitulo se muestran los resultados obtenidos de la experimentación con el sistema en varios equipos.
- **Capítulo 8**: Conclusiones extraídas y posibles líneas de desarrollo futuras a partir del estado final del proyecto.
- **Capítulo 9**: Sección en la que figuran los anexos relacionados con este proyecto.
- **Capitulo 10**: Bibliografía

2. Estado de la técnica.

En este capítulo se explica primero brevemente la evolución histórica de las tarjetas, para centrar después el estudio en las Smart Card o también conocidas como tarjetas inteligentes. Se detallará su estructura interna, sus componentes, como se protegen los datos que contienen, y especialmente se mostraran las pautas seguidas en la comunicación con los dispositivos lectores. En la segunda parte de este capítulo se hablará de los dispositivos lectores, se dará un vistazo a que tipos de estos existen y a sus características más importantes. Y por último, la tercera sección tratará sobre microcontroladores, evolución, características y las familias más populares de estos.

2.1 Smart Card.

Hoy en día, sería tan difícil imaginar un mundo sin tarjetas inteligentes como un mundo sin teléfonos móviles. Sin embargo, esto no siempre fue así, todo tiene un principio, y el de estos dispositivos fue en la década de los 50 en Estados Unidos.

La primera tarjeta de plástico de pago que se extendió fue lanzada por Diners Club a principios de la segunda mitad del siglo XX, pero fue con la entrada en este sector de VISA y MasterCard cuando realmente se produjo la gran expansión de esta tecnología, primero en Estados Unidos, luego en Europa y finalmente por el resto del mundo. Inicialmente estos dispositivos eran simples tarjetas de plástico serigrafiadas con textos estampados en relieve, la autenticidad de las tarjetas se basaba en la dificultad de reproducir su diseño.

La primera fue inventada por Frank McNamara en 1950, quien emitió su tarjeta para 200 clientes para que la pudieran utilizar en 27 restaurantes de Nueva York, de donde vino la designación de Diners' Club. En 1958 American Express presentó su versión de una tarjeta de crédito universal. En 1958 el Bank of América emitió la primera tarjeta bancaria, la BankAmericard (actual VISA).

Figura 2.1 Diners´Club, inventada por Frank McNamara en 1950

El primer avance tecnológico importante se produjo con la inserción de una banda magnética sobre estos dispositivos, esta banda magnética no solo permitía la lectura de

datos digitales guardados en una tarjeta, sino que aumentaba muy considerablemente las características de seguridad de esta.

La banda magnética fue inventado por IBM en 1960, está compuesta por partículas ferromagnéticas incrustadas en una matriz de resina, es capaz de almacenar cierta cantidad de información mediante una codificación determinada que polariza dichas partículas. La banda magnética es grabada o leída mediante contacto físico pasándola a través de una cabeza lectora/escritora gracias al fenómeno de la inducción magnética. Fue inventado por IBM en 1960.

En aplicaciones estándar de tarjetas identificación, como las usadas para las transacciones financieras, la información contenida en la banda magnética se organiza en diferentes pistas. El formato y estructura de datos de estas pistas están regulados por los estándares internacionales ISO7813 (para las pistas 1 y 2) e ISO4909 (para la pista 3).

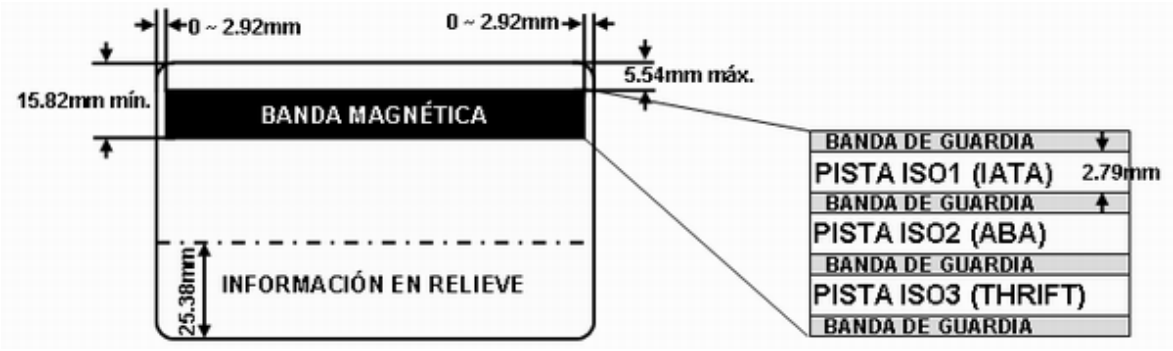

Figura 2.2 Tarjeta con banda magnética.

La siguiente generación de tarjetas son las ICC, tarjetas con circuito integrado. Tienen el aspecto de la *figura 2.3*. El circuito integrado presente en muchas ICC es simplemente una memoria EPROM serie, lo que supone que este circuito únicamente puede almacenar datos y estos datos solo pueden ser escritos una vez (Característica de la memoria tipo EPROM), por lo tanto estas tarjetas no son reutilizables son desechadas una vez consumido su saldo. Las tarjetas para las cabinas telefónicas son un ejemplo de esta tecnología. Los datos almacenados en las tarjetas de este tipo pueden ser leídos sin ningún tipo de restricciones, aunque pueden ser cifrados antes de ser escritos. Existen tarjetas de memoria que aportan control de acceso mediante PIN (Personal Identificación Number o simplemente Clave de Acceso), pero siguen siendo básicamente memorias.

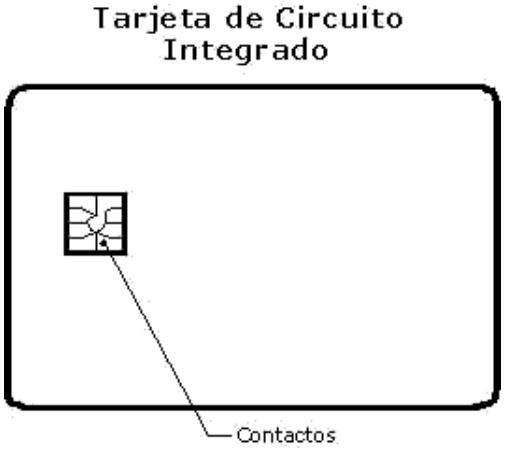

Figura 2.3 Aspecto de una ICC

Con el mismo aspecto y sin dejar de ser tarjetas con circuito integrado (ICC integrated circuit card), al adaptar a este ICC un microprocesador, se obtiene un avance importantísimo, y aumenta enormemente la cantidad de aplicaciones para las que son útiles estas tarjetas, asimismo los mecanismos de seguridad aumentan en complejidad y potencia. A este tipo de tarjeta se la conoce como Smart Card, y cuya traducción al español suele ser tarjeta inteligente o tarjeta microprogramada, probablemente sean las más usadas, son utilizadas en GSM, como tarjetas de crédito, DNI electrónico, control de personal, etc.

El último avance significativo en el mundo de las tarjetas son las ICCs sin contactos, estos dispositivos se comunican con los lectores mediante radiofrecuencia, aunque ya se usan, se sigue investigado en este campo y en sus posibilidades. Actualmente se comercializan también tarjetas hibridas, tarjetas sin contactos al a cual se le agrega un segundo chip de contacto. Ambos pueden ser o chips microprocesadores o simples chips de memoria. El chip sin contacto es generalmente usado en aplicaciones que requieren transacciones rápidas. Por ejemplo el transporte, mientras que el chip de contacto es generalmente utilizado en aplicaciones que requieren de alta seguridad como las bancarias. Un ejemplo de tarjeta híbrida es la tarjeta de identificación MyKad en Malasia.

La tarjeta de estudiante de la UPCT es de tipo Smart Card, es decir una tarjeta cuyo circuito integrado está compuesto de una memoria más un microprocesador, por este motivo los siguientes 4 puntos tratarán sobre la estructura interna, la seguridad, los mecanismos de comunicación y la programación de este tipo de tarjetas.

2.1.1 Estructura interna.

En este apartado se exponen los elementos internos de la tarjeta y se explica cual es la función de cada elemento. La memoria interna protagoniza un apartado propio y extenso por tratarse de un elemento sumamente importante.

2.1.1.1 Componentes

Como se ha comentado con anterioridad una Smart Card es un soporte de plástico de dimensiones determinadas más un circuito integrado (en adelante ICC), dentro de este están todos los elementos necesarios para dar soporte a las aplicaciones de la tarjeta.

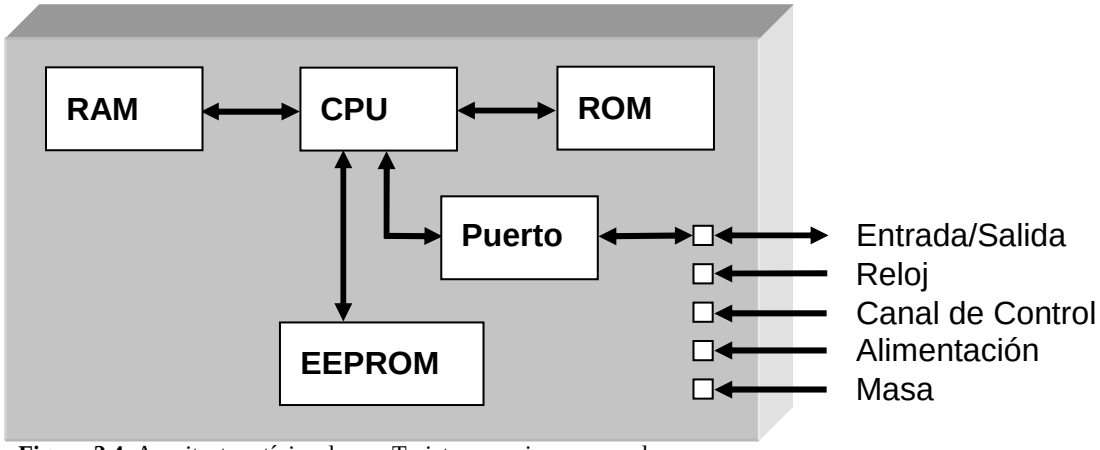

Figura 2.4 Arquitectura típica de una Tarjeta con microprocesador.

A continuación se detallan estos elementos y su función:

- CPU: Es el procesador, suele ser de un ancho de palabra de 8 bits, funciona a unos 5 MHz y a 5 voltios. Puede tener módulos hardware para operaciones criptográficas.

- ROM: Es la memoria interna, en la que está el sistema operativo, las rutinas del protocolo de comunicaciones los algoritmos de seguridad de alto nivel por software. Su capacidad oscila entre los 12 y 30 KB y no se puede reescribir, se inicializa durante el proceso de fabricación.

- RAM: Se trata de la memoria volátil que necesita el procesador.

- EPROM o EEPROM: Memoria de almacenamiento en la que se encuentra el sistema de ficheros, las claves de seguridad, los datos usados por las aplicaciones y las aplicaciones que se ejecutan en la tarjeta. El acceso a esta memoria está protegido por el sistema operativo.

- Puerto de entrada/salida, normalmente consisten en un simple registro.

2.1.1.2 La memoria

La memoria EEPROM, es el disco duro de la Tarjeta. En los Sistemas Operativos modernos, su división es la siguiente:

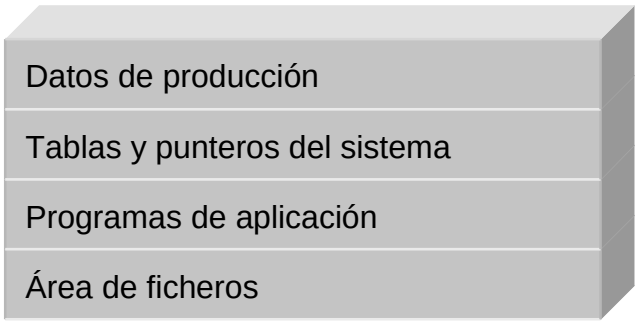

Figura 2.5 Ejemplo de división de la EEPROM.

- Primero están almacenados algunos datos de producción que pueden ser números fijos, y por lo tanto no varían, y se utilizan para funciones específicas. Esta zona suele estar protegida frente a accesos no autorizados por algún mecanismo hardware. Esta zona habitualmente es de 32 bytes.

- Después de esta zona, están las tablas y los punteros del sistema. Estos se graban durante la fase de fabricación y en conjunto, con el programa de la ROM, forman el Sistema Operativo. Para asegurar que el sistema funciona perfectamente, esta sección de la EEPROM está protegida con un checksum, que es calculado antes de cada acceso a esta zona. Si se encuentra algún error en la comparación, el Sistema Operativo dejará de usar esta zona de memoria.

- La siguiente zona contiene los códigos de las aplicaciones adicionales del sistema. En algunos casos esta zona, está protegida frente a posibles alteraciones mediante el uso de un checksum. Los datos contenidos también pueden ser algoritmos que no están almacenados en la ROM por falta de espacio.

- Al final de la memoria EEPROM, esta la zona que contiene la estructura de ficheros. Esta zona no está protegida por ningún mecanismo pero generalmente existen módulos hardware o software orientados a proteger ficheros. En esta zona es donde el usuario podrá grabar sus aplicaciones y sus datos. El tiempo de escritura en la EEPROM es alto y varía entre una y otra marca (dato orientativo 1msg/byte).

La RAM, es la memoria donde el microprocesador realiza sus operaciones. En casos de necesidad, el SO puede utilizar parte de la EEPROM como RAM, con la desventaja de que la escritura será mucha más lenta.

2.1.1.2.1 El Sistema de Ficheros.

Las Tarjetas actuales, incluyen auténticos sistemas de administración de ficheros que siguen una estructura jerárquica. Estos programas están muy minimizados para reducir el uso de memoria.

Los Sistemas Operativos más recientes, están orientados a trabajar con objetos, esto quiere decir, que todos los datos referentes a un fichero, están contenidos en él mismo. Los ficheros están divididos en dos secciones distintas: la primera es la cabecera y contiene datos referentes a la estructura del fichero y a las condiciones de acceso del mismo. La otra sección es el cuerpo del fichero, que contiene los datos útiles.

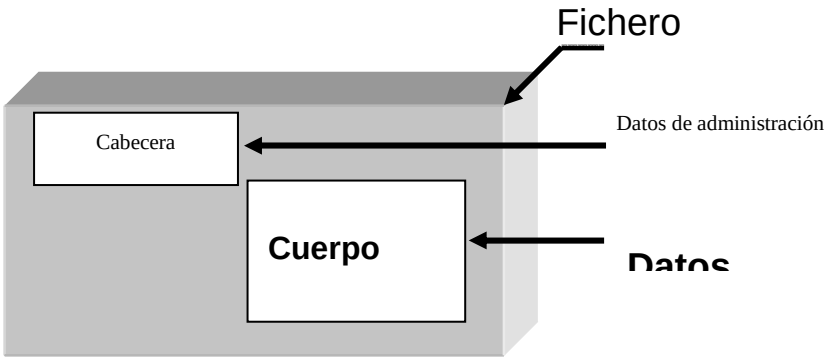

Figura 2.6 Estructura de un fichero en una Tarjeta Inteligente.

Otra consecuencia del funcionamiento orientado a objetos, es que para efectuar cambios en el contenido de un fichero, éste debe ser primero seleccionado con la correspondiente instrucción. Esta estructura está especificada en el ISO/IEC 7816−4 y es similar a los sistemas DOS o UNIX. Se tiene un directorio raíz (en DOS c:\) y a partir de él cuelgan los ficheros y directorios restantes. En las Tarjetas, tanto el directorio raíz con los demás directorios, son tratados como ficheros especiales, así, el directorio raíz es llamado Fichero Maestro o MF (Master File), los directorios, Ficheros Dedicados o DFs (Dedicated Files) y los ficheros normales se llaman Ficheros Elementales o EFs (Elementary Files). De estos últimos, podemos diferenciar los EF del sistema y los del usuario.

Es bastante habitual guardar todos los datos referentes a una determinada aplicación con un mismo Directorio (Fichero Dedicado). Con esto se consiguen unas estructuras más claras y organizadas, si el usuario desea aumentar el número de aplicaciones de su Tarjeta, bastará con crear un nuevo Directorio (DF) que contenga los nuevos datos. Si la Tarjeta se usa para una sola aplicación, los datos del usuario pueden estar contenidos simplemente en el Fichero Maestro (Directorio Raíz).

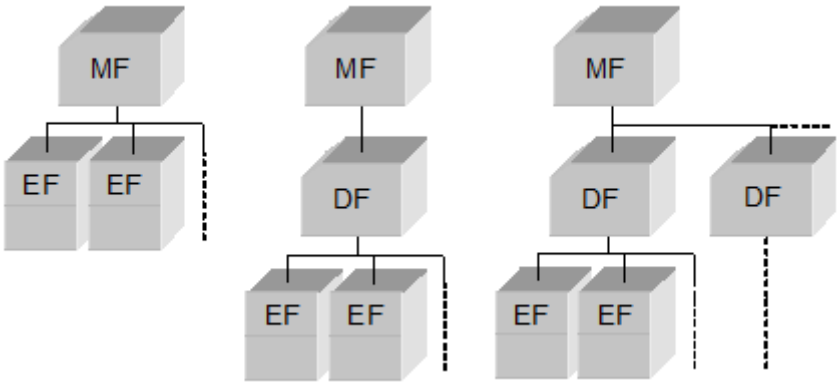

Figura 2.7 Diferentes estructuras de organización de datos.

Con respecto a la figura anterior, el diagrama de la izquierda y el centro corresponden a ejemplos típicos de tarjetas de una sola aplicación y el de la derecha a tarjetas de varias aplicaciones.

A continuación se detallan las características más importantes de los elementos que forman el sistema de archivos:

- Fichero Maestro o MF (Directorio raíz): Es el directorio raíz y es seleccionado automáticamente después de iniciar la Tarjeta. En él están contenidos todos los directorios y ficheros. El Fichero Maestro representa a toda la memoria disponible de la Tarjeta para almacenar datos. Solo existe uno en la Tarjeta.

- Fichero Dedicado o DF (Directorio normal): Son directorios normales, usados para tener estructuras de ficheros más claras y organizadas. Según el SO de la Tarjeta, estará permitido el anidar o no directorios. Cuando el anidamiento está permitido, estará restringido por la capacidad de la memoria.

- Fichero Elemental o EF (Ficheros normales): Hay diferentes tipos y estructuras para este tipo de fichero. Algunos tipos no tendrán una estructura interpretable por la Tarjeta, otros podrán tener diferentes estructuras y otros sólo podrán tener una sola estructura.

2.1.1.2.2 Estructuras de los Ficheros Elementales.

Aquí, se explicará algunas de las diferentes estructuras internas, que pueden tener los Ficheros Elementales.

- Estructura transparente: Estos ficheros no poseen ningún tipo de estructura y los datos se pueden leer o escribir byte a byte, por medio de un puntero que se

desplaza a través del mismo. El tamaño mínimo que pueden tener es de 1 byte, mientras que el máximo no está especificado.

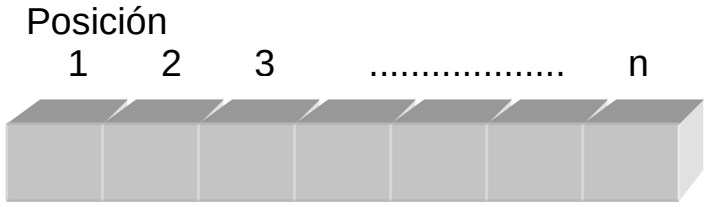

Figura 2.8 Estructura de un fichero transparente**.**

- Estructura lineal fija: Esta estructura está basada en una serie de celdas de igual longitud, que están unidas en forma de matriz. La unidad más pequeña a la que se tiene acceso es de una celda, y no se puede acceder a fracciones de la misma. A la primera celda se le identifica con el número 01h mientras que el número más alto permitido es FEh, estando FFh reservado para usos futuros. El tamaño de cada celda puede variar entre 1 y 254 bytes, siendo las celdas de una misma matriz del mismo tamaño.

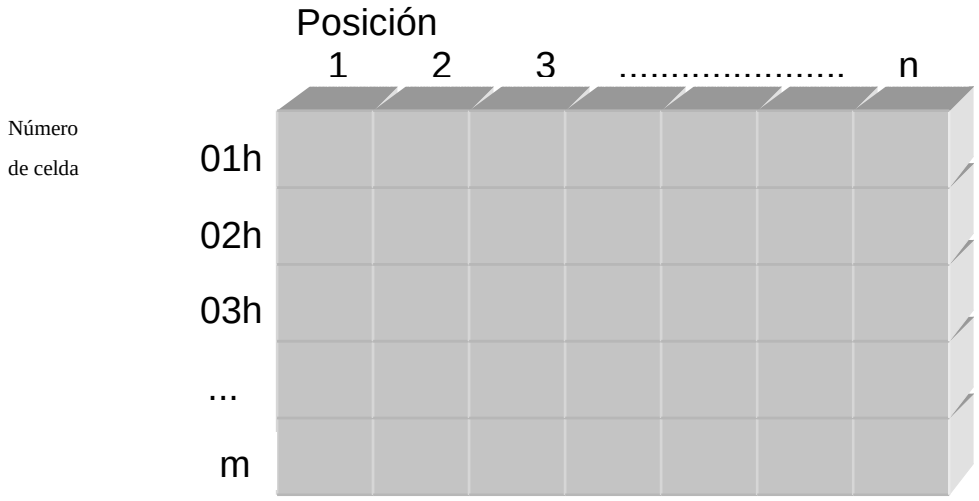

Figura 2.9 Estructura de un fichero lineal fijo.

- Estructura lineal variable: Debido a las restricciones de espacio de memoria y como los datos almacenados en las celdas pueden tener una longitud variable, fue creado este nuevo tipo de estructura, que es útil para optimizar el uso de la memoria. Cada celda puede tener un tamaño variable, en función de los datos que se almacenen en

ella. El tipo de numeración y el tamaño de cada celda son exactamente iguales que en el caso anterior.

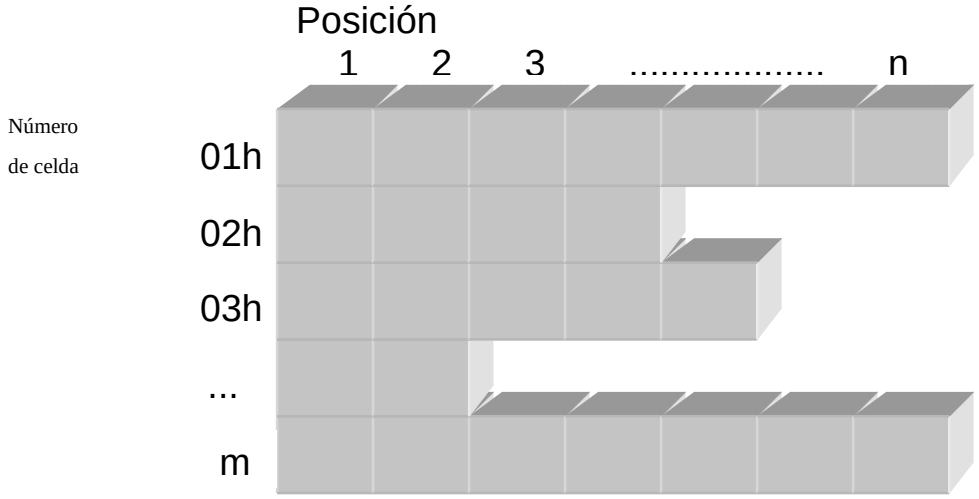

Figura 2.10 Estructura de un fichero lineal variable.

- Estructura cíclica: Esta estructura de datos está basada en un fichero lineal fijo. En este caso existe un puntero que indica cuál fue el último conjunto de datos al que se accedió. Una vez que el puntero llega a la última celda, éste es puesto, por el Sistema Operativo de la Tarjeta en la primera celda (01h) de la matriz.

2.1.1.2.3 Tipos de Ficheros Elementales.

Aquí, se verán los diferentes tipos de ficheros que pueden existir, según la Tarjeta habrá más o menos tipos.

- Ficheros del sistema: Estos ficheros, normalmente son usados para almacenar claves de acceso, para proteger a Directorios y Ficheros Elementales de usos desautorizados. También se usan para ejecutar determinadas aplicaciones. El acceso a estos ficheros está protegido por el SO (No confundir estos ficheros con los Ficheros proporcionados por la Tarjeta). Existen dos maneras distintas de integrar estos ficheros dentro del Sistema de Archivos, el método de ISO, consiste en ocultarlos dentro del Fichero Dedicado correspondiente a la aplicación (en el caso de que cuelguen del MF, dicho fichero estará dentro del MF). Otro método, propuesto por el ETSI (Instituto Europeo para la Estandarización de la Telecomunicaciones), consiste en dejar visible estos ficheros y asignarles un identificador con el cual puedan ser seleccionados. Como ya se ha dicho, el SO, depende de la filosofía del fabricante, por ello según la marca y tipo de la Tarjeta, usará uno de estos métodos u otro propio del fabricante.

- Ficheros ejecutables: En contraste con otros Sistemas Operativos, no es frecuente en la industria de las Tarjetas Inteligentes almacenar programas ejecutables. Sin

embargo, esta es una de las principales funciones de cualquier sistema. Existen varias razones por las que esta función ha estado ausente en las Tarjetas durante bastante tiempo. La razón principal, es que teniendo en cuenta que el microprocesador de la Tarjeta Inteligente, no tiene protección de memoria de ningún tipo, tan pronto como el contador de programa sea cargado con la dirección del fichero ejecutable, el control de la Tarjeta pasará automáticamente al fichero ejecutable y este podrá acceder a toda la memoria, saltándose los mecanismos de seguridad y acceder a cualquier dato almacenado. La creación de este tipo de ficheros, se puede realizar de dos maneras:

1. La primera consiste, en almacenar el código del programa en un fichero ejecutable. Para activar el programa basta con seleccionar dicho fichero y enviar la orden **EXECUTE**, en algunos casos también deben satisfacerse las condiciones de acceso. La respuesta generada por el programa se devuelve como parte de la respuesta a la orden de ejecución.

2. El otro método de creación de ficheros ejecutables, está descrito en el estándar EN 726−3. De acuerdo con esta norma, los Ficheros Dedicados contienen todos los datos referentes a una determinada aplicación y también pueden incluir instrucciones específicas para esa aplicación. El DF posee un área donde almacena el fichero ejecutable que es administrado por el Sistema Operativo. Una vez seleccionado el DF y enviado larden de **EXECUTE**, el SO comprobará antes de ejecutar cada instrucción, si dicha instrucción pertenece al conjunto de instrucciones específicas para esa aplicación, en caso afirmativo se ejecuta la instrucción pedida. En el caso de que la instrucción no pertenezca a la aplicación actual, el Sistema Operativo ignora la orden.

Este tipo de fichero también se puede usar para ampliar el Sistema Operativo de las Tarjetas. Por ejemplo, para añadir algoritmos criptográficos.

- Fichero del usuario: A este tipo de fichero, acudirá el usuario para guardar los datos que necesite su aplicación, por ejemplo, los datos personales del usuario, el número de veces que realizó una operación, etc. Puede tener cualquier tipo de estructura de las antes mencionadas.

- Ficheros proporcionados por la Tarjeta: Son ficheros que la Tarjeta ofrece al usuario para una utilidad determinada, como puede ser: almacenar claves 3DES, claves públicas, códigos secretos... También hay ficheros para utilizar la Tarjeta como monedero o para realizar transacciones. La estructura de cada fichero dependerá del SO.

2.1.1.2.4 Atributos de los Ficheros Elementales.

Los atributos se definen a la vez que se crea el fichero y generalmente no se pueden modificar con posterioridad.

- Atributo de lectura múltiple. También se le conoce como W.O.R.M. (Write Once, Read Many), escribe una vez, lee muchas veces. Cuando un fichero posee este atributo, sólo se podrá escribir en él una vez, pero podrá ser leído cuantas veces como se desee. Esta característica de escribir una sola vez, debe ser soportada por el hardware de la EEPROM o también se puede incorporar a través de una función software. El propósito de este atributo es proteger los datos importantes contenidos en la Tarjeta frente a posibles sobre escrituras.

- Atributo de escritura múltiple .Permite que el contenido del fichero pueda ser alterado muchas veces. El número máximo de alteraciones, depende de los ciclos de lectura y escritura de la EEPROM. Se utiliza en los ficheros que se usan mucho.

- Atributo de corrección de errores. Este atributo se usa para datos que son particularmente importantes (por ejemplo las claves) y que requieren algún tipo de código para la detección de errores. Este atributo permite corregir errores producidos en la memoria EEPROM.

2.1.1.2.5 Identificación y selección de los ficheros (File Identifier o FID).

Todos los ficheros (DF y MF también son ficheros) poseen un identificador o FID de 2 bytes de longitud, que se usa para poder seleccionarlos. Por razones históricas el MF tiene el FID 3F00h. El valor FFFFh está reservado para aplicaciones futuras. Los valores del FID de cada fichero deben escogerse de tal manera, que dos EFs que estén dentro del mismo DF (o MF) no deben tener el mismo FID y tampoco está permitido que un DF posea un FID, igual al de un EF o DF que estén colgado de él, o en el nivel inferior en la jerarquía de ficheros.

Los FID son asignados por el propio usuario, y se ha de seguir algún tipo de norma para su asignación, para evitar posibles problemas.

Los DFs tienen otro identificador adicional, el nombre del DF, por ejemplo DatosUsuario.

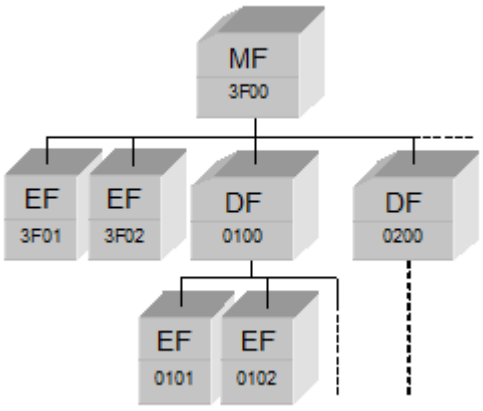

Figura 2.11 Ejemplo de asignación FIDs.

Debido a la estructuración en objeto del Sistema Operativo, es necesario seleccionar los ficheros antes de acceder a ellos. Esto sirve para indicar al sistema con qué fichero se desea trabajar. Esto presenta el inconveniente de que no es posible seleccionar dos ficheros a la vez.

El Fichero Maestro, se puede seleccionar desde cualquier lugar de la jerarquía de ficheros, usando el identificador 3F00h.

Sólo se pueden seleccionar los Ficheros Dedicados, que estén dentro del directorio de trabajo actual, esto es, si por ejemplo se tiene seleccionado el MF, solo se podrán seleccionar los DFs que cuelguen de él. También se pueden seleccionar DFs que se encuentren en el mismo nivel jerárquico del sistema de ficheros. Por ejemplo, seleccionar un DF que cuelgue del MF, desde otro DF que también cuelgue del MF. La selección puede ser, usando su FID o su nombre de directorio.

Sólo se pueden seleccionar los Ficheros Elementales, que estén dentro del directorio de trabajo actual, por ejemplo, si se tiene seleccionado el MF, sólo se podrán seleccionar los EFs que cuelguen de él. Esta selección puede ser explícita, usando el FID del Fichero Elemental. O implícita, usando sólo los 5 bits menos significativos del FID. Este último método, permite seleccionar un fichero y al mismo tiempo acceder a él, usando sólo una instrucción. Este método sólo sería efectivo, si se ha seguido una estrategia de asignación de los FID, compatible con este método de selección.

Todos los ficheros, en su cabecera, tienen unas condiciones de acceso, que deben ser cumplidas antes de realizar cualquier operación sobre el fichero. Estas condiciones, son puestas cuando se crea el fichero y como norma general, no se pueden modificar más tarde. Por ejemplo, los EFs pueden estar protegidos por dos condiciones de acceso, una que bloquea la operación de leer en el EF y otro la operación de escribir. El MF o los DFs, pueden estar protegidos por una sola condición de acceso, que bloquea el poder crear EFs sobre ellos. El encargado de controlar este acceso, es el administrador de ficheros.

Estas condiciones de acceso, son normalmente códigos secretos, que el usuario que intente usar dicho fichero, debe de conocer. El ejemplo más claro de este tipo de condición de acceso es el PIN.

2.1.2 Seguridad.

En este punto, veremos qué mecanismos de seguridad tienen las Smart Card, para protegerse de ataques físicos y lógicos y qué servicios criptográficos, ofrecen a los usuarios.

2.1.2.1 Seguridad Física.

Los ataques físicos, son aquellos que intentan explotar las vulnerabilidades físicas, que presenta todo circuito electrónico, tales como:

- El Consumo de energía: Las instrucciones, pueden ser rastreadas midiendo el consumo de energía mientras el código está siendo ejecutado, el microprocesador de la Tarjeta Inteligente puede evitar este ataque, generando consumos erráticos o realizando operaciones fingidas entretanto. Esto incluye almacenar bits en celdas fingidas, para evitar la detección de localizaciones físicas de memoria.

- Voltaje Alto/Bajo: Un cambio repentino en la tensión de alimentación, puede provocar un mal funcionamiento del micro de la Tarjeta, produciendo tal vez, una validación de un PIN no válido. Para evitar esto, el micro posee una referencia del voltaje para compararlo con el de alimentación, si es demasiado alto o bajo, la Tarjeta no debe funcionar.

- La Extracción del chip: Los circuitos integrados de las Tarjetas Inteligentes no deben funcionar al ser extraídos, esto evita ataques mediante microscopios electrónicos; cuando el chip se introduce en la Tarjeta, se somete a una fuerza mecánica permanente, detectando esta fuerza el chip, puede comprobar si ha sido o no extraído de la Tarjeta.

- Borrado parcial: Aunque están protegidas contra la luz ultravioleta y los Rayos X, teóricamente es posible borrar celdas de memoria; por ello, hay bits centinela situados en la memoria aleatoriamente, que serán comprobados por el microprocesador cada cierto tiempo y que en caso de haber sido borrados, indicarán al micro que la Tarjeta debe dejar de funcionar.

2.1.2.2 Seguridad Lógica.

Los ataques lógicos, son aquello que intentan explotar las vulnerabilidades del Software. Para evitar este tipo de ataques, se dispone de:

- Un Sistema Operativo muy robusto, prácticamente impenetrable, aunque esto, en seguridad es una utopía.

- Autenticación Interna. El terminal, envía un número aleatorio a la Tarjeta, esta lo cifra, usando una clave que sólo conocen la Tarjeta y el terminal. El resultado de esta operación es enviado al terminal, que realiza la misma operación con el número aleatorio y compara el resultado con el obtenido por la Tarjeta y si ambos son iguales la Tarjeta queda validada, frente al mundo exterior.

- Autenticación Externa. El terminal, solicita a la Tarjeta un número aleatorio, lo cifra con una clave secreta que sólo conocen la Tarjeta y el terminal. El resultado es enviado hacia la Tarjeta, esta realiza la misma operación y compara ambos resultados, si son iguales, el terminal queda validado frente a la Tarjeta.

- Claves de acceso, para realizar operaciones sobre ficheros, como pueden ser: crear un DF, escribir en un EF, leer un EF, etc.

- Todos los códigos secretos, enviados a la Tarjeta, pueden ser encriptados.

- Todas la PDU enviadas a la Tarjeta, pueden ser enviadas con un checksum encriptado, asegurando así la integridad de la PDU.

2.1.2.3 Criptografía en las Tarjetas Inteligentes.

Dependiendo del tipo de tarjeta y del fabricante, el número de operaciones criptográficas que la tarjeta puede realizar variará. Las diferentes operaciones posibles que puede tener una Smart Card se muestran en los siguientes puntos:

- Generación de claves DES o 3DES.
- Cifrado DES o 3DES.
- Generación de claves RSA de distintas longitudes.
- Encriptación con clave pública o privada.
- Funciones Hash (MD5, SHA, SSL...).
- Firma digital y su verificación.
- Cifrado y descifrado de envoltorios digitales (cifrar los datos con DES y encriptar la clave DES con RSA).

Como norma general, las Tarjetas sólo ofrecen: generar claves RSA, firma digital y su verificación.

2.1.3 Comunicación.

Debido a la existencia de un único canal de comunicaciones entre la Tarjeta y el lector de Tarjetas, ambos deben turnarse para llevar a cabo la transmisión de datos.

Toda comunicación que se realice con una Smart Card es iniciada siempre por el exterior, esto quiere decir, que la Tarjeta nunca transmite información sin que se haya producido antes una petición externa. Esto equivale a una relación maestro-esclavo, siendo el lector el maestro y la tarjeta el esclavo.

Cada vez que se inserta una tarjeta en el lector, la tarjeta inicia un reset de encendido y envía un paquete de datos (PDU) de respuesta del reset, llamada ATR (Answer To Reset) hacia el lector. Esta respuesta, contiene información referente a cómo ha de ser la comunicación Tarjeta-lector: estructura de los datos intercambiados, protocolo de transmisión, etc.

La transmisión de datos entre ambos dispositivos es asíncrona, esto significa, que cada byte enviado debe estar provisto de algún mecanismo de sincronización. Toda comunicación comienza siempre con un bit de inicio, después el byte de datos, luego un bit de paridad, para detectar errores en el byte de datos y por ultimo 1 o 2 bits de parada. El tiempo que ha de esperarse para poder enviar otro carácter vendrá indicado en la PDU ATR. Siempre que no se esté transmitiendo información por la línea, ésta debe estar a 5 voltios.

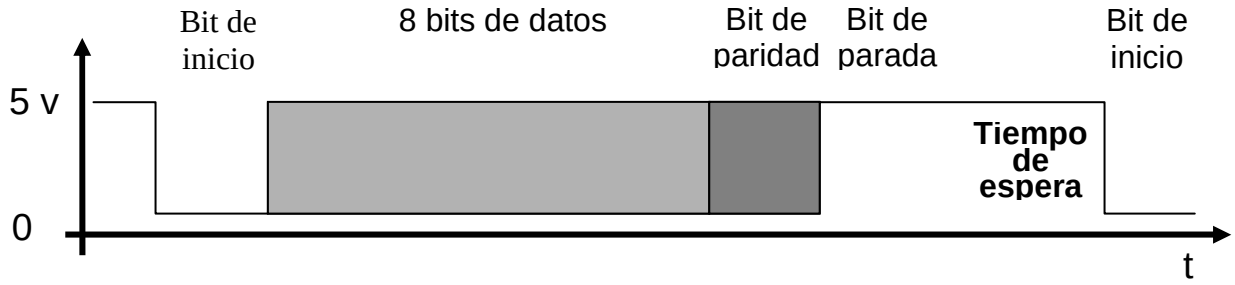

Figura 2.12 Estructura del paquete de datos transmitido.

La duración nominal de un bit, se define como unidad elemental de tiempo o ETU. Esta medida posee un valor inicial, que depende de si la Tarjeta posee reloj interno o por lo contrario necesita de uno externo (que es aplicado a través del contacto CLK). Si la Tarjeta posee su propio reloj, el valor del ETU es 1/9600sg. Esto produce una tasa de transmisión de datos de 9600 bps. Cuando la Tarjeta necesita un reloj externo, el valor del ETU = 372 / fi [sg]. Donde la variable "**fi**", es la frecuencia del reloj externo y su valor está entre 1 y 5 MHz

2.2.3.1. Protocolos de comunicación.

Entre la respuesta ATR y la primera orden enviada, el terminal puede transmitir una instrucción de selección del tipo de protocolo o PTS (Protocol Type Selection). Esto sucede cuando la Tarjeta especifica más de un protocolo en la respuesta al reset y el terminal no sabe cuál ha de usar.

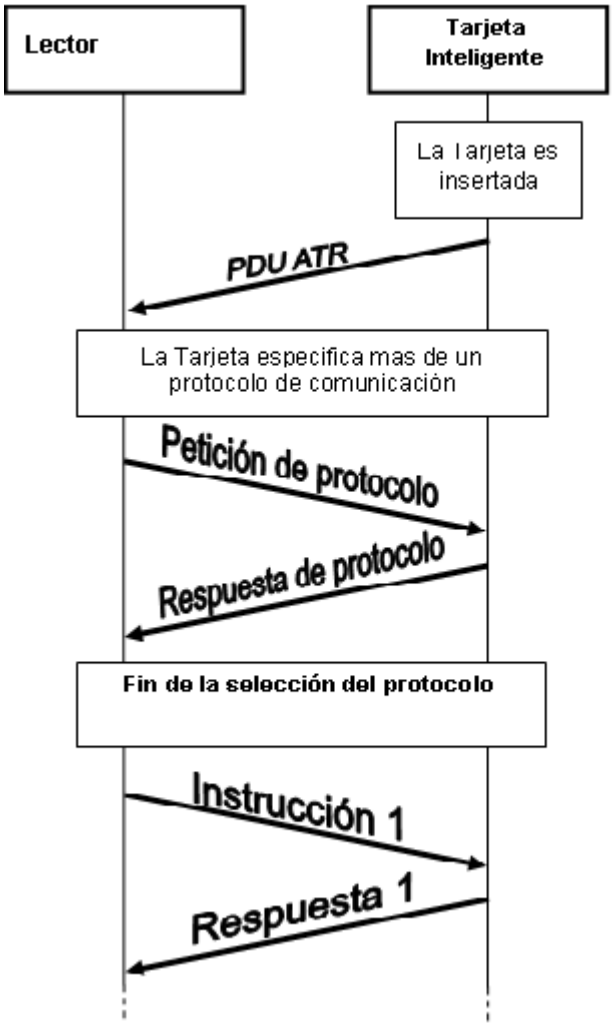

Figura 2.13 Secuencia de inicio y transferencia de datos entre Tarjeta y lector.

Los protocolos de comunicación, especifican con precisión, el formato de las instrucciones, las respuestas a las mismas y el procedimiento a seguir en caso de que se produzcan errores durante la transmisión. La siguiente tabla describe los rasgos más importantes de los distintos tipos de protocolos especificados por ISO:

Protocolo	Significado
T=0	Transmisión byte a byte de manera asíncrona y half duplex
T=1	Transmisión por bloques de manera asíncrona y half duplex
T=2	Transmisión por bloques de manera asíncrona y full duplex
T=3	Transmisión full duplex aún sin especificar

T=4	Expansión del protocolo T = 0
T=5 ... T=13	Aún sin especificar
T=14	Para funciones nacionales, no está especificado por ISO
T=15	Aún sin especificar

Tabla 2.1 Protocolo de transmisión-significado.

Los protocolos más utilizados son: el T=0 que fue diseñado en 1989, y el T=1 que fue creado en 1992.

Protocolo T0

El protocolo T0 fue creado en Francia, durante la fase inicial del desarrollo de las Smart Card y fue el primero en incluirse dentro de un estándar internacional. Es un protocolo orientado al uso del byte, esto quiere decir que la unidad mínima de información procesada es un byte.

Un APDU (*Application Protocol Data Unit*) es un comando con parámetros, cada uno de estos comandos es la unidad mínima de comunicación entre tarjeta y lector.

La tarjeta expone una serie de comandos que pueden ser invocados. Estos comandos interactúan con los ficheros que subyacen a cada aplicación de la tarjeta y proporcionan un resultado. Desde el terminal se invocan estos comandos a través de cualquiera de las APIs de programación. La estructura de un comando APDU está definida por el estándar ISO/IEC 7816

Las PDUs consisten en una cabecera, que incluye un byte para especificar la clase de la instrucción (CLA), otro para indicar la instrucción (INS) y tres bytes opcionales, que actúan como referencias (P1, P2 y P3). A la cabecera le sigue opcionalmente una sección de datos.

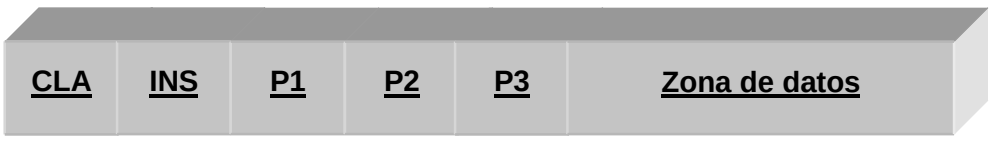

Figura 2.14 Formato de la instrucción del protocolo T=0.

El byte CLA es la clase de la instrucción (GPS, ISO...). El valor FFh está reservado para la selección del tipo de protocolo. El byte INS especifica el código de la instrucción. Este campo sólo es válido si el bit menos significativo es igual a cero, y el más

significativo tiene cualquier valor distinto a 6 y 9.

P1 y P2 son referencias, por ejemplo, una dirección que completa el código de la instrucción. El byte P3 especifica el número de bytes que serán transmitidos en la sección de datos (instrucción de escritura) o los bytes que serán leídos (instrucción de lectura). Cuando P3 es igual a cero y la orden es de lectura, la Tarjeta enviará una cabecera de 256 bytes.

Existen cuatro tipos distintos de comandos APDUs para llamadas y dos tipos para respuesta, seguidamente se explican los campos que tiene o puede tener cada comando. En las *figuras 2.15, 2.16, 2.17* y *2.18* se representan cada uno de estos cuatro comandos de llamada y en las *figuras 2.19 y 2.20* se representan los dos tipos de respuesta posible.

Cabecera: un byte para denotar la clase (CLA), un byte para denotar la instrucción (INS), dos bytes para denotar los parámetros (P1-P2)
Campo Lc: se compone 0, 1 o 3 bytes si la longitud del campo de datos es mayor que cero. En caso contrario estará ausente.1
Campo de datos: bytes correspondientes a la longitud de la cadena de datos a enviar. Si no hay datos a enviar, no se especificará ningún valor en este campo.
Campo Le: número de bytes esperados en la respuesta. Si no se espera ninguna cadena de bytes en la respuesta, no se especificará ningún valor en este campo.
Campo de datos de la respuesta: cadena de bytes correspondiente a la longitud enviada en el comando. Si el campo Le del comando enviados no tenía asignado ningún valor, este campo no existirá.
Campo de Status Word: 2 bytes denotados como SW1 y SW2 que indican el estado del proceso.

Figura 2.15 Cabecera de un comando APDU.

Figura 2.16 APDU. Cabecera y longitud de datos.

Figura 2.17 APDU. Cabecera, longitud de datos y datos a enviar.

Figura 2.18 APDU. Cabecera, longitud de datos a recibir y a enviar y datos a enviar.

Figura 2.19 Respuesta con los datos recibidos y el *Status Byte*.

Figura 2.20 Status Bytes.

La tarjeta siempre incluye en su respuesta uno de los más de 50 códigos de estado diferentes que existen, conocidos como *Status Word*, los bytes SW1 y SW2 componen ese código.

En caso de que se detecte un error en la transmisión de algún byte, se debe proceder al reenvío del byte dañado. El único mecanismo de detección de errores es el bit de paridad. Cuando el receptor detecta un error después de recibir el byte de paridad, debe poner la línea de comunicaciones a nivel bajo (0 voltios) al menos durante un ETU y como máximo dos. Esto sirve para que el dispositivo transmisor, compruebe si los datos se recibieron correctamente. Si la línea, después de transmitir los datos, está a nivel alto, quiere decir que la recepción fue correcta, en caso contrario, se procede al reenvío del byte erróneo después de finalizar la señal de error.

Protocolo T1

Este protocolo es orientado a la transmisión de bloques de manera asíncrona half duplex. Tomando como referencia el modelo **OSI**, este protocolo opera en el nivel dos o capa de enlace. La característica principal es que permite formar paquetes con los bytes de datos. Con esto se consigue:

- Control de flujo.
- Corrección de errores.
- Encadenamiento de paquetes.

Otra ventaja importante de este protocolo es la posibilidad de administrar en ambos sentidos el flujo de datos. El T=0, no permite que la tarjeta tome el control de las comunicaciones y sólo puede enviar información por la línea de datos, cuando el terminal los haya solicitado. El T=1 elimina esta barrera y permite que la Tarjeta envíe comandos de igual manera que el terminal lector.

Las desventajas que presenta este tipo de protocolo son:

- Mayor complejidad del Sistema Operativo.
- Incremento en el uso de la memoria de la tarjeta.

2.1.4 Programación

Se puede hablar de dos ámbitos distintos en cuanto a la programación de tarjetas:

1. Programación de aplicaciones **para el chip** de la tarjeta.
2. Programación de aplicaciones **para los sistemas** en los que se utiliza la tarjeta.

2.1.4.1 Programación interna.

Se trata de aplicaciones que se almacenan y ejecutan dentro del chip de la tarjeta cuando ésta recibe alimentación eléctrica de un lector. Este tipo de programación es de muy bajo nivel y depende normalmente del tipo y proceso de fabricación de las propias tarjetas. En la mayoría de las tarjetas inteligentes el sistema operativo de la tarjeta y las aplicaciones que van dentro del chip se cargan en el propio proceso de fabricación y no pueden ser luego modificadas una vez que la tarjeta ha sido fabricada.

Una excepción clara a este caso pueden ser las Java Cards, que son tarjetas que en el proceso de fabricación incorporan un sistema operativo y una máquina virtual Java específica para este entorno. Una vez fabricada la tarjeta, los desarrolladores pueden implementar mini-aplicaciones (applets) Java para ser cargadas en la tarjeta.

2.1.4.2 Programación de aplicaciones para uso de Smart Card

Se trata de aplicaciones que se ejecutan en ordenadores (en un sentido genérico, ya que pueden ser TPVs empotrados, cajeros automáticos, PCs de escritorio, etc.) a los que se conecta un lector de tarjetas en el que se inserta (o aproxima si es un lector sin contactos) una tarjeta inteligente. Estas aplicaciones se comunican con el lector, el cual se comunica con la tarjeta y sus aplicaciones.

La programación de este tipo de aplicaciones se lleva a cabo mediante el uso de una interfaz de programación de aplicaciones API (del inglés *application programming interface*), que es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Existen varias APIs de programación estandarizadas para comunicarse con los lectores de tarjetas inteligentes desde un ordenador. Las principales son:

- PC/SC (*Personal Computer/Smart Card*), especificado por PC/SC Workgroup. Totalmente implementado para maquinas con sistema operativo Microsoft Windows.

- MUSCLE que proporciona una implementación casi completa de PC/SC para los sistemas operativos GNU Linux-UNIX.

- OCF (*OpenCard Framework*), especificado por el grupo de empresas OpenCard. Este entorno intenta proporcionar un diseño orientado a objetos fácilmente extensible y modular. El consorcio OpenCard publica el API y proporciona una implementación de referencia en Java. Existe un adaptador para que OCF trabaje sobre PC/SC.

En cualquier caso y como ya se ha visto, el modelo de programación que utilizan las tarjetas inteligentes está basado en protocolos de petición-respuesta. La tarjeta (su software) expone una serie de comandos que pueden ser invocados. Estos comandos interactúan con los ficheros que subyacen a cada aplicación de la tarjeta y proporcionan un resultado. Desde el terminal se invocan estos comandos a través de cualquiera de las APIs antes descritas, estas componen los APDUs y son enviados a la tarjeta para que ésta responda, la API también se encargara interpretar la respuesta y devolverla en su formato.

PC/SC

PC/SC (*Personal Computer/Smart Card*) es un conjunto de especificaciones para la integración de tarjetas inteligentes en ordenadores personales. En particular se define un API de programación que permite a los desarrolladores trabajar de forma uniforme con lectores de tarjetas de distintos fabricantes siempre que cumplan con la especificación.

El API de PC/SC está incorporada en sistemas Microsoft Windows 200x/XP y disponible también Microsoft Windows NT/9x. También hay una implementación libre, de código abierto, llamada PC/SC Lite (proyecto MUSCLE) para sistemas operativos GNU Linux.

Las especificaciones **PC/SC** se desarrollan por un comité (PC/SC Workgroup) en el que participan varias empresas del sector como: Gemalto(Axalto+Gemplus), Infineon, Microsoft o Toshiba.

La especificación se divide en 9 partes que contienen los requisitos detallados de interoperabilidad de dispositivos compatibles, información de diseño, interfaces de programación, etc. Este es el contenido de cada una de esas partes:

- Parte 1. Introducción y visión general de la arquitectura
- Parte 2. Requisitos de interoperabilidad para las tarjetas y los lectores
- Parte 3. Requisitos de interoperabilidad para los lectores
- Parte 4. Consideraciones de diseño de los lectores e información de referencia
- Parte 5. Definición de la interfaz del *Resource Manager*
- Parte 6. Definición de la interfaz del *Service Provider*
- Parte 7. Consideraciones de diseño para el desarrollo de aplicaciones
- Parte 8. Recomendación para la implementación de servicios de seguridad y privacidad con tarjetas inteligentes

- Parte 9. Lectores con capacidades extendidas

El API de programación es sencilla y proporciona funcionalidades simples. A continuación se enumeran las principales para dar una visión general del modo en que se programa usando este API:

- *SCardEstablishContext* y *SCardReleaseContext* (inicialización y liberación de recursos)
- *SCardConnect*, *SCardReconnect* y *SCardDisconnect* (conexión y desconexión a una tarjeta en un lector)
- *SCardListReaderGroups*, *SCardListReaders*, ... (exploración de lectores conectados)
- *SCardGetAttrib* (obtención de propiedades de los lectores conectados)
- *SCardTransmit* (envío de comandos APDU's a una tarjeta conectada)

OpenCard

El marco OpenCard (OCF) es un middleware de tarjetas inteligentes implementado en Java. Este marco permite a una aplicación java acceder a una tarjeta inteligente usando objetos y métodos que suponen una mayor abstracción al programador a la hora de fabricar, enviar y recibir los comandos APDUs a la tarjeta, siempre y cuando la tarjeta este implementada tal y como se define en la norma ISO/IEC 7816-4, -8 y -9.

2.2 Lectores de Smart Card

Un lector de tarjetas es un dispositivo capaz de leer y algunos también escribir en una tarjeta. Estos dispositivos se puede presentar de formas diferentes, existen lectores para conexión RS-232, PCCard (PCMCIA), USB y PS/2. Existen numerosos fabricantes y un amplio abanico de precios.

En el mercado se encuentran numerosos fabricantes como Bit4id, C3PO, Gemalto o Kalysis, por citar algunos.

2.2.1 Tipos de lector

Aunque también incorporan un lector los TPVs, como los llamados “cajeros automáticos”, esta obra mostrara los aspectos más relevantes de los dispositivos que se conectan o integran a un ordenador, ya que el proyecto en cuestión trata de explicar cómo hacer un control de presencia mediante dispositivos estándar que se conecten a un ordenador.

Existen tres tipos de dispositivos clasificándolos principalmente por su aspecto y por la forma en que se conectan o integran al ordenador.

1- Los integrados en el teclado.
2- Los dispositivos USB
3- Los dispositivos con conexión RS-232
4- Los dispositivos de tarjeta PCMCIA

2.2.1.1. Lectores integrados

Los lectores que se integran en el teclado son más caros, porque para adquirir el lector hay que comprar un teclado que ya lo lleve incorporado, con lo que se suma el coste del mismo.

Su ventaja se aprecia si se emplean las tarjetas inteligentes con asiduidad, porque siempre están a mano del usuario y no hay que preocuparse de enchufar o desenchufar ningún cable. Sin embargo, ofrecen menos portabilidad.

Figura 2.21 Ejemplo de dispositivo integrado.

2.2.1.2. Lectores USB

El dispositivo USB acostumbra a ser de reducidas dimensiones lo permite trasladarlo con facilidad si se tiene pensado usarlo en más de un lugar. Este dispositivo consta de un pequeño adaptador en el que se introduce la tarjeta inteligente y un cable que se conecta al ordenador en un puerto USB. El precio de este tipo de dispositivo suele oscilar entre los 20 y 70 euros dependiendo de marca, modelo y establecimiento.

Figura 2.22 Ejemplo de dispositivos lectores por USB.

2.3.1.3. Lectores RS-232

Los dispositivos que usan el interface RS-232 son idénticos a los que usan interface USB, evidentemente los diferencia el modo en que se conectan al ordenador ya sea por un puerto USB o RS-232, pero el resto de características, imagen, dimensión, versatilidad pueden ser exactamente las mismas. De hecho numerosos fabricantes ofrecen el mismo modelo de lector con distintas interfaces USB o RS-232.

Figura 2.23 Conector rs-232.

2.3.1.4 Lectores de tarjeta PCMI

Los lectores basados en tarjetas PCMCIA se dirigen a los usuarios de ordenadores portátiles. Su gran baza radica en que se pueden usar con más comodidad, sobre todo en

movimiento, que los dispositivos USB porque se integran dentro del cuerpo del portátil.

Figura 2.24 Ejemplo de lector PCMCIA.

2.3.2. Compatibilidad

En principio, cualquiera de los lectores de tarjetas se puede utilizar en los sistemas operativos más habituales (Windows, Mac OS X y GNU/Linux), siempre que estos lectores cumplan las especificaciones ISO y en especial el estándar 7816 que se ha comentado con anterioridad.

Estos dispositivos suelen traer de fábrica un disco con los controladores necesarios para funcionar en el ordenador.

Todos los siguientes dispositivos que han sido evaluados y soportan el Resource Manager PC/SC de Microsoft Windows:

Bull SmarTLP0
Bull SmarTLP3
Activ Card
Smart ReaderGemplus
GCR410Gemplus GCR410-P
De la Rue DE128
C3PO LTC31
C3PO LTC32
C3PO LTC36
CHIPKIT & CHIPKIT PLUS de Chipmedia
Schlumberger Reflex 20 (No evaluado por la FNMT-RCM)
Schlumberger Reflex 72 (No evaluado por la FNMT-RCM)
INFOS LFD
Utimaco CardMan 4000 (PCMCIA)
Bit4id Ibérica - bit4id miniLECTOR USB
Fujitsu-Siemens SmartCard Reader External, USB
Fagor SCR-1 USB
Athena ASEDrive IIIe USB V2
Omnikey Cardman 3121
SCR3310v2.0 USB

SCR3311 USB
SCR243 (PCMCIA)
C3PO TLTC2 USB
APD KBD 103 PS/SC
Teclado Preh MC147
Fujitsu-Siemens KBPC CX E
DATA PROF. Modelo K107B (Heng-Yu)
C3PO KBR36 USB
Hengyu K107C
Omnikey Cardman 4040

El hecho de que otros lectores no aparezcan en esta lista no significa que no cumplan con el estándar.

2.3 Microcontroladores.

Un controlador es un dispositivo que se emplea en el gobierno de uno o varios procesos. Aunque el concepto de controlador ha continuado inalterable a través del tiempo, su implementación física ha variado frecuentemente. Hace tres décadas, los controladores se construían exclusivamente con componentes de lógica discreta; posteriormente se utilizaron los microprocesadores, que se rodeaban con chips de memoria y E/S sobre una tarjeta de circuito impreso. En la actualidad, todos los elementos del controlador se han podido incluir en un chip que recibe el nombre de **microcontrolador**.

En definitiva, un microcontrolador es un circuito integrado programable que contiene todos los componentes de un computador. Se utiliza para controlar el funcionamiento de una tarea determinada y, debido a su reducida medida, suele ir incorporado en el propio dispositivo que gobierna. Esta última característica es la que le confiere la denominación de controlador incrustado (*embedded controller*).

El microcontrolador es un computador dedicado. En su memoria solamente reside un programa destinado a gobernar una aplicación determinada; sus líneas de entrada/salida soportan la conexión de los sensores y actuadores del dispositivo a controlar y todos los recursos complementarios disponibles tienen como única finalidad atender sus requerimientos. Una vez programado y configurado el microcontrolador solamente sirve para gobernar la tarea asignada.

Un microcontrolador posee todos los componentes de un computador pero con unas características fijas que no pueden alterarse. Todos disponen de los bloques esenciales: procesador, memoria de datos y de instrucciones, módulos de E/S, oscilador de reloj y módulos controladores de periféricos. Además de estos elementos, existen una serie de recursos especiales que los fabricantes pueden ofertar, algunos amplían las capacidades de las memorias, otros incorporan nuevo recursos y hay quienes reducen las prestaciones al mínimo para aplicaciones muy simples. Depende del programador el encontrar el modelo mínimo que se ajuste a sus requerimientos y así minimizar el coste, el hardware y el software. Algunos de los principales recursos específicos que incorporan los

microcontroladores son:

- Temporizadores (*Timers*).
- Perro guardián (*Watchdog*).
- Protección frente a fallo de alimentación (*Brown-out*).
- Estado de bajo consumo.
- Conversores AD y DA.
- Modulador de anchura de pulsos PWM.
- Comparadores analógicos.
- Puertos de E/S digital.
- Puertos de comunicación: serie, CAN, USB, I2C,…

Los microcontroladores pueden clasificarse según su arquitectura, que puede ser Von Neumann o Harvard. La **arquitectura Von Neumann** se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único. Hay aspectos positivos en esta configuración como los accesos a tablas almacenadas en memoria ROM y un set de instrucciones más ortogonal. El bus de direcciones es usado para identificar qué posición de memoria está siendo accedida, mientras que el bus de datos es utilizado para trasladar información entre la CPU y alguna dirección de memoria o viceversa. Con un único sistema de buses, la arquitectura Von Neumann es usada secuencialmente para acceder a instrucciones de la memoria de programa y ejecutarlas regresando desde/hacia la memoria de datos. Esto significa que el ciclo de instrucción no puede solaparse con ningún acceso a la memoria de datos.

Una desventaja de esta arquitectura podría ser que el contador de programa o algún otro registro se corrompieran y apuntaran a la memoria de datos y se tomara ésta momentáneamente como memoria de programa. Consecuentemente se ejecutaría una instrucción no deseada o un error en la decodificación de la instrucción.

La **Arquitectura Harvard** se caracteriza por disponer de dos memorias independientes, una que contiene sólo instrucciones y otra con sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias. Una de las ventajas de esta arquitectura es que la operación del microcontrolador puede ser controlada más fácilmente si se presentara una anomalía en el contador de programa. Existe otra arquitectura que permite accesos a tablas de datos desde la memoria de programa. Esta arquitectura es la llamada arquitectura Harvard modificada.

Esta última arquitectura es la dominante en los microcontroladores actuales ya que la memoria de programa es usualmente ROM, OTP, EPROM o FLASH mientras que la memoria de datos es usualmente RAM. Consecuentemente, las tablas de datos pueden estar en la memoria de programa sin que sean perdidas cada vez que el sistema es apagado. Otra ventaja importante en la arquitectura Harvard modificada es que las transferencias de datos pueden ser solapadas con los ciclos de decodificación de instrucciones. Esto quiere decir que la siguiente instrucción puede ser cargada de la memoria de programa mientras se está ejecutando una instrucción que accede a la memoria de datos. La desventaja de la arquitectura Harvard modificada podría ser que se requieren instrucciones especiales para acceder a valores en memoria RAM y ROM haciendo la programación un poco complicada.

Las principales ventajas que se pueden encontrar en el uso de microcontroladores son:

- Gestión eficiente de procesos.
- Aumento de la fiabilidad.
- Reducción del tamaño, consumo y coste.
- Mayor flexibilidad (únicamente se requiere la reprogramación).

Existe una gran diversidad de microcontroladores. Quizá la clasificación más importante sea entre microcontroladores de 4, 8, 16 o 32 bits. Aunque las prestaciones de los microcontroladores de 16 y 32 bits son superiores a los de 4 y 8 bits, la realidad es que los microcontroladores de 8 bits dominan el mercado y los de 4 bits se resisten a desaparecer. La razón de esta tendencia es que los microcontroladores de 4 y 8 bits son apropiados para la gran mayoría de las aplicaciones, lo que hace innecesario emplear microcontroladores más potentes y consecuentemente más caros.

En cuanto a las técnicas de fabricación, cabe decir que prácticamente la totalidad de los microcontroladores actuales se fabrican con tecnología CMOS4 (*Complementary Metal Oxide Semiconductor*). Esta tecnología supera a las técnicas anteriores por su bajo consumo y alta inmunidad al ruido.

El número de productos que funcionan en base a uno o varios microcontroladores aumenta de forma exponencial. La industria informática acapara gran parte de los microcontroladores que se fabrican. Casi todos los periféricos del computador, desde el ratón o el teclado hasta la impresora, son regulados por el programa de un microcontrolador.

Los electrodomésticos (desde hornos y lavadoras hasta televisores y vídeos) incorporan también numerosos microcontroladores e, igualmente, los sistemas de supervisión, vigilancia y alarma en los edificios, utilizan estos chips para optimizar el rendimiento de los ascensores, calefacción, aire acondicionado, etc.

Las comunicaciones y sus sistemas de transferencia de información utilizan

profusamente estos pequeños computadores, incorporándolos en los grandes automatismos y en los teléfonos móviles.

La instrumentación y la electromedicina son dos campos idóneos para la implantación de estos circuitos integrados. Finalmente, una importante industria consumidora de microcontroladores es la de la automoción, que los aplica en el control de la climatización, la seguridad y los frenos ABS. En la siguiente figura se puede ver la utilización de los microcontroladores en distintos sectores de consumo.

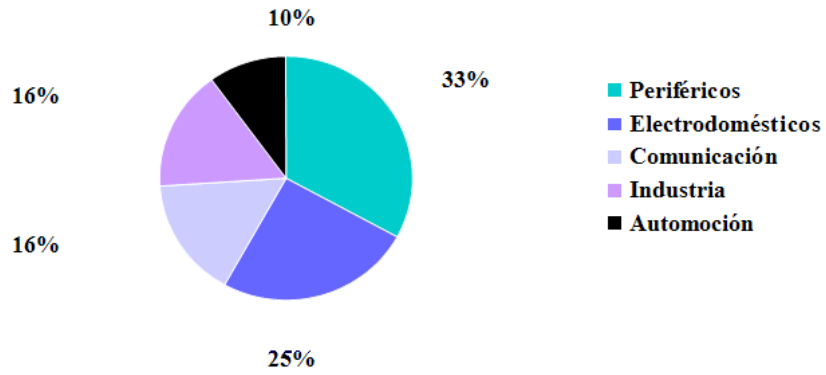

Figura 2.25 Utilización de microcontroladores por sectores.

Algunos de los principales fabricantes de microcontroladores son Microchip, Motorola, Intel, Atmel, Siemens, Philips, Hitachi o Nacional Semiconductor, entre otros. De entre todos los fabricantes expuestos, Microchip es el que más diversidad posee, cuenta actualmente con 159 microcontroladores distintos además de todas sus versiones según encapsulado.

A continuación se exponen algunos de los microcontroladores más populares:

- 8048 (Intel). Es el padre de los microcontroladores actuales, el primero de todos. Su precio, disponibilidad y herramientas de desarrollo hacen que todavía sea muy utilizado.

- 8051 (Intel y otros). Es sin duda el microcontrolador más popular. Fácil de programar pero potente. Está bien documentado y posee cientos de variantes e incontables herramientas de desarrollo.

- 80186, 80188 y 80386 EX (Intel). Versiones en microcontrolador de los microprocesadores 8086 y 8088. Su principal ventaja es que permiten aprovechar las herramientas de desarrollo para PC.

- 68HC11 (Motorola y Toshiba). Es un microcontrolador de 8 bits potente y

con gran cantidad de variantes.

- 683xx (Motorola). Surgido a partir de la familia 68k, a la que se incorporan algunos periféricos. Son microcontroladores de altísimas prestaciones.

- PIC (Microchip). Familia de microcontroladores que gana popularidad día a día. Fueron los primeros microcontroladores RISC

Es preciso resaltar en este punto que existen innumerables familias de microcontroladores, cada una de las cuales posee un gran número de variantes.

3. Análisis de alternativas.

En este capítulo se exponen aquellos aspectos del proyecto sujetos a distintas alternativas y las características

3.1. Lector de Smart Card

En puntos anteriores en esta obra ya se ha hablado sobre los dispositivos lectores y sus principales características.

Los requisitos técnicos para la ejecución del proyecto son únicamente que el dispositivo lector cumpla con la especificación ISO 7816. De modo que sea cual sea su presentación, dimensiones o interface con la que se conecte al ordenador el dispositivo puede ser válido.

3.2 Microcontroladores

En cuanto a las alternativas que se presentan con respecto al microcontrolador habrá que tener en cuenta los dos siguientes puntos:

- Requisitos de la aplicación a realizar.
- Lenguaje y herramientas de desarrollo.

3.2.1 Requisitos

Antes de seleccionar un microcontrolador es imprescindible analizar los requisitos de la aplicación que se quiere desarrollar. A continuación se indican algunos de los aspectos que normalmente hay que tener en cuenta a la hora de realizar la elección:

- **Procesamiento de datos**: puede ser necesario que el microcontrolador realice

cálculos críticos en un tiempo limitado. En ese caso debemos asegurarnos de seleccionar un dispositivo suficientemente rápido para ello. Por otro lado, habrá que tener en cuenta la precisión de los datos a manejar: si no es suficiente con un microcontrolador de 8 bits, puede ser necesario acudir a microcontroladores de 16 o 32 bits, o incluso a hardware de coma flotante.

- **Entrada/Salida**: para determinar las necesidades de Entrada/Salida del sistema es conveniente conocer el diagrama de bloques del mismo, de tal forma, que sea sencillo identificar la cantidad y tipo de señales a controlar. Una vez realizado este análisis puede ser necesario añadir periféricos externos o cambiar a otro microcontrolador más adecuado a ese sistema.

- **Consumo**: algunos productos que incorporan microcontroladores están alimentados con baterías. Lo más conveniente en un caso como éste puede ser que el microcontrolador esté en estado de bajo consumo pero que despierte ante la activación de una señal (una interrupción) y ejecute el programa adecuado para procesarla.

- **Memoria**: en cuanto a la cantidad de memoria necesaria se debe hacer una estimación de cuánta memoria volátil y no volátil es necesaria y si es conveniente disponer de memoria no volátil modificable.

- **Ancho de palabra**: el criterio de diseño debe ser seleccionar el microcontrolador de menor ancho de palabra que satisfaga los requerimientos de la aplicación. Usar un microcontrolador de 4 bits supondrá una reducción en los costes importante, mientras que uno de 8 bits puede ser el más adecuado si el ancho de los datos es de un byte. Los microcontroladores de 16 y 32 bits, debido a su elevado coste, deben reservarse para aplicaciones que requieran altas prestaciones.

- **Diseño de la placa**: la selección de un microcontrolador concreto condicionará el diseño de la placa. Deberá tenerse en cuenta el encapsulado del mismo.

Además de todo lo mencionado también es importante tener en cuenta la documentación y herramientas de desarrollo disponibles para cada microcontrolador.

Según volumen de ventas y diversidad de modelos se puede establecer como principales fabricantes a los siguientes:

- Microchip Technology Corp.

- STMicroelectronics

- Atmel Corp.

- Motorola Semiconductors Corp.

De todos los fabricantes expuestos, Microchip es el que más diversidad posee, cuenta actualmente con 159 microcontroladores distintos además de todas sus versiones según encapsulado.

3.2.2 Programación de microcontroladores

A la hora de realizar la programación del microcontrolador, se pueden utilizar lenguajes de bajo nivel (Ensamblador) o lenguajes de alto nivel (C, BASIC…).
Los lenguajes de alto nivel surgen de la necesidad de realizar programas más complejos, cuya implementación en ensamblador requeriría mucho tiempo y dificultad. Los lenguajes de alto nivel funcionan con un compilador, que se encarga de descomponer las instrucciones en ensamblador en instrucciones de bajo nivel. Otra ventaja es que para realizar un programa en un lenguaje de alto nivel no es necesario conocer la arquitectura interna del microcontrolador.

Los principales lenguajes de programación de microcontroladores son:

Ensamblador

Se dice que es un lenguaje de bajo nivel debido a que sus instrucciones son exactamente las que el procesador ejecuta.

El uso del lenguaje ensamblador posee las siguientes ventajas:

- Aprovechamiento más eficiente de los recursos del microcontrolador.

- Depende del programador el obtener un código optimizado tanto en velocidad como en tiempo de ejecución.

- Debido a su número relativamente bajo de instrucciones, es fácil y rápido llegar a dominarlo por completo.

Su principal problema es la poca potencia de las instrucciones que ejecuta, limitándose a sumas, restas, desplazamiento, etc. Es ideal para aprender la arquitectura interna del microcontrolador y para programas sencillos.

BASIC

El lenguaje BASIC es un lenguaje de alto nivel cuya principal ventaja es su sencillez, siendo un lenguaje apropiado para iniciarse en la programación a alto nivel. Su mayor defecto es que es un lenguaje “no estructurado”, lo que significa que se puede saltar a cualquier parte del programa usando la instrucción GOTO, lo que impide la realización de código estructurado.

C

C es el lenguaje más popular para la programación de alto nivel en microcontroladores, y uno de los más potentes. Con respecto a BASIC, posee la ventaja de ser estructurado y

además tiene gran cantidad de librerías. También permite introducir código en ensamblador dentro del programa. Es más difícil de aprender que el BASIC pero mucho más potente. Además existen varios compiladores que poseen abundantes librerías para todo tipo de dispositivos. Su principal desventaja es la implícita a cualquier lenguaje de alto nivel, poco control sobre el código generado lo que no permite código optimizado y además lo hace mas difícil de depurar.

Como conclusión, el ensamblador es el lenguaje óptimo para programas de poco alcance, con PICs pequeños, donde se busque la velocidad y sencillez. Además permite depurar el código generado línea por línea. Si se necesita un lenguaje de alto nivel para realizar programas más complejos, puede ser útil empezar con BASIC ya que es más sencillo de aprender, pero proporciona malos hábitos de programación al ser de tipo “no estructurado” y no es tan potente como C.

3.2.3 Interconexión entre Microcontrolador y Ordenador

En cualquier ordenador existe una parte muy importante llamada subsistema de Entrada/Salida, que es la que hace posible la comunicación con el mundo exterior. Este sistema está formado por varios dispositivos periféricos que proporcionan un medio para intercambiar datos con el exterior y que se comunican con el procesador a través de una serie de módulos llamados de E/S. Cualquiera de estos módulos contiene una serie de controladores que se encargan de manejar el funcionamiento de uno o varios periféricos.

Los módulos de E/S no deben conectar directamente el periférico con el bus del sistema, sino que tienen que poseer una cierta inteligencia para poder realizar la comunicación entre el periférico y el procesador de forma eficiente. Si se observan algunas de las características del subsistema de Entrada/Salida será posible darse cuenta de esta necesidad:

- Existe una gran diversidad de periféricos que utilizan métodos de operación diferentes. No sería lógico que la CPU tuviera que incorporar toda la lógica necesaria para controlar este rango de dispositivos.

- La velocidad de transferencia de los datos de los periféricos es a menudo mucho más lenta que la que tiene el procesador con el sistema de memoria, por lo tanto resulta poco práctico usar el bus del sistema de alta velocidad para comunicarse directamente con los periféricos.

- A menudo los periféricos utilizan formatos y longitudes de palabra de datos diferentes a los que utiliza el procesador. Debe haber por tanto algún mecanismo para adecuar las señales de ambos dispositivos.

Los módulos de E/S establecen una serie de reglas (llamadas *interfaces*) que les permiten por un lado conectarse con la CPU y la memoria a través del bus del sistema o del de expansión y, por otro lado, conectarse con los dispositivos periféricos a través de

enlaces dedicados para datos. Estos enlaces se caracterizan porque son más lentos, tienen una menor longitud de palabra y menores velocidades de transferencia de datos. Su diseño se basa en un estándar para permitir la interconexión de dispositivos de diferentes fabricantes.

A continuación se describirán algunos de los estándares más importantes utilizados en la interconexión del ordenador y los periféricos, haciendo para ello una distinción entre interfaces serie e interfaces paralelas:

- **Interfaz serie:** Se utiliza una única línea para transmitir los datos.

- **Interfaz paralela:** Se utilizan varias líneas de datos para transmitir múltiples bits de forma simultánea.

3.2.3.1 Interfaces 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 y también impresoras.

En la transmisión serie se van transfiriendo los bits de información de uno en uno a través de una línea de datos, pudiendo ser las transferencias síncronas o asíncronas.

Si se utilizan señales adicionales (reloj o señales de petición y reconocimiento) para indicar cuándo el bit siguiente es válido, entonces se dice que la transmisión se realiza de forma síncrona. La principal ventaja de este tipo de transferencias es que el receptor puede funcionar a varias frecuencias de reloj (siempre que no sobrepase su frecuencia máxima de funcionamiento). Simplemente bastará con retrasar el envío de la señal de reconocimiento para ralentizar el protocolo. En las transferencias asíncronas, por el contrario, tanto el receptor como el transmisor deben funcionar a la misma frecuencia. 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*), paridad impar (*even*), siempre un nivel alto (*mark*) o siempre un nivel bajo (*space*).

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.

3.2.3.1.1 La interfaz RS-232.

Este estándar lo incorporan todos los ordenadores personales 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 International Telephone and Telegraph*). En él se definen todas las características mecánicas, eléctricas y los protocolos 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. Debido a la gran diferencia de voltaje que existe entre los niveles altos y bajos, se permiten tasas de transferencia de hasta 115.200 baudios si la longitud del cable es de unas pocas decenas de metros.

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.

3.2.3.1.2 El Bus Serie Universal (USB).

El USB es un estándar (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 CD-ROMS).

La especificación de este estándar ha sido respaldada por las empresas líderes mundiales en el campo de la informática: Intel, IBM, DEC, Microsoft, Compac, 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 (*hubs*) 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.

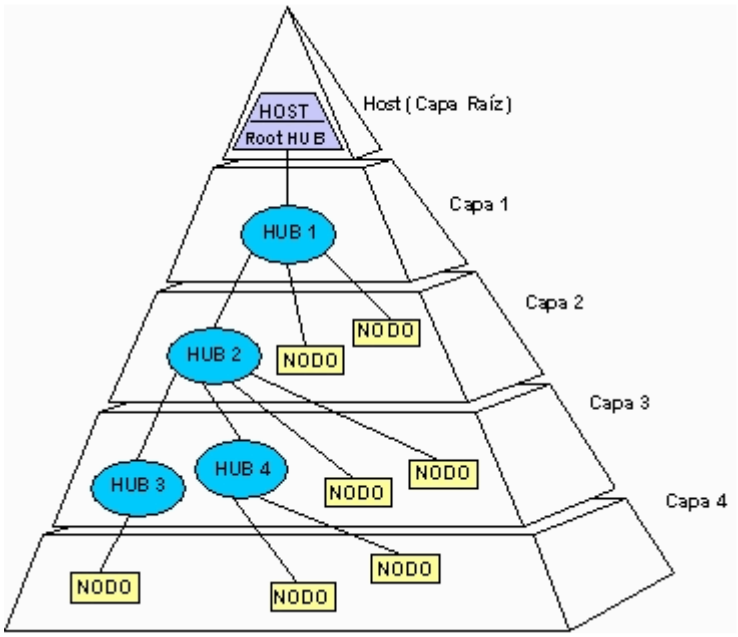

Figura 3.1 Topología de una conexión USB

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 tan sencilla. 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 auditorías 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 2.2). Los datos se 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 3 a 5 amperios por razones de seguridad, 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.

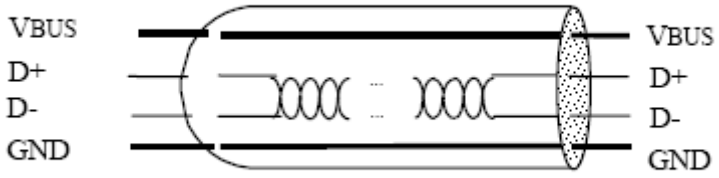

Figura 3.2 Formato del tipo de cable.

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.

3.2.3.1.3 El estándar IEEE 1394 o FireWire.

Apple y Sony inventaron el FireWire a mediados de los 90 y lo desarrollaron hasta convertirlo en el estándar multiplataforma IEEE 1394. FireWire es una tecnología para la entrada/salida de datos en serie a alta velocidad y la conexión de dispositivos digitales como videocámaras o cámaras fotográficas digitales que ha sido ampliamente adoptado por fabricantes de periféricos digitales como Sony, Canon, JVC y Kodak.

FireWire es uno de los estándares de periféricos más rápidos que se han desarrollado, característica que lo hace ideal para su uso con periféricos del sector multimedia (como cámaras de vídeo) y otros dispositivos de alta velocidad como, por ejemplo, lo último en unidades de disco duro e impresoras. Se ha convertido en la interfaz preferida de los sectores de audio y vídeo digital, ya que reúne numerosas ventajas, entre las que se encuentran la elevada velocidad, la flexibilidad de la conexión y la capacidad de conectar un máximo de 63 dispositivos. Además de cámaras y equipo de vídeo digital, la amplia gama de productos FireWire comprende reproductores de vídeo digital, sistemas domésticos para el ocio, sintetizadores de música, escáneres y unidades de disco duro.

Con un ancho de banda 30 veces mayor que el conocido estándar de periféricos USB 1.1, el FireWire 400 se ha convertido en el estándar más respetado para la transferencia de datos a alta velocidad. Apple fue el primer fabricante de ordenadores que incluyó FireWire en toda su gama de productos. Una vez más, Apple ha vuelto a subir las apuestas duplicando la velocidad de transferencia con su implementación del estándar IEEE 1394b o FireWire 800.

La velocidad sobresaliente del FireWire 800 frente al USB 2.0 convierte al primero en un medio mucho más adecuado para aplicaciones que necesitan mucho ancho de banda, como las de gráficos y vídeo, que a menudo consumen cientos o incluso miles de megabytes de datos por archivo.

Algunas de las características más importantes del FireWire son:

- Flexibles opciones de conexión. Admite un máximo de 63 dispositivos con cables de hasta 4,25 metros.

- Distribución en el momento. Fundamental para aplicaciones de audio y vídeo, donde un fotograma que se retrasa o pierde la sincronización arruina un trabajo.

- Alimentación por el bus. Mientras el USB 2.0 permite la alimentación de dispositivos sencillos que consumen un máximo de 2,5 W, como un ratón, los dispositivos FireWire pueden proporcionar o consumir hasta 45 W, más que suficiente para discos duros de alto rendimiento y baterías de carga rápida.

- Es conectable/desconectable en uso. Lo que significa que no se necesita desactivar un dispositivo para conectarlo o desconectarlo y que no es necesario reiniciar el ordenador.

- Funciona tanto con Mac como con PC. Lo que garantiza la compatibilidad con una larga lista de productos con FireWire a precios razonables.

3.2.3.2 Interfaces paralelas.

Los ordenadores personales incorporan tradicionalmente un puerto paralelo consistente en un conector DB25. Este tipo de interfaz se caracteriza porque se envían simultáneamente los bits de datos por medio de diferentes líneas. Desde siempre se ha considerado la interfaz paralela como el puerto utilizado para conectar la impresora, pero desde comienzos de la década de los noventa se viene utilizando con otros fines, ya sea para comunicar diferentes sistemas informáticos o bien para conectar dispositivos de almacenamiento masivo. La clave para su expansión fue la utilización de estándares que permitían la comunicación bidireccional por las líneas de datos.

3.2.3.2.1 La interfaz Centronics.

Inicialmente se diseñó una interfaz con 36 pines, que utilizaba la casa Centronics Data Computer Corporation en sus impresoras. Sin embargo, la interfaz Centronics de los ordenadores personales actuales fue diseñada por Epson Corporation.

La interfaz consta de 8 pines para datos más 5 señales que controlan la impresora y cinco que vienen de la misma. Se utilizan voltajes TTL con señales no balanceadas, por lo que son susceptibles de recibir ruido y producir errores. El bus soporta tasas de transferencia de datos de hasta 100 Kbyte/s.

Actualmente se han diseñado dos estándares que tratan de aumentar el ancho de banda de la interfaz Centronics sin perder la compatibilidad con el mismo, permitiendo además la comunicación bidireccional. Son las interfaces ECP (*Extended Capabilities Port*) y EPP (*Enhanced Capabilities Port*) que se definen en el estándar del IEEE 1284. ECP se utiliza en las impresoras y escáner, puesto que permite mayores tasas de transferencia con protocolos sencillos, mientras que EPP sirve para los demás dispositivos en donde se necesita un control de errores más exhaustivo.

3.2.3.2.2 El estándar IEEE 1284.

Este nuevo estándar define 5 modos de transferencia de datos, desde el viejo Centronics hasta dos métodos que permiten la comunicación bidireccional entre el ordenador y el dispositivo. Debido a que los protocolos se implementan por hardware, EPP y ECP permiten tasas de transferencia de datos mucho mayores, llegando incluso al Megabyte por segundo. Los 5 modos de transferencia de datos que define el estándar son: modo compatible, 4 bits, 8 bits (modo byte), ECP y EPP.

El estándar describe el formato de las señales, la asignación de pines y los mecanismos de detección y corrección de errores, sin embargo las funciones de la BIOS, la interfaz software y el control de los puertos están a cargo de los fabricantes.

El puerto paralelo se configura inicialmente en el modo compatible. Después se establece un diálogo con el periférico para decidir el modo de funcionamiento final, aunque debido a la facilidad con la que se puede cambiar el modo, es posible realizar transferencias cambiando los modos de emisión y de recepción de datos de forma dinámica. Los modos byte, ECP y EPP son opcionales en el estándar.

3.2.3.2.3 Small Computer Systems Interface (SCSI).

La interfaz SCSI es una interfaz paralela, con 8, 16 o 32 líneas de datos, que se utiliza para comunicar dispositivos rápidos, como discos CD-ROM, dispositivos de audio y dispositivos de almacenamiento externo de datos. Normalmente se considera a la configuración SCSI como un bus (conexión multipunto), sin embargo, los dispositivos están conectados entre sí formando una conexión daisy-chain. Cada dispositivo tiene dos conectores, uno de entrada y otro de salida. El comienzo del bus se conecta con el host y el último dispositivo incorpora un terminado para evitar problemas de reflexiones de las señales. Los dispositivos funcionan de forma independiente y pueden intercambiar datos tanto entre sí como con el host.

Este bus puede soportar múltiples procesadores y múltiples dispositivos periféricos. Soporta hasta 8 dispositivos, de los cuales cada uno puede tener 8 unidades lógicas, cada una de las cuales soporta 256 subunidades lógicas.

La especificación original se llamó SCSI-1 y usaba 8 líneas de datos a una frecuencia de 5 MHz, permitiendo una transferencia de datos de 5 Mb/s. SCSI-1 soporta hasta 7 dispositivos que pueden ser encadenados al bus.

En 1991 surgió una extensión estándar, el SCSI-2, que incrementaba el número de líneas de datos a 16 o 32 bits e incrementaba la frecuencia de reloj a 10 MHz Así se logran tasas de transferencia máxima de hasta 40 Mbytes/s.

Las transferencias en el bus siempre tienen lugar entre un iniciador (dispositivo que manda comandos) y un objetivo (dispositivo que ejecuta los comandos). Normalmente el host es el iniciador y el controlador del dispositivo es el objetivo, aunque puede haber algún dispositivo que sea ambas cosas a la vez. Las señales que se transmiten por el bus pueden estar implementadas utilizando un solo cable cada una y compartiendo una masa común en el caso de un *single-ended* SCSI o utilizando dos cables cada una en el caso del *differential* SCSI. El primero se utiliza para distancias menores a 6 metros y el segundo para distancias menores a 25 metros. Los conectores son de 50 pines.

3.3 Programación de aplicaciones

En este punto se muestran las opciones manejables para la realización del código de la aplicación, es decir, los posibles lenguajes en que se puede programar la aplicación que se ejecutara en el ordenador para llevar cabo el control de presencia.

C: Es un lenguaje de programación creado en 1969 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell basándose en los lenguajes BCPL y B. Al igual que sus predecesores, es un lenguaje orientado al desarrollo de sistemas operativos, aunque se ha convertido en uno de los lenguajes de propósito general más usados. Es útil para la programación de microcontroladores, sistemas operativos, drivers, etc. Pero su gran inconveniente es que no es orientado a objetos, como si lo son C++ y C#.

C++: Lenguaje de programación orientado a objetos desarrollado en los años 80 como extensión del lenguaje C (de hecho, su nombre significa "incremento de C"). Se dice que C++ abarca los tres paradigmas de la programación: La programación estructurada, la programación genérica y la programación orientada a objetos.

C#: A diferencia de C++ donde sus instrucciones se traducen en código máquina al ser compiladas, el código generado a partir de C# pertenece a MSIL, que forma parte de la plataforma .NET. C# es una evolución de C++ diseñado por Microsoft para ser el lenguaje más importante en la plataforma .NET.

Visual Basic: Es un lenguaje de programación desarrollado por el alemán Alan Cooper para Microsoft. El lenguaje de programación es un dialecto de BASIC, con importantes agregados. Su primera versión fue presentada en 1991, con la intención de simplificar la programación utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y, en cierta medida, también la programación misma. Desde el 2001 Microsoft ha propuesto abandonar el desarrollo basado en la API Win32 y pasar a trabajar sobre un framework o marco común de librerías independiente de la versión del sistema operativo, .NET Framework, a través de Visual Basic .NET (y otros lenguajes como C Sharp (C#) de fácil transición de código entre ellos).

JAVA: Se trata de un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

4. Elección de componentes hardware y software

En este capitulo se enumeran los componentes hardware y software elegidos para el desarrollo del proyecto, así como los motivos de esta elección.

4.1 Dispositivos Lectores

Como ha sido explicado en el anterior apartado, cualquier lector que cumpla la norma ISO 78/16 seria valido, por tanto los motivos para la elección de los dos lectores utilizados en la realización del proyecto han sido la disponibilidad de uno y el coste económico del otro.

Los lectores utilizados durante la realización, testeo y validación de la aplicación han sido el Gem PC twin de Gemalto y el LTC-31 de C3PO.

4.1.1 Lector GemPc Twin de Gemalto

Este dispositivo ha sido usado debido a la disponibilidad por parte del director de este proyecto de un lector de este tipo.

Figura 4.1 Lector GemPC Twin.

Sus características más destacables son las siguientes:

- Dimensiones: 68x63x12
- Temperatura: Almacenamiento-> desde -25°C hasta 60°
  Operación-> de 5 a 55°
- Peso: 0,15 kg
- Ciclos inserción: 100.000
- Conexión: USB 0 serial(conmutable)
- Tasa de transferencia: USB -> 12Mbps
- PC / SC 98, ME, 2K, XP, Server 2003, x64, CE 4.1, 4,2, 5,0 /95OSR2 y NT4, Vista, interfaz en serie
  PC / SC: Linux (RedHat, Suse, Debian)
  PC / SC: MacOS (10,3., 10,4, 10,5, 10,6) (sólo USB)
  CT-API: 2K, XP, Server 2003

4.1.2 Lector LTC-31 de C3PO

Este dispositivo cumple con las especificaciones técnicas necesarias siendo uno de los más económicos del mercado, razón principal por la que ha sido empleado en la realización de este proyecto.

Figura 4.2 Lector LTC-31

Sus principales características son:

- Dispositivo lector y grabador de tarjeta chip conectado a PC.
- Instalación en modo Plug and Play para Windows.
- Diseño de reducidas dimensiones y poco peso que facilita su portabilidad.
- Base plana para su posible sujeción a otros dispositivos (monitor, teclado, etc.).
- Carcasa de PVC ligero y resistente a golpes.
- Presentaciones y acabados opcionales para grandes volúmenes.
- Diseñado especialmente para su utilización en entornos de certificación con firma digital.

Características técnicas:

- Actualización de firmware de forma remota
- Provisto de una CPU CMOS de 8 Bits
- Provisto de dos LEDS señalizadores: uno verde y otro rojo.
- Alimentación 5V DC del propio PC.
- Soporta velocidades de comunicación entre 9.600 bps y 1.500.000 bps

El dispositivo es compatible con los entornos más utilizados:

- Windows 98, ME
- Windows 2000, XP, Vista y 7
- Windows XP (64-bit) y Windows Vista (64-bit)
- Linux
- Mac OS X
- Otras UNIX soportadas por libccid

Soporta las tarjetas más utilizadas actualmente en el mercado:

- DNI electrónico
- Tarjetas asíncronas protocolo T=0 (GSM, VisaCash, Euro6000, Tarjeta Criptográfica, etc.)
- Tarjetas asíncronas protocolo T=1 (German Geldkarte)
- Tarjetas síncronas basadas en SLE4442(C3P2K) y SLE4428

Especificaciones soportadas:

- PC/SC "Interoperability Specifications for ICCs and Personal Computer Systems".
- ISO 7816
- Especificaciones EMV (EuroPay / MasterCard / Visa)
- Device Class Specification for USB Chip/Smart Card Interface Devices (CCID)
- Homologado con el Sistema de Identificación Segura de Secuware, compatible con el DNIe

4.2 Microcontrolador

En esta sección se explica el microcontrolador elegido para la realización de este proyecto, sus principales características y los motivos por los que ha sido elegido. Se muestra también la elección de interface de conexión entre microcontrolador y ordenador, motivos y características esenciales.

4.2.1 Microcontrolador PIC16F84A

A la hora de elegir un fabricante para el microcontrolador, se ha decidido utilizar los microcontroladores de la familia PIC, de Microchip Technology. Los PIC son los microcontroladores más extendidos actualmente en el mercado, además la disposición de una amplia gama de estos microcontroladores por parte de la UPCT y su estudio en

asignaturas y prácticas en la carrera de ingeniería técnica de telecomunicación convierte a esta familia de microcontroladores en la mejor opción para la realización de este proyecto.

Microchip dispone de 3 familias de microcontroladores, los de gama baja, media y alta. Así, hay microcontroladores sencillos y baratos para aplicaciones sencillas, y otros más complejos para aplicaciones de más envergadura.

Los requerimientos del proyecto a la hora de elegir un microcontrolador no son demasiado altos, de modo que el microcontrolador usado en la realización de este proyecto ha sido el PIC16f84A, en la elección de este micro se han tenido en cuenta las necesidades de procesamiento de datos, entrada/salida, ancho de palabra etc.…

Las principales características de este PIC de gama media son:

- Repertorio de 35 Instrucciones.
- Instrucciones de un solo ciclo excepto las de salto que necesitan dos.
- Memoria de programa Flash de 1 K x 14 bits.
- Memoria RAM dividida en 2 áreas: 22 registros de propósito específico (SFR) y 68 de propósito general (GPR) como memoria de datos.
- 15 registros de funciones especiales.
- Memoria de datos RAM de 68 bytes (68 registros de propósito general).
- Memoria de datos EEPROM de 64 bytes.
- Contador de programa de 13 bit (lo que en teoría permitiría direccionar 4 KB de memoria, aunque el 16F84 solo dispone de 1KB de memoria implementada).
- Pila con 8 niveles de profundidad.
- Modos de direccionamiento directo, indirecto y relativo.
- ALU de 8 bits y registro de trabajo W del que normalmente recibe un operando que puede ser cualquier registro, memoria, puerto de Entrada/Salida o el propio código de instrucción.
- 4 fuentes de interrupciones:
    1. A través del pin RB0/INT.
    2. Desbordamiento del temporizador TMR0.
    3. Interrupción por cambio de estado de los pins 4:7 del Puerto B.
    4. Completada la escritura de la memoria EEPROM.
- 1.000.000 de ciclos de borrado/escritura de la memoria EEPROM.
- 40 años de retención de la memoria EEPROM.
- 13 pins de E/S con control individual de dirección.
- PortA de 5 bits <RA0:RA4>.
- PortB de 8 bits <RB0:RB7>.
- Contador/Temporizador TMR0 de 8 bits con divisor programable.
- Power-on Reset (POR).
- Power-up Timer (PWRT).
- Oscillator Start-up Timer (OST).
- Watchdog Timer (WDT).
- Protección de código.
- Modo de bajo consumo SLEEP.
- Puede operar bajo 4 modos diferentes de oscilador.
- Programación en serie a través de dos pins.
- Tecnología de baja potencia y alta velocidad CMOS Flash/EEPROM.

1. Características eléctricas
2. Temperatura ambiente máxima para funcionamiento de -55°C to +125°C.
3. Tensión máxima de VDD respecto a VSS de -0,3 a +7,5V.
4. Tensión de cualquier patilla con respecto a VSS (excepto VDD, MCLR, y RA4) de -0,3V a (VDD + 0.3V).
5. Tensión en MCLR con respecto a VSS -0,3 a +14V.
6. Tensión en RA4 con respecto a VSS -0,3 a +8,5V.
7. Disipación de potencia total de 800 mW.
8. Máxima corriente de salida a VSS 150 mA.
9. Máxima corriente de salida de VDD 100 mA.
10. Máxima corriente del puerto "A" como fuente, 50 mA.
11. Máxima corriente del puerto "A" como sumidero, 80 mA.
12. Máxima corriente del puerto "B" como fuente, 100 mA.
13. Máxima corriente del puerto "B" como sumidero, 150 mA.
14. Máxima corriente que puede suministrar una sola salida como fuente o sumidero, 25 mA.

- Rango de alimentación:
  - de 4 a 5,5 v en configuración de oscilador XT, RC y LP.
  - de 4,5 a 5.5 v en configuración de oscilador HS.
- Consumo típico:

  16F84A:
  - de 1,8 a 4.5 mA en configuración de oscilador RC y XT (FOSC=4 MHz, VDD=5,5V).
  - de 3 a 10 mA en configuración de oscilador RC y XT durante la programación de la FLASH (FOSC=4MHz, VDD=5,5V).

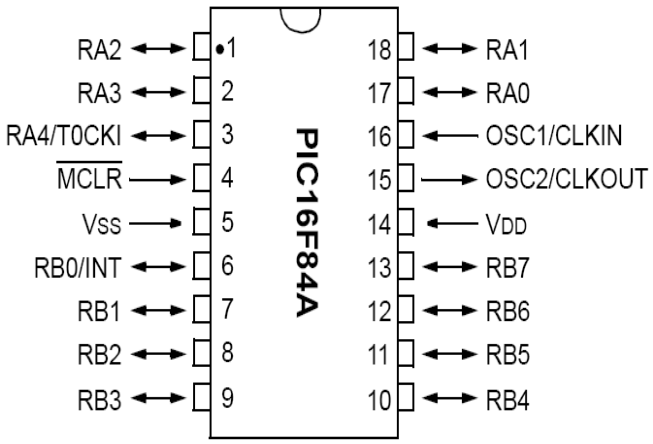

Figura 4.3 Disposición de las patillas del PIC16f84

En la siguiente tabla se describen las funciones de las patillas del encapsulado de dieciocho patillas del PIC 16f84a:

Nombre	**Nº**	**Tipo**	**Descripción**
OSC1/CLKIN	16	I	Entrada del oscilador a cristal/Entrada de la fuente de reloj externa

OSC2/CLKOUT	15	O	Salida del oscilador a cristal. En el modo RC, es una salida con una frecuencia de ¼ OSC1
MCLR	4	I/P	Reset/Entrada del voltaje de programación.
RA0	17	I/O	Puerto A bidireccional, bit 0
RA1	18	I/O	Puerto A bidireccional, bit 1
RA2	1	I/O	Puerto A bidireccional, bit 2
RA3	2	I/O	Puerto A bidireccional, bit 3
RA4/T0CKI	3	I/O	También se utiliza para la entra de reloj para el TMR0
RB0/INT	6	I/O	Puerto B bidireccional, bit 0 Puede seleccionarse para entrada de interrupción externa
RB1	7	I/O	Puerto B bidireccional, bit 1
RB2	8	I/O	Puerto B bidireccional, bit 2
RB3	9	I/O	Puerto B bidireccional, bit 3
RB4	10	I/O	Puerto B bidireccional, bit 4 Interrupción por cambio de estado
RB5	11	I/O	Puerto B bidireccional, bit 5 Interrupción por cambio de estado
RB6	12	I/O	Puerto B bidireccional, bit 6 Interrupción por cambio de estado
RB7	13	I/O	Puerto B bidireccional, bit 7, Interrupción por cambio de estado
Vss	5	P	Tierra de referencia
Vdd	14	P	Alimentación

Tabla 4.1 Descripción de las patillas del encapsulado del PIC16f84a

4.2.2 Interconexión entre Microcontrolador y Ordenador. Interface RS-232

En cuanto a la interconexión entre el microcontrolador y ordenador se ha optado por un interface serie, concretamente el interface RS-232. Los motivos para esta decisión son principalmente la presencia de este interface presumiblemente en todos los ordenadores de sobremesa, que el dispositivo lector será de tipo USB y no se desea copar los puertos USB del equipo ya que estos son cada vez mas usados por los periféricos y la disposición del entrenador Pic school del hardware necesario para la implementación del interface.

4.3 Entorno y lenguaje de programación

Las aplicaciones que llevan a cavo el control de acceso y la gestión de la base de datos de usuarios han sido escritas en lenguaje C++, cualquier lenguaje de los presentados en capítulos anteriores podría haber sido usado, la decisión de usar este lenguaje se debe a que la información prestada por el departamento de ATICA de la Universidad de Murcia, necesaria para poder trabajar con la tarjeta de estudiante, se proporcionó en este lenguaje. El entorno de desarrollo usado para la realización de estas aplicaciones ha sido Visual Studio .NET 2003.

La programación del microcontrolador se ha realizado en lenguaje ensamblador, la edición y grabación del software para el PIC, se ha llevado a cabo con las herramientas MPLAB IDE v8.88 y WinPIC 800

4.4 La tarjeta de estudiante de la Universidad Politécnica de Cartagena

La tarjeta de estudiante de la Universidad Politécnica de Cartagena es una Smart Card con chip WG10. Está ideada para aplicaciones de identificación de alta seguridad, y homologada por la Fábrica Nacional de Moneda y Timbre. El potente sistema de seguridad con criptografía simétrica (algoritmo DES) hace que esta tarjeta se use en documentos de identidad y tarjetas de corporaciones como es este caso.

Figura 4.4 Aspecto de la tarjeta de estudiante UPCT.

En la máscara de la tarjeta se implementan, entre otras, las siguientes funciones:

- Gestión de PIN y claves de seguridad.
- Funciones de crédito y débito seguras.
- Algoritmo DES (Data Encryption Standard).
- Mecanismos de seguridad.
- Claves diversificadas por cada tarjeta.
- Integridad en la gestión de los datos para proteger la transacción.
- Función de búsqueda de transacciones y saldos.
- Cumplimiento de la norma ISO7816 1/2/3/4.
- Capacidad funcional de uso de SAM (Security Access Module).
- Posibilidad de gestión ON line o OFF line.
- Capacidad para multi-operadores y multi-aplicaciones.

Características técnicas:

- RAM de 256 bytes.
- Frecuencia máxima de reloj: 5.0 Mhz.
- Tiempo mínimo de ejecución de una instrucción: 800 ns con una frecuencia de reloj de 5 Mhz.
- Voltaje de alimentación: 5 V ± 10%.
- Consumo de corriente: 4.0 mA (typ).
- Número de pins: 5.
- Márgenes de temperatura: 0º - 55ºC.
- Protocolo de comunicación T=0.
- 9600 bits por segundo.

EEPROM capacidad:

- 8192 bytes.

Es importante destacar que tanto el DNI como otros datos del titular de la tarjeta se encuentran en la zona no protegida de la memoria, por lo que para acceder a estos datos no es necesario introducir el código PIN, lo que facilita la realización de este proyecto.

Toda la información necesaria para trabajar con esta tarjeta ha sido facilitada por el departamento de ATICA de la Universidad de Murcia, quien ha participado activamente en la implantación de esta tarjeta en el ámbito universitario.

5 Diseño

Como se ha comentado en capítulos anteriores, el objetivo de esta obra es la creación de un sistema para el control de acceso de estudiantes de la UPCT a uno o varios recintos por medio de la tarjeta de estudiante. El diseño de este sistema es básicamente el siguiente:

1. Estudiante poseedor de una Smart Card (Tarjeta estudiante UPCT) inserta su Smart Card en el dispositivo lector indicando el deseo de acceder a un recinto.

2. El dispositivo lector lee el dni de la Smart Card del alumno y envía la información al ordenador al que está conectado por USB.

3. En el ordenador se ejecuta una aplicación que tiene acceso a una base de datos de alumnos y permisos asignados a estos. En caso de que el alumno en cuestión tenga permiso para acceder al recinto solicitado, el ordenador envía por medio de su interface RS-232 la información de la puerta que se debe abrir al PIC.

4. El PIC recibe los datos del ordenador sobre la puerta que se debe abrir y envía un pulso se unos tres segundos aproximadamente a la puerta en cuestión (de 1 a 8 puertas).

5.1 Diseño del hardware

En el siguiente esquema se muestra los distintos componentes que integran este proyecto y el flujo de información que existe entre ellos:

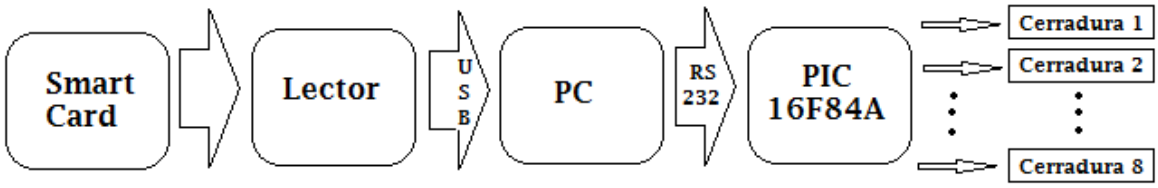

Figura 5.1 Flujo de información del sistema.

Como se puede observar en la figura anterior, los elementos hardware implicados en este proyecto son Smart Card, Lector de Smart Card, Bus USB, PC, Bus rs232, PIC16f84A y 8 cerraduras eléctricas. Tanto la Smart Card, que es la tarjeta de estudiante UPCT, como el lector con su propia conexión USB al ordenador y el mismo ordenador son elementos con su propio diseño y que no necesita especificarse en esta obra. El hardware que se debe diseñar es el concerniente a la comunicación entre el PC y el PIC y entre el PIC y las cerraduras.

En capítulos anteriores se ha hablado del interface serie RS-232 y como se ha explicado, los niveles de tensión que se utilizan en el interface rs232 no son niveles TTL como los de los usados por el PIC, por tanto para realizar la comunicación por este interface será necesario entre otros elementos un convertidor de niveles.

El MAX-232 es un conversor de nivel TTL/RS232. Sólo es necesario este circuito integrado y 4 condensadores para convertir niveles TTL a RS232 y viceversa. La interfaz mínima con el MAX232 entre un dispositivo con salida serie TTL o CMOS y el conector RS232 se muestra en la *Figura 5.2.*

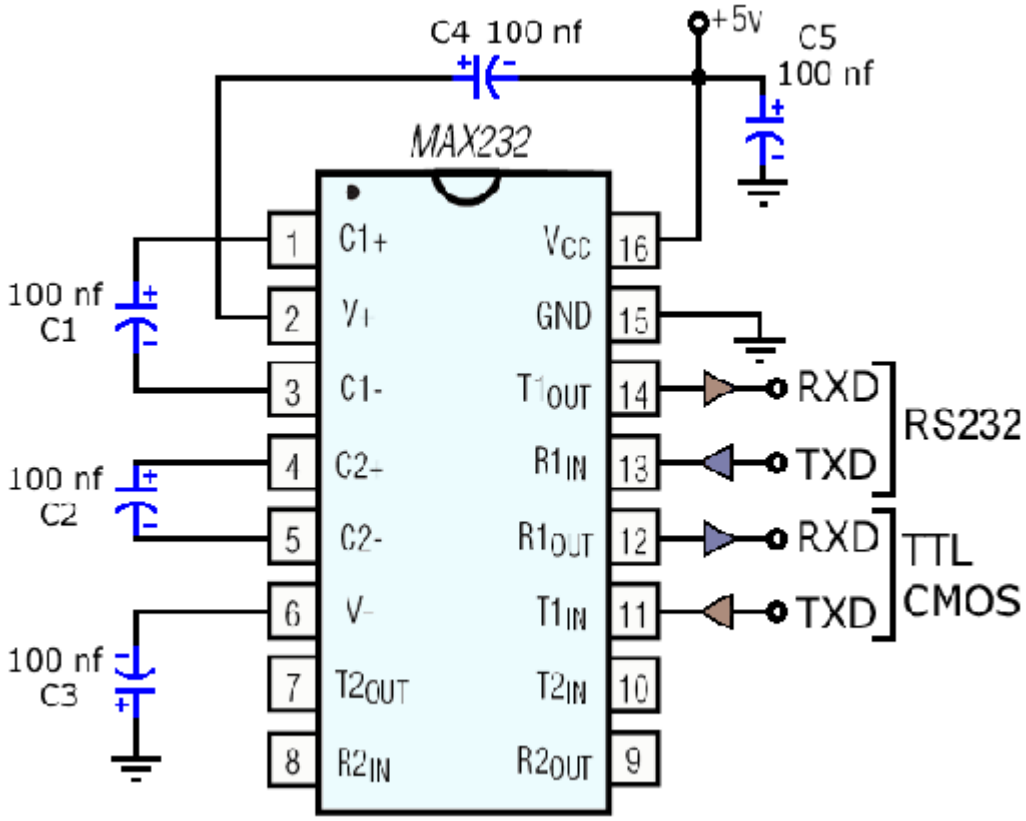

Figura 5.2 Conexión mínima RS232/TTL con un MAX232. Con los condensadores de 100 nF se puede llegar hasta los 64 Kbps, si se colocan de 1 microfaradio puede llegar hasta 120Kbps.

Puesto que el PIC solo recibe datos desde el ordenador, solo se necesita una conexión entre el pic y el circuito convertidor. En la Figura 5.3 se muestra las conexiones necesarias entre el circuito convertidor de niveles, el PIC y el conector RS-232.

Es importante saber que no cualquier cable serie vale para la conexión entre el PC y el PIC. Debe utilizarse un cable macho-hembra no cruzado.

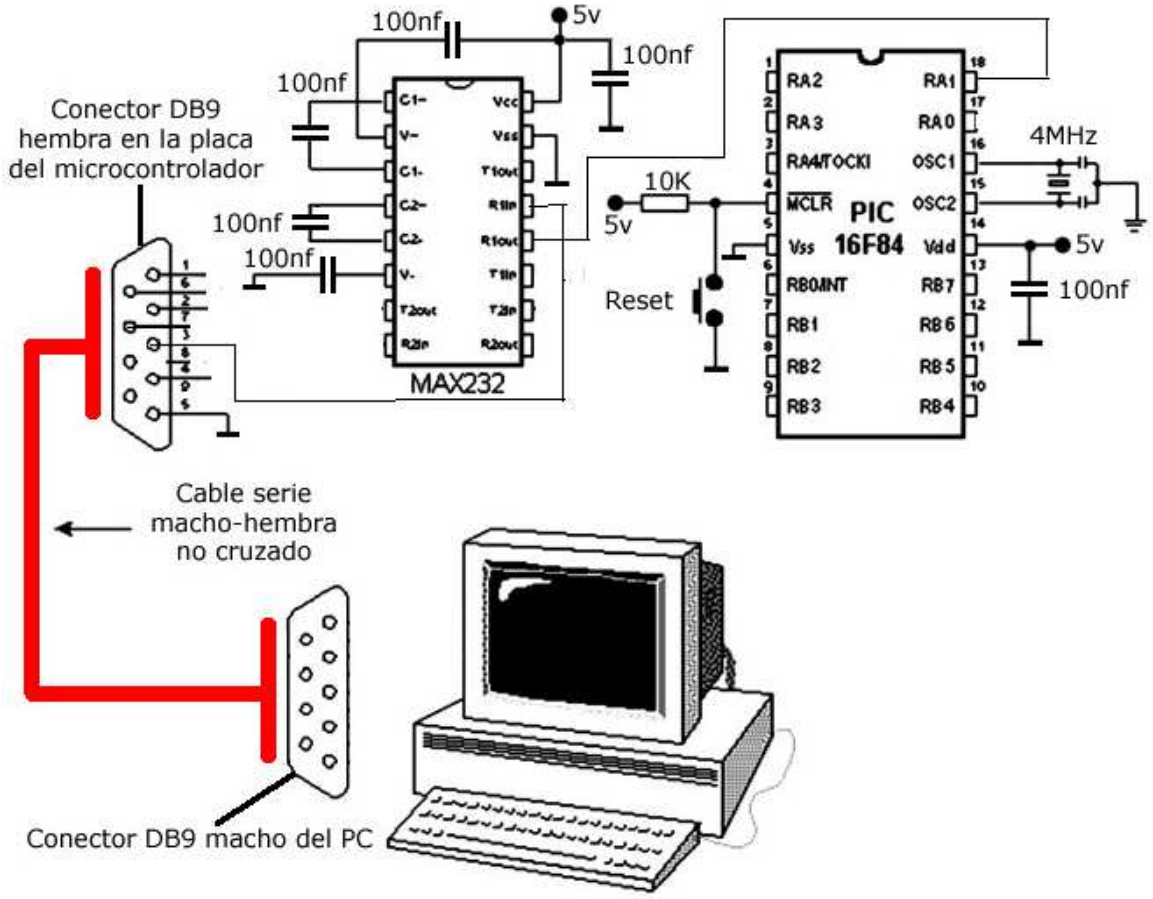

Figura 5.3 Conexión entre PC/MAX-232/PIC16f84.

La configuración del puerto COM en el ordenador debe de ser la siguiente:

- Bits por segundo = 4800 baudios.
- Bits de datos = 8.
- Paridad: ninguna.
- Bits de parada = 2.
- Control de flujo: ninguno.

Con esta configuración el tiempo por bit, dato que será muy importante en la implementación, será:

4800 baudios -> 1/4800 seg -> 208,3 microsegundos

En cuanto a la conexión del PIC con las cerraduras, esta se realizará por medio del Puerto B, cada patilla del Puerto B, de RB(0) a RB(7), estará conectada con una cerradura.

5.2 Diseño del software

El diseño del software se abordará desde cuatro puntos definidos independientemente:

1. Aplicación para el control de acceso.

2. Aplicación para el registro y gestión de la base de datos de alumnos y permisos.

3. Aplicación para el microcontrolador.

4. Base de datos

5.2.1 Aplicación – Control de Acceso.

Esta aplicación será la que el alumno verá en el terminal dispuesto para el control de acceso y por tanto la principal interface del sistema. La interfaz de usuario debe proveer la funcionalidad necesaria para una buena comunicación entre el hombre y la máquina. Un buen diseño agiliza el uso de una aplicación, tanto al visualizar aquella información más importante, como al realizar aquellas tareas más habituales con sólo unos pocos clicks de ratón. El diseño de la interfaz debe realizarse de manera que esta resulte simple e intuitiva de utilizar.

Lo primero que deberá realizar la aplicación será establecer, de forma totalmente transparente al usuario, una tubería para obtener los datos de la tarjeta de estudiante insertada en el dispositivo lector. Para ello PC/SC proporciona una API (winscard.dll) que contiene un conjunto de funciones que permiten comunicarse con la tarjeta. Una vez se tiene el DNI del alumno se quiere que la aplicación compruebe los permisos asignados a ese alumno en la base de datos, para dar acceso al alumno al recinto o denegarlo. También debe la aplicación, en caso de tener el alumno permiso afirmativo, enviar vía RS-232 la información sobre el recinto al que se quiere acceder al PIC.

Otras funcionalidades que deberá implementar la aplicación son:

- Mostrar el DNI del alumno unos instantes cuando se introduce la tarjeta.
- Mostrar el nombre del alumno en caso de estar registrado durante unos instantes.
- Comunicar brevemente si ha ocurrido algún tipo de error.
- Mostrar los posibles recintos a los que el alumno puede desear acceder y que tenga capacidad de seleccionar el deseado.
- Mostrar el estado de la aplicación y dar instrucciones al usuario por medio de la barra de estado de la aplicación.
- Mostrar los últimos 20 accesos de usuarios registrados

5.2.2 Aplicación – Registro del control de acceso.

Esta aplicación será la encargada de gestionar la base de datos de alumnos y los permisos que estos tienen asignados. Solo el usuario root o administrador debe tener acceso a ella. Las funcionalidades que deberá implementar son:

- Añadir un registro (dar de alta un alumno, con DNI y nombre).
- Modificar permisos asignados a un registro.
- Buscar un registro (por medio de la clave principal DNI) y mostrar datos asociados.
- Eliminar historial de un registro.
- Eliminar un registro.

5.2.3 Aplicación para el PIC.

Esta aplicación será la que este corriendo en el PIC, lo que debe hacer es permanecer a la espera de la recepción del dato sobre el recinto al que se quiere acceder y una vez recibido el dato, interpretarlo y dar un pulso de aproximadamente tres segundos en la patilla del PORTB que corresponda a ese recinto.

5.2.4 Diseño de la base de datos.

La base de datos de alumnos debe registrar el dni del alumno, que será usado como clave principal, nombre y apellidos del alumno por separado, permisos y restricciones que tiene el alumno asignados y un historial de accesos.

Toda esta información se debe estructurar en dos tablas unidas por la clave principal. La primera de las tablas contendrá los datos personales del alumno y los permisos y restricciones asignados, y la segunda contendrá una serie de registros correspondientes cada uno a un acceso del alumno a un recinto, cada uno de estos registros debe contener tanto la identificación del alumno que realiza el acceso como información sobre el recinto al que se accede y la fecha y hora en que se realiza el acceso.

DNI	Nombre	Apellido 1	Apellido 2	Permisos	Restricciones

Figura 5.4 Información que debe ser contenida en la tabla alumnos-permisos

DNI	Recinto al que se accede	Fecha	Hora

Figura 5.5 Datos que debe contener cada uno de los registros que formarán el historial del alumno, tabla de accesos

La relación entre estas dos tablas debe estar basada en el campo clave principal DNI, y la relación será "Uno a varios", es decir el campo "DNI" no puede tener duplicados en la tabla Alumnos-permisos y este mismo campo "DNI" podrá tener numerosas entradas iguales en la tabla "accesos", cada una de las cuales se corresponderá con un acceso que el alumno realiza. En la *Figura 5.6* se representa de forma visual la relación que debe de existir entre las dos tablas que formarán la base de datos.

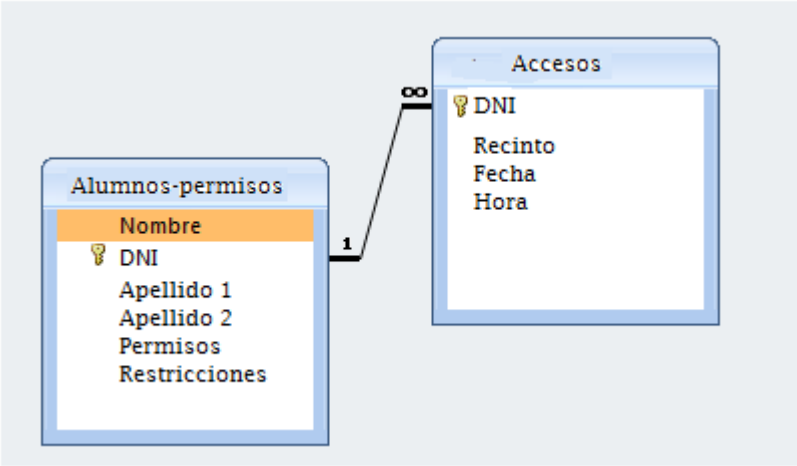

Figura 5.6 Relación entre las tablas alumnos y accesos.

6 Implementación.

En este capítulo se explica como se ha llevado a cabo el proceso de implementación del sistema para el control de acceso, objeto de esta obra.

6.1 Implementación del hardware.

Como se mencionó en el capitulo anterior, el hardware que se debe diseñar e implementar es de el que concierne al PIC y sus conexiones con el ordenador y las cerraduras, el proceso es montar el circuito en una placa de prueba para comprobar que la selección de los componentes y el diseño del esquema eléctrico son correctos. Esta placa de prueba se crea colocando los componentes sobre una protoboard de montaje según el esquema diseñado. Una vez comprobado el correcto funcionamiento del sistema, el último paso sería la fabricación del circuito impreso (PCB), para este proyecto no será necesaria la creación de dicho circuito, el sistema se montará, probará y expondrá en la placa de prueba.

Para el montaje del circuito se ha usado la placa (Picschool) donde se tiene casi todo lo necesario para hacer la interconexión de elementos. Tiene una protoboard donde realizaremos el diseño del circuito, su propia fuente de alimentación, un convertidor MAX-232 con un conector DB9 hembra para la interface RS-232, 8 leds que representarán las 8 cerraduras, todos los puertos del microprocesador se pueden conectar fácilmente a la protoboard y por lo tanto poder interactuar con ellos fácilmente, así como sus señales Vcc a 5V y GND a tierra.

Figura 6.1 Placa Picschool.

A continuación se representa el esquema eléctrico del interface RS-232 con el adaptador MAX-232 tal y como se encuentra en la placa Picsschool:

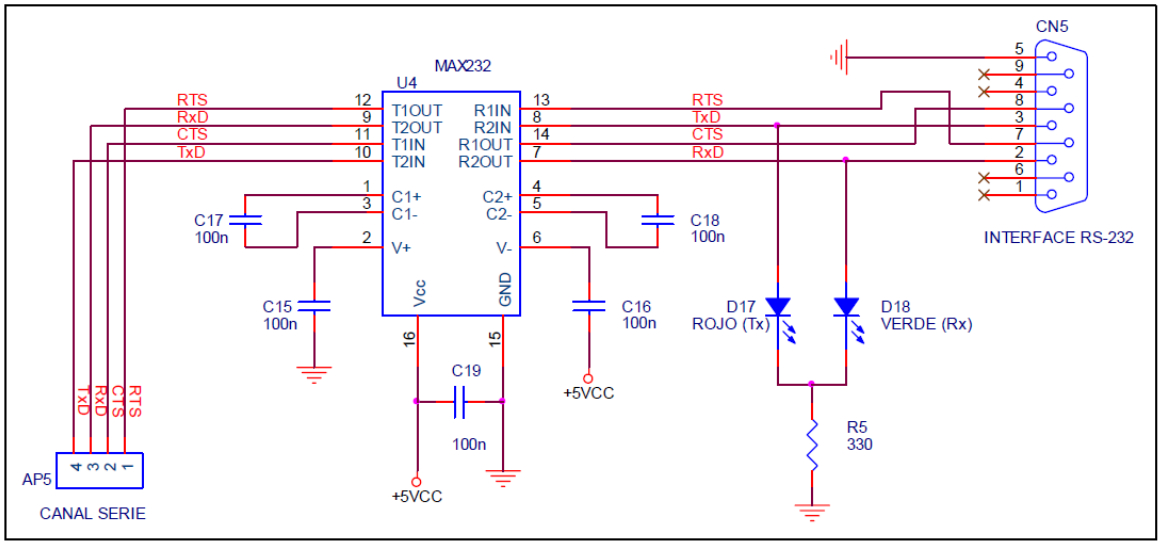

Figura 6.2 Esquema eléctrico del interface RS-232 con el adaptador MAX-232

Según el esquema eléctrico mostrado en la *Figura 6.2* el interface está formado por

el popular adaptador de niveles MAX- 232. Mediante el conector AP5 disponemos de las señales de transmisión y recepción (TxD y RxD) así como las de control de flujo CTS y RTS. Estas señales proceden del microcontrolador. El conector CN5 es un conector DB9 hembra estándar, que permite realizar la conexión con el periférico serie, en el caso de este proyecto el ordenador. Mediante un led bicolor (D17/D18) se monitoriza todo tipo de transmisión y/o recepción.

Para la recepción de los datos del PC, se conectará la patilla RA(1) a AP5(3oRxD) Tal y como se muestra en *la figura 6.3.*

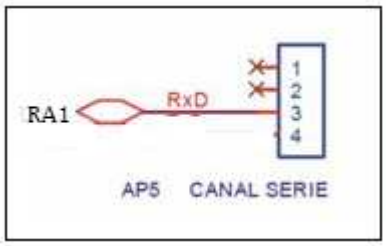

Figura 6.3 Conexión de RA1 con AP5(3), RxD.

Conexión del PIC con las cerraduras.

Figura 6.4 Salidas digitales incorporadas en Picschool.

La placa Picschool incorpora 8 leds los cuales emularan 8 cerraduras eléctricas. Estos leds son accesibles mediante el conector AP17 tal y como se muestra en el esquema de la *figura 6.5*, y se pueden conectar individualmente a cualquiera de las líneas del microcontrolador. Estas líneas son capaces de suministrar del orden de 25mA por lo que no es necesario ningún circuito adicional de amplificación excepto las resistencias de absorción contenidas en el pack RP1. Un nivel lógico “1” por cualquiera de esas líneas provoca el encendido del led correspondiente. Un nivel “0” lo apaga. Es una forma muy simple y económica de reflejar el estado binario de las líneas de salida, donde cada led simula la carga que se desea controlar.

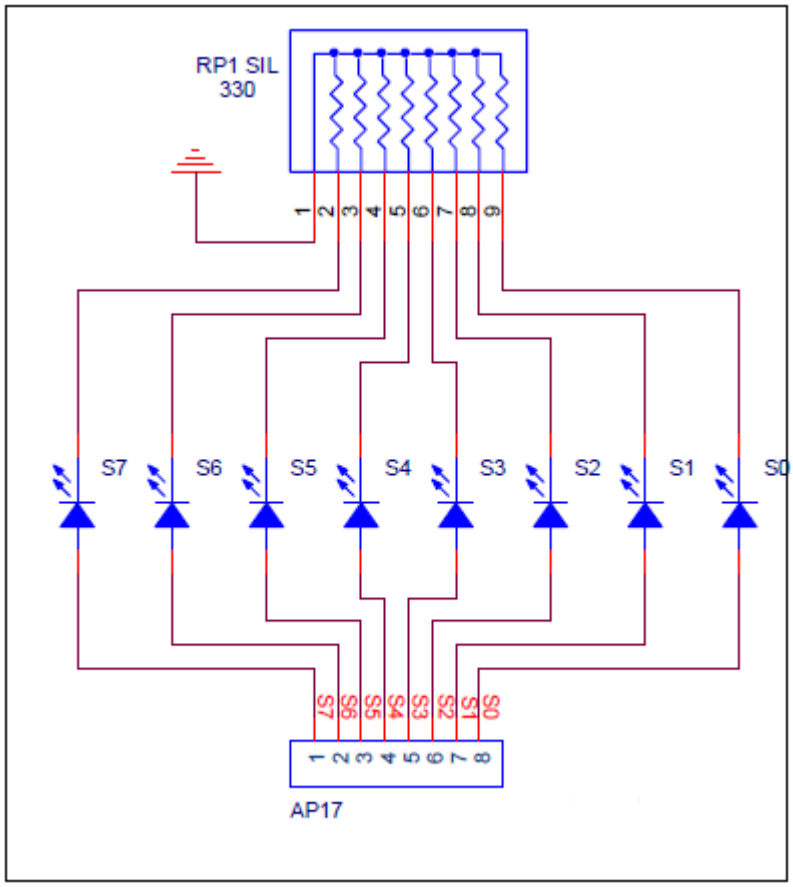

Figura 6.5 Esquema eléctrico de salidas digitales, leds.

El puerto B será el encargado de transmitir la salida a la cerradura correspondiente, de modo que cada patilla estará conectada a un led por medio del conector AP17 y coincidiendo RB(0)->S0, RB(1)->S1 etc.…

La *figura 6.6* muestra el esquema de conexión entre el puerto b del PIC y el conector AP17 de las salidas digitales incorporadas en la placa picschool:

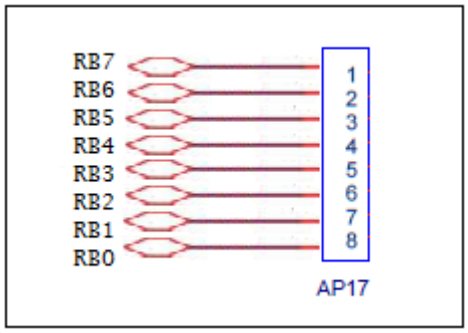

Figura 6.6 Esquema de conexión entre PORTB y las salidas digitales.

6.2 Implementación del software.

La implementación del software relativo al sistema para el control de acceso, objeto de este proyecto, se expondrá en cuatro partes. La primera parte será la de la aplicación interface usuario, es decir la aplicación que usará el alumno para identificarse e intentar acceder a un recinto, la segunda parte se trata de la aplicación diseñada para el usuario administrador, con la que se gestionará la base de datos de alumnos y permisos, la tercera parte trata de la implementación de la propia base de datos y la cuarta sección será para el software del PIC.

6.2.1 Implementación de aplicación –Control de Acceso-

La comunicación con la tarjeta y obtención del dni del alumno ha sido posible gracias a la colaboración del departamento de ATICA de la universidad de Murcia, el cual facilito la información necesaria para el envío de los correctos comandos APDU a la tarjeta para que esta devuelva la información solicitada.

Estos son los comandos APDU que se deben usar para la comunicación con la tarjeta de estudiante UPCT:

- APDU Select MF: Seleccionar el Master File.
  #define APDU_SELECT_MF {0x00, 0xa4, 0x04 …

- APDU Select DF WG10: Seleccionar fichero dedicado al sistema operativo WG10.
  #define APDU_SELECT_WG10 {0x00, 0xa4…

- APDU Select DF: Seleccionar fichero dedicado a aplicación de la tarjeta
  #define APDU_SELECT_DF_RELEASE {0x00, 0xa4…

- APDU_GET_RESPONSE: este es el APDU de obtención de respuesta, l es el número de bytes a leer. Por limitaciones del protocolo T=0, la tarjeta no puede recibir y enviar datos en el mismo comando. Por lo tanto, este comando se debe de ejecutar tras otro comando de tipo 4, recibe los datos entrantes, procesa el comando y prepara la respuesta. El S.O. devuelve un código "61 XXh", donde XX es la longitud que se debe especificar en el campo L2 de APDU_GET_RESPONSE
  #define APDU_GET_RESPONSE(l) {0x00, 0xc0, 0x00, 0x00, l};

- APDU Lectura binaria: permite leer el contenido de un fichero transparente (binario), el cual tiene un tamaño de unidad de datos variable (la unidad de escritura son los bits), se especifica la longitud del fichero en bloques de 256 bits y el offset, por último se indica la longitud de los datos que se quieren leer.
  #define APDU_READ_BINARY(len,offset) { 0x00, 0xb0, ((offset / 0x100) & (~0x80)), (offset & 0xff), len}

- APDU Select Serial.
  #define APDU_SELECT_SERIAL {0x00, 0xA4, 0x02…

- APDU Select EF DNI: fichero elemental en el que se encuentra el dni

#define APDU_SELECT_DNI_EF {0x00, 0xa4, 0x02, …

- APDU Select EF Caducidad: fichero elemental en que está la fecha de caducidad

#define APDU_SELECT_CAD_EF {0x00, 0xa4, 0x02, 0x00…

- APDU Select EF NAC: fichero elemental para la fecha de nacimiento.

#define APDU_SELECT_NAC_EF {0x00, 0xa4…

- APDU Read Binary Saldo

#define APDU_READ_BINARY_SALDO {0x00, …

- APDU Select EF Clave RSA

#define APDU_SELECT_PRIVKEY_EF {0x00, 0xa4, 0x02…

- APDU Select EF Certificado

#define APDU_SELECT_CERT_EF {0x00, 0xa4, 0x02, …

- APDU Select EF Certificado 2

#define APDU_SELECT_CERT2_EF {0x00, 0xa4, 0x02, …

- APDU Select EF Certificado CA

#define APDU_SELECT_CA_EF {0x00, …

- APDU Select EF Certificado CA2

#define APDU_SELECT_CA2_EF {0x00, 0xa4, …

- APDU Verify PIN

//#define APDU_VERIFY_PIN(a) {0x00, 0x20, 0x00, 0x00…

- APDU Select EF SEG

#define APDU_SELECT_SEG_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x14}

- APDU Select EF Response

#define APDU_SELECT_Response_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x21}

- APDU Select EF KEY

#define APDU_SELECT_KEY_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x22}

- APDU Select EF NT

#define APDU_SELECT_NT_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x23}

- APDU Select EF PIN

#define APDU_SELECT_PIN_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x24}

- APDU_STANDAR_UPDATE_BINARY_HEADER

#define APDU_STANDARD_UPDATE_BINARY_HEADER(len,offset) { 0x00, …

Para enviar estos comandos a la tarjeta y recibir respuesta de ella a través del lector se usa la API PC/SC, y la librería "winscard.dll" en la que se encuentran las funciones necesarias para establecer comunicación con el lector y enviar y recibir datos de la tarjeta.

Algunas de las funciones usadas son estas:

- ```
  LONG WINAPI SCardTransmit(SCARDHANDLE hCard,LPCSCARD_IO_REQUEST
  oSendPci, LPCBYTE pbSendBuffer,DWORD bSendLength,
  LPSCARD_IO_REQUESTioRecvPci, LPBYTE pbRecvBuffer, LPDWORD
  pcbRecvLength );
  ```

- ```
  LONG WINAPI SCardReleaseContext( SCARDCONTEXT hContext );
  ```

- ```
  LONG WINAPI SCardEstablishContext(DWORD dwScope, LPCVOID
  pvReserved1, LPCVOID pvReserved2, LPSCARDCONTEXT phContext );
  ```

- ```
  LONG WINAPI SCardListReaders( SCARDCONTEXT hContext, LPCTSTR
  mszGroups, LPTSTR mszReaders, LPDWORD pcchReaders );
  ```

- ```
  LONG WINAPI SCardFreeMemory(SCARDCONTEXT hContext,LPCVOID pvMem);
  ```

- ```
  LONG WINAPI SCardGetStatusChange(SCARDCONTEXT hContext, DWORD
  dwTimeout, LPSCARD_READERSTATE rgReaderStates, DWORD cReaders );
  ```

- ```
  LONG WINAPI SCardConnect(SCARDCONTEXT hContext, LPCTSTR szReader,
  DWORD dwShareMode, DWORD dwPreferredProtocols, LPSCARDHANDLE
  hCard, LPDWORD pdwActiveProtocol );
  ```

- ```
  LONG WINAPI SCardDisconnect(SCARDHANDLE hCard,DWORD disposition);
  ```

Por medio de estas funciones se comprueba primero si existe algún lector conectado al equipo y en caso afirmativo se establece un contexto de comunicaciones con el lector. Cuando un lector está conectado este informará de un cambio en su estado, por ejemplo tarjeta presente, a la aplicación que procederá en este caso a solicitar la información sobre la tarjeta insertada.

Introducida una tarjeta en el lector caven dos posibilidades:

1. Tarjeta correcta. Smart Card estudiante UPCT.
2. Tarjeta errónea. Tarjeta distinta a la requerida o tarjeta UPCT dañada.

Si ocurre el segundo caso, y la tarjeta no es la tarjeta de estudiante UPCT o está dañada, la aplicación mostrará el mensaje "Tarjeta errónea o dañada" y en la barra de estado de la aplicación se darán instrucciones de retirar la tarjeta. Para mostrar mensajes en la barra de estado se usa el siguiente código:

```
CString Text = "Extraiga la tarjeta.";
((CMainFrame*)AfxGetMainWnd())->StatusBarMessage(Text.GetBuffer (0));
```

En caso de ser la tarjeta correcta, la aplicación solicitará el dni, y obtenido el dni del estudiante se comprobará en la base de datos si ese estudinte está registrado y si tiene permisos para acceder al recinto solicitado.

Para acceder a la base de datos se necesita una API que ofrezca la capacidad para leer y escribir en una base de datos Access. En este caso se usa DAO (Data Access Objects), se trata de una API que ofrece la capacidad de programar aplicaciones independientes de cualquier sistema de administración de bases de datos (DBMS).

La tecnología DAO es conocida por los programadores de bases de datos que utilizan Microsoft Access Basic o Microsoft Visual Basic. DAO utiliza el motor de bases de datos Microsoft Jet para proporcionar un conjunto de objetos de acceso a datos: objetos de base de datos, objetos de definición de tabla, objetos de definición de consulta y objetos de conjunto de registros, entre otros.

DAO proporciona el mejor rendimiento con archivos .mdb creados en Microsoft Access, pero también puede tener acceso a orígenes de datos ODBC (Open Database Connectivity) a través de DAO y del motor de bases de datos Microsoft Jet.

Pasos para acceder a registros de una base de datos por medio de DAO :

1. Crear un puntero aun objeto CDaoDatabase.

   - `CDaoDatabase *pDao = new CDaoDatabase;`

2. Abrir la base de datos por medio del puntero al objeto CDaoDatabase.

   - `pDao->Open ("Alumnos.mdb");`

3. Crear referencia a objeto CDaoRecordset vinculado al objeto CDaoDatabase.

   - `CDaoRecordset r(pDao);`

4. Crear referencia a objeto COleVariant para recoger los datos de los registros

   - `COleVariant v;`

5. Abrir tabla y obtener datos de los registros con la referencia a CDaoRecordset.

   - `r.Open(dbOpenDynaset,LPCTSTR(“ SQL ”),dbReadOnly );`
   - `r.GetFieldValue ("Nombre", v);`

```
//(v) puede tener los siguientes tipos
      //bstrVal -> BSTR (una cadena).
      //bVal -> unsigned char
      //iVal -> short
      //lVal = long
      //fltVal -> float
      //dblVal -> double
```

6. Cerrar el vinculo a la tabla y a la base de datos

- r.Close();
- pDao->Close(); // Cerrar la conexión con la BBDD
- delete pDao; // liberar memoria:

De la comprobación en la base de datos se pueden obtener los siguientes resultados:

1. Alumno no registrado.
2. Alumno registrado con permiso negativo para acceder al recinto deseado.
3. Alumno registrado con permiso afirmativo para acceder al recinto deseado.

En el primer caso el procedimiento será informar al alumno de que no está registrado y dar instrucciones para que retire la tarjeta. Además en la sección “Registro” del interface, donde aparecen los últimos 20 intentos de acceso, aparecerá una línea indicando día, hora, recinto al que se deseaba acceder, dni del alumno y el mensaje “Acceso denegado”.

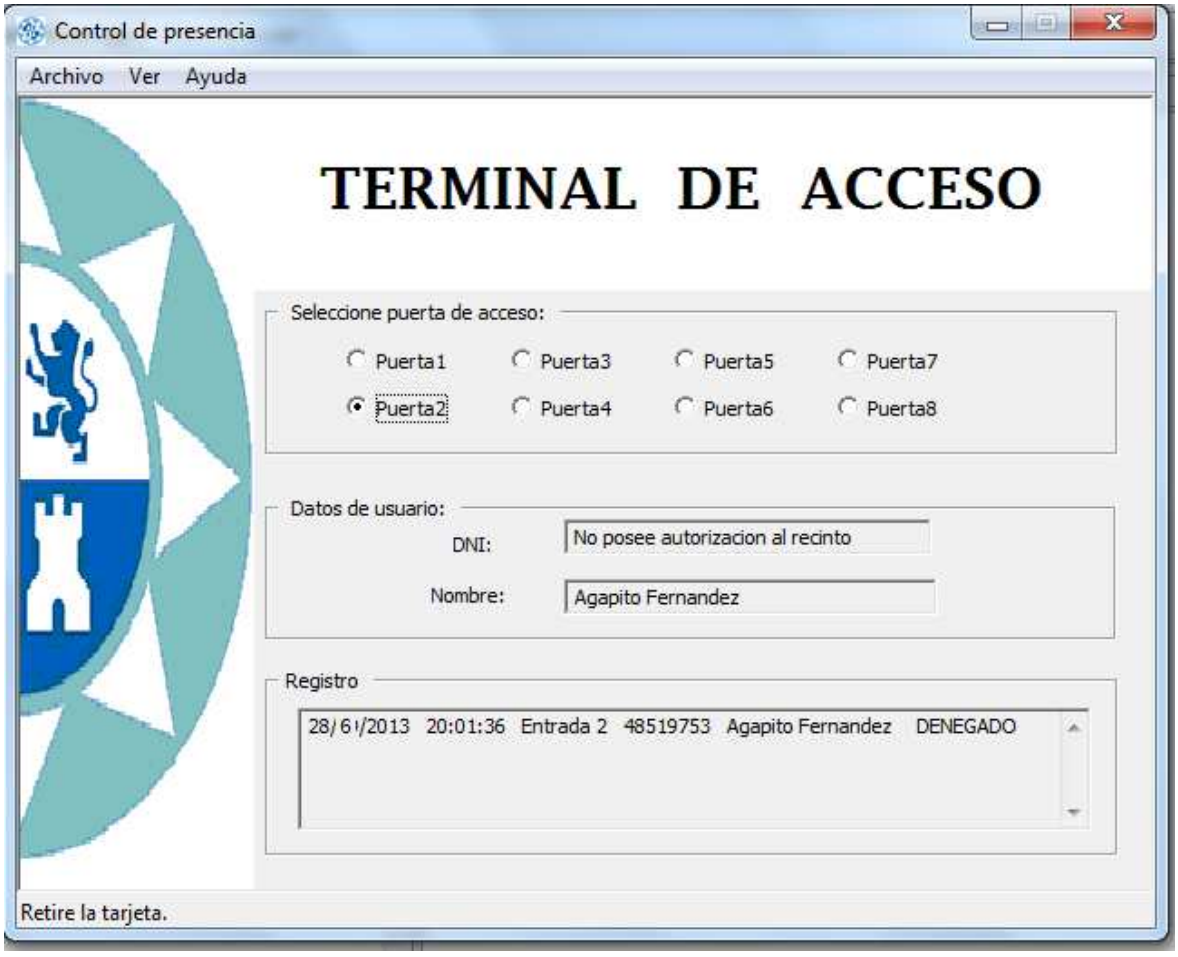

Figura 6.7 Interface de la aplicación –Control de Acceso-.

En el segundo caso, el procedimiento será poner el nombre del alumno en la sección “Nombre” de la interface y en la sección “DNI” aparecerá el texto “No posee autorización

al recinto", todo esto durante unos pocos segundos, informando al alumno de que no tiene permisos suficientes para acceder al recinto y dándole indicaciones para que retire la tarjeta en la barra de estado. En la sección "Registro" del interface, aparecerá una línea indicando día, hora, recinto al que se deseaba acceder, dni, nombre del alumno y el mensaje "Denegado".

En el tercer caso, el alumno está registrado y tiene suficientes permisos para acceder al recinto, el procedimiento será mostrar el dni y el nombre del alumno unos segundos, crear una entrada en la sección "Registro" del interface con la misma información que en el caso anterior cambiando el mensaje a "Valido" y enviar al PIC la información sobre el recinto al que el alumno desea acceder para que accione la cerradura y permita abrir la puerta.

Para el envío de la información desde la aplicación al PIC, habrá que configurar una salida serie, un puerto COM y enviar la información a través de el.

El puerto COM serie se crea y abre del mismo modo que un archivo o cualquier dispositivo de entrada/salida, con la función CreateFile():

- ```
  WINAPI CreateFile (LPCTSTR lpFileName, dwDesiredAccess,DWORD
  dwShareMode,LPSECURITY_ATTRIBUTES lpSecurityAttributes, DWORD
  dwCreationDisposition, DWORD dwFlagsAndAttributes,MANGO
  hTemplateFile );
  ```

El código para crear y configurar el puerto COM en esta aplicación es el siguiente:

```
hComPort = CreateFile(_T("COM9:"),GENERIC_READ | GENERIC_WRITE,0,0,
           OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);

DCB dcb;    // Estructura de datos de configuración de puesto
dcb.DCBlength=sizeof(DCB);
GetCommState(hComPort, &dcb); // Recuperar la configuracion actual
dcb.BaudRate = CBR_4800;
dcb.ByteSize = 8;
dcb.Parity= NOPARITY;
dcb.StopBits = TWOSTOPBITS;
dcb.fBinary= TRUE;
SetCommState(hComPort, &dcb);
```

Como se puede apreciar de las líneas de código anteriores la configuración del puerto será:

- Velocidad: 4800 baudios
- Bits de datos: 8
- Paridad: ninguna
- Bits de parada: 2

Con el puerto COM9 configurado, se procede al envió de los datos al PIC por medio de la función WriteFile().
```

- ```
  BOOL WINAPI WriteFile(HANDLE hFile, LPCVOID lpBuffer,DWORD
        nNumberOfBytesToWrite, LPDWORD lpNumberOfBytesWritten,
        LPOVERLAPPED lpOverlapped );
  ```

En el caso de esta aplicación la llamada a esta función tiene la siguiente forma:

- ```
  WriteFile( hComPort , &puerta_n  , 1  , &dwBytesRead, NULL);
  ```

Donde "hComPort" es el identificador del puerto COM9, "puerta_n" contiene la información que se desea transmitir, y "1" es el numero de bytes que se van a transmitir.

Terminada la transmisión se cerrará el puerto con ClosHandle();

- ```
  BOOL WINAPI CloseHandle(HANDLE hObject );
  ```

6.*2.2 Implementación de aplicación –Registro del Control de Acceso-*

La aplicación "Registro Control de Acceso" es básicamente una interface para la gestión de la base de datos. Como se comento en la etapa de diseño una interfaz debe proveer la funcionalidad necesaria para una buena comunicación entre el hombre y la máquina. Esta interface debe visualizar la información requerida por el usuario administrador y realizar las tareas más habituales con sólo unos pocos clicks de ratón.

Esta interfaz debe realizarse de manera que esta resulte simple e intuitiva de utilizar. La aplicación "Registro Control de Acceso" se ha implementado teniendo en cuenta estos requisitos y su aspecto es que se muestra en la *figura 6.8.*

Tal y como se ve en la *figura 6.8* se pueden identificar cinco grupos o apartados. Uno de ellos es el grupo "Datos Personales" en el que se mostrará la información sobre los datos personales (nombre y apellidos) de un registro cuya clave de búsqueda será el dni.

Los campos DNI, Nombre, Apellido 1 y Apellido 2 son de tipo CEdit, cajas en las que se puede mostrar o introducir texto, el campo DNI se utiliza para introducir el dni que se desee buscar o agregar.

Un registro solo se puede buscar o añadir por medio de la clave DNI, que debe constar de ocho caracteres numéricos.

El grupo "Periodos restringidos " lo forman tres casillas de verificación y dos subgrupos "Inicio de restricción " y "Fin de restricción". Este grupo, como su nombre indica, es el encargado de mostrar y gestionar las tres posibles restricciones temporales que se le pueden asignar a un alumno, y las cuales son:

1. Fines de semana: Si esta casilla esta activa, el alumno tendrá denegado el permiso de acceso los fines de semana.
  ```

2. Agosto: Si esta casilla esta activa, el alumno tendrá denegado el permiso de acceso todo el mes de Agosto.

3. Intervalo Temporal: Con esta casilla activada, el alumno tendrá restringido el acceso en el intervalo temporal comprendido entre “Inicio de restricción” y “Fin de restricción.”

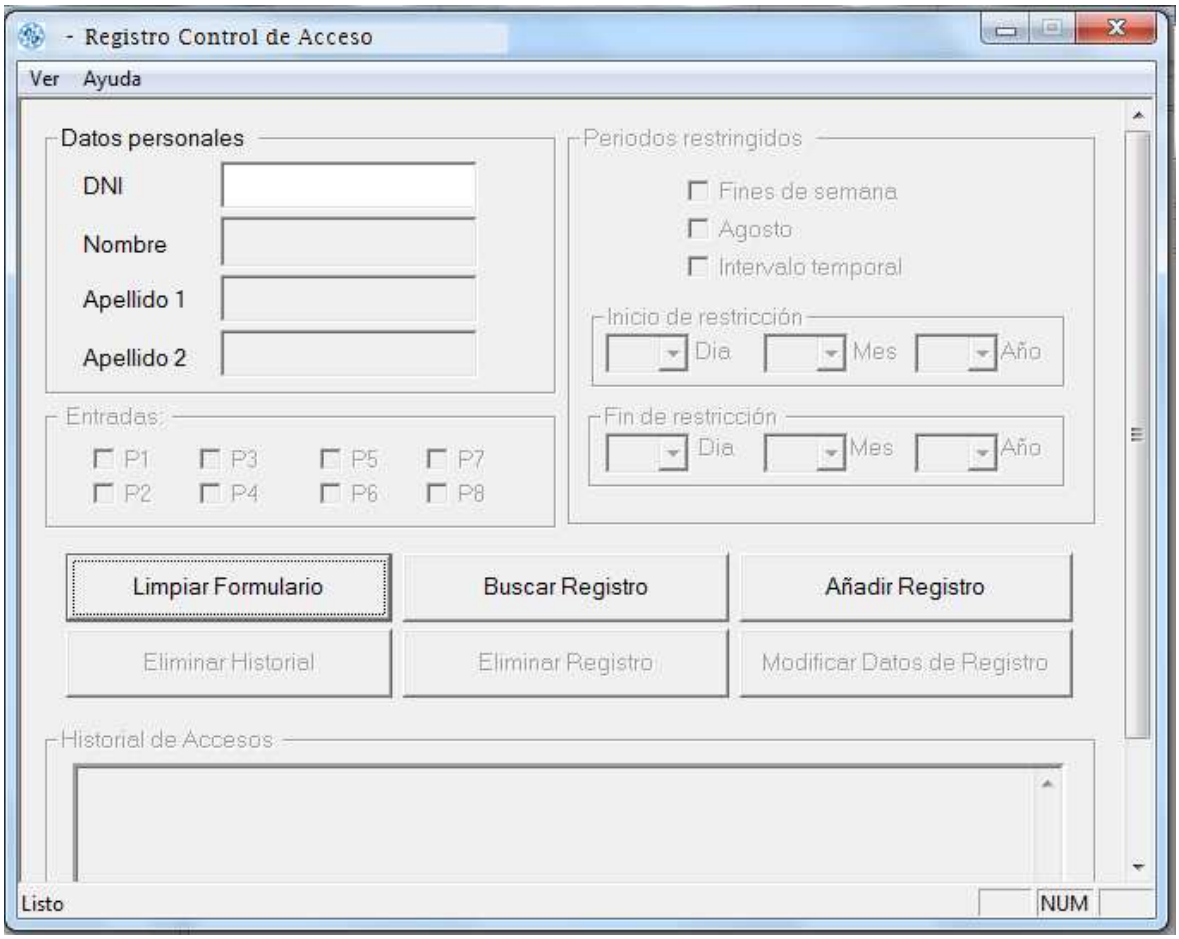

Figura 6.8 Interface de la aplicación –Registro Control de Acceso-.

El grupo “Entradas” lo forman ocho calillas de verificación, cada una de ellas indica si el alumno tiene permiso de acceso a cada uno de los ocho recintos cuyo acceso puede gestionar el sistema de control de acceso desarrollado en este proyecto.

El siguiente grupo lo forman seis botones, estos son los encargados de la funcionalidad.

1. **Limpiar Formulario**: Como el su mismo nombre indica, este botón se encarga de limpiar todos los campos de edición así como las casillas de verificación para poner la aplicación en su estado inicial. En estado inicial solo tres botones están activos, **Limpiar Formulario**, **Buscar Registro** y **Añadir Registro**.

2. **Buscar Registro**: Pulsado este botón se realizará la búsqueda del registro cuya clave DNI coincida con la introducida en el apartado DNI del formulario. En el caso de que el dni introducido sea erróneo o no esté registrado en la base de datos aparecerá una caja de dialogo informando del suceso. Si el dni es correcto y está registrado, aparecerá toda la información sobre este registro, datos personales, restricciones, permisos e historial. Esta información aparecerá en modo "*no-modificable*" para evitar posibles cambios no deseados. Al realizar una búsqueda con resultado positivo, automáticamente se activan los botones "**Eliminar Registro** " y "**Modificar Datos de Registro**" .

3. **Añadir Registro**: Si la clave DNI introducida es valida, al pulsar este botón se añadirá a la base de datos un registro con esta clave principal y se mostrará en un mensaje indicando que el registro ha sido añadido y las instrucciones para modificar datos y permisos de este registro.

4. **Eliminar Historial**: Pulsando este botón se eliminará el historial de accesos de un registro previamente seleccionado mediante "**Buscar Registro**", antes de realizar la operación la aplicación mostrara un mensaje de confirmación para evitar posibles errores.

5. **Eliminar Registro**: Este botón se usa para eliminar completamente un registro de la base de datos. El registro debe haber sido seleccionado previamente mediante "**Buscar Registro"**, por supuesto, antes de realizar la operación de eliminación se solicitará confirmación mediante una caja de dialogo informativa.

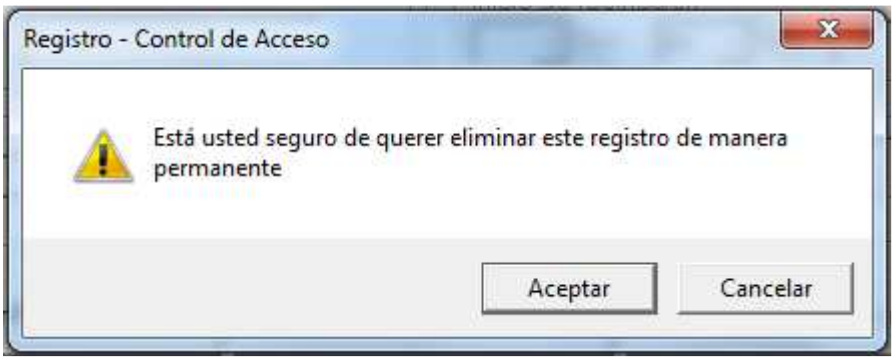

Figura 6.9 Caja de dialogo con solicitud de confirmación.

6. **Modificar Datos de Registro**: Este botón concede la capacidad de cambiar los datos asociados a un registro, datos personales, permisos y restricciones. Al pulsarlo aparecerán los botones "Guardar" y "Cancelar", para guardar los cambios realizados sobre un registro o no hacerlo.

El último grupo es el grupo "Historial de Accesos", esta compuesto por una caja de texto en la que se mostrará una entrada por cada acceso que haya realizado el alumno a un recinto, cada entrada contendrá información sobre la hora del acceso y el recinto al que se accede. En esta caja de texto no se podrá escribir ni modificar el

texto, solo mostrarlo por parte de la aplicación.

6.2.3 Implementación de la base de datos.

La base de datos ha sido realizada con el programa Access 2007 del paquete Office 2007 de Microsoft.

La base de datos, con nombre “alumnos.mdb”, está formada por dos tablas. La tabla “Alumnos” y la tabla “Picadas”.

1. Tabla “Alumnos”: Esta tabla contiene información sobre los datos personales del alumno, sus permisos y restricciones. Además cada registro en esta tabla puede contener muchos registros de la tabla “Picadas”. La clave principal es el dni del alumno, el nombre de este campo es “DNI”, es de tipo numérico, único y necesario (se requiere). El resto de los campos son ”Nombre”, ”Apellido1”, ”Apellido2”, ”Puerta”, “Agosto”, “Fin_de_semana”, “Periodo_temporal”, “FechaA” y “FechaB”, todos son de tipo texto, no únicos y no necesarios.

    - DNI: contiene el dni del alumno, es de tipo numérico con ocho caracteres, clave principal y por tanto único y requerido.

    - Nombre: este campo contiene el nombre del alumno, el mismo nombre se puede encontrar en distintos registros y pueden existir registros sin nombre.

    - Apellido1: Contiene el primer apellido del alumno, como el campo Nombre, el mismo apellido se puede encontrar en distintos registros y pueden existir registros este campo.

    - Apellido2: Campo que contiene el segundo apellido del alumno y con características similares al anterior.

    - Puerta: Este campo contiene los números de puerta a los que el alumno tiene permiso de acceso. Los números de puerta son del 1 al 8, de tener acceso a varias puertas el campo contendrá los números de puerta separados por coma y no necesariamente ordenados.

    - Agosto: Este campo indica si la restricción de acceso los fines de semana está activa para ente alumno. Un 1 indica restricción activa y un 0 restricción no activada.

    - Fin_de_semana: Campo que indica si el alumno tiene activada la restricción de entrada los fines de semana. Un 1 indica restricción activa y un 0 restricción no activada.

    - Periodo_temporal: Se usa este campo para indicar que el alumno tiene una

restricción temporal activa. Un 1 indica restricción activa y un 0 restricción no activada.

- FechaA: Cuando la restricción de periodo temporal está activada, este campo indica la fecha de inicio de la restricción, el formato será dd/mm/aa

- FechaB: Cuando la restricción de periodo temporal está activada, este campo indica la fecha de fin de la restricción, el formato será dd/mm/aa

2. Tabla "Picadas": Esta tabla esta compuesta por registros que se crean cada vez que un alumno accede a un recinto. "Picada" es el nombre que recibe coloquialmente el hecho de introducir la tarjeta identificativa de un trabajador en un lector para acceder a su lugar de trabajo, de ahí el nombre de esta tabla. Los campos que contiene son "DNI", "Fecha", "Hora" y "Puerta".

    - DNI: Clave principal, tipo numérico, y requerido. Por la naturaleza de esta tabla, pueden existir numerosos registros con el mismo dni.

    - Fecha: De tipo texto y formato dd/mm/aa es requerido. Contiene la fecha en que se realiza el acceso.

    - Hora: De tipo texto y formato hh/mm/ss es requerido. Contiene la hora en que se realiza el acceso.

    - Puerta: Este campo contiene el dato sobre el recinto al que se accede. Es de tipo texto aunque contiene el número de puerta al que se accede.

La relación existente entre ambas tablas se obtiene a través del campo "DNI" de estas. La relación se define como "1 es a muchos", es decir, un dni puede aparecer una única vez el la tabla "Alumnos" y muchas veces en la tabla "Picadas", de este modo el registro de un alumno en la tabla "Alumnos", identificado por su dni, puede contener muchas registros de accesos presentes en la tabla "picadas". Las relaciones existentes entre ambas bases de datos se exponen fácilmente en la *figura 6.10.*

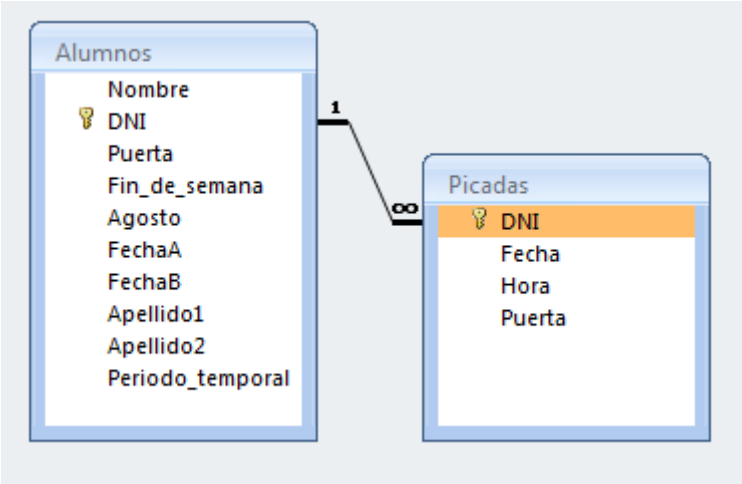

Figura 6.10 Diagrama relacional de la base de datos.

6.2.4 Implementación del software para el PICs.

Como se ha comentado en capítulos anteriores, el software del PIC se ha implementado en lenguaje ensamblador. El software que se debe implementar debe realizar la siguiente tarea: esperar la llegada de un dato, cuando se recibe el dato interpretarlo y activar la cerradura (led) correspondiente durante unos tres segundos.

En la figura 6.11 se representa el envío de un carácter a través del puerto COM.

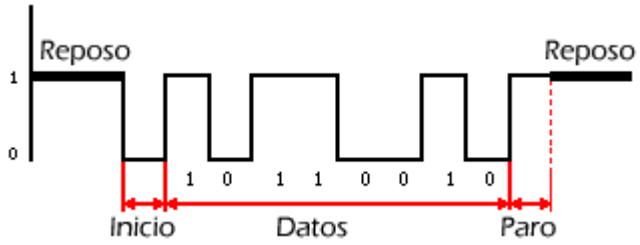

Figura 6.11 Envío del carácter ASCII ‘M’ a través del puerto COM

El carácter ASCII “M” es el numero 77, y en binario "01001101", de la figura 6.11 se puede extraer que en estado de reposo la línea de conexión permanece a 1 lógico, antes de comenzar el envío de datos la línea se pone en estado 0 lógico durante un tiempo igual al tiempo empleado para enviar un bit, después se envían los bits comenzando por el de menor peso, para indicar el fin del dato, la línea permanece en estado lógico 1 durante un tiempo de bit.

Lo primero a tener en cuenta antes de la implementación es la configuración del puerto COM, que será la siguiente:

- Bits por segundo = 4800 baudios
- Bits de datos = 8
- Paridad: ninguna
- Control de flujo: ninguno
- Bits de parada = 2

Puesto que se trata de una transmisión serie, se debe saber el tiempo que se tarda en transmitir un bit, esta información es de suma importancia a la hora de determinar cuando se debe tomar la información sobre el estado del dato (1 o 0).

- 4800 baudios -> 1/4800 seg -> 208,3 microsegundos

Por lo tanto se sabe que cada bit transmitido estará durante un tiempo de 208,3 microsegundos en la patilla de entrada del PIC.

Los includes necesarios son <RETARDOS.INC> y <P16F84A.INC>, en la librería retardos.inc se encuentran múltiples subrutinas de retardos, desde 4 microsegundos hasta 20 segundos. Además se pueden implementar otras subrutinas muy fácilmente.

Fuses, configuran opciones externas de hardware para la programación:

```
__CONFIG    _CP_OFF & _PWRTE_ON & _WDT_ON & _XT_OSC
```

La configuración de los puertos A y B del pic será la siguiente: RA1 entrada y el resto del PORTA salida y PORB se configurará entero como salida.

```
BSF     STATUS,RP0 ;
movlw B'00010'      ;
MOVWF       TRISA           ; Ra1 entrada
movlw B'00000000'   ;
MOVWF       TRISB           ; PuertoB=Salida
```

Una vez configurados los puertos, se pasa a esperar la llegada de un dato, se permanece testeando la entrada RA1 y cuando su estado cambie y tengamos un "0" este será el bit de inicio de trama, entonces se carga el literal 8, en el contador nBITS_RS232, que será el encargado de contar el número de bit que se está recibiendo, se introduce un retardo de 312 microsegundos, aproximadamente 1,5 veces el tiempo de bit y entonces se realiza la lectura del bit, porque se supone que en este tiempo, el bit de inicio ya se ha terminado de transmitir y el primer bit está transmitiéndose.

```
testearRA1
      btfsc PORTA,1           ; Comprobar RA1, Salta si RA1=0.
```

```
      goto  testearRA1        ;

      movlw d'8'              ; Número de bits a recibir.
      movwf nBITS_RS232 ;

      call  Retardo_200micros ; Retardo 312microsegundos
      call  Retardo_100micros ;
      call  Retardo_5micros   ;
      call  Retardo_5micros   ;
```

Se lee el bit y se introduce en el registro DatoRecibido, los bits se introducen por la izquierda del registro haciendo rotar a este hacia la derecha, después se introduce un retardo igual al tiempo de bit, se decrementa el contador nBITS_RS232 para comprobar si es el último bit, en caso de no ser el último se vuelve a leer el siguiente bit.

```
RS232_LeeBit
      bcf   STATUS,C          ; leer  pin. En principio supone que es 0.
      btfsc PORTA,1           ; ¿Realmente es cero?
      bsf   STATUS,C          ; No, pues cambia a "1".
      rrf   DatoRecibido,F    ; Introduce bit en  registro de lectura.
      call  Retardo_5micros         ;
      call  Retardo_200micros ;
      decfsz nBITS_RS232,F    ; Comprueba que es el último bit.
      goto  RS232_LeeBit      ; no es último  pasa a leer el siguiente.
```

En caso de tratarse del último bit se introduce un retardo igual a dos veces el tiempo de bit y se mueve el dato al registro acumulador W, y se llama a la rutina “puerta”, esta rutina devolverá en W la salida del PORTB, activando la cerradura (led) deseada.

```
call  Retardo_200micros ;
      call  Retardo_200micros ;
      call  Retardo_5micros   ; Esperar los 2 bits de Stop.
      call  Retardo_5micros   ;
      call  Retardo_5micros   ;
      movf  DatoRecibido,W    ; El resultado en el registro W.
call  puerta                  ;
      movwf PORTB             ;
```

Tras activar la salida correspondiente se introduce un retardo de aproximadamente tres segundos tras el cual todas las salidas serán desactivadas y se volverá al estado de esperar llegada de dato.

```
movlw B'10010110'       ; 150
      movwf c3s               ; 20 milisegundos x50 = 1 segundo
temporizacion3s
      clrwdt                        ; x150 =3 segundos
      call  Retardo_20ms            ;
      decfsz      c3s,1             ;
      goto  temporizacion3s         ;
      movlw B'00000000'       ;
      movwf PORTB             ;
      goto  testearRA1        ;
```

7 Experimentación.

Una vez finalizada la implementación, se deben realizar una serie de pruebas para evaluar el correcto funcionamiento del sistema, tanto de la aplicación para el control de acceso con los dispositivos asociados (Tarjeta, Lector y PIC) como de la aplicación para la gestión de la base de datos.

Las pruebas realizadas son:

- Instalación del sistema en distintas máquinas.
- Comprobación del correcto funcionamiento.

7.1. Instalación y comprobación del sistema.

El periférico se ha probado en varias máquinas con distintos sistemas operativos. El objetivo de estas pruebas era comprobar que el periférico era reconocido como un dispositivo USB y que todo el proceso de instalación y comunicación periférico-host era correcto.

Las máquinas donde se han llevado a cabo las pruebas son:

- ASUS con procesador Intel Core i3 a 2.53 GHz, memoria RAM de 4 GB y sistema operativo Windows 7 Home Premium.
- OKI con procesador Pentium Dual Core T2330 a 1.6 GHz, memoria RAM de 1 GB
- Sony Vaio con procesador Intel Pentium M a 1 GHz, memoria RAM de 1 GB y sistema operativo Windows XP Profesional.
- Compaq con procesador Intel Pentium III a 1 GHz, memoria RAM de 256 MB y sistema operativo Windows XP Home Edition.

En todos ellos se han instalado y ejecutado correctamente las aplicaciones “Control de ACceso” y “Registro Control de Acceso”.

En algunos casos ha sido necesario instalar paquete redistribuible de *.NET Framework* versión 1.1 que incluye lo necesario para ejecutar aplicaciones desarrolladas con *.NET Framework*.

Para el caso de los ordenadores portátiles que a menudo no incorporan interface RS-232 se ha usado un cable conversor USB/RS232, una vez instalado el controlador para este cable en el ordenador, las maquinas conectadas con el envían y reciben información por el puerto COM como si se tratase de una interface rs-232 normal, la transformación de las señales se realiza de forma opaca a ambas maquinas.

Para la comprobación del sistema para el control de acceso es necesario verificar que las aplicaciones realizan de forma correcta las funciones que se indicaron en el diseño.

Las pruebas realizadas consisten en:

- Agregar un alumno a la base de datos (inicialmente vacía).
- Dotar de permisos para acceder a varios recintos a ese alumno.
- Introducir la tarjeta estudiante del alumno agregado solicitando el acceso a un recinto al que tiene permiso y observar la respuesta con la correcta activación del led indicado.
- Introducir la tarjeta estudiante del alumno agregado solicitando el acceso a un recinto al que no tiene permiso y observar la respuesta negativa al acceso.
- Consultar el historial del alumno registrado.
- Modificar los datos del alumno registrado.
- Eliminar el registro del alumno agregado.

Se ha conseguido realizar con éxito todas las operaciones que estaban previstas en la etapa de diseño del sistema control de acceso.

8 Conclusiones y Líneas futuras.

Después de haber realizado las pruebas y haber examinado los resultados obtenidos, se pueden extraer conclusiones sobre el proyecto. Tanto el PIC como las aplicaciones software desarrolladas satisfacen todos los objetivos marcados inicialmente.
El resultado del proyecto ha sido la obtención de un sistema para el control de acceso a uno o varios recintos mediante lector estándar de tarjeta inteligente, tarjeta de estudiante UPCT y un microcontrolador.

Aunque se han cumplido los objetivos del proyecto, es posible mejorarlo añadiendo nuevas funcionalidades.

Como trabajo futuro se propone:

- Ampliar el sistema para controlar el uso del estudiante de ordenadores dentro de los recintos a los que tiene acceso.

- Usando esta tecnología se podría crear un sistema para el control de asistencia a clases o exámenes, de forma informatizada, en detrimento de las "típicas hojas de firmas".

- En un apartado más electrónico, ya que se expone en este proyecto toda la información referente a la tarjeta, al lector y a como se realiza la comunicación entre ellos (APDUS, protocolos de comunicación, etc.), se podría llevar a cabo un proyecto en el que la finalidad fuera conseguir comunicarnos con la tarjeta de estudiante mediante un PIC, es decir fabricar el lector de tarjetas inteligentes propio de la UPCT.

9 Anexos

9.1 Código de la aplicación –Control de Acceso-

alen.cpp: define los comportamientos de las clases para la aplicación.

```
#include "stdafx.h"
#include "alen.h"
#include "MainFrm.h"

#include "alenDoc.h"
#include "alenView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CalenApp

BEGIN_MESSAGE_MAP(CalenApp, CWinApp)
	ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
	// Comandos de documento estándar basados en archivo
	ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
	ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()


// Construcción de CalenApp
```

```
CalenApp::CalenApp()
{
	// TODO: agregar aquí el código de construcción,
	// Colocar toda la inicialización importante en InitInstance
}


// El único objeto CalenApp

CalenApp theApp;

// Inicialización de CalenApp

BOOL CalenApp::InitInstance()
{
	// Windows XP requiere InitCommonControls() si un manifiesto de
	// aplicación especifica el uso de ComCtl32.dll versión 6 o
posterior para habilitar
	// estilos visuales. De lo contrario, se generará un error al crear
ventanas.
	InitCommonControls();

	CWinApp::InitInstance();

	// Inicializar bibliotecas OLE
	if (!AfxOleInit())
	{
		AfxMessageBox(IDP_OLE_INIT_FAILED);
		return FALSE;
	}
	AfxEnableControlContainer();
	// Inicialización estándar
	// Si no utiliza estas características y desea reducir el tamaño
	// del archivo ejecutable final, debe quitar
	// las rutinas de inicialización específicas que no necesite
	// Cambie la clave del Registro en la que se almacena la
configuración
	// TODO: debe modificar esta cadena para que contenga información
correcta
	// como el nombre de su compañía u organización
	SetRegistryKey(_T("Aplicaciones generadas con el Asistente para
aplicaciones local"));
	LoadStdProfileSettings(4);  // Cargar opciones de archivo INI
estándar (incluidas las de la lista MRU)
	// Registrar las plantillas de documento de la aplicación. Las
plantillas de documento
	//  sirven como conexión entre documentos, ventanas de marco y
vistas
	CSingleDocTemplate* pDocTemplate;
	pDocTemplate = new CSingleDocTemplate(
		IDR_MAINFRAME,
		RUNTIME_CLASS(CalenDoc),
		RUNTIME_CLASS(CMainFrame),       // Ventana de marco MDI
principal
		RUNTIME_CLASS(CalenView));
	if (!pDocTemplate)
		return FALSE;
	AddDocTemplate(pDocTemplate);
```

```
	// Analizar línea de comandos para comandos Shell estándar, DDE,
Archivo Abrir
	CCommandLineInfo cmdInfo;
	ParseCommandLine(cmdInfo);
	// Enviar comandos especificados en la línea de comandos. Devolverá
FALSE si
	// la aplicación se inició con los modificadores /RegServer,
/Register, /Unregserver o /Unregister.
	if (!ProcessShellCommand(cmdInfo))
		return FALSE;
	// Se ha inicializado la única ventana; mostrarla y actualizarla
	m_pMainWnd->ShowWindow(SW_SHOW);
	m_pMainWnd->UpdateWindow();
	// Llamar a DragAcceptFiles sólo si existe un sufijo
	//  En una aplicación SDI, esto debe ocurrir después de
ProcessShellCommand
	return TRUE;
}



// Cuadro de diálogo CAboutDlg utilizado para el comando Acerca de

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Datos del cuadro de diálogo
	enum { IDD = IDD_ABOUTBOX };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    //
Compatibilidad con DDX/DDV

// Implementación
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// Comando de la aplicación para ejecutar el cuadro de diálogo
void CalenApp::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
}
```

```
// Controladores de mensaje de CalenApp
// Aquí termina el código de la clase alen.cpp
```

alenView.cpp: implementación de la clase CalenView

```
#include "stdafx.h"
#include "alen.h"
#include "tarjeta.h"
#include "alenDoc.h"
#include "alenView.h"
#include ".\alenview.h"
#include "MainFrm.h"
#include "Bytes.h"
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <afxdb.h>
#include <sstream>

using namespace std;
#include <string.h>
#include <time.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CalenView

IMPLEMENT_DYNCREATE(CalenView, CFormView)

BEGIN_MESSAGE_MAP(CalenView, CFormView)
	ON_WM_TIMER()
	ON_BN_CLICKED(IDC_BUTTON1, OnBnClickedButton1)
	ON_BN_CLICKED(IDC_BUTTON2, OnBnClickedButton2)
	ON_EN_CHANGE(IDC_EDIT1, OnEnChangeEdit1)
	ON_EN_CHANGE(IDC_EDIT2, OnEnChangeEdit2)
END_MESSAGE_MAP()

// Construcción o destrucción de CalenView

CalenView::CalenView()
	: CFormView(CalenView::IDD)
{
	n_registros = 0;
	flags=1;		// indicador de estado anterior 0->con tarjeta,
1->sin tarjeta
	ctiempovisible=0; //controlara el tiempo que permanecen visibles
nombre y dni


}

CalenView::~CalenView()
```

```
{
}

void CalenView::DoDataExchange(CDataExchange* pDX)
{
	CFormView::DoDataExchange(pDX);
}

BOOL CalenView::PreCreateWindow(CREATESTRUCT& cs)
{
	// TODO: modificar aquí la clase Window o los estilos cambiando
	//  CREATESTRUCT cs

	return CFormView::PreCreateWindow(cs);
}

void CalenView::OnInitialUpdate()
{

	//ESTABLECEMOS EL TEMPORIZADOR
	SetTimer(ID_1SEGUNDO,1000,NULL);

	CFormView::OnInitialUpdate();
	GetParentFrame()->RecalcLayout();
	ResizeParentToFit();
}


// Diagnósticos de CalenView

#ifdef _DEBUG
void CalenView::AssertValid() const
{
	CFormView::AssertValid();
}

void CalenView::Dump(CDumpContext& dc) const
{
	CFormView::Dump(dc);
}

CalenDoc* CalenView::GetDocument() const // La versión de no depuración
es en línea
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CalenDoc)));
	return (CalenDoc*)m_pDocument;
}
#endif //_DEBUG


// Controladores de mensaje de CalenView



void CalenView::OnTimer(UINT nIDEvent){

	ctiempovisible++;
	if(ctiempovisible>1){
		SetDlgItemText(IDC_EDIT2,LPCTSTR(" "));
		SetDlgItemText(IDC_EDIT1,LPCTSTR(" "));
```

```
            ctiempovisible=0;
      }

      time_t rawtime;  //  usado para obtener la fecha del sistema.
      char* fecha ;
      int a=0;
      Bytes *dni;
      Bytes d2 =Bytes();
      d2.len=11;
      dni=&d2;
      dni->len=11;
      //dni=NULL
      char nombre_leido[40];
      nombre_leido[0]='\0';
      char registro3[70];
      registro3[0]='\0';
      unsigned char *d;
      unsigned char d1[11];
    for(int i =0;i<11 ; i++){
        d1[i]='\0';
      }
      d1[10]='\0';
      d=d1;


      Tarjeta estudiante1 = Tarjeta();

      a=estudiante1.connect();// Establecer la conexion con el lector

      if (a == -1){

                  SetDlgItemText(IDC_EDIT1,LPCTSTR("Error de
establecimiento de contexto."));
                  SetDlgItemText(IDC_EDIT2,LPCTSTR(""));
                  CString sReplacementText = "Aborte de inmediato la
aplicacion y vuelba a intentarlo mas tarde.";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));
                  estudiante1.~Tarjeta();

      }else if(a== -2){
                  SetDlgItemText(IDC_EDIT1,LPCTSTR(""));
                  SetDlgItemText(IDC_EDIT2,LPCTSTR(""));
                  CString sReplacementText = "Conecte un lector al
equipo.";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));
                  estudiante1.~Tarjeta();

      }else if(a== -3){

                  SetDlgItemText(IDC_EDIT1,LPCTSTR("Error al leer el
registro estatus."));
                  CString sReplacementText = "Estraiga la targeta, espere
unos segundos y reintentelo";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));
                  estudiante1.disconnect();
                  estudiante1.~Tarjeta();
                  flags=1;
```

```
      }else if(a== -4){

                  SetDlgItemText(IDC_EDIT1,LPCTSTR("Error seleccionar UM
del registro UM."));
                  CString sReplacementText = "Estraiga la targeta, espere
unos segundos y reintentelo";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));
                  estudiante1.disconnect();
                  estudiante1.~Tarjeta();
                  flags=1;

      }else if(a==-7){
                  SetDlgItemText(IDC_EDIT1,LPCTSTR("Error al enviar
APDU."));
                  CString sReplacementText = "Estraiga la targeta, espere
unos segundos y reintentelo";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));
                  estudiante1.disconnect();
                  estudiante1.~Tarjeta();
                  flags=1;

      }else if(a==-6){        // a=-6 -> cuando no hay tarjeta
            if(flags==1){           //Si en el estado anteriior habia
tarjeta y haora no hay
                  SetDlgItemText(IDC_EDIT1,LPCTSTR(""));
                  SetDlgItemText(IDC_EDIT2,LPCTSTR(""));

                  CString sReplacementText = "Espere unos segundos antes
de insretar tarjeta.";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));
                  estudiante1.disconnect();
                  Sleep(4000);
                  estudiante1.~Tarjeta();
                  flags=0;
            }else{

                        CString sReplacementText = "Introduzca la tarjeta
de estudiante.";
                        ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));
                        estudiante1.~Tarjeta();
                        flags=0;

            }
      }else{

            if(flags==1){

                  CString sReplacementText = "DNI ya registrado, extraiga
la targeta.";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));
                  estudiante1.disconnect();
                  estudiante1.~Tarjeta();

            }else{
```

```
                  CString sReplacementText = "Procediendo a la lectura
del dni";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));
                  a=estudiante1.readDNI(&dni);  //
                  unsigned char*(d)=dni->bytes();
                  if(d==NULL){
                        SetDlgItemText(IDC_EDIT1,LPCTSTR("Error de
lectura, espere unos segundos."));
                        Sleep(3000);
                        a=estudiante1.readDNI(&dni);  //
                        unsigned char*(d)=dni->bytes();
                        if(d==NULL){
                              estudiante1.disconnect();
                              estudiante1.~Tarjeta();
                     CString sReplacementText = "Estraiga la targeta.";
                              ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));
                              SetDlgItemText(IDC_EDIT1,LPCTSTR("Retire la
tarjeta y vuelva a intentarlo."));
                              flags=1;
                        }else{

                              estudiante1.disconnect();
                              estudiante1.~Tarjeta();
                              time_t rawtime;
                              char* fecha ;
                              time ( &rawtime );
                              fecha = ctime (&rawtime);     // formato de
la variable fecha: Sat May 20 12:13:00 2010
                              char mesc[3];
                              int mesn;
                              char  dia[3];
                              char  ao[5];
                              char hora[9];
                              char  fechad[12];
                              char  mes[4];
                              for(int i =4;i<7 ; i++){
                                    mes[i-4]=fecha[i];
                              }
                              mes[3] ='\0';
                              if(strcmp(mes,"Jan")==0){
                                    mesc[0]='1';
                                    mesc[1]='\0';
                              }
                              if(strcmp(mes,"Feb")==0){
                                    mesc[0]='2';
                                    mesc[1]='\0';
                              }
                              if(strcmp(mes,"Mar")==0){
                                    mesc[0]='3';
                                    mesc[1]='\0';
                              }
                              if(strcmp(mes,"Apr")==0){
                                    mesc[0]='4';
                                    mesc[1]='\0';
                              }
                              if(strcmp(mes,"May")==0){
                                    mesc[0]='5';
```

```
                              mesc[1]='\0';
                        }
                        if(strcmp(mes,"JUN")==0){
                              mesc[0]='6';
                              mesc[1]='\0';
                        }
                        if(strcmp(mes,"Jul")==0){
                              mesc[0]='7';
                              mesc[1]='\0';
                        }
                        if(strcmp(mes,"Agu")==0){
                              mesc[0]='8';
                              mesc[1]='\0';
                        }
                        if(strcmp(mes,"Sep")==0){
                              mesc[0]='9';
                              mesc[1]='\0';
                        }
                        if(strcmp(mes,"Oct")==0){
                              mesc[0]='1';
                              mesc[1]='0';
                              mesc[2]='\0';
                        }
                        if(strcmp(mes,"Nov")==0){
                              mesc[0]='1';
                              mesc[1]='1';
                              mesc[2]='\0';
                        }
                        if(strcmp(mes,"Dec")==0){
                              mesc[0]='1';
                              mesc[1]='2';
                              mesc[2]='\0';
                        }
                        for(int i =8; i<10 ; i++){
                              dia[i-8]=fecha[i];
                        }
                        dia[2] ='\0';

                        for(int i =20;i<24 ; i++){
                              ao[i-20]=fecha[i];

                        }
                        ao[4] ='\0';

                        for(int i =11; i<19 ; i++){        // Dar
formato a la hora
                              hora[i-11]=fecha[i];
                        }
                        hora[8]='\0';          // Tengo la hora en
formato hh:mm:ss en "hora"
                        fechad[0]= '\0';
                        strcat(fechad,dia);
                        strcat(fechad,"/");
                        strcat(fechad,mesc);
                        strcat(fechad,"/");
                        strcat(fechad,ao);     // Tengo la fecha en
formato dd/mm/aaaa en "fechad"
                        //************************************
                        SetDlgItemText(IDC_EDIT1,LPCTSTR(fechad));
                        SetDlgItemText(IDC_EDIT2,LPCTSTR(hora));
```

```
                              Sleep(4000);

                              //    Abrir la base d datos y trabajar con
ella//

      ////////////////////////////////////////////
                              CDaoDatabase *pDao = new CDaoDatabase;
                              pDao->Open ("Alumnos.mdb");
                              COleVariant v;
                              CDaoRecordset r(pDao);
                              char iip[42];
                              CString sd ="SELECT * FROM Alumnos WHERE
dni =";     //33 caracteres
                              for(int l =0; l<33 ; l++){
                                    iip[l]=sd[l];
                              }
                              iip[33]='\0';
                              char opl[9];
                              opl[0]=d[2];
                              opl[1]=d[3];
                              opl[2]=d[4];
                              opl[3]=d[5];
                              opl[4]=d[6];
                              opl[5]=d[7];
                              opl[6]=d[8];
                              opl[7]=d[9];
                              opl[8]='\0';
                              strcat( iip,opl);

                              SetDlgItemText(IDC_EDIT1,LPCTSTR(d));
                              int no_registrado = 0 ;
                              try{
                                    r.Open(dbOpenDynaset,LPCTSTR(iip)
,dbReadOnly );
                                    r.GetFieldValue ("Nombre", v);
                                    //(v) puede tener los siguientes
tipos
                                    //bstrVal -> BSTR (una cadena).
                                    //bVal -> unsigned char
                                    //iVal -> short
                                    //lVal = long
                                    //fltVal -> float
                                    //dblVal -> double
                                    strcat(nombre_leido
,(char*)v.bstrVal);

      SetDlgItemText(IDC_EDIT2,LPCTSTR(v.bstrVal));

      SetDlgItemText(IDC_EDIT1,LPCTSTR(opl));

                                    r.Close();
                              }catch(CDaoException* e ) {
                                    long number =e->m_pErrorInfo-
>m_lErrorCode;
                                    std::ostringstream stm ;
                                    stm << number ;
                                    const char* cstr = stm.str().c_str()
;
                                    CString tt = cstr;
```

```
                                    if(number == 3021){

      SetDlgItemText(IDC_EDIT2,LPCTSTR("Nombre no registrado"));
                                          no_registrado =1;
                                          r.Close();

                                    }else{

      SetDlgItemText(IDC_EDIT2,LPCTSTR("Algun problema en BBDD"));
                                          AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                                          r.Close();
                                    }
                                    e->Delete();
                              }
                              if(no_registrado == 1){
                                    try{
                                          r.Open(dbOpenDynaset,"SELECT *
FROM Alumnos ",dbAppendOnly);
                                          r.AddNew();
                                          r.SetFieldValue ("DNI",
LPCTSTR(opl)); //-> variable contiene el //dato a insertar

      r.SetFieldValue("Nombre",LPCTSTR("Nombre no registrado"));
                                          r.Update ();
                                          r.Close();
                                          strcat(nombre_leido ,"Nombre no
registrado");
                                    }catch(CDaoException* e){
                                          AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                                          e->Delete();
                                          r.Close();
                                    }

                              }
                              try{
                                    r.Open(dbOpenDynaset,"SELECT * FROM
Picadas ",dbAppendOnly);
                                    r.AddNew();
                                    r.SetFieldValue ("DNI",
LPCTSTR(opl)); //-> variable contiene el //dato a insertar

      r.SetFieldValue("Fecha",LPCTSTR(fechad));

      r.SetFieldValue("Hora",LPCTSTR(hora));
                                    r.Update ();
                              }catch(CDaoException* e){
                                    AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                                    e->Delete();
                              }
                              pDao->Close();    //Para cerrar la conexion
con la BBDD y liberar memoria:
                              delete pDao;
                              flags=1;
                              ctiempovisible=0;
```

```
      //*****************************************************************\
\
      //*****************************************************************\
\
      // ******************** Escribir en el campo "Datos registrados"
\\
      //    edit3 = matriz de 20x70
                              registro3[0] ='\0';
                              strcat(registro3 ,"\n");
                              strcat(registro3 ,fechad);
                              strcat(registro3 ,"\t");
                              strcat(registro3 ,hora);
                              strcat(registro3 ,"\t \t");
                              strcat(registro3 ,opl);
                              strcat(registro3 ,"\t");
                              strcat(registro3 ,nombre_leido);
                              strcat(registro3 ,"\r\n");
                              if(n_registros<20)
                                    n_registros++;
                              else{
                                    for(int v =0; v<19 ; v++){
                                          for(int z=0 ; z<70; z++)
                                 edit3[v][z]=edit3[v+1][z];
                                    }
                              }
                              char v_escribir[1420];
                              v_escribir[0]='\0';

                                    for(int i=0; i<71 ; i++){
                               edit3[n_registros-1][i]=registro3[i];
                                          if(registro3[i]=='\0')
                                                break;
                              }

                                    for(int i=0 ; i < n_registros ; i++){
                                          strcat(v_escribir,edit3[i]);
                              }

                                    SetDlgItemText ( IDC_EDIT3, "");
                                    SendDlgItemMessage ( IDC_EDIT3,
EM_REPLACESEL, 0, LPARAM(v_escribir) );
      ////////////*********************************************\\\\\\\\\
\\\\\\\\\\\\\\\\\\\
      ///////////*************************************************\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\

                        }
                  }else{

                        estudiante1.disconnect();
                        estudiante1.~Tarjeta();

                              time_t rawtime;
// lo usaremos para obtener la fecha del sistema.
                              char* fecha ;
                              time ( &rawtime );
                              fecha = ctime (&rawtime);     // formato de
la variable fecha: Sat May 20 12:13:00 2010
                              char mesc[3];
                              int mesn;
```

```
char  dia[3];
char  ao[5];
char hora[9];
char  fechad[12];
char  mes[4];
for(int i =4;i<7 ; i++){
      mes[i-4]=fecha[i];
}
mes[3] ='\0';
if(strcmp(mes,"Jan")==0){
      mesc[0]='1';
      mesc[1]='\0';
}
if(strcmp(mes,"Feb")==0){
      mesc[0]='2';
      mesc[1]='\0';
}
if(strcmp(mes,"Mar")==0){
      mesc[0]='3';
      mesc[1]='\0';
}
if(strcmp(mes,"Apr")==0){
      mesc[0]='4';
      mesc[1]='\0';
}
if(strcmp(mes,"May")==0){
      mesc[0]='5';
      mesc[1]='\0';
}
if(strcmp(mes,"JUN")==0){
      mesc[0]='6';
      mesc[1]='\0';
}
if(strcmp(mes,"Jul")==0){
      mesc[0]='7';
      mesc[1]='\0';
}
if(strcmp(mes,"Agu")==0){
      mesc[0]='8';
      mesc[1]='\0';
}
if(strcmp(mes,"Sep")==0){
      mesc[0]='9';
      mesc[1]='\0';
}
if(strcmp(mes,"Oct")==0){
      mesc[0]='1';
      mesc[1]='0';
      mesc[2]='\0';
}
if(strcmp(mes,"Nov")==0){
      mesc[0]='1';
      mesc[1]='1';
      mesc[2]='\0';
}
if(strcmp(mes,"Dec")==0){
      mesc[0]='1';
      mesc[1]='2';
      mesc[2]='\0';
}
```

```
                              for(int i =8; i<10 ; i++){
                                    dia[i-8]=fecha[i];
                              }
                              dia[2] ='\0';

                              for(int i =20;i<24 ; i++){
                                    ao[i-20]=fecha[i];

                              }
                              ao[4] ='\0';

                              for(int i =11; i<19 ; i++){        // Dar
formato a la hora
                                    hora[i-11]=fecha[i];
                              }
                              hora[8]='\0';          // Tengo la hora en
formato hh:mm:ss en "hora"
                              fechad[0]= '\0';
                              strcat(fechad,dia);
                              strcat(fechad,"/");
                              strcat(fechad,mesc);
                              strcat(fechad,"/");
                              strcat(fechad,ao);     // Tengo la fecha en
formato dd/mm/aaaa en "fechad"



                              //    Abrir la base d datos y trabajar con
ella//

      ////////////////////////////////////////////
                              CDaoDatabase *pDao = new CDaoDatabase;
                              pDao->Open ("Alumnos.mdb");
                              COleVariant v;
                              CDaoRecordset r(pDao);
                              int es_nuevo =0;
                              char iip[42];
                              CString sd ="SELECT * FROM Alumnos WHERE
dni =";     //33 caracteres
                              for(int l =0; l<33 ; l++){
                                    iip[l]=sd[l];
                              }
                              iip[33]='\0';
                              char opl[9];
                              opl[0]=d[2];
                              opl[1]=d[3];
                              opl[2]=d[4];
                              opl[3]=d[5];
                              opl[4]=d[6];
                              opl[5]=d[7];
                              opl[6]=d[8];
                              opl[7]=d[9];
                              opl[8]='\0';
                              strcat( iip,opl);

                              SetDlgItemText(IDC_EDIT3,LPCTSTR("\n Antes
de abrir la base"));
                              SetDlgItemText(IDC_EDIT1,LPCTSTR(opl));

                              try{
```

```
                                    r.Open(dbOpenDynaset,LPCTSTR(iip)
,dbReadOnly );
                                    r.GetFieldValue ("Nombre", v);
                                    //(v) puede tener los siguientes
tipos
                                    //bstrVal -> BSTR (una cadena).
                                    //bVal -> unsigned char
                                    //iVal -> short
                                    //lVal = long
                                    //fltVal -> float
                                    //dblVal -> double
                                    strcat(nombre_leido
,(char*)v.bstrVal);

      SetDlgItemText(IDC_EDIT2,LPCTSTR(v.bstrVal));

      SetDlgItemText(IDC_EDIT1,LPCTSTR(opl));
                                    r.Close();
                              }catch(CDaoException* e ) {
                                    long number =e->m_pErrorInfo-
>m_lErrorCode;
                                    std::ostringstream stm ;
                                    stm << number ;
                                    const char* cstr = stm.str().c_str()
;
                                    CString tt = cstr;

                                    if(number == 3021){

      SetDlgItemText(IDC_EDIT2,LPCTSTR("Nombre no registrado"));
                                          es_nuevo=1;
                                          strcat(nombre_leido ,"Nombre no
registrado");
                                          r.Close();
                                    }else{

      SetDlgItemText(IDC_EDIT2,LPCTSTR("Algun problema en BBDD"));
                                          AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                                          e->Delete();
                                          r.Close();
                                    }
                                    e->Delete();
                              }
                              if(es_nuevo==1){
                                    try{
                                          r.Open(dbOpenDynaset,"SELECT *
FROM Alumnos ",dbAppendOnly);
                                          r.AddNew();
                                          r.SetFieldValue ("DNI",
LPCTSTR(opl)); //-> variable contiene el //dato a insertar

      r.SetFieldValue("Nombre",LPCTSTR("Nombre no registrado"));
                                          r.Update ();
                                          r.Close();
                                    }catch(CDaoException* e){
                                          AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                                          e->Delete();
                                          r.Close();
```

```
                                    }

                              }

                              try{
                                    r.Open(dbOpenDynaset,"SELECT * FROM
Picadas ",dbAppendOnly);
                                    r.AddNew();
                                    r.SetFieldValue ("DNI",
LPCTSTR(opl)); //-> variable contiene el //dato a insertar

      r.SetFieldValue("Fecha",LPCTSTR(fechad));

      r.SetFieldValue("Hora",LPCTSTR(hora));
                                    r.Update ();
                              }catch(CDaoException* e){
                                    AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                                    e->Delete();
                              }
                              pDao->Close();    //Para cerrar la conexion
con la BBDD y liberar memoria:
                              delete pDao;
                              flags=1;
                              ctiempovisible=0;



      //*****************************************\\

      //*****************************************\\
                              //  Escribir en el campo "Datos
registrados" \\
                              registro3[0] ='\0';
                              strcat(registro3 ,"\n");
                              strcat(registro3 ,fechad);
                              strcat(registro3 ,"\t");
                              strcat(registro3 ,hora);
                              strcat(registro3 ,"\t \t");
                              strcat(registro3 ,opl);
                              strcat(registro3 ,"\t");
                              strcat(registro3 ,nombre_leido);
                              strcat(registro3 ,"\r\n");
                              char v_escribir[1420];
                              v_escribir[0]='\0';
                              if(n_registros<20){     // se mostrará un
maximo de 20 registros en pantalla.
                                    n_registros++;
                              }else{
                                    for(int v =0; v<19 ; v++){
                                          for(int z=0; z<70 ;z++){
                                    edit3[v][z]= edit3[v+1][z];
                                          }
                                    }
                              }
                              for(int i=0; i<71 ; i++){
                        edit3[n_registros-1][i]=registro3[i];
                                    if(registro3[i]=='\0')
                                          break;
                              }
```

```
                              for(int i=0 ; i < n_registros ; i++){
                                    strcat(v_escribir,edit3[i]);
                              }

                              SetDlgItemText ( IDC_EDIT3, "");
                              SendDlgItemMessage ( IDC_EDIT3,
EM_REPLACESEL, 0, LPARAM(v_escribir) );

      ////////////*****************************\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\

      ////////////*****************************\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\

                  }



            }

      }

}




void CalenView::OnBnClickedButton1(){



      flags =0;
      ////////////////////////////

      time_t rawtime;  // lo usaremos para obtener la fecha del sistema.
      char* fecha ;
      ofstream f2;      // puntero al fichero de texto usado como base de
datos
      int a=27;
      Bytes *dni;
      unsigned char *d;
      char aux2[11];
      char aux1[11];
      aux1[10]='\0';
      aux2[10]='\0';
      int tags = 0;     // Sera el indicador de estado anterior: 0->sin
tarjeta , 1->con tarjeta
      Tarjeta estudiante1 = Tarjeta();

      a=0;
      dni =NULL;
      a=estudiante1.connect();

      if (a == -1){

                  SetDlgItemText(IDC_EDIT1,LPCTSTR("Error de
establecimiento de contexto."));
```

```
                  CString sReplacementText = "Aborte de inmediato la
aplicacion y vuelba a intentarlo mas tarde.";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));

      }else if(a== -2){

                  SetDlgItemText(IDC_EDIT1,LPCTSTR("No se encuentra
lector."));
                  CString sReplacementText = "Conecte un lector al
equipo.";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));

      }else if(a== -3){

                  SetDlgItemText(IDC_EDIT1,LPCTSTR("Error al leer el
registro estatus."));
                  CString sReplacementText = "Estraiga la targeta, espere
unos segundos y reintentelo";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));

      }else if(a==-6){
                  SetDlgItemText(IDC_EDIT1,LPCTSTR("No hay tarjeta."));
                  CString sReplacementText = "Introduzca la tarjeta de
estudiante.";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));

      }else {
                  CString sReplacementText = "Procediendo ala lectura del
dni";
                  ((CMainFrame*)AfxGetMainWnd())-
>StatusBarMessage(sReplacementText.GetBuffer(0));

                  a=estudiante1.readDNI(&dni);
                  d=dni->bytes();
                  a=dni->length();

                  //Imprimir el dni por pantalla
                  SetDlgItemText(IDC_EDIT1,LPCTSTR(d));


      //*********************************************************\\
                  //Obtener la fecha del sistema
                  time_t rawtime;  // lo usaremos para obtener la fecha
del sistema.
                  char* fecha ;
                  time ( &rawtime );
                  fecha = ctime (&rawtime);     // formato de la variable
fecha: Sat May 20 12:13:00 2010
                  SetDlgItemText(IDC_EDIT2,LPCTSTR(fecha));
                  Sleep(4000);
                  char mesc[3];
                  int mesn;
                  char  dia[3];
                  char  ao[5];
                  char hora[9];
                  char  fechad[12];
```

```
char  mes[4];
for(int i =4;i<7 ; i++){
      mes[i-4]=fecha[i];
}
mes[3] ='\0';
if(strcmp(mes,"Jan")==0){
      mesc[0]='1';
      mesc[1]='\0';
}
if(strcmp(mes,"Feb")==0){
      mesc[0]='2';
      mesc[1]='\0';
}
if(strcmp(mes,"Mar")==0){
      mesc[0]='3';
      mesc[1]='\0';
}
if(strcmp(mes,"Apr")==0){
      mesc[0]='4';
      mesc[1]='\0';
}
if(strcmp(mes,"May")==0){
      mesc[0]='5';
      mesc[1]='\0';
}
if(strcmp(mes,"JUN")==0){
      mesc[0]='6';
      mesc[1]='\0';
}
if(strcmp(mes,"Jul")==0){
      mesc[0]='7';
      mesc[1]='\0';
}
if(strcmp(mes,"Agu")==0){
      mesc[0]='8';
      mesc[1]='\0';
}
if(strcmp(mes,"Sep")==0){
      mesc[0]='9';
      mesc[1]='\0';
}
if(strcmp(mes,"Oct")==0){
      mesc[0]='1';
      mesc[1]='0';
      mesc[2]='\0';
}
if(strcmp(mes,"Nov")==0){
      mesc[0]='1';
      mesc[1]='1';
      mesc[2]='\0';
}
if(strcmp(mes,"Dec")==0){
      mesc[0]='1';
      mesc[1]='2';
      mesc[2]='\0';
}
for(int i =8; i<10 ; i++){
      dia[i-8]=fecha[i];
}
dia[2] ='\0';
```

```
                  for(int i =20;i<24 ; i++){
                        ao[i-20]=fecha[i];

                  }
                  ao[4] ='\0';

                  for(int i =11; i<19 ; i++){        // Dar formato a la
hora
                        hora[i-11]=fecha[i];
            }
                  hora[8]='\0';
                  fechad[0]= '\0';
                  strcat(fechad,dia);
                  strcat(fechad,"/");
                  strcat(fechad,mesc);
                  strcat(fechad,"/");
                  strcat(fechad,ao);
                  // Tengo la fecha en formato dd/mm/aaaa en "fechad"
                  // Tengo la hora en formato hh:mm:ss en "hora"

      //*****************************************************************
***\\

      //*****************************************************************
***\\


                  //Abrir la base d datos y trabajar con ella//
                  ////////////////////////////////////////////
                  CDaoDatabase *pDao = new CDaoDatabase;
                  //    pDao->Open
("C:\Users\josegarcia\Desktop\Bases2003\Alumnos.mdb");
                  pDao->Open ("alumnos.mdb");
                  COleVariant v;
                  CDaoRecordset r(pDao);
                  char iip[42];
                  CString sd ="SELECT * FROM Alumnos WHERE dni =";
      //33 caracter
                  for(int l =0; l<33 ; l++){
                        iip[l]=sd[l];
                  }
                  iip[33]='\0';

                  SetDlgItemText(IDC_EDIT1,LPCTSTR(iip));
                  char opl[9];
                  opl[0]=d[2];
                  opl[1]=d[3];
                  opl[2]=d[4];
                  opl[3]=d[5];
                  opl[4]=d[6];
                  opl[5]=d[7];
                  opl[6]=d[8];
                  opl[7]=d[9];
                  opl[8]='\0';
                  strcat( iip,opl);
                  SetDlgItemText(IDC_EDIT1,LPCTSTR(iip));

                  try{
                        //SELECT * FROM alumnos WHERE dni = "
```

```
                        r.Open(dbOpenDynaset,LPCTSTR(iip) ,dbReadOnly );
                        //r.MoveFirst ();
                        //while(!r.IsEOF ()){
                        r.GetFieldValue ("Nombre", v);
                        //(v) puede tener los siguientes tipos
                        //bstrVal -> BSTR (una cadena).
                        //bVal -> unsigned char
                        //iVal -> short
                        //lVal = long
                        //fltVal -> float
                        //dblVal -> double
                        //por ejemplo: v.bstrVal;
                        SetDlgItemText(IDC_EDIT2,LPCTSTR(v.bstrVal));
                        r.Close();


                  }catch(CDaoException* e ) {
                        long number =e->m_pErrorInfo->m_lErrorCode;
                        std::ostringstream stm ;
                        stm << number ;
                        const char* cstr = stm.str().c_str() ;
                        CString tt = cstr;

                        if(number == 3021){
                              SetDlgItemText(IDC_EDIT2,LPCTSTR("Nombre no
registrado"));
                        }else{
                              SetDlgItemText(IDC_EDIT2,LPCTSTR("Algun
problema"));
                              //SetDlgItemText(IDC_EDIT1,LPCTSTR(cstr));
                              //AfxMessageBox(cstr);
                              //SetDlgItemText(IDC_EDIT2,LPCTSTR(e-
>m_pErrorInfo->m_lErrorCode));
                              // eSTO ES LO QUE TENIAMOS ANTES
                              AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                        }
                  e->Delete();
                  }

                  try{
                        r.Open(dbOpenDynaset,"SELECT * FROM Picadas
",dbAppendOnly);
                        r.AddNew();
                        r.SetFieldValue ("DNI", LPCTSTR(opl)); //->
variable contiene el //dato a insertar
                        r.SetFieldValue("Fecha",LPCTSTR(fechad));
                        r.SetFieldValue("Hora",LPCTSTR(hora));
                        r.Update ();
                        //r.SetBookmark (r.GetLastModifiedBookmark());
                  }catch(CDaoException* e){
                        AfxMessageBox(e->m_pErrorInfo->m_strDescription);
                        e->Delete();
                  }


                        //Para cerrar la conexion y liberar memoria:
                  pDao->Close();
                  delete pDao;
      ////////////////7 Trabajo con la base de datos terminado
```

```
	}
}

void CalenView::OnBnClickedButton2(){


	//*************************************************************\\
			//Obtener la fecha del sistema
			time_t rawtime;  // lo usaremos para obtener la fecha
del sistema.
			char* fecha ;
			time ( &rawtime );
			fecha = ctime (&rawtime);     // formato de la variable
fecha: Sat May 20 12:13:00 2010
			char mesc[3];
			int mesn;
			char  dia[3];
			char  ao[5];
			char hora[9];
			char  fechad[12];
			char  mes[4];
			for(int i =4;i<7 ; i++){
				mes[i-4]=fecha[i];
			}
			mes[3] ='\0';
			if(strcmp(mes,"Jan")==0){
				mesc[0]='1';
				mesc[1]='\0';
			}
			if(strcmp(mes,"Feb")==0){
				mesc[0]='2';
				mesc[1]='\0';
			}
			if(strcmp(mes,"Mar")==0){
				mesc[0]='3';
				mesc[1]='\0';
			}
			if(strcmp(mes,"Apr")==0){
				mesc[0]='4';
				mesc[1]='\0';
			}
			if(strcmp(mes,"May")==0){
				mesc[0]='5';
				mesc[1]='\0';
			}
			if(strcmp(mes,"JUN")==0){
				mesc[0]='6';
				mesc[1]='\0';
			}
			if(strcmp(mes,"Jul")==0){
				mesc[0]='7';
				mesc[1]='\0';
			}
			if(strcmp(mes,"Agu")==0){
				mesc[0]='8';
				mesc[1]='\0';
```

```
                  }
                  if(strcmp(mes,"Sep")==0){
                        mesc[0]='9';
                        mesc[1]='\0';
                  }
                  if(strcmp(mes,"Oct")==0){
                        mesc[0]='1';
                        mesc[1]='0';
                        mesc[2]='\0';
                  }
                  if(strcmp(mes,"Nov")==0){
                        mesc[0]='1';
                        mesc[1]='1';
                        mesc[2]='\0';
                  }
                  if(strcmp(mes,"Dic")==0){
                        mesc[0]='1';
                        mesc[1]='2';
                        mesc[2]='\0';
                  }
                  for(int i =8; i<10 ; i++){
                        dia[i-8]=fecha[i];
                  }
                  dia[2] ='\0';

                  for(int i =20;i<24 ; i++){
                        ao[i-20]=fecha[i];

                  }
                  ao[4] ='\0';

                  for(int i =11; i<19 ; i++){        // Dar formato a la
hora
                        hora[i-11]=fecha[i];
            }
                  hora[8]='\0';
                  fechad[0]= '\0';
                  strcat(fechad,dia);
                  strcat(fechad,"/");
                  strcat(fechad,mesc);
                  strcat(fechad,"/");
                  strcat(fechad,ao);
                  // Tengo la fecha en formato dd/mm/aaaa en "fechad"

      //Abrir la base d datos y trabajar con ella//
                  ////////////////////////////////////////////
                  CDaoDatabase *pDao = new CDaoDatabase;
                  //    pDao->Open
("C:\Users\josegarcia\Desktop\Bases2003\Alumnos.mdb");
                  pDao->Open ("Alumnos.mdb");
                  COleVariant v;
                  CDaoRecordset r(pDao);
                  char iip[42];
                  CString sd ="SELECT * FROM Alumnos WHERE dni =";
      //33 caracter
                  for(int l =0; l<33 ; l++){
                        iip[l]=sd[l];
                  }
                  iip[33]='\0';
```

```
                  strcat( iip,"47777777");

                  try{
                        //SELECT * FROM alumnos WHERE dni = "
                        r.Open(dbOpenDynaset,LPCTSTR(iip) ,dbReadOnly );
                        //r.MoveFirst ();
                        //while(!r.IsEOF ()){
                        r.GetFieldValue ("Nombre", v);
                        //(v) puede tener los siguientes tipos
                        //bstrVal -> BSTR (una cadena).
                        //bVal -> unsigned char
                        //iVal -> short
                        //lVal = long
                        //fltVal -> float
                        //dblVal -> double
                        //por ejemplo: v.bstrVal;
                        SetDlgItemText(IDC_EDIT2,LPCTSTR(v.bstrVal));
                        r.Close();
                        //r.MoveNext();
                  }catch(CDaoException* e ) {
                        long number =e->m_pErrorInfo->m_lErrorCode;
                        std::ostringstream stm ;
                        stm << number ;
                        const char* cstr = stm.str().c_str() ;
                        CString tt = cstr;

                        if(number == 3021){
                              SetDlgItemText(IDC_EDIT1,LPCTSTR("Nombre no
registrado"));
                              r.Close();
                        }else{
                              SetDlgItemText(IDC_EDIT2,LPCTSTR("Algun
problema en la base de datos "));
                              AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                        }
                  e->Delete();
                  }

                  try{
                        r.Open(dbOpenDynaset,"SELECT * FROM Picadas
",dbAppendOnly);
                        r.AddNew();
                        r.SetFieldValue ("DNI", LPCTSTR("47777777")); //-
> variable contiene el //dato a insertar
                        r.SetFieldValue("Fecha",LPCTSTR(fechad));
                        r.SetFieldValue("Hora",LPCTSTR(hora));
                        r.Update ();
                        //r.SetBookmark (r.GetLastModifiedBookmark());
                  }catch(CDaoException* e){
                        SetDlgItemText(IDC_EDIT2,LPCTSTR("Salta excepcion
en el try de picadas"));
                        AfxMessageBox(e->m_pErrorInfo->m_strDescription);
                        e->Delete();
                  }

                  //Para cerrar la conexion y liberar memoria:
                  pDao->Close();
                  delete pDao;
      ////////////////7 Trabajo con la base de datos terminado
```

```
}

void CalenView::OnEnChangeEdit1()
{
      // TODO:  If this is a RICHEDIT control, the control will not
      // send this notification unless you override the
CFormView::OnInitDialog()
      // function and call CRichEditCtrl().SetEventMask()
      // with the ENM_CHANGE flag ORed into the mask.

      // TODO:  Add your control notification handler code here
}
void CalenView::OnEnChangeEdit2()
{
      // TODO:  If this is a RICHEDIT control, the control will not
      // send this notification unless you override the
CFormView::OnInitDialog()
      // function and call CRichEditCtrl().SetEventMask()
      // with the ENM_CHANGE flag ORed into the mask.

      // TODO:  Add your control notification handler code here
}
// Final de la clase CalenView.cpp
```

alenDoc.cpp: implementación de la clase CalenDoc

```
#include "stdafx.h"
#include "alen.h"

#include "alenDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CalenDoc

IMPLEMENT_DYNCREATE(CalenDoc, CDocument)

BEGIN_MESSAGE_MAP(CalenDoc, CDocument)
END_MESSAGE_MAP()


// Construcción o destrucción de CalenDoc

CalenDoc::CalenDoc()
{
      // TODO: agregar aquí el código de construcción único

}
```

```
CalenDoc::~CalenDoc()
{
}

BOOL CalenDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;

	// TODO: agregar aquí código de reinicio
	// (los documentos SDI volverán a utilizar este documento)

	return TRUE;
}



// Serialización de CalenDoc

void CalenDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{
		// TODO: agregar aquí el código de almacenamiento
	}
	else
	{
		// TODO: agregar aquí el código de carga
	}
}


// Diagnósticos de CalenDoc

#ifdef _DEBUG
void CalenDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CalenDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG


// Comandos de CalenDoc
// Fin de la clase CalenDoc
```

Bytes.cpp: Implementación de la clase Bytes, esta clase es usada para traducir a formato legible las respuestas a las peticiones a la tarjeta.

```
#include "stdafx.h"

#include "Bytes.h"

// Class Byte
Bytes::Bytes(){
	strBytes=NULL;
	len=0;
}

Bytes::Bytes(unsigned char *strBytes, int length, int cp){
	if (cp==0) {
		this->strBytes=strBytes;
		this->len=length;
		return;
	}
	//else
	this->strBytes=new unsigned char [length];
	memcpy(this->strBytes,strBytes,length);
	this->len=length;
}

Bytes::Bytes(const Bytes &right){
	strBytes=new unsigned char[right.len];
	len=right.len;
	memcpy(strBytes,right.strBytes,right.len);
}

Bytes& Bytes::operator= (const Bytes &right){
	if (strBytes!=NULL) {
		delete [] strBytes;
	}
	strBytes=new unsigned char[right.len];
	len=right.len;
	memcpy(strBytes,right.strBytes,right.len);
	return (*this);
}

void Bytes::setBytes(unsigned char *strBytes, int length){
	this->len = length;
	this->strBytes = strBytes;
}


Bytes::~Bytes()
{
	if (len!=0)
		delete [] strBytes;
	return;
}
```

MainFrm.cpp: implementación de la clase CMainFrame que está derivada de CFrameWnd, controla todas las características del marco de una aplicación SDI (Single Document Interface)

```
// MainFrm.cpp: implementación de la clase CMainFrame
//

#include "stdafx.h"
#include "alen.h"

#include "MainFrm.h"
#include <afxpriv.h>
#include <afxpriv.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CMainFrame


IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
            //{{AFX_MSG_MAP(CMainFrame)

            ON_MESSAGE(WM_SETMESSAGESTRING, OnSetMessageString)

            //}}AFX_MSG_MAP
            ON_WM_CREATE()
      END_MESSAGE_MAP()

/*
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
      ON_WM_CREATE()
END_MESSAGE_MAP()
*/
CString m_sStatusBarString;  //para cambiar los mensajes de la barra de
estado de debajo

//**************************************************************************
**
// OnSetMessageString
//**************************************************************************
**
LRESULT CMainFrame::OnSetMessageString(WPARAM wParam, LPARAM lParam)
{
      UINT nIDLast = m_nIDLastMessage;
      m_nFlags &= ~WF_NOPOPMSG;

      CWnd* pMessageBar = GetMessageBar();
      if (pMessageBar != NULL)
      {
            CString sMsg;
```

```
            CString strMessage;

            // set the message bar text
            if (lParam != 0)
            {
                  ASSERT(wParam == 0);    // can't have both an ID and a
string
                              m_sStatusBarString = (LPCTSTR)lParam;
                              sMsg = m_sStatusBarString;
            }
            else if (wParam != 0)
            {
                  // map SC_CLOSE to PREVIEW_CLOSE when in print preview
mode
                  if (wParam == AFX_IDS_SCCLOSE && m_lpfnCloseProc !=
NULL)
                        wParam = AFX_IDS_PREVIEW_CLOSE;

                  // get message associated with the ID indicated by
wParam
                              if (wParam == AFX_IDS_IDLEMESSAGE)
                                    sMsg = m_sStatusBarString;
else
{
      GetMessageString(wParam, strMessage);
      sMsg = strMessage;
}
            }

            pMessageBar->SetWindowText(sMsg);

            // update owner of the bar in terms of last message selected
            CFrameWnd* pFrameWnd = pMessageBar->GetParentFrame();
            if (pFrameWnd != NULL)
            {
                  m_nIDLastMessage = (UINT)wParam;
                  m_nIDTracking = (UINT)wParam;
            }
      }

      m_nIDLastMessage = (UINT)wParam;          // new ID (or 0)
      m_nIDTracking = (UINT)wParam;             // so F1 on toolbar
buttons work

      return nIDLast;
}



//********************************************************************
**
// StatusBarMessage
//********************************************************************
**
void CMainFrame::StatusBarMessage(TCHAR * fmt)
{
      TCHAR buffer[256]= _T("");

      CStatusBar* pStatus = (CStatusBar*)
            GetDescendantWindow(AFX_IDW_STATUS_BAR);
```

```
	va_list argptr;
	va_start(argptr, fmt);
	_vstprintf(buffer, fmt, argptr);
	va_end(argptr);

	m_sStatusBarString = buffer;

	SetMessageText((LPCTSTR)m_sStatusBarString);
	return;
}



static UINT indicators[] =
{
	ID_SEPARATOR,           // Indicador de línea de estado
	ID_INDICATOR_CAPS,
	ID_INDICATOR_NUM,
	//ID_INDICATOR_SCRL,
};


// Construcción o destrucción de CMainFrame

CMainFrame::CMainFrame()
{

	// TODO: agregar aquí el código de inicialización adicional de
miembros
}

CMainFrame::~CMainFrame()
{
}


int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;
/*
	if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD |
WS_VISIBLE | CBRS_TOP
		| CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY |
CBRS_SIZE_DYNAMIC) ||
		!m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
	{
		TRACE0("No se pudo crear la barra de herramientas\n");
		return -1;      // No se pudo crear
	}
*/
	if (!m_wndStatusBar.Create(this) ||
		!m_wndStatusBar.SetIndicators(indicators,
		  sizeof(indicators)/sizeof(UINT)))
	{
		TRACE0("No se pudo crear la barra de estado\n");
		return -1;      // No se pudo crear
	}
```

```
	// TODO: eliminar estas tres líneas si no desea que la barra de
herramientas se pueda acoplar
	//m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
	//EnableDocking(CBRS_ALIGN_ANY);
	//DockControlBar(&m_wndToolBar);

	return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
	if( !CFrameWnd::PreCreateWindow(cs) )
		return FALSE;
	cs.style &= ~FWS_ADDTOTITLE;
	// TODO: modificar aquí la clase Window o los estilos cambiando
	//  CREATESTRUCT cs

	return TRUE;
}


// Diagnósticos de CMainFrame

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
	CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
	CFrameWnd::Dump(dc);
}

#endif //_DEBUG


// Controladores de mensaje de CMainFrame
// Final de la clase CMainframe
```

stdafx.cpp: archivo de código fuente que contiene solo las inclusiones estándar

```
// stdafx.cpp: archivo de código fuente que contiene solo las inclusiones
estándar
// alen.pch será el encabezado precompilado
// stdafx.obj contendrá la información de tipos precompilada

#include "stdafx.h"
```

Tarjeta.cpp: implementación de la clase Tarjeta.

```
#include "stdafx.h"

#include "tarjeta.h"
#include <string.h>
#include <stdlib.h>
#include <direct.h>
#include <stdio.h>
#include <conio.h>

// APDU Select MF
#define APDU_SELECT_MF {0x00, 0xa4, 0x04, 0x00, 0x04, 0x52, 0x4f, 0x4f,
0x54}

// APDU Select DF WG10
#define APDU_SELECT_WG10 {0x00, 0xa4, 0x04, 0x00, 0x07, 0x50, 0x20, 0x00,
0xff, 0x00, 0x01, 0x01}

// APDU Select DF de la aplicación de la tarjeta.
//#ifdef UMU_RELEASE
#define APDU_SELECT_DF_RELEASE {0x00, 0xa4, 0x04, 0x00, 0x07, 0x50, 0x20,
0x00, 0xff, 0x00, 0x04, 0x01}


// APDU Obtención de respuesta, l es el número de bytes a leer.
/* Por limitaciones del protocolo T=0, la tarjeta no puede recibir y
enviar datos en el mismo
   comando. Por lo tanto, este comando se debe de ejecutar tras otro
comando de tipo 4, recibe
   los datos entrantes,procesa el comando y prepara la respuesta. El S.O.
devuelve un código
   "61 XXh", donde xx es la longitud que se debe especificar en el campo
L2 de APDU_GET_RESPONSE
*/
#define APDU_GET_RESPONSE(l) {0x00, 0xc0, 0x00, 0x00, l};

/* APDU Lectura binaria, permite leer el contenido de un fichero
transparente (binario), el cual
   tiene un tamaño de unidad de datos variable (la unidad de escritura
son los bits), se especifica
   la longitud del fichero en bloques de 256 bits (no se que es el & con
0xff), y el offset, por
   último se indica la longitud de los datos que se quieren leer.*/
#define APDU_READ_BINARY(len,offset) { 0x00, 0xb0, ((offset / 0x100) &
(~0x80)), (offset & 0xff), len}

//APDU Select Serial
#define APDU_SELECT_SERIAL {0x00, 0xA4, 0x02, 0x00, 0x02, 0x2F, 0x02};
//APDU Read Binary Saldo
#define APDU_READ_BINARY_SALDO {0x00, 0xB0, 0x00, 0x00, 0x0B};
// APDU Select EF Clave RSA
#define APDU_SELECT_PRIVKEY_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x14}
// APDU Select EF Certificado
#define APDU_SELECT_CERT_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x15}
// APDU Select EF Certificado 2
#define APDU_SELECT_CERT2_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x16}
// APDU Select EF Certificado CA
```

```
#define APDU_SELECT_CA_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x17}
// APDU Select EF Certificado CA2
#define APDU_SELECT_CA2_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x18}
// APDU Verify PIN
//#define APDU_VERIFY_PIN(a) {0x00, 0x20, 0x00, 0x00, 0x08, 0x30, 0x30,
0x30, 0x30, a[0], a[1], a[2], a[3]}
#define APDU_VERIFY_PIN(a) {0x00, 0x20, 0x00, 0x00, 0x08, a[0], a[1],
a[2], a[3], a[4], a[5], a[6], a[7]}
// APDU Select EF DNI
#define APDU_SELECT_DNI_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x11}
// APDU Select EF Caducidad
#define APDU_SELECT_CAD_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x12}
// APDU Select EF NAC
#define APDU_SELECT_NAC_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x13}
// APDU Select EF SEG
#define APDU_SELECT_SEG_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x14}
// APDU Select EF Response
#define APDU_SELECT_Response_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00,
0x21}
// APDU Select EF KEY
#define APDU_SELECT_KEY_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x22}
// APDU Select EF NT
#define APDU_SELECT_NT_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x23}
// APDU Select EF PIN
#define APDU_SELECT_PIN_EF {0x00, 0xa4, 0x02, 0x00, 0x02, 0x00, 0x24}
//APDU_STANDAR_UPDATE_BINARY_HEADER
#define APDU_STANDARD_UPDATE_BINARY_HEADER(len,offset) { 0x00, 0xD6,
((offset / 0x100) & (~0x80)), (offset & 0xff), len}

#define SC4K_TAM_BLOQUE_READ  0x80  // tamaño del bloque de lectura
#define TAM_BLOQUE_WRITE            0x20
#define TAM_PRIVKEY                       1016  // tamaño de la clave
#define TAM_CER                     1016        // tamaño del certificado
(0x3B8)
#define TAM_CA                      1016        // tamaño del certificado
CA (0x3C8)
#define TAM_CER_DEL                 10
#define TAM_CER_CA_DEL        10
#define TAM_PK_DEL                  100

#define SCARD_E_SECURITY            0x6982
// --- Valores de la función scard_verifyPINDlgProc
#define SC4K_PIN_VERIFIED           0x0         // pin verificado
#define SC4K_PIN_CLOSED             0x1         // pin bloqueado
#define SC4K_PIN_LONG_ERROR         0x2         // especificado un pin de
longitud errónea
#define SC4K_PIN_ERROR              0x3         // error en la
verificación del pin
#define SC4K_PIN_NO_VERIFIED  0x4         // pin incorrecto
#define TAM_PIN                           8           // tamaño del pin
(reales 8 pero útiles 4)

static const char rnd_seed[] = "año 2001 Facultad de Informatica de la
Universidad de Murcia.FX";


//LRESULT CALLBACK verifyPINDlgProc(HWND hDlg, UINT message, WPARAM
wParam, LPARAM lParam);
long send_APDU(long hsCard, Bytes *apdu, unsigned char **status);
HINSTANCE hinst;
```

```
int longCer,longCA;


/*************************************************************/
Tarjeta::Tarjeta() {
            hContext=NULL;
            hsCard=0;
}
/*************************************************************/
Tarjeta::~Tarjeta() {
      if (hContext)
            SCardReleaseContext(hContext);
      if (hsCard)
            SCardDisconnect(hsCard,SCARD_LEAVE_CARD);
}
/*************************************************************************/
// sendAPDU-->utiliza la librería winscard para mandar el apdu a la
tarjeta,
// que usa el SO WG10, el formato de la apdu ha de ser IEEE-7816
long Tarjeta::sendAPDU(Bytes *apdu, unsigned char **status){

      long lReturn;     // código de respuesta PC/SC
      *status=new unsigned char[2];
    unsigned long dwStatusLength = sizeof(status);
    unsigned long lpdwExtraBytes = 0;

    // --- Envío el APDU a la tarjeta
    lReturn = SCardTransmit(hsCard, SCARD_PCI_T0, apdu->bytes(), apdu-
>length(), NULL, *status, &dwStatusLength);

    if (lReturn == SCARD_S_SUCCESS){
            if (!( ((*status)[0]) == 0x90 && ((*status)[1])== 0x00)){
                  if ( ((*status)[0])== 0x61){
                lpdwExtraBytes = (unsigned int) (*status)[1];
                        //printf("Se pueden leer estos bytes de la
tarjeta :%d\n",(*status)[1]);
                        //en status[1] están los bytes que se pueden leer
tras el envío de la APDU, con getResponse
                  }else {
                        // --- Retorno SW1 y SW2 como código de error
                lReturn = MAKELONG(MAKEWORD(((*status)[1]),
((*status)[0])), 0x0000);
                        //printf("LReturn:%ld\n",lReturn);
                        }
        }
    }
      //LogFilter("status [0]=%2X, [1]=%2X\n",(*status)[0],(*status)[1]);
    return lReturn;
}


/*************************************************************************
************/
long Tarjeta::getResponse(IN unsigned long dwLength, OUT unsigned char
*lplpbResponse){
    long lResult;
      unsigned char *lpbResponse = NULL;

      // --- Compruebo la longitud de datos a leer
```

```
    if (dwLength > 256)
        return SCARD_E_INVALID_PARAMETER;

      // --- Reservo memoria para los datos a leer + 2 por el codigo del
estado de respuesta
    if ((lpbResponse = (unsigned char *) LocalAlloc(LMEM_ZEROINIT,
dwLength + 2)) == NULL)
        return ERROR_OUTOFMEMORY;

    // --- Construyo el APDU GetResponse
    unsigned long dwStatusLen = dwLength + 2;
    unsigned char apdu[] = APDU_GET_RESPONSE((unsigned char) dwLength);

    // --- Envío el APDU a la tarjeta
    lResult = SCardTransmit(hsCard, SCARD_PCI_T0, apdu, sizeof(apdu),
NULL, lpbResponse, &dwStatusLen);

    if (lResult == SCARD_S_SUCCESS){
            // --- Compruebo que se ha leído lo deseado
        if (dwStatusLen == dwLength + 2){
            if (lpbResponse[dwLength] == 0x90 && lpbResponse[dwLength +
1] == 0x00)
                        memcpy(lplpbResponse,lpbResponse,dwLength);

            else {
                // --- Retorno SW1 y SW2 como código de error
                lResult = MAKELONG(MAKEWORD(lpbResponse[dwLength + 1],
lpbResponse[dwLength]), 0x000);
                        //printf("Error en getResponse: %02x, %02x\n",
lpbResponse[dwLength + 1], lpbResponse[dwLength]);
                 }
        }else {
            // --- Retorno SW1 y SW2 como código de error
            lResult = MAKELONG(MAKEWORD(lpbResponse[1], lpbResponse[0]),
0x000);
//                printf("Error en getResponse codigos hex: %02x,
%02x\n", lpbResponse[1], lpbResponse[2]);
            }
    }else {
//          printf("GetResponse-Error en transmit\n");
    }


      // --- Libero memoria
      if (lpbResponse)
            LocalFree(lpbResponse);

    return lResult;
}




// --- scard_getRead --------
//
//    Lee EF binario, delimitado por un desplazamiento de bytes dado a
partir del comienzo del DF
//  en el que estemos.
```

```
//getRead-->
//                1-Tamaño de los datos a leer.
//                2-Desplazamiento dentro del df a partir de donde se
lee.
//                3-Lugar donde se ponen los datos leidos.
long Tarjeta::getRead (IN unsigned char dwLength, IN unsigned short
offset, OUT unsigned char * lplpbResponse){
    long lResult;
      unsigned char *lpbResponse = NULL;

      // --- Compruebo la longitud de datos a leer
    if (dwLength > 256)
        return SCARD_E_INVALID_PARAMETER;

      // --- Reservo memoria para los datos a leer
    if ((lpbResponse = (unsigned char *) LocalAlloc(LMEM_ZEROINIT,
dwLength + 2)) == NULL)
        return ERROR_OUTOFMEMORY;

    // --- Construyo APDU ReadBinary
    unsigned long dwStatusLen = dwLength + 2;
    unsigned char apdu[] = APDU_READ_BINARY((unsigned char)
dwLength,offset);

    // --- Envío el APDU a la tarjeta
    lResult = SCardTransmit(hsCard, SCARD_PCI_T0, apdu, sizeof(apdu),
NULL, lpbResponse, &dwStatusLen);

    if (lResult == SCARD_S_SUCCESS){
            // --- Compruebo que se ha leído lo deseado
        if ((long) dwStatusLen == dwLength + 2){
            if (lpbResponse[dwLength] == 0x90 && lpbResponse[dwLength +
1] == 0x00)
                        memcpy(lplpbResponse,lpbResponse,dwLength);
            else
                        // --- Retorno SW1 y SW2 como código de error
                lResult = MAKELONG(MAKEWORD(lpbResponse[dwLength +
1],lpbResponse[dwLength]), 0x000);
        }else
                // --- Retorno SW1 y SW2 como código de error
            lResult = MAKELONG(MAKEWORD(lpbResponse[1], lpbResponse[0]),
0x000);
    }

      // --- Libero memoria
      if(lpbResponse)
            LocalFree(lpbResponse);

    return lResult;
}


int compara(IN unsigned char selectAPDU[],IN unsigned long lSelectAPDU,
unsigned char apdutocompare[]){
      int i;

      for (i = 0 ; i < (int)lSelectAPDU; i++){
            if(apdutocompare == NULL) return false;
      //    printf("compara <%x> con
<%x>\n",selectAPDU[i],apdutocompare[i]);
```

```
            if(selectAPDU[i] != apdutocompare[i]) return false;
      }

      return true;
}




// --- scard_getField --------
//
//    Lectura del contenido de un EF Transparente.
//
//  Lo uso
int Tarjeta::getField(IN unsigned char selectAPDU[], IN unsigned long
lSelectAPDU, OUT unsigned char* content, OUT unsigned long* lContent)
{
      unsigned long dwExtraBytes = 0;
      unsigned char *lpbResult = NULL;
      unsigned long lReturn;
      unsigned char *status=NULL;
      unsigned char *contaux = NULL;
      SCardBeginTransaction(hsCard);           //establece el modo de
acceso exclusivo

      // --- Envio APDU de selección del EF
      //LogFilter("entra a getField()\n");
      Bytes *command=new Bytes(selectAPDU,lSelectAPDU);
      //LogFilter("la longitud del apdu es %d\n", lSelectAPDU);
      if ((lReturn = sendAPDU(command, &status)) != SCARD_S_SUCCESS){

            /*LogFilter("comando:\n");
            for(int i = 0; i < command->length(); i++)
                  LogFilter("[%d]=<%x>",i,command->bytes()[i]);
            LogFilter("\n");*/

            delete(command);
            return lReturn;
      }

      delete(command);
      dwExtraBytes=(unsigned int) status[1];

      // --- Obtengo la información del EF
      if((lpbResult = (unsigned char *)LocalAlloc(LMEM_ZEROINIT,
dwExtraBytes)) == NULL){
            return ERROR_OUTOFMEMORY;
      }
      if ((lReturn = getResponse(dwExtraBytes, lpbResult)) ==
SCARD_S_SUCCESS) {
            // --- Obtengo la longitud de los datos del EF
            unsigned char apduaux[] = APDU_SELECT_DNI_EF;
            unsigned char apduaux1[] = APDU_SELECT_NAC_EF;
            unsigned char apduaux2[] = APDU_SELECT_CAD_EF;
            *lContent = 3;
            if(compara(selectAPDU,lSelectAPDU,apduaux)){
                  *lContent = 10;
                  contaux = content;
```

```
                  //LogFilter("Es DNI, lContent: %d\n", *lContent);
                  //printf("Es dni\n");

            }
            else if(compara(selectAPDU,lSelectAPDU,apduaux1)){
                  *lContent = 5;
                  contaux = content;
                  //printf("Es nac, lReturn:%d\n", lReturn);
            }
            else if(compara(selectAPDU,lSelectAPDU,apduaux2)){
                  *lContent = 7;
                  contaux = content;
                  //LogFilter("Es cad, lContent: %d\n", *lContent);
            }
            else{
                  //reservamos espacio para los 3 primeros bytes (Tipo y
LONGITUD)
                  if((contaux = (unsigned char
*)LocalAlloc(LMEM_ZEROINIT, 3)) == NULL){
                        //printf("Es otro tipo de dato el que hay que
reservar.\n");
                        LocalFree(lpbResult);
                        return ERROR_OUTOFMEMORY;
                  }
            }

            //leemos primeros 10, 5 o 7 bytes, para leer DNI, NAC o CAD o
bien 3 para leer longitud de EF.
            //printf("3.lReturn:%d\n", lReturn);
            //LogFilter("GetField primera parte: lee %d bytes\n",
*lContent);
            if((lReturn = getRead((unsigned char) *lContent, (unsigned
short) 0, contaux)) != SCARD_S_SUCCESS){
                  LocalFree(lpbResult);
      //          printf("sale al leer %d bytes\n", lContent);
                  return lReturn;
            }
            if(*lContent == 3){

      //**************
/*    clock_t start, finish;

      start = clock();
*/    //**************

                  //si no es CAD, ni NAC, ni DNI seguimos leyendo lo que
corresponda en los bytes 1 y 2.
                  *lContent = 0;
                  bool primerEF = false;

                  unsigned char apduaux3[] = APDU_SELECT_CERT_EF;
                  if(compara(selectAPDU,lSelectAPDU,apduaux3)){
                        primerEF = true;
      //guardamos longitud certificado usuario
                        if(contaux[0] == 0x02) longCer =
MAKEWORD(contaux[1], contaux[2])+3;
//                      printf("Es cert\n");
                  }

                  unsigned char apduaux4[] = APDU_SELECT_CA_EF;
```

```
                  if(compara(selectAPDU,lSelectAPDU,apduaux4)){
                        primerEF = true;
      //guardamos longitud certificado CA
                        if(contaux[0] == 0x02) longCA =
MAKEWORD(contaux[1], contaux[2])+3;
                        //LogFilter("Es ca, longitud del fichero:
%d\n",longCA);
                  }

                  unsigned char apduaux5[] = APDU_SELECT_PRIVKEY_EF;
                  if(compara(selectAPDU,lSelectAPDU,apduaux5)){
                        *lContent = MAKEWORD(contaux[1], contaux[2]);
                  }

                  if(primerEF == false){  //obtenemos la longitud
restante

                        unsigned char apduaux6[] = APDU_SELECT_CERT2_EF;
                        if(compara(selectAPDU,lSelectAPDU,apduaux6)){
                              *lContent = longCer - TAM_CER;
//                            printf("Es cert2\n");
                        }

                        unsigned char apduaux7[] = APDU_SELECT_CA2_EF;
                        if(compara(selectAPDU,lSelectAPDU,apduaux7)){
                              *lContent = longCA - TAM_CA;
                              //LogFilter("Es ca2\n");
                        }
                        //caso CER1 y CA1
                  }else if(*lContent == 0) *lContent =
MAKEWORD(contaux[1], contaux[2])+3;

                  //LogFilter("El valor calculado de lContent es: %d\n",
*lContent);
                  if(*lContent > 1016) *lContent = 1016;    //para cuando
vaya fragmentado

            //ANTES LEIA TODO: *lContent = MAKEWORD(lpbResult[9],
lpbResult[8]);

                  if(content != NULL) {
                        unsigned int count =
(*lContent)/SC4K_TAM_BLOQUE_READ;   // número de bloques
                        unsigned int resto =
(*lContent)%SC4K_TAM_BLOQUE_READ;   // resto
                        unsigned char i = 0;
                        //printf("count: %d\n",count);
                        //printf("resto: %d\n",resto);
                        if(count == 0)
                        // --- No existe ningún bloque, por tanto tampoco
se tienen en cuenta los bytes del SO
                              resto = (*lContent);
                        else
                        {
                        // --- Leo los bloques
                              for(; i < count; i++) {
                                    if((lReturn = getRead
(SC4K_TAM_BLOQUE_READ, (unsigned short) i*SC4K_TAM_BLOQUE_READ, content +
i*SC4K_TAM_BLOQUE_READ)) != SCARD_S_SUCCESS){
                                          LocalFree(lpbResult);
```

```
                                        return lReturn;
                                  }
                            }
                      }

                      if(resto != 0) {
                      // --- Leo el resto
                            if((lReturn = getRead((unsigned char)
resto, (unsigned short) i*SC4K_TAM_BLOQUE_READ, content +
i*SC4K_TAM_BLOQUE_READ)) != SCARD_S_SUCCESS){
                                  LocalFree(lpbResult);
                                  return lReturn;
                            }
                      }

      //**************
/*    finish = clock();
      double duration = (double)(finish - start) / CLOCKS_PER_SEC;
      printf("%2.5f tarda en leer %d bloques del certificado.\n",
duration, count );
*/    //**************

                  }//if(content != NULL)...
//                printf("sale0, contaux: %d\n", contaux);
//                if(contaux == NULL)printf("contaux es null");
                  LocalFree(contaux);

            }//   if(*lContent == 3)...
      }//if ((lReturn = getResponse...

      if (lpbResult)
            LocalFree(lpbResult);

      // --- Libero memoria
      SCardEndTransaction(hsCard, SCARD_LEAVE_CARD );      //finaliza el
modo de acceso exclusivo
      return lReturn;
}

int Tarjeta::disconnect(){
            SCardDisconnect(hsCard,SCARD_LEAVE_CARD);
            if(   SCardDisconnect(hsCard,SCARD_LEAVE_CARD) ==
SCARD_S_SUCCESS )
                  return 1;
            else
                  return -1;
}

int Tarjeta::connect(){
      SCARD_READERSTATE rgscState[1];
      DWORD cReaders = 1;
      //Conexión PC/SC con la tarjeta
      char *pmszReaders = NULL;
      char *lpReader = NULL;
      unsigned long count, lReturn;
      unsigned long cch = SCARD_AUTOALLOCATE;
      bool found = false;
      SCARDHANDLE hCard;
      unsigned long dwExtraBytes = 0;
```

```
	if (hContext!=NULL) {
		if (hContext)
			SCardReleaseContext(hContext);
		if (hsCard)
			SCardDisconnect(hsCard,SCARD_LEAVE_CARD);
	}

	// ---Establecer Contexto
	lReturn = SCardEstablishContext(SCARD_SCOPE_USER, NULL, NULL,
&hContext);
	if ( SCARD_S_SUCCESS != lReturn ) {
//		printf("Failed SCardEstablishContext\n");

		if (hContext)
			SCardReleaseContext(hContext);
		hContext=NULL;
		return -1;
	}else{
//		printf("Success tras SCardEstablishContext.\n ");
	}

	// --- Recupera la lista de lectores.
	if ((lReturn = SCardListReaders(hContext, NULL,
(LPTSTR)&pmszReaders, &cch)) != SCARD_S_SUCCESS) {
//		printf("Error Lista lectores\n");
		if (pmszReaders)
			lReturn = SCardFreeMemory( hContext,pmszReaders );
		if (hContext)
			SCardReleaseContext(hContext);

		hContext=NULL;
		return -2;
	}

	// --- Se sitúa en el primer lector
	lpReader =pmszReaders;

	// --- Comprobamos lector a lector si tiene una tarjeta insertada y
si ésta tiene un DF wg10
	unsigned char *status=NULL;
	unsigned long dwActiveProtocol; // ignorado
	// --- Conectamos con la tarjeta
	if((lReturn = SCardConnect(hContext, lpReader, SCARD_SHARE_SHARED,
			SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1, &(hCard),
&dwActiveProtocol)) == SCARD_S_SUCCESS) {

//		printf("Success tras SCardConnect.\n");
		/* Veamos el estado y su ATR */
		char mszReaderNames[200];
		DWORD cchReaderLen=200;
		DWORD dwState;
		DWORD dwProtocol;
		BYTE ATR[32];
		DWORD bcAtrLen;
		hsCard=(long)hCard;
		//unsigned char *d1;

		lReturn=SCardStatus(hsCard,
mszReaderNames,&cchReaderLen,&dwState,&dwProtocol, ATR,&bcAtrLen);
```

```
            if (lReturn==SCARD_S_SUCCESS){
//                printf("Estado del lector: %s recuperado
correctamente\n",mszReaderNames);



            // Examine retrieved status elements.
            // Look at the reader name and card state.

            switch ( dwState ){
                  case SCARD_ABSENT:
                        //printf("Card absent.\n");
                        break;
                  case SCARD_PRESENT:
                        //printf("Card present.\n");
                        break;
                  case SCARD_SWALLOWED:
                        //printf("Card swallowed.\n");
                        break;
                  case SCARD_POWERED:
                        //printf("Card has power.\n");
                        break;
                  case SCARD_NEGOTIABLE:
                  //    printf("Card reset and waiting PTS
negotiation.\n");
                        break;
                  case SCARD_SPECIFIC:
                  //    printf("Card has specific communication protocols
set.\n");
                        break;
                  default:
                        //printf("Unknown or unexpected card state.\n");
                        break;
            }


            }

            if(lReturn!=SCARD_S_SUCCESS) {
                  //printf("Error al leer status\n");
//                printf("Failed SCardStatus, returned: %d\n", lReturn);
                  if (pmszReaders)

                        lReturn = SCardFreeMemory(hContext, pmszReaders);
                  if (hContext)
                        SCardReleaseContext(hContext);
                  hContext=NULL;
                  return -3;
            }

            hsCard=(long)hCard;
            unsigned char selDF[] = APDU_SELECT_DF_RELEASE;
            Bytes *command=new Bytes(selDF,sizeof(selDF));
            lReturn=sendAPDU(command, &status);

            if (lReturn!= SCARD_S_SUCCESS) {
//                printf("Failure Select UM DF, returned: %d\n",
lReturn);
                  if (status)
```

```
                        delete [] status;
                  if (command)
                        delete command;
                  return -4 ; //Error al seleccionar UM del registro DF
                  //continue;
            } else {
//                printf("Success Select UM DF\n");
//                printf("Esto es status:%s\n",status);
                  if (status){
//                      printf(" hago delete[] status \n");
                        delete [] status;
                  }
                  if (command)
                        delete command;
                  found=true;
                  // --- Devolver handle PC/SC de la tarjeta seleccionada
            }
      }
            // --- Nos situamos en el siguiente lector
            //printf("Cada vez que nos situamos en el siguiente lector
\n");
      lpReader = lpReader + strlen(lpReader) + 1;

//    printf("Liberar memoria despues de send_apdu()\n");
      // --- Liberamos memoria
      lReturn = SCardFreeMemory( hContext, pmszReaders );

      // --- No se ha encontrado ninguna tarjeta
      if ((!found) || (lReturn != SCARD_S_SUCCESS)) {
//          printf("No se ha encontrado ninguna tarjeta, lReturn: %d\n",
lReturn);
            if (hContext)
                  SCardReleaseContext(hContext);
            hContext=NULL;
            return -6;
      }else{
            // --- Devolver handle PC/SC de la tarjeta seleccionada
            hsCard=(long)hCard;
            return (long) hCard;
            //Seleccionar el DF del CSP
//          printf("Seleccionando DF CSP\n");
            unsigned char selDF[]=APDU_SELECT_DF_RELEASE;
            unsigned char *status=NULL;

            Bytes *command=new Bytes(selDF,sizeof(selDF));
            lReturn=sendAPDU(command, &status);
            if (lReturn!= SCARD_S_SUCCESS) {
//                printf("Si se produce error en send apduu \n");
                  if (status)
                        delete [] status;
                  if (command)
                        delete command;
                  if (hContext)
                        SCardReleaseContext(hContext);
                  return -7;
            }
            delete [] status;
            delete command;
//          printf("Antes del return 1\n");
```

```
            return 1;
      }
//    printf("Llegamos a la ultima sentencia del connect() \n");
      return 1;
}




// sendAPDU
long send_APDU(long hsCard, Bytes *apdu, unsigned char **status){
    long lReturn;            // código de respuesta PC/SC
      *status=new unsigned char[2];
    unsigned long dwStatusLength = sizeof(status);

    unsigned long lpdwExtraBytes = 0;

    // --- Envío el APDU a la tarjeta
    lReturn = SCardTransmit(hsCard, SCARD_PCI_T0, apdu->bytes(), apdu-
>length(), NULL, *status, &dwStatusLength);
    if (lReturn == SCARD_S_SUCCESS)
    {
            //printf("succes\n");
        if (!( ((*status)[0]) == 0x90 && ((*status)[1])== 0x00))
        {
           if ( ((*status)[0])== 0x61)
                lpdwExtraBytes = (unsigned int) (*status)[1];
                       //printf("hola:%d\n",lpdwExtraBytes);
                       else {
                       // --- Retorno SW1 y SW2 como código de error
                lReturn = MAKELONG(MAKEWORD(((*status)[1]),
((*status)[0])), 0x0000);
                       //printf("LReturn:%ld\n",lReturn);
                       }
        }
            //printf("status [0]=%2X,
[1]=%2X\n",(*status)[0],(*status)[1]);
    }

    return lReturn;
}




int Tarjeta::readDNI(Bytes **dni) {
      unsigned long lReturn;
      unsigned long dwExtraBytes = 0;
      unsigned char content[11];
      unsigned long lContent = 0;
      unsigned char *status=NULL;

      if (selectDFRelease()!=1)
            return -1;

      //printf("Vamos a pasar a leer el Certificado\n");
```

```
	unsigned char apdu[] = APDU_SELECT_DNI_EF;
	lReturn = getField(apdu, sizeof(apdu), content, &lContent);
	if (lReturn==SCARD_S_SUCCESS) {

		unsigned char *cadu = (BYTE *) LocalAlloc(LMEM_ZEROINIT, 11);
		memcpy(cadu,content, lContent);
		cadu[11]='\0';

		*dni=new Bytes(cadu, lContent);//cambio de lcontent+1 a
lcontent
	} else
		return -1;

	return 1;
}


int Tarjeta::selectEF(IN unsigned char selectAPDU[], IN unsigned long
lSelectAPDU){
	unsigned long dwExtraBytes;
	unsigned char *lpbResult = NULL;
	unsigned long lReturn;
	unsigned char *status=NULL;
	unsigned long lContent=0;

	SCardBeginTransaction(hsCard);

	// --- Envio APDU de selección del EF
	Bytes *command=new Bytes(selectAPDU,lSelectAPDU);
	if ((lReturn = sendAPDU(command, &status)) != SCARD_S_SUCCESS)
		return -1;

	dwExtraBytes=(unsigned int) status[1];
	// --- Obtengo la información del EF
	if((lpbResult = (unsigned char *)LocalAlloc(LMEM_ZEROINIT,
dwExtraBytes)) == NULL)
		//return ERROR_OUTOFMEMORY;
		return -2;
	if ((lReturn = getResponse(dwExtraBytes, lpbResult)) ==
SCARD_S_SUCCESS) {

		// --- Obtengo la longitud de los datos del EF
		//lContent = MAKEWORD(lpbResult[9], lpbResult[8]);
		//printf("La longitud del EF es: %d\n",lContent);
	} else {
		LocalFree(lpbResult);
		return -3;
	}
	SCardEndTransaction(hsCard,SCARD_LEAVE_CARD);
	LocalFree(lpbResult);
	return 1;
}

/*****************************************************/
int Tarjeta::selectDFRelease() {
		unsigned char selDF[]=APDU_SELECT_DF_RELEASE;
		unsigned char *status=NULL;
		unsigned long lReturn;
		unsigned long dwExtraBytes;
```

```
            Bytes *command = new Bytes(selDF, sizeof(selDF));
            lReturn=sendAPDU(command, &status);
            if (lReturn!= SCARD_S_SUCCESS) {
                  //fprintf(stderr,"\nERROR seleccionando DF\n");
                  delete command;
                  delete [] status;
                  return -1;
            } else {
                  if (status==NULL) {
                        delete command;
                        return -1;
                  }
                  if (status[0]==0x90) {
                        delete [] status;
                        delete command;
                        return -2;
                  }else if (status[0]==0x61) {
                        dwExtraBytes=(unsigned int) status[1];
                        //printf("Hay que enviar un Obtener Respuesta.
Bytes extra: %d, %d\n",status[1],dwExtraBytes);
                        // --- Obtengo la información del EF
                        unsigned char *lpbResult=NULL;
                        if ((lpbResult = (unsigned char
*)malloc(sizeof(char)*dwExtraBytes)) == NULL) {
                              //return ERROR_OUTOFMEMORY;
                              return -3;
                        }
                        if ((lReturn = getResponse(dwExtraBytes,
lpbResult)) == SCARD_S_SUCCESS) {
                              //_debug_printToHex
(lpbResult,dwExtraBytes);
                        }
                        else {
//                            printf("Error al obtener el getResponse:
%d\n",lReturn);
                              delete command;
                              return -4;
                        }
                  }
                  //printf("Conexion correcta\n");
                  delete [] status;
                  delete command;
            }

            return 1;
}

// Final de la clase Tarjeta
```

alen.h: archivo de encabezado principal para la aplicación alen

```
//
#pragma once
```

```
#ifndef __AFXWIN_H__
      #error incluye 'stdafx.h' antes de incluir este archivo para PCH
#endif

#include "resource.h"       // Símbolos principales


// CalenApp:
// Consulte la sección alen.cpp para obtener información sobre la
implementación de esta clase
//

class CalenApp : public CWinApp
{
public:
      CalenApp();


// Reemplazos
public:
      virtual BOOL InitInstance();

// Implementación
      afx_msg void OnAppAbout();
      DECLARE_MESSAGE_MAP()
};

extern CalenApp theApp;
```

alenDoc.h: interfaz de la clase CalenDoc.

```
// alenDoc.h: interfaz de la clase CalenDoc
//


#pragma once

class CalenDoc : public CDocument
{
protected: // Crear sólo a partir de serialización
      CalenDoc();
      DECLARE_DYNCREATE(CalenDoc)

// Atributos
public:

// Operaciones
public:

// Reemplazos
      public:
      virtual BOOL OnNewDocument();
      virtual void Serialize(CArchive& ar);

// Implementación
public:
```

```
      virtual ~CalenDoc();
#ifdef _DEBUG
      virtual void AssertValid() const;
      virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Funciones de asignación de mensajes generadas
protected:
      DECLARE_MESSAGE_MAP()
};
```

alenView.h: interfaz de la clase CalenView.

```
#pragma once


class CalenView : public CFormView
{
protected: // Crear sólo a partir de serialización
      CalenView();
      DECLARE_DYNCREATE(CalenView)

public:
      enum{ IDD = IDD_ALEN_FORM };

// Atributos
public:
      int flags;
      int ctiempovisible;
      CalenDoc* GetDocument() const;
      int n_registros ;
      char edit3[20][70];
// Operaciones
public:

// Reemplazos
      public:
virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:
      virtual void DoDataExchange(CDataExchange* pDX);    //
Compatibilidad con DDX/DDV
      virtual void OnInitialUpdate(); // Se llama la primera vez después
de la construcción

// Implementación
public:
      virtual ~CalenView();
#ifdef _DEBUG
      virtual void AssertValid() const;
      virtual void Dump(CDumpContext& dc) const;
#endif

protected:
```

```
// Funciones de asignación de mensajes generadas
protected:
	DECLARE_MESSAGE_MAP()
public:
	afx_msg void OnBnClickedButton1();
	afx_msg void OnBnClickedButton2();
	afx_msg void OnTimer(UINT nIDEvent);
	afx_msg void OnEnChangeEdit1();
	afx_msg void OnEnChangeEdit2();
};

#ifndef _DEBUG  // Versión de depuración en alenView.cpp
inline CalenDoc* CalenView::GetDocument() const
   { return reinterpret_cast<CalenDoc*>(m_pDocument); }
#endif
```

Bytes.h: fichero de cabecera de la clase Bytes.

```
#ifndef _Bytes_H_
#define _Bytes_H_


#ifdef __cplusplus
extern "C" {
#endif

	#include <stdio.h>
	#include <string.h>
	#include <stdlib.h>

	class Bytes {
		public:
		  Bytes();
		  Bytes(const Bytes &right);
		  Bytes(unsigned char *strBytes, int length, int cp=1);
		  Bytes& operator= (const Bytes&);
		  ~Bytes();
		  void setBytes(unsigned char *strBytes, int length);
		  unsigned char *bytes();
		  int length();

	  public:
		  unsigned char *strBytes;
		  int len;
	};

	// Class Byte


	//## Get and Set Operations for Class Attributes (inline)
	inline unsigned char *Bytes::bytes()
	{
		if(strBytes!=NULL)
			return strBytes;
		else return NULL;
	}
```

```
      inline int Bytes::length()
      {
        return len;
      }

#ifdef __cplusplus
}
#endif

#endif
```

MainFrm.h: interfaz de la clase CMainFrame.

```
// MainFrm.h: interfaz de la clase CMainFrame
//


#pragma once
class CMainFrame : public CFrameWnd
{

protected: // Crear sólo a partir de serialización
      CMainFrame();
      DECLARE_DYNCREATE(CMainFrame)

// Atributos
public:

// Operaciones
public:

// Reemplazos
public:
      virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
      void StatusBarMessage(TCHAR * fmt);


// Implementación
public:
      virtual ~CMainFrame();
#ifdef _DEBUG
      virtual void AssertValid() const;
      virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // Miembros incrustados de la barra de control
      CStatusBar  m_wndStatusBar;
      CToolBar    m_wndToolBar;

// Funciones de asignación de mensajes generadas
public:


      //{{AFX_MSG(CMainFrame)

      afx_msg LRESULT OnSetMessageString(WPARAM wParam, LPARAM lParam);
```

```
        //}}AFX_MSG
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        DECLARE_MESSAGE_MAP()



};
```

stdafx.h: archivo de inclusión para archivos de inclusión estándar del sistema,
o archivos de inclusión específicos del proyecto utilizados frecuentemente,
pero cambiados rara vez

```
// stdafx.h: archivo de inclusión para archivos de inclusión estándar del
sistema,
// o archivos de inclusión específicos del proyecto utilizados
frecuentemente,
// pero cambiados rara vez

#pragma once

#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN         // Excluir material rara vez utilizado de
encabezados de Windows
#endif

// Modificar las siguientes secciones define si su objetivo es una
plataforma distinta a las especificadas a continuación.
// Consulte la referencia MSDN para obtener la información más reciente
sobre los valores correspondientes a diferentes plataformas.
#ifndef WINVER                       // Permitir el uso de características
específicas de Windows 95 y Windows NT 4 o posterior.
#define WINVER 0x0400        // Cambiar para establecer el valor
apropiado para Windows 98 y Windows 2000 o posterior.
#endif

#ifndef _WIN32_WINNT         // Permitir el uso de características
específicas de Windows NT 4 o posterior.
#define _WIN32_WINNT 0x0400        // Cambiar para establecer el valor
apropiado para Windows 98 y Windows 2000 o posterior.
#endif

#ifndef _WIN32_WINDOWS       // Permitir el uso de características
específicas de Windows 98 o posterior.
#define _WIN32_WINDOWS 0x0410 // Cambiar para establecer el valor
apropiado para Windows Me o posterior.
#endif

#ifndef _WIN32_IE            // Permitir el uso de características
específicas de Internet Explorer 4.0 o posterior.
#define _WIN32_IE 0x0400     // Cambiar para establecer el valor
apropiado para IE 5.0 o posterior.
#endif
```

```
#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS     // Algunos constructores
CString serán explícitos

// Desactiva la ocultación de MFC para algunos mensajes de advertencia
comunes y, muchas veces, omitidos de forma consciente
#define _AFX_ALL_WARNINGS

#include <afxwin.h>         // Componentes principales y estándar de MFC
#include <afxext.h>         // Extensiones de MFC
#include <afxdisp.h>        // Clases de automatización de MFC

////////////////// Todods mis includes
//#include <winscard.h>/////******************** para mi tal
#include <windows.h>

# include <winscard.h>



#include <afxdtctl.h>         // Compatibilidad MFC para controles
comunes de Internet Explorer 4
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                 // Compatibilidad MFC para controles
comunes de Windows
#endif // _AFX_NO_AFXCMN_SUPPORT
#include <afxdb.h>         // ODBC

#include <atlbase.h>
#include <afxoledb.h>
#include <atlplus.h>
#include <afxdao.h>
```

Tarjeta.h: fichero de cabecera de la clase tarjeta.

```
#ifndef _Tarjeta_H_
#define _Tarjeta_H_
//////#include <windows.h>
#include "stdafx.h"
#include <winscard.h>

#ifdef __cplusplus
extern "C" {
#endif



      #ifndef _Bytes_H_
            #include "Bytes.h"
      #endif

      class Tarjeta{

            public:
                  Tarjeta();
                  ~Tarjeta();
```

```
                  int connect();
                  //Return:
                  //1: OK

                  int disconnect();

                  int readDNI(Bytes **dni);
                  //Return:
                  //1: OK
                  //-20: Not enough memory


            //    int checkStructureSC();
                  //Return:
                  //1: OK
                  //-20: Not enough memory

            //    int erase();
                  //Return:
                  //1: OK

            protected:
                  long sendAPDU(IN Bytes *apdu, OUT unsigned char
**status);
                  long getResponse(IN unsigned long dwLength, OUT
unsigned char *lplpbResponse);
                  long getRead (IN unsigned char dwLength, IN unsigned
short offset, OUT unsigned char * lplpbResponse);
                  int getField(IN unsigned char selectAPDU[], IN unsigned
long lSelectAPDU, OUT unsigned char* content, OUT unsigned long*
lContent);
                  int selectEF(IN unsigned char selectAPDU[], IN unsigned
long lSelectAPDU);
                  int setStandardField(unsigned char selectAPDU[],
unsigned long lSelectAPDU, unsigned char *content, unsigned long
lContent, unsigned short offset);
                  int selectCSPSEG();
                  int selectDFRelease();



            private:
                  SCARDCONTEXT hContext;
                  long hsCard;

      };


#ifdef __cplusplus
}
#endif

#endif
```

9.2 Código de la Aplicación –Registro Control de Acceso-

MainFrm.cpp: Contiene la definición de la clase CMainFrame, que está derivada de CFrameWnd, clase que controla las características del marco de una aplicación SDI(Single Documents Interface).

```
// MainFrm.cpp: implementación de la clase CMainFrame
//

#include "stdafx.h"
#include "Registro - Control de Acceso.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
	ON_WM_CREATE()
END_MESSAGE_MAP()

static UINT indicators[] =
{
	ID_SEPARATOR,           // Indicador de línea de estado
	ID_INDICATOR_CAPS,
	ID_INDICATOR_NUM,
	ID_INDICATOR_SCRL,
};


// Construcción o destrucción de CMainFrame

CMainFrame::CMainFrame()
{
	// TODO: agregar aquí el código de inicialización adicional de
miembros
}

CMainFrame::~CMainFrame()
{
}


int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;
	/*
	if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD |
WS_VISIBLE | CBRS_TOP
		| CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY |
CBRS_SIZE_DYNAMIC) ||
		!m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
	{
		TRACE0("No se pudo crear la barra de herramientas\n");
		return -1;      // No se pudo crear
```

```
	}
/*/
	if (!m_wndStatusBar.Create(this) ||
		!m_wndStatusBar.SetIndicators(indicators,
		  sizeof(indicators)/sizeof(UINT)))
	{
		TRACE0("No se pudo crear la barra de estado\n");
		return -1;		// No se pudo crear
	}
	// TODO: eliminar estas tres líneas si no desea que la barra de
herramientas se pueda acoplar
	m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
	EnableDocking(CBRS_ALIGN_ANY);
	DockControlBar(&m_wndToolBar);

	return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{

	if( !CFrameWnd::PreCreateWindow(cs) )
		return FALSE;
	// TODO: modificar aquí la clase Window o los estilos cambiando
	//  CREATESTRUCT cs
	cs.style &= ~(	WS_MAXIMIZEBOX | WS_THICKFRAME )  ;

	return TRUE;
}


// Diagnósticos de CMainFrame

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
	CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
	CFrameWnd::Dump(dc);
}

#endif //_DEBUG


// Controladores de mensaje de CMainFrame
```

Registro-Control de Acceso.cpp: Es el fichero fuente principal, contiene la definición de de la aplicación. Contiene la definición de la clase CRegistroControldeAccesoApp.

```
// Registro - Control de Acceso.cpp : define los comportamientos de las
clases para la aplicación.
//

#include "stdafx.h"
#include "Registro - Control de Acceso.h"
```

```
#include "MainFrm.h"

#include "Registro - Control de AccesoDoc.h"
#include "Registro - Control de AccesoView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CRegistroControldeAccesoApp

BEGIN_MESSAGE_MAP(CRegistroControldeAccesoApp, CWinApp)
	ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
	// Comandos de documento estándar basados en archivo
	ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
	ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()


// Construcción de CRegistroControldeAccesoApp

CRegistroControldeAccesoApp::CRegistroControldeAccesoApp()
{
	// TODO: agregar aquí el código de construcción,
	// Colocar toda la inicialización importante en InitInstance
}


// El único objeto CRegistroControldeAccesoApp

CRegistroControldeAccesoApp theApp;

// Inicialización de CRegistroControldeAccesoApp

BOOL CRegistroControldeAccesoApp::InitInstance()
{
	// Windows XP requiere InitCommonControls() si un manifiesto de
	// aplicación especifica el uso de ComCtl32.dll versión 6 o
posterior para habilitar
	// estilos visuales. De lo contrario, se generará un error al crear
ventanas.
	InitCommonControls();

	CWinApp::InitInstance();

	// Inicializar bibliotecas OLE
	if (!AfxOleInit())
	{
		AfxMessageBox(IDP_OLE_INIT_FAILED);
		return FALSE;
	}
	AfxEnableControlContainer();
	// Inicialización estándar
	// Si no utiliza estas características y desea reducir el tamaño
	// del archivo ejecutable final, debe quitar
	// las rutinas de inicialización específicas que no necesite
	// Cambie la clave del Registro en la que se almacena la
configuración
```

```
	// TODO: debe modificar esta cadena para que contenga información
correcta
	// como el nombre de su compañía u organización
	SetRegistryKey(_T("Aplicaciones generadas con el Asistente para
aplicaciones local"));
	LoadStdProfileSettings(4);  // Cargar opciones de archivo INI
estándar (incluidas las de la lista MRU)
	// Registrar las plantillas de documento de la aplicación. Las
plantillas de documento
	//  sirven como conexión entre documentos, ventanas de marco y
vistas
	CSingleDocTemplate* pDocTemplate;
	pDocTemplate = new CSingleDocTemplate(
		IDR_MAINFRAME,
		RUNTIME_CLASS(CRegistroControldeAccesoDoc),
		RUNTIME_CLASS(CMainFrame),       // Ventana de marco MDI
principal
		RUNTIME_CLASS(CRegistroControldeAccesoView));
	if (!pDocTemplate)
		return FALSE;
	AddDocTemplate(pDocTemplate);
	// Analizar línea de comandos para comandos Shell estándar, DDE,
Archivo Abrir
	CCommandLineInfo cmdInfo;
	ParseCommandLine(cmdInfo);
	// Enviar comandos especificados en la línea de comandos. Devolverá
FALSE si
	// la aplicación se inició con los modificadores /RegServer,
/Register, /Unregserver o /Unregister.
	if (!ProcessShellCommand(cmdInfo))
		return FALSE;
	// Se ha inicializado la única ventana; mostrarla y actualizarla
	m_pMainWnd->ShowWindow(SW_SHOW);
	m_pMainWnd->UpdateWindow();
	// Llamar a DragAcceptFiles sólo si existe un sufijo
	//  En una aplicación SDI, esto debe ocurrir después de
ProcessShellCommand
	return TRUE;
}



// Cuadro de diálogo CAboutDlg utilizado para el comando Acerca de

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Datos del cuadro de diálogo
	enum { IDD = IDD_ABOUTBOX };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    //
Compatibilidad con DDX/DDV

// Implementación
protected:
	DECLARE_MESSAGE_MAP()
};
```

```
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// Comando de la aplicación para ejecutar el cuadro de diálogo
void CRegistroControldeAccesoApp::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
}

// Controladores de mensaje de CRegistroControldeAccesoApp
```

Registro-Control de AccesoDoc.cpp: Este fichero creado automáticamente es útil cuando se necesite añadir datos especiales o implementar las órdenes de cargar o guardar del menú Archivo.

```
// Registro - Control de AccesoDoc.cpp: implementación de la clase
CRegistroControldeAccesoDoc
//

#include "stdafx.h"
#include "Registro - Control de Acceso.h"

#include "Registro - Control de AccesoDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CRegistroControldeAccesoDoc

IMPLEMENT_DYNCREATE(CRegistroControldeAccesoDoc, CDocument)

BEGIN_MESSAGE_MAP(CRegistroControldeAccesoDoc, CDocument)
END_MESSAGE_MAP()


// Construcción o destrucción de CRegistroControldeAccesoDoc

CRegistroControldeAccesoDoc::CRegistroControldeAccesoDoc()
{
	// TODO: agregar aquí el código de construcción único

}

CRegistroControldeAccesoDoc::~CRegistroControldeAccesoDoc()
```

```
{
}

BOOL CRegistroControldeAccesoDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;

	// TODO: agregar aquí código de reinicio
	// (los documentos SDI volverán a utilizar este documento)

	return TRUE;
}



// Serialización de CRegistroControldeAccesoDoc

void CRegistroControldeAccesoDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{
		// TODO: agregar aquí el código de almacenamiento
	}
	else
	{
		// TODO: agregar aquí el código de carga
	}
}


// Diagnósticos de CRegistroControldeAccesoDoc

#ifdef _DEBUG
void CRegistroControldeAccesoDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CRegistroControldeAccesoDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG


// Comandos de CRegistroControldeAccesoDoc
```

Registro_Control_de_AccesoView.cpp: Este archivo contiene la implementación de la clase CRegistroControldeAccesoView, encargada de gestionar los ojetos que formaran la vista del documento.

```
// Registro - Control de AccesoView.cpp: implementación de la clase
CRegistroControldeAccesoView
//
```

```
#include <afxdb.h>        // ODBC
#include <atlbase.h>
#include <afxoledb.h>
#include <atlplus.h>
#include <afxdao.h>
#include "stdafx.h"
#include "Registro - Control de Acceso.h"

#include "Registro - Control de AccesoDoc.h"
#include "Registro - Control de AccesoView.h"
#include ".\registro - control de accesoview.h"

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <afxdb.h>
#include <sstream>

#include <iostream>
#include <fstream>
using namespace std;
#include <string.h>


#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CRegistroControldeAccesoView

IMPLEMENT_DYNCREATE(CRegistroControldeAccesoView, CFormView)

BEGIN_MESSAGE_MAP(CRegistroControldeAccesoView, CFormView)


      //ON_CBN_SELCHANGE(IDC_DIA_INICIO, OnCbnSelchangeDiaInicio)




      ON_BN_CLICKED(IDC_INTERVALO, OnBnClickedIntervalo)
      ON_BN_CLICKED(IDC_CHANGE_DATA, OnBnClickedChangeData)

      ON_BN_CLICKED(IDC_LIMPIAR, OnBnClickedLimpiar)

      ON_BN_CLICKED(IDC_BUSCAR_REG, OnBnClickedBuscarReg)

      ON_BN_CLICKED(IDC_ADD_REG, OnBnClickedAddReg)


      ON_BN_CLICKED(IDC_ELIMINAR_HIS, OnBnClickedEliminarHis)
      ON_BN_CLICKED(IDC_ELIMINAR_REG, OnBnClickedEliminarReg)
END_MESSAGE_MAP()

// Construcción o destrucción de CRegistroControldeAccesoView

CRegistroControldeAccesoView::CRegistroControldeAccesoView()
      : CFormView(CRegistroControldeAccesoView::IDD)
```

```
	, StringYearInicio(_T(""))
	, StringDiaInicio(_T(""))
	, StringMesInicio(_T(""))
	, StringDiaFin(_T(""))
	, StringMesFin(_T(""))
	, StringYearFin(_T(""))
	, bCheckIntervalo(FALSE)
	, stringDni(_T(""))
	, stringNombre(_T(""))
	, stringApellido1(_T(""))
	, stringApellido2(_T(""))
	, stringHistorial(_T(""))
	, bAgosto(FALSE)
	, bFinSem(FALSE)
	, bGuardarCancelar(false)
	, bP1(FALSE)
	, bP2(FALSE)
	, bP3(FALSE)
	, bP4(FALSE)
	, bP5(FALSE)
	, bP6(FALSE)
	, bP7(FALSE)
	, bP8(FALSE)
{
	// TODO: agregar aquí el código de construcción

}

CRegistroControldeAccesoView::~CRegistroControldeAccesoView()
{
}

void CRegistroControldeAccesoView::DoDataExchange(CDataExchange* pDX)
{
	CFormView::DoDataExchange(pDX);

	DDX_CBString(pDX, IDC_YEAR_INICIO, StringYearInicio);
	DDX_CBString(pDX, IDC_DIA_INICIO, StringDiaInicio);
	DDX_CBString(pDX, IDC_MES_INICIO, StringMesInicio);
	DDX_CBString(pDX, IDC_DIA_FIN, StringDiaFin);
	DDX_CBString(pDX, IDC_MES_FIN, StringMesFin);
	DDX_CBString(pDX, IDC_YEAR_FIN, StringYearFin);
	DDX_Check(pDX, IDC_INTERVALO, bCheckIntervalo);
	DDX_Text(pDX, IDC_DNI_BOX, stringDni);
	DDX_Text(pDX, IDC_NOMBRE_BOX, stringNombre);
	DDX_Text(pDX, IDC_APELLIDO1_BOX, stringApellido1);
	DDX_Text(pDX, IDC_APELLIDO2_BOX, stringApellido2);
	DDX_Text(pDX, IDC_HISTORIAL, stringHistorial);
	DDX_Check(pDX, IDC_AGOSTO, bAgosto);
	DDX_Check(pDX, IDC_FIN_SEMANA, bFinSem);
	DDX_Check(pDX, IDC_P1, bP1);
	DDX_Check(pDX, IDC_P2, bP2);
	DDX_Check(pDX, IDC_P3, bP3);
	DDX_Check(pDX, IDC_P4, bP4);
	DDX_Check(pDX, IDC_P5, bP5);
	DDX_Check(pDX, IDC_P6, bP6);
	DDX_Check(pDX, IDC_P7, bP7);
	DDX_Check(pDX, IDC_P8, bP8);
}
```

```
BOOL CRegistroControldeAccesoView::PreCreateWindow(CREATESTRUCT& cs)
{
	// TODO: modificar aquí la clase Window o los estilos cambiando
	//  CREATESTRUCT cs

	return CFormView::PreCreateWindow(cs);
}

void CRegistroControldeAccesoView::OnInitialUpdate()
{
	CFormView::OnInitialUpdate();
	GetParentFrame()->RecalcLayout();
	ResizeParentToFit();
}


// Diagnósticos de CRegistroControldeAccesoView

#ifdef _DEBUG
void CRegistroControldeAccesoView::AssertValid() const
{
	CFormView::AssertValid();
}

void CRegistroControldeAccesoView::Dump(CDumpContext& dc) const
{
	CFormView::Dump(dc);
}

CRegistroControldeAccesoDoc* CRegistroControldeAccesoView::GetDocument()
const // La versión de no depuración es en línea
{
	ASSERT(m_pDocument-
>IsKindOf(RUNTIME_CLASS(CRegistroControldeAccesoDoc)));
	return (CRegistroControldeAccesoDoc*)m_pDocument;
}
#endif //_DEBUG


// Controladores de mensaje de CRegistroControldeAccesoView




void CRegistroControldeAccesoView::OnBnClickedIntervalo()
{
	// TODO: Add your control notification handler code here
//	CButton * auxButton;
	CEdit * auxEdit;
	CWnd * auxGroup;

	/// Comprobar si la casilla de verificación está activada
	CButton * m_ctlCheck = (CButton *) GetDlgItem (IDC_INTERVALO);
	int ChkBox = m_ctlCheck-> GetCheck ();


	if (ChkBox == BST_UNCHECKED){
```

```
            /// Casilla desactIvada, desactivar los CEdit, Text  y los
GroupBox
            auxEdit = (CEdit*)GetDlgItem (IDC_DIA_INICIO);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_MES_INICIO);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_INICIO);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_DIA_FIN);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_MES_FIN);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_FIN);
            auxEdit->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_INI);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_FIN);
            auxGroup->EnableWindow(FALSE);
            /// Static Text
            auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_INICIO);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_FIN);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_INICIO);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_FIN);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_INICIO);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_FIN);
            auxGroup->EnableWindow(FALSE);

      }else if (ChkBox == BST_CHECKED){
            /// Casilla actvada, activar los CEdit y los GroupBox
            //SetDlgItemText(IDC_APELLIDO2_BOX ,LPCTSTR("Intervalo
activado"));
            auxEdit = (CEdit*)GetDlgItem (IDC_DIA_INICIO);
            auxEdit->EnableWindow(TRUE);
            auxEdit = (CEdit*)GetDlgItem (IDC_MES_INICIO);
            auxEdit->EnableWindow(TRUE);
            auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_INICIO);
            auxEdit->EnableWindow(TRUE);
            auxEdit = (CEdit*)GetDlgItem (IDC_DIA_FIN);
            auxEdit->EnableWindow(TRUE);
            auxEdit = (CEdit*)GetDlgItem (IDC_MES_FIN);
            auxEdit->EnableWindow(TRUE);
            auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_FIN);
            auxEdit->EnableWindow(TRUE);
            auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_INI);
            auxGroup->EnableWindow(TRUE);
            auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_FIN);
            auxGroup->EnableWindow(TRUE);
            /// Static Text
            auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_INICIO);
            auxGroup->EnableWindow(TRUE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_FIN);
            auxGroup->EnableWindow(TRUE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_INICIO);
            auxGroup->EnableWindow(TRUE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_FIN);
```

```
			auxGroup->EnableWindow(TRUE);
			auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_INICIO);
			auxGroup->EnableWindow(TRUE);
			auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_FIN);
			auxGroup->EnableWindow(TRUE);
		}
}



void CRegistroControldeAccesoView::OnBnClickedChangeData()
{
		// TODO: Add your control notification handler code here
		CButton * auxButton;
		CEdit * auxEdit;
		CWnd * auxGroup;
		char *ptr;
		int iAux1;
		char sAux1 [20];
		CString csAux1;
		CString puerta;
		CString agosto;
		CString str;
		int iAlumnosPicadas;

		CDaoDatabase *pDao = new CDaoDatabase;
		CDaoRecordset r(pDao);


		if(bGuardarCancelar==false)
		{


			/// Desactivar el CEdit DNI, el dni no puede ser modificado
			auxEdit = (CEdit*)GetDlgItem (IDC_DNI_BOX);
			auxEdit->EnableWindow(FALSE);

			/// Los campos nombre y apellidos se podran cambiar
			auxEdit = (CEdit*)GetDlgItem (IDC_NOMBRE_BOX);
			auxEdit->SetReadOnly(FALSE);
			auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO1_BOX);
			auxEdit->SetReadOnly(FALSE);
			auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO2_BOX);
			auxEdit->SetReadOnly(FALSE);


			/// Activar el grupo "Periodos restringidos"
			auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_RES);
			auxGroup->EnableWindow(TRUE);

			/// Activar las casillas de verificacion de restricciones
			auxButton = (CButton *) GetDlgItem (IDC_FIN_SEMANA);
			auxButton->EnableWindow(TRUE);
			auxButton = (CButton *) GetDlgItem (IDC_AGOSTO);
			auxButton->EnableWindow(TRUE);
			auxButton = (CButton *) GetDlgItem (IDC_INTERVALO);
			auxButton->EnableWindow(TRUE);
			/// Comprobar estado del boton "Intervalo":
```

```
            ///   Activado->activar grupos "ini-fin",  Desactivado-
>desactivar grupos
            int ChkBox = auxButton-> GetCheck ();
            if (ChkBox == BST_UNCHECKED){

                  auxEdit = (CEdit*)GetDlgItem (IDC_DIA_INICIO);
                  auxEdit->EnableWindow(FALSE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_MES_INICIO);
                  auxEdit->EnableWindow(FALSE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_INICIO);
                  auxEdit->EnableWindow(FALSE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_DIA_FIN);
                  auxEdit->EnableWindow(FALSE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_MES_FIN);
                  auxEdit->EnableWindow(FALSE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_FIN);
                  auxEdit->EnableWindow(FALSE);


            }else if (ChkBox == BST_CHECKED)
            {
                  auxEdit = (CEdit*)GetDlgItem (IDC_DIA_INICIO);
                  auxEdit->EnableWindow(TRUE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_MES_INICIO);
                  auxEdit->EnableWindow(TRUE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_INICIO);
                  auxEdit->EnableWindow(TRUE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_DIA_FIN);
                  auxEdit->EnableWindow(TRUE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_MES_FIN);
                  auxEdit->EnableWindow(TRUE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_FIN);
                  auxEdit->EnableWindow(TRUE);
                  /// Static Text
                  auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_INICIO);
                  auxGroup->EnableWindow(TRUE);
                  auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_FIN);
                  auxGroup->EnableWindow(TRUE);
                  auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_INICIO);
                  auxGroup->EnableWindow(TRUE);
                  auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_FIN);
                  auxGroup->EnableWindow(TRUE);
                  auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_INICIO);
                  auxGroup->EnableWindow(TRUE);
                  auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_FIN);
                  auxGroup->EnableWindow(TRUE);
                  /// Activar las etiquetas de grupo
                  auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_INI);
                  auxGroup->EnableWindow(TRUE);
                  auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_FIN);
                  auxGroup->EnableWindow(TRUE);

            }


            /// El historial permanece desactivado.
            /// Activar los cedit Nombre y apellidos y el grupo "Datos
Personales"
            auxEdit = (CEdit*)GetDlgItem (IDC_NOMBRE_BOX);
            auxEdit->EnableWindow(TRUE);
```

```
            auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO1_BOX);
            auxEdit->EnableWindow(TRUE);
            auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO2_BOX);
            auxEdit->EnableWindow(TRUE);

            /// Activar el grupo y las casillas de verificación
"Entradas"
            auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_ENT);
            auxGroup->EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_P1);
            auxButton-> EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_P2);
            auxButton-> EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_P3);
            auxButton-> EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_P4);
            auxButton->EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_P5);
            auxButton-> EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_P6);
            auxButton-> EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_P7);
            auxButton-> EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_P8);
            auxButton-> EnableWindow(TRUE);



            /// Desactivar los botones  [Eliminar Historial],[Eliminar
Registro] , [Limpiar] y [Buscar Registro]
            auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_HIS);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_REG);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_LIMPIAR);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_BUSCAR_REG);
            auxButton-> EnableWindow(FALSE);

            /// Activar Botones Modificar->Cancelar  ,  Nuebo Registro-
>GUARDAR
            auxButton = (CButton *) GetDlgItem (IDC_ADD_REG);
            auxButton-> EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_CHANGE_DATA);
            auxButton-> EnableWindow(TRUE);
            SetDlgItemText(IDC_ADD_REG ,LPCTSTR("GUARDAR"));
            SetDlgItemText(IDC_CHANGE_DATA ,LPCTSTR("CANCELAR"));

            /// Lanzamos un MessajeBox con las instrucciones para cambiar
los datos de un registro
            str= "Usted ha pulsado [Modificar Datos].\nSi desea cambiar
el estado de este registro, realice dichos cambios y pulse [GUARDAR].\nSi
no desea guardar los cambios pulse [CANCELAR].";
                  AfxMessageBox (str);

            bGuardarCancelar = true;
      }///fin del if(bGuardarCancelar==false)
      else{
            /// En este caso el usuario ha pulsado [CANCELAR]. Se vuelve
al estado inicial
```

```
            /// Los botones añadir y modificar vuelven a su estado
inicial
            SetDlgItemText(IDC_ADD_REG ,LPCTSTR("Añadir Registro"));
            SetDlgItemText(IDC_CHANGE_DATA ,LPCTSTR("Modificar Datos de
Registro"));
            bGuardarCancelar = false;

            /// Realizar las mismas operaciones que al pulsar [BUSCAR]
            /// Comprobación correcta. Buscar en la base de datos el DNI
            pDao->Open ("alumnos.mdb");
            COleVariant v;
            /// Este flags indica si estamos trabajando en la tabla
alumnos o picadas
            iAlumnosPicadas=0;
            /// Construccion de la sentencia sql
            CString sd = "SELECT * FROM Alumnos WHERE dni =";    //33
caracteres
            sd = sd+ stringDni ;
            try{
            /// Comprobar si existe ese dni en la base de datos
                  r.Open(dbOpenDynaset,LPCTSTR(sd) , dbReadOnly );
                  r.GetFieldValue ("Nombre", v );
                  stringNombre = (char*)v.bstrVal;
                  r.GetFieldValue ("Apellido1", v);
                  stringApellido1= (char*)v.bstrVal;
                  r.GetFieldValue ("Apellido2", v);
                  stringApellido2= (char*)v.bstrVal;
                  r.GetFieldValue ("Puerta", v);
                  /// ACTIVAR/DESACTIVAR las entradas a las que este
registro tiene acceso
                  puerta = (char*)v.bstrVal;
                  bool abPuertas[8];
                  char numeros[8] = {'1','2','3','4','5','6','7','8'};
                  for(int i =0; i<8 ; i++){
                 if( (ptr = strchr( puerta , numeros[i] )) != NULL ){
                              abPuertas[i]=true;
                        }
                  }
                  auxButton = (CButton *) GetDlgItem (IDC_P1);

                  if( abPuertas[0]==true ){
                        auxButton->SetCheck(BST_CHECKED);
                  }else{
                        auxButton->SetCheck(BST_UNCHECKED);
                  }
                  auxButton = (CButton *) GetDlgItem (IDC_P2);
                  if( abPuertas[1]==true){
                        auxButton->SetCheck(BST_CHECKED);
                  }else{
                        auxButton->SetCheck(BST_UNCHECKED);
                  }
                  auxButton = (CButton *) GetDlgItem (IDC_P3);
                  if( abPuertas[2]==true){
                        auxButton->SetCheck(BST_CHECKED);
                  }else{
                        auxButton->SetCheck(BST_UNCHECKED);
                  }
                  auxButton = (CButton *) GetDlgItem (IDC_P4);
                  if( abPuertas[3]==true){
                        auxButton->SetCheck(BST_CHECKED);
```

```
}else{
	auxButton->SetCheck(BST_UNCHECKED);
}
auxButton = (CButton *) GetDlgItem (IDC_P5);
if( abPuertas[4]==true){
	auxButton->SetCheck(BST_CHECKED);
}else{
	auxButton->SetCheck(BST_UNCHECKED);
}
auxButton = (CButton *) GetDlgItem (IDC_P6);
if( abPuertas[5]==true){
	auxButton->SetCheck(BST_CHECKED);
}else{
	auxButton->SetCheck(BST_UNCHECKED);
}
auxButton = (CButton *) GetDlgItem (IDC_P7);
if( abPuertas[6]==true){
	auxButton->SetCheck(BST_CHECKED);
}else{
	auxButton->SetCheck(BST_UNCHECKED);
}
auxButton = (CButton *) GetDlgItem (IDC_P8);
if( abPuertas[7]==true){
	auxButton->SetCheck(BST_CHECKED);
}else{
	auxButton->SetCheck(BST_UNCHECKED);
}

/// Comporbar la restricción sobre el mes de "Agosto"
r.GetFieldValue ("Agosto", v);
agosto = (char*)v.bstrVal;
auxButton = (CButton *) GetDlgItem (IDC_AGOSTO);
if(   (ptr = strchr( agosto , '1' )) != NULL  ){
	/// En este caso el periodo está restringido
	auxButton->SetCheck(BST_CHECKED);
	bAgosto=TRUE;
}else{
	auxButton->SetCheck(BST_UNCHECKED);
	bAgosto=FALSE;
}

/// Comporbar la restricción sobre "Fin de semana"
r.GetFieldValue ("Fin_de_semana", v);
agosto = (char*)v.bstrVal;
auxButton = (CButton *) GetDlgItem (IDC_FIN_SEMANA);
if(   (ptr = strchr( agosto , '1' )) != NULL  ){
	/// En este caso el periodo está restringido
	bFinSem=TRUE;
	auxButton->SetCheck(BST_CHECKED);
}else{
	auxButton->SetCheck(BST_UNCHECKED);
	bFinSem=FALSE;
}

/// Ccomprobar si tiene acctivada la restricción
temporal
r.GetFieldValue ("Periodo_temporal", v);
agosto = (char*)v.bstrVal;
auxButton = (CButton *) GetDlgItem (IDC_INTERVALO);
if(   (ptr = strchr( agosto , '1' )) != NULL  ){
```

```
                        /// En este caso el periodo está restringido
                        /// Recoger datos de inicio y fin de restriccion
                        r.GetFieldValue ("FechaA", v);
                        csAux1=(char*)v.bstrVal;
                        char aux[3];
                        aux[2]='\0';
                        char* cAux = csAux1.GetBuffer( csAux1.GetLength()
);
                        aux[0]=cAux[0];
                        aux[1]=cAux[1];
                        StringDiaInicio=aux;
                        aux[0]=cAux[3];
                        aux[1]=cAux[4];
                        StringMesInicio=aux;
                        aux[0]=cAux[6];
                        aux[1]=cAux[7];
                        StringYearInicio=aux;
                        /// Ahora con la fecha de fin:
                        r.GetFieldValue ("FechaB", v);
                        csAux1=(char*)v.bstrVal;
                        cAux = csAux1.GetBuffer( csAux1.GetLength() );
                        aux[0]=cAux[0];
                        aux[1]=cAux[1];
                        StringDiaFin=aux;
                        aux[0]=cAux[3];
                        aux[1]=cAux[4];
                        StringMesFin=aux;
                        aux[0]=cAux[6];
                        aux[1]=cAux[7];
                        StringYearFin=aux;

                        auxButton->SetCheck(BST_CHECKED);
                        bCheckIntervalo = TRUE;
                  }else{
                        /// En este caso  "Intervalo" no está activado
                        auxButton->SetCheck(BST_UNCHECKED);
                        bCheckIntervalo = FALSE;
                  }///Fin del if() de Intervalo activado
                  /// Representar los datos en el form
                  /// Mostrar el historial para este registro
                  stringHistorial= _T("");      // Limpiar Historial
                  r.Close();  /// Cerramos el vinculo a la tabla Alumnos
                  pDao->Open ("Picadas.mdb"); /// Crear vinculo a tabla
"Picadas"
                  sd = "SELECT * FROM Picadas WHERE dni =";//33
caracteres
                  sd = sd+ stringDni ;
                  /// Este flags indica si estamos trabajando en la tabla
alumnos o picadas
                  iAlumnosPicadas=1;
                  r.Open(dbOpenDynaset,LPCTSTR(sd) , dbReadOnly );
                  r.MoveFirst();
                  while(      !r.IsEOF () ) {
                        r.GetFieldValue ( "Fecha" , v );
                        stringHistorial = stringHistorial +
(char*)v.bstrVal;
                        stringHistorial = stringHistorial + "  " ;
                        r.GetFieldValue ( "Hora" , v );
                        stringHistorial = stringHistorial +
(char*)v.bstrVal;
```

```
                        stringHistorial = stringHistorial + "  " ;
                        r.GetFieldValue ( "Puerta" , v );
                        stringHistorial = stringHistorial + "Acceso a
entrada nº " ;
                        stringHistorial = stringHistorial +
(char*)v.bstrVal;
                        stringHistorial = stringHistorial + "\r\n" ;
                        r.MoveNext();
                  }
                  r.Close();
                  ////////////////////////////////////////////
                  /// Actualizar los CEdit para mostrar los datos
                  UpdateData(false);
                  /// Mostrar el historial
                  auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_HIS);
                  auxGroup->EnableWindow(TRUE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_HISTORIAL);
                  auxEdit->EnableWindow(TRUE);
                        //SetDlgItemText(IDC_HISTORIAL
,LPCTSTR(stringNombre));

                  /// Si todo ha salido bien pongo los datos en sus
CEdit, luego activo los
                  /// botones [Eliminar Historial],[Eliminar Registro] y
[Cambiar datos]
                  auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_HIS);
                  auxButton->EnableWindow(TRUE);
                  auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_REG);
                  auxButton->EnableWindow(TRUE);
                  auxButton = (CButton *) GetDlgItem (IDC_CHANGE_DATA);
                  auxButton->EnableWindow(TRUE);
                  auxButton = (CButton *) GetDlgItem (IDC_LIMPIAR);
                  auxButton->EnableWindow(TRUE);
                  /// Desactivo los votones [Buscar registro]  y [Añadir
Registro]
                  auxButton = (CButton *) GetDlgItem (IDC_BUSCAR_REG);
                  auxButton->EnableWindow(FALSE);
                  auxButton = (CButton *) GetDlgItem (IDC_ADD_REG);
                  auxButton->EnableWindow(FALSE);

            }catch(CDaoException* e ) {
                  long number =e->m_pErrorInfo->m_lErrorCode;
                  std::ostringstream stm ;
                  stm << number ;
                  const char* cstr = stm.str().c_str() ;
                  CString tt = cstr;
                  if(number == 3021){
                        /// Este codigo indica que el registro no existe
en la base de datos
                        if(iAlumnosPicadas==0){

                              str = "[CHANGE DATA] El DNI introducido no
esta registrado en la base de datos";
                              AfxMessageBox(str);

                        }else{
                              /// En este caso ell error ha ocurrido al
no encontrar registros en la tabla picadas.
                              /// Actualizar los CEdit para mostrar los
datos
```

```
                              stringHistorial= _T("Este usuario no tiene
accesos registrados en su historial");
                              UpdateData(false);
                              auxGroup = (CWnd*)GetDlgItem
(IDC_GRUPO_HIS);
                              auxGroup->EnableWindow(TRUE);
                              auxEdit = (CEdit*)GetDlgItem
(IDC_HISTORIAL);
                              auxEdit->EnableWindow(TRUE);
                              /// Si todo ha salido bien activo los
                              /// botones [Eliminar Historial],[Eliminar
Registro] y [Cambiar datos]
                              auxButton = (CButton *) GetDlgItem
(IDC_ELIMINAR_HIS);
                              auxButton->EnableWindow(TRUE);
                              auxButton = (CButton *) GetDlgItem
(IDC_ELIMINAR_REG);
                              auxButton->EnableWindow(TRUE);
                              auxButton = (CButton *) GetDlgItem
(IDC_CHANGE_DATA);
                              auxButton->EnableWindow(TRUE);
                              auxButton = (CButton *) GetDlgItem
(IDC_LIMPIAR);
                              auxButton->EnableWindow(TRUE);
                              /// Desactivo los votones [Buscar registro]
y [Añadir Registro]
                              auxButton = (CButton *) GetDlgItem
(IDC_BUSCAR_REG);
                              auxButton->EnableWindow(FALSE);
                              auxButton = (CButton *) GetDlgItem
(IDC_ADD_REG);
                              auxButton->EnableWindow(FALSE);



                        }
                        e->Delete();
                        r.Close();
                  }else{
                        //SetDlgItemText(IDC_APELLIDO1_BOX,LPCTSTR("Algun
problema en BBDD"));
                        AfxMessageBox(e->m_pErrorInfo->m_strDescription);
                        e->Delete();
                        r.Close();
                  }
            }

            /// Si todo ha salido bien pongo los datos en sus CEdit,
luego activo los
            /// botones [Eliminar Historial],[Eliminar Registro] y
[Cambiar datos]
            auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_HIS);
            auxButton->EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_REG);
            auxButton->EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_CHANGE_DATA);
            auxButton->EnableWindow(TRUE);
            auxButton = (CButton *) GetDlgItem (IDC_LIMPIAR);
            auxButton->EnableWindow(TRUE);
```

```
            /// Desactivo los votones [Buscar registro]  y [Añadir
Registro]
            auxButton = (CButton *) GetDlgItem (IDC_BUSCAR_REG);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_ADD_REG);
            auxButton->EnableWindow(FALSE);

            /// Desactivar los grupos "restringidos", "historial",
"inicio", "fin", "Entradas"
            auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_RES);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_HIS);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_INI);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_FIN);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_ENT);
            auxGroup->EnableWindow(FALSE);

            ///Desactivar el grupo y EL CEdit "datos personales", "DNI";
Desactivar los CEdit nombre y apellidos
            auxGroup = (CWnd*)GetDlgItem (IDC_DATOS_PERSONALES);
            auxGroup->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_DNI_BOX);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_NOMBRE_BOX);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO1_BOX);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO2_BOX);
            auxEdit->EnableWindow(FALSE);

            /// Desactivar las casillas de verificación de restricción
            auxButton = (CButton *) GetDlgItem (IDC_FIN_SEMANA);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_AGOSTO);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_INTERVALO);
            auxButton->EnableWindow(FALSE);

            /// Desactivar los ComboBox "inicio" y "fin" de restricción
            auxEdit = (CEdit*)GetDlgItem (IDC_DIA_INICIO);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_MES_INICIO);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_INICIO);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_DIA_FIN);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_MES_FIN);
            auxEdit->EnableWindow(FALSE);
            auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_FIN);
            auxEdit->EnableWindow(FALSE);

            /// Desactivar las etiquetas "Dia", "Mes", "Año"...  Static
Text
            auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_INICIO);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_FIN);
```

```
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_INICIO);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_FIN);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_INICIO);
            auxGroup->EnableWindow(FALSE);
            auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_FIN);
            auxGroup->EnableWindow(FALSE);

            /// Limpiar y desactivar las casillas de verificacion de
Puertas con acceso permitido
            auxButton = (CButton *) GetDlgItem (IDC_P1);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_P2);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_P3);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_P4);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_P5);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_P6);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_P7);
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_P8);
            auxButton->EnableWindow(FALSE);


      }/// Fin del Else<-if(bGuardar...=true)




}



void CRegistroControldeAccesoView::OnBnClickedLimpiar()
{
      // TODO: Add your control notification handler code here
      CButton * auxButton;
      CEdit * auxEdit;
      CWnd * auxGroup;



      /// Limpiar todos los CEdit;  "grupo restringido",
"datospersonales" y "historial"
      StringYearInicio = _T(" ");   // Limpiar restringciones
      StringDiaInicio = _T(" ");
      StringMesInicio = _T(" ");
      StringDiaFin = _T(" ");
      StringMesFin = _T(" ");
      StringYearFin = _T(" ");
```

```
	stringDni = _T("");			//Limpiar datos personales
	stringNombre= _T("");
	stringApellido1= _T("");
	stringApellido2= _T("");
	stringHistorial= _T("");	// Limpiar Historial

	// Actualizar los CEdit;  view.cpp -> form
	UpdateData(false);

	/// El campo DNI se podrá escribir
	auxEdit = (CEdit*)GetDlgItem (IDC_DNI_BOX);
	auxEdit->SetReadOnly(FALSE);

	/// Limpiar las casillas de verificacion del "Grupo restringido"
	bCheckIntervalo = FALSE;
	auxButton = (CButton *) GetDlgItem (IDC_INTERVALO);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton = (CButton *) GetDlgItem (IDC_FIN_SEMANA);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton = (CButton *) GetDlgItem (IDC_AGOSTO);
	auxButton-> SetCheck( BST_UNCHECKED );

	/// Limpiar y desactivar las casillas de verificacion de Puertas
con acceso permitido
	auxButton = (CButton *) GetDlgItem (IDC_P1);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_P2);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_P3);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_P4);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_P5);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_P6);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_P7);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_P8);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);



	/// Desactivar los grupos "restringidos", "historial", "inicio",
"fin", "Entradas"
	auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_RES);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_HIS);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_INI);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_FIN);
```

```
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_ENT);
	auxGroup->EnableWindow(FALSE);

	/// Activar el grupo y EL CEdit "datos personales", "DNI";
Desactivar los CEdit nombre y apellidos
	auxGroup = (CWnd*)GetDlgItem (IDC_DATOS_PERSONALES);
	auxGroup->EnableWindow(TRUE);
	auxEdit = (CEdit*)GetDlgItem (IDC_DNI_BOX);
	auxEdit->EnableWindow(TRUE);
	auxEdit = (CEdit*)GetDlgItem (IDC_NOMBRE_BOX);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO1_BOX);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO2_BOX);
	auxEdit->EnableWindow(FALSE);

	/// Desactivar las casillas de verificación
	auxButton = (CButton *) GetDlgItem (IDC_FIN_SEMANA);
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_AGOSTO);
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_INTERVALO);
	auxButton->EnableWindow(FALSE);

	/// Desactivar los ComboBox "inicio" y "fin" de restricción
	auxEdit = (CEdit*)GetDlgItem (IDC_DIA_INICIO);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_MES_INICIO);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_INICIO);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_DIA_FIN);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_MES_FIN);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_FIN);
	auxEdit->EnableWindow(FALSE);

	/// Desactivar las etiquetas "Dia", "Mes", "Año"...  Static Text
	auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_INICIO);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_FIN);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_INICIO);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_FIN);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_INICIO);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_FIN);
	auxGroup->EnableWindow(FALSE);

	/// ACTIVAR/DESACTIVAR botones de la aplicacion
	/// Desactivar los botones  [Eliminar Historial],[Eliminar
Registro] y [Cambiar datos]
	auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_HIS);
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_REG);
	auxButton->EnableWindow(FALSE);
```

```
	auxButton = (CButton *) GetDlgItem (IDC_CHANGE_DATA);
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_LIMPIAR);
	auxButton->EnableWindow(TRUE);
	auxButton = (CButton *) GetDlgItem (IDC_BUSCAR_REG);
	auxButton->EnableWindow(TRUE);
	auxButton = (CButton *) GetDlgItem (IDC_ADD_REG);
	auxButton->EnableWindow(TRUE);


}


void CRegistroControldeAccesoView::OnBnClickedBuscarReg()
{
	//CDaoDatabase *pDao = new CDaoDatabase;
	char *ptr;
	int iAux1;
	char sAux1 [20];
	CString csAux1;
	CString puerta;
	CString agosto;
	CString str;
	CButton * auxButton;
	CEdit * auxEdit;
	CWnd * auxGroup;
	int iAlumnosPicadas;
	CDaoDatabase *pDao = new CDaoDatabase;
	CDaoRecordset r(pDao);

	UpdateData(true);	// Actualizar las CEdit;   form -> view.cpp

	iAux1= strlen(stringDni);	//Obtener el numero de digitos del
campo DNI
	if( iAux1 == 8 ){
		/// Comprobación correcta. Buscar en la base de datos el DNI
		pDao->Open ("alumnos.mdb");
		COleVariant v;
		/// El campo DNI solo se podrá leer
		auxEdit = (CEdit*)GetDlgItem (IDC_DNI_BOX);
		auxEdit->SetReadOnly(TRUE);

		iAlumnosPicadas =0; /// Flags que indicará, en que tabla
estamos buscando los datos
		/// Construccion de la sentencia sql
		CString sd = "SELECT * FROM Alumnos WHERE dni =";	//33
caracteres
		sd = sd+ stringDni ;
		try{
		/// Comprobar si existe ese dni en la base de datos
			r.Open(dbOpenDynaset,LPCTSTR(sd) , dbReadOnly );
			r.GetFieldValue ("Nombre", v );
			stringNombre = (char*)v.bstrVal;
			r.GetFieldValue ("Apellido1", v);
			stringApellido1= (char*)v.bstrVal;
			r.GetFieldValue ("Apellido2", v);
			stringApellido2= (char*)v.bstrVal;
			r.GetFieldValue ("Puerta", v);
			/// ACTIVAR/DESACTIVAR las entradas a las que este
registro tiene acceso
```

```
  puerta = (char*)v.bstrVal;
  bool abPuertas[8];
  char numeros[8] = {'1','2','3','4','5','6','7','8'};
  for(int i =0; i<8 ; i++){
if( (ptr = strchr( puerta , numeros[i] )) != NULL ){
              abPuertas[i]=true;
        }
  }
  auxButton = (CButton *) GetDlgItem (IDC_P1);
  if( abPuertas[0]==true ){
        bP1=TRUE;
        auxButton->SetCheck(BST_CHECKED);
  }else{
        auxButton->SetCheck(BST_UNCHECKED);
  }
  auxButton = (CButton *) GetDlgItem (IDC_P2);
  if( abPuertas[1]==true){
        bP2=TRUE;
        auxButton->SetCheck(BST_CHECKED);
  }else{
        auxButton->SetCheck(BST_UNCHECKED);
  }
  auxButton = (CButton *) GetDlgItem (IDC_P3);
  if( abPuertas[2]==true){
        auxButton->SetCheck(BST_CHECKED);
        bP3=TRUE;
  }else{
        auxButton->SetCheck(BST_UNCHECKED);
  }
  auxButton = (CButton *) GetDlgItem (IDC_P4);
  if( abPuertas[3]==true){
        bP4=TRUE;
        auxButton->SetCheck(BST_CHECKED);
  }else{
        auxButton->SetCheck(BST_UNCHECKED);
  }
  auxButton = (CButton *) GetDlgItem (IDC_P5);
  if( abPuertas[4]==true){
        bP5=TRUE;
        auxButton->SetCheck(BST_CHECKED);
  }else{
        auxButton->SetCheck(BST_UNCHECKED);
  }
  auxButton = (CButton *) GetDlgItem (IDC_P6);
  if( abPuertas[5]==true){
        bP6=TRUE;
        auxButton->SetCheck(BST_CHECKED);
  }else{
        auxButton->SetCheck(BST_UNCHECKED);
  }
  auxButton = (CButton *) GetDlgItem (IDC_P7);
  if( abPuertas[6]==true){
        bP7=TRUE;
        auxButton->SetCheck(BST_CHECKED);
  }else{
        auxButton->SetCheck(BST_UNCHECKED);
  }
  auxButton = (CButton *) GetDlgItem (IDC_P8);
  if( abPuertas[7]==true){
        bP8=TRUE;
```

```
                        auxButton->SetCheck(BST_CHECKED);
                  }else{
                        auxButton->SetCheck(BST_UNCHECKED);
                  }

                  /// Comporbar la restricción sobre el mes de "Agosto"
                  r.GetFieldValue ("Agosto", v);
                  agosto = (char*)v.bstrVal;
                  auxButton = (CButton *) GetDlgItem (IDC_AGOSTO);
                  if(   (ptr = strchr( agosto , '1' )) != NULL  ){
                        /// En este caso el periodo está restringido
                        auxButton->SetCheck(BST_CHECKED);
                        bAgosto=TRUE;
                  }else{
                        auxButton->SetCheck(BST_UNCHECKED);
                        bAgosto=FALSE;
                  }

                  /// Comporbar la restricción sobre "Fin de semana"
                  r.GetFieldValue ("Fin_de_semana", v);
                  agosto = (char*)v.bstrVal;
                  auxButton = (CButton *) GetDlgItem (IDC_FIN_SEMANA);
                  if(   (ptr = strchr( agosto , '1' )) != NULL  ){
                        /// En este caso el periodo está restringido
                        bFinSem=TRUE;
                        auxButton->SetCheck(BST_CHECKED);
                  }else{
                        auxButton->SetCheck(BST_UNCHECKED);
                        bFinSem=FALSE;
                  }

                  /// Ccomprobar si tiene acctivada la restricción
temporal
                  r.GetFieldValue ("Periodo_temporal", v);
                  agosto = (char*)v.bstrVal;
                  auxButton = (CButton *) GetDlgItem (IDC_INTERVALO);
                  if(   (ptr = strchr( agosto , '1' )) != NULL  ){
                        /// En este caso el periodo está restringido
                        /// Recoger datos de inicio y fin de restriccion
                        r.GetFieldValue ("FechaA", v);
                        csAux1=(char*)v.bstrVal;
                        char aux[3];
                        aux[2]='\0';
                        char* cAux = csAux1.GetBuffer( csAux1.GetLength()
);
                        aux[0]=cAux[0];
                        aux[1]=cAux[1];
                        StringDiaInicio=aux;
                        aux[0]=cAux[3];
                        aux[1]=cAux[4];
                        StringMesInicio=aux;
                        aux[0]=cAux[6];
                        aux[1]=cAux[7];
                        StringYearInicio=aux;
                        /// Ahora con la fecha de fin:
                        r.GetFieldValue ("FechaB", v);
                        csAux1=(char*)v.bstrVal;
                        cAux = csAux1.GetBuffer( csAux1.GetLength() );
                        aux[0]=cAux[0];
                        aux[1]=cAux[1];
```

```
                        StringDiaFin=aux;
                        aux[0]=cAux[3];
                        aux[1]=cAux[4];
                        StringMesFin=aux;
                        aux[0]=cAux[6];
                        aux[1]=cAux[7];
                        StringYearFin=aux;

                        auxButton->SetCheck(BST_CHECKED);
                        bCheckIntervalo = TRUE;
                  }else{
                        /// En este caso  "Intervalo" no está activado
                        auxButton->SetCheck(BST_UNCHECKED);
                        bCheckIntervalo = FALSE;
                  }///Fin del if() de Intervalo activado

                  iAlumnosPicadas = 1;



                  /// Representar los datos en el form
                  /// Mostrar el historial para este registro
                  stringHistorial= _T("");      // Limpiar Historial


                  r.Close();  /// Cerramos el vinculo a la tabla Alumnos
                  pDao->Open ("Picadas.mdb"); /// Crear vinculo a tabla
"Picadas"

                  sd = "SELECT * FROM Picadas WHERE dni =";//33
caracteres
                  sd = sd+ stringDni ;
                  r.Open(dbOpenDynaset,LPCTSTR(sd) , dbReadOnly );
                  r.MoveFirst();
                  while(      !r.IsEOF () ) {
                        r.GetFieldValue ( "Fecha" , v );
                        stringHistorial = stringHistorial +
(char*)v.bstrVal;
                        stringHistorial = stringHistorial + "  " ;
                        r.GetFieldValue ( "Hora" , v );
                        stringHistorial = stringHistorial +
(char*)v.bstrVal;
                        stringHistorial = stringHistorial + "  " ;
                        r.GetFieldValue ( "Puerta" , v );
                        stringHistorial = stringHistorial + "Acceso a
entrada nº " ;
                        stringHistorial = stringHistorial +
(char*)v.bstrVal;
                        stringHistorial = stringHistorial + "\r\n" ;
                        r.MoveNext();
                  }
                  r.Close();
                  pDao->Close();
                  delete pDao;
                  /// Actualizar los CEdit para mostrar los datos
                  UpdateData(false);
                  auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_HIS);
                  auxGroup->EnableWindow(TRUE);
                  auxEdit = (CEdit*)GetDlgItem (IDC_HISTORIAL);
```

```
                auxEdit->EnableWindow(TRUE);
                      //SetDlgItemText(IDC_HISTORIAL
,LPCTSTR(stringNombre));

                /// Si todo ha salido bien activo los
                /// botones [Eliminar Historial],[Eliminar Registro] y
[Cambiar datos]
                auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_HIS);
                auxButton->EnableWindow(TRUE);
                auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_REG);
                auxButton->EnableWindow(TRUE);
                auxButton = (CButton *) GetDlgItem (IDC_CHANGE_DATA);
                auxButton->EnableWindow(TRUE);
                auxButton = (CButton *) GetDlgItem (IDC_LIMPIAR);
                auxButton->EnableWindow(TRUE);
                /// Desactivo los votones [Buscar registro]  y [Añadir
Registro]
                auxButton = (CButton *) GetDlgItem (IDC_BUSCAR_REG);
                auxButton->EnableWindow(FALSE);
                auxButton = (CButton *) GetDlgItem (IDC_ADD_REG);
                auxButton->EnableWindow(FALSE);

           }catch(CDaoException* e ) {
                long number =e->m_pErrorInfo->m_lErrorCode;
                std::ostringstream stm ;
                stm << number ;
                const char* cstr = stm.str().c_str() ;
                CString tt = cstr;
                if(number == 3021){
                      if(iAlumnosPicadas == 0){
                            /// Este codigo indica que el registro no
existe en la base de datos
                            str = "El DNI introducido no esta
registrado en la base de datos";
                            AfxMessageBox(str);

                      }else{
                            /// En este caso ell error ha ocurrido al
no encontrar registros en la tabla picadas.
                            /// Actualizar los CEdit para mostrar los
datos
                            stringHistorial= _T("Este usuario no tiene
accesos registrados en su historial");
                            UpdateData(false);
                            auxGroup = (CWnd*)GetDlgItem
(IDC_GRUPO_HIS);
                            auxGroup->EnableWindow(FALSE);
                            auxEdit = (CEdit*)GetDlgItem
(IDC_HISTORIAL);
                            auxEdit->EnableWindow(TRUE);
                            /// Si todo ha salido bien activo los
                            /// botones [Eliminar Historial],[Eliminar
Registro] y [Cambiar datos]

                            auxButton = (CButton *) GetDlgItem
(IDC_ELIMINAR_REG);
                            auxButton->EnableWindow(TRUE);
                            auxButton = (CButton *) GetDlgItem
(IDC_CHANGE_DATA);
                            auxButton->EnableWindow(TRUE);
```

```
                              auxButton = (CButton *) GetDlgItem
(IDC_LIMPIAR);
                              auxButton->EnableWindow(TRUE);
                              /// Desactivo los votones [Buscar registro]
y [Añadir Registro]
                              auxButton = (CButton *) GetDlgItem
(IDC_BUSCAR_REG);
                              auxButton->EnableWindow(FALSE);
                              auxButton = (CButton *) GetDlgItem
(IDC_ADD_REG);
                              auxButton->EnableWindow(FALSE);

                        }
                        e->Delete();
                        r.Close();
                        pDao->Close();
                        delete pDao;
                  }else{
                        //SetDlgItemText(IDC_APELLIDO1_BOX,LPCTSTR("Algun
problema en BBDD"));
                        AfxMessageBox(e->m_pErrorInfo->m_strDescription);
                        e->Delete();
                        r.Close();
                        pDao->Close();
                        delete pDao;
                        }
            }



      }else{      /// Fin de:    if( iAux1 == 8 ){...
            /// Si el dni no cumple el formato se lanza un MessageBox y
se limpian los CEdit
            str= "DNI incorrecto. El DNI consta de 8 digitos sin
letra.\nIntroduzca de nuevo los datos.";
            AfxMessageBox (str);
            //SetDlgItemText(IDC_APELLIDO1_BOX ,LPCTSTR(
itoa(iAux1,sAux1,10)  ));

            /// Limpiar todos los CEdit;  "grupo restringido",
"datospersonales" y "historial"
            stringDni = _T("");          //Limpiar datos personales
            stringNombre= _T("");
            stringApellido1= _T("");
            stringApellido2= _T("");
      }
}


void CRegistroControldeAccesoView::OnBnClickedAddReg()
{
      CButton * auxButton;
      CEdit * auxEdit;
      CWnd * auxGroup;

      /// Primero comprobar que en el estado actual en que se encuentra
el boton
      if(bGuardarCancelar==false)
      {
```

```
            //tring s ((LPCTSTR) cs);
            int iAux1;
            char sAux1 [20];
            CString str;
            CDaoDatabase *pDao = new CDaoDatabase;
            CDaoRecordset r(pDao);
            /// En este caso el boton se encuentra en el estado:"añadir
registro"
            UpdateData(true);// Actualizar las CEdit;   form -> view.cpp

            /// Comprobar que los datos son correctos
            iAux1= strlen(stringDni);    //Obtener el numero de digitos
del campo DNI
            if( iAux1 == 8 ){
                  /// Comprobación correcta. Añadir registro en la base
de datos

                  //    Abrir la base d datos y trabajar con ella//
                  ////////////////////////////////////////////
                  pDao->Open ("Alumnos.mdb");
                  COleVariant v;
                  CDaoRecordset r(pDao);
                  int es_nuevo =0;
                  /// Construir la sentencia de busqueda en la base de
datos
                  CString sd = "SELECT * FROM Alumnos WHERE dni =";
      //33 caracteres
                  sd = sd+ stringDni ;
                  try{
                        /// Comprobar si ya existe ese dni en la base de
datos
                        r.Open(dbOpenDynaset,LPCTSTR(sd) , dbReadOnly );
                        r.GetFieldValue ("Nombre", v );
                  }catch(CDaoException* e ) {
                        long number =e->m_pErrorInfo->m_lErrorCode;
                        std::ostringstream stm ;
                        stm << number ;
                        const char* cstr = stm.str().c_str() ;
                        CString tt = cstr;
                        if(number == 3021){

      //SetDlgItemText(IDC_APELLIDO2_BOX,LPCTSTR("Dentro d la 3001"));
                              /// Este codigo indica que el registro no
existe previamente, por lo tanto pasamos a añadirlo
                              es_nuevo =1; /// Flags que indica  que el
registro es nuevo
                              e->Delete();
                              r.Close();
                        }else{

      SetDlgItemText(IDC_APELLIDO1_BOX,LPCTSTR("Algun problema en
BBDD"));
                              AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                              e->Delete();
                              r.Close();
                              pDao->Close();    //Para cerrar la conexion
con la BBDD y liberar memoria:
                              delete pDao;
                        }
```

```
                  }

                  if(es_nuevo == 1){
                        try{
                              r.Open(dbOpenDynaset,"SELECT * FROM Alumnos
",dbAppendOnly);
                              r.AddNew();
                              r.SetFieldValue ("DNI",
LPCTSTR(stringDni)); //-> variable contiene el //dato a insertar

      r.SetFieldValue("Nombre",LPCTSTR(stringNombre));

      r.SetFieldValue("Apellido1",LPCTSTR(stringApellido1));

      r.SetFieldValue("Apellido2",LPCTSTR(stringApellido2));
                              //r.SetFieldValue("Fecha",LPCTSTR(fechad));
                              //r.SetFieldValue("Hora",LPCTSTR(hora));

      //r.SetFieldValue("Puerta",LPCTSTR(stringPuerta));
                              r.Update ();
                              r.Close();


      ////////////////////////////////////////////////////////////////
////////////////
                              /// Una vez añadido a la base de datoso se
lanza un MessageBox y se limpian los CEdit
                              str = " Nuevo registro añadido con clave
principal:\n DNI: ";
                              str = str + stringDni ;
                              if( strlen(stringNombre)> 1 )
                              {
                                    str = str + "\n Nombre: " ;
                                    str = str + stringNombre  ;
                              }
                              if( strlen(stringApellido1)> 1 )
                              {
                                    str = str + "\n Apellido1: " ;
                                    str = str + stringApellido1  ;
                              }
                              if( strlen(stringApellido2)> 1 )
                              {
                                    str = str + "\n Apellido2: " ;
                                    str = str + stringApellido2  ;
                              }
                              str = str + "\n Acceso permitido a
entradas: <NINGUNA>\n Para dar permisos de acceso a este usuario pulse\n
[Buscar Registro] y [Modificar Datos de Registro]";
                              AfxMessageBox (str);
                        }catch(CDaoException* e){
                                    AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                                    e->Delete();
                                    pDao->Close();    //Para cerrar la
conexion con la BBDD y liberar memoria:
                                    delete pDao;
                                    r.Close();
                        }
```

```
                        pDao->Close();    //Para cerrar la conexion con
la BBDD y liberar memoria:
                        delete pDao;

                  /// Fin de: if(es_nuevo == 1){...
                  }else{
                        /// El hilo de ejecución llegará aquí si el DNI
ya existe.
                        str= "Error al generar nuevo registro.\nYa existe
registro con  DNI: ";
                        str= str + stringDni;
                        AfxMessageBox (str);
                        /// Cerrar la conexion con la BBDD y liberar
memoria:
                        r.Close();
                        pDao->Close();
                        delete pDao;

                        //Limpiar los CEdit "datos personales"
                        stringDni = _T("");
                        stringNombre= _T("");
                        stringApellido1= _T("");
                        stringApellido2= _T("");

                  }


            }else{
                  /// Si el dni no cumple el formato se lanza un
MessageBox y se limpian los CEdit
                  str= "DNI incorrecto. El DNI consta de 8 digitos sin
letra.\nIntroduzca de nuevo los datos.";
                  AfxMessageBox (str);
                  //SetDlgItemText(IDC_APELLIDO1_BOX ,LPCTSTR(
itoa(iAux1,sAux1,10)  ));

                  /// Limpiar todos los CEdit;  "grupo restringido",
"datospersonales" y "historial"
                  stringDni = _T("");          //Limpiar datos personales
                  stringNombre= _T("");
                  stringApellido1= _T("");
                  stringApellido2= _T("");
            }

      }else{

      ////////////////////////////////////////////////////////////////////
//

      ////////////////////////////////////////////////////////////////////
//
            /// En este caso el boton se encuentra en el estado:
"GUARDAR"  ////

      ////////////////////////////////////////////////////////////////////
/

      ////////////////////////////////////////////////////////////////////
/
            int continuar =1;
```

```
            char  cfechaA [11];     /// Variable para almacenar la fecha
de inicio de restriccion
            cfechaA [0]= '\0';
            char cfechaB[11] ;      /// Variable para almacinar la fecha
de  fin de restricción
            cfechaB[0]='\0';
            char  sPuerta[24] ;     /// Variable para almacenar las
puertas a las que tendra acceso un usuario
            sPuerta[0]='\0';
            int iAux1;
            char sAux1 [20];
            CString str;
            CDaoDatabase *pDao = new CDaoDatabase;
            CDaoRecordset r(pDao);
            pDao->Open ("Alumnos.mdb");


            UpdateData(true);// Actualizar las CEdit;   form -> view.cpp

            ///  Preparar los datos para introducirlos en la base de
datos
            /// 1º-> Comprobar si está activada la casilla "Restricción
Intervalo"
            /// 2º-> Comprobar si las fechas son validas
            /// 3º-> Construir el string sobre las puertas vaidas
            if(bCheckIntervalo==TRUE){
                  if(atoi(StringYearInicio)>atoi(StringYearFin) ){
                        str = "Imposible guardar datos. Periodos de
intervalo de restricción mal definidos.\n";
                        AfxMessageBox (str);
                        continuar = 0;
                  }else if(
(atoi(StringYearInicio)==atoi(StringYearFin))&(atoi(StringMesInicio)>atoi
(StringMesFin))   ){
                        str = "Imposible guardar datos. Periodos de
intervalo de restricción mal definidos.\n";
                        AfxMessageBox (str);
                        continuar=0;
                  }else if(
(atoi(StringYearInicio)==atoi(StringYearFin))&(atoi(StringMesInicio)==ato
i(StringMesFin))& ( atoi(StringDiaInicio)>atoi(StringDiaFin)) ){
                        str = "Imposible guardar datos. Periodos de
intervalo de restricción mal definidos.\n";
                        AfxMessageBox (str);
                        continuar =0;
                  }else{
                        /// En este caso las fechas son validas
                        /// Los datos sobre las fechas estaran en cfechaA
y cfechaB
                        cfechaA[0]= StringDiaInicio[0];
                        cfechaA[1]= StringDiaInicio[1];
                        cfechaA[2]= '/';
                        cfechaA[3]= StringMesInicio[0];
                        cfechaA[4]= StringMesInicio[1];
                        cfechaA[5]= '/';
                        cfechaA[6]= StringYearInicio[0];
                        cfechaA[7]= StringYearInicio[1];
                        cfechaA[8]= '\0';
                        cfechaB[0]= StringDiaFin[0];
                        cfechaB[1]= StringDiaFin[1];
```

```
                        cfechaB[2]= '/';
                        cfechaB[3]= StringMesFin[0];
                        cfechaB[4]= StringMesFin[1];
                        cfechaB[5]= '/';
                        cfechaB[6]= StringYearFin[0];
                        cfechaB[7]= StringYearFin[1];
                        cfechaB[8]= '\0';
                        continuar =1;  /// Indicador de que los datos son
correctos
                  }
            }else{
                  /// En este caso la casilla "Intervalo temporal" está
desactivada
                  cfechaA[0]='\0';
                  cfechaB[0]='\0';
            }

            if(continuar==1){
                  if(bP1==TRUE){
                        strcat(sPuerta ,"1,");
                  }
                  if(bP2==TRUE){
                        strcat(sPuerta ,"2,");
                  }
                  if(bP3==TRUE){
                        strcat(sPuerta ,"3,");
                  }
                  if(bP4==TRUE){
                        strcat(sPuerta ,"4,");
                  }
                  if(bP5==TRUE){
                        strcat(sPuerta ,"5,");
                  }
                  if(bP6==TRUE){
                        strcat(sPuerta ,"6,");
                  }
                  if(bP7==TRUE){
                        strcat(sPuerta ,"7,");
                  }
                  if(bP8==TRUE){
                        strcat(sPuerta ,"8");
                  }

                  CString sd = "SELECT * FROM Alumnos WHERE dni =";
      //33 caracteres
                  sd = sd+ stringDni ;
                  try{
                        //dbDenyWrite Otros usuarios no pueden modificar
ni agregar registros.
                        // dbAppendOnly
                        r.Open(dbOpenDynaset,LPCTSTR(sd),dbDenyWrite);
                        r.Edit();
                        r.SetFieldValue("Nombre",LPCTSTR(stringNombre));

      r.SetFieldValue("Apellido1",LPCTSTR(stringApellido1));

      r.SetFieldValue("Apellido2",LPCTSTR(stringApellido2));
                        r.SetFieldValue("Puerta",LPCTSTR(sPuerta));
                        if(bAgosto==TRUE)
                              r.SetFieldValue("Agosto",LPCTSTR("1"));
```

```
                        else
                              r.SetFieldValue("Agosto",LPCTSTR("0"));
                        if(bFinSem==TRUE)

      r.SetFieldValue("Fin_de_semana",LPCTSTR("1"));
                        else

      r.SetFieldValue("Fin_de_semana",LPCTSTR("0"));
                        if(bCheckIntervalo==TRUE)

      r.SetFieldValue("Periodo_temporal",LPCTSTR("1"));
                        else

      r.SetFieldValue("Periodo_temporal",LPCTSTR("0"));
                        r.SetFieldValue("FechaA",LPCTSTR(cfechaA));
                        r.SetFieldValue("FechaB",LPCTSTR(cfechaB));

                        r.Update ();
                        r.Close();
                        pDao->Close();    //Para cerrar la conexion con
la BBDD y liberar memoria:
                        delete pDao;


                  }catch(CDaoException* e){
                              AfxMessageBox(e->m_pErrorInfo-
>m_strDescription);
                              e->Delete();
                              r.Close();
                              pDao->Close();    //Para cerrar la conexion
con la BBDD y liberar memoria:
                              delete pDao;
                  }


                  /// Comprobar si el historial está limpio para
activar/desactivar el boton ELIMINAR_HIS
                  if(         57 == stringHistorial.GetLength() ){
                        auxButton = (CButton *) GetDlgItem
(IDC_ELIMINAR_HIS);
                        auxButton->EnableWindow(FALSE);

                  }else{
                        auxButton = (CButton *) GetDlgItem
(IDC_ELIMINAR_HIS);
                        auxButton->EnableWindow(TRUE);
                  }


                  ////SetDlgItemText(IDC_APELLIDO2_BOX
,LPCTSTR("Validar,OK"));

                  /// Se ha realizado la grabación de la base de datos
                  /// Los botones [GUARDAR],[CANCELAR]->[AÑADIR
REGISTRO][MODIFICAR REGISTRO]
                  bGuardarCancelar = false;
                  SetDlgItemText(IDC_ADD_REG ,LPCTSTR("Añadir
Registro"));
                  SetDlgItemText(IDC_CHANGE_DATA ,LPCTSTR("Modificar
Datos de Registro"));
```

```
                /// Activar los botones [Limpiar], [modificar datos],
[Eliminar]
                auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_REG);
                auxButton->EnableWindow(TRUE);
                auxButton = (CButton *) GetDlgItem (IDC_CHANGE_DATA);
                auxButton->EnableWindow(TRUE);
                auxButton = (CButton *) GetDlgItem (IDC_LIMPIAR);
                auxButton->EnableWindow(TRUE);
                auxButton = (CButton *) GetDlgItem (IDC_BUSCAR_REG);
                auxButton->EnableWindow(FALSE);
                auxButton = (CButton *) GetDlgItem (IDC_ADD_REG);
                auxButton->EnableWindow(FALSE);

                /// Desactivar las casillas de verificacion de Puertas
con acceso permitido
                auxButton = (CButton *) GetDlgItem (IDC_P1);
                auxButton->EnableWindow(FALSE);
                auxButton = (CButton *) GetDlgItem (IDC_P2);
                auxButton->EnableWindow(FALSE);
                auxButton = (CButton *) GetDlgItem (IDC_P3);
                auxButton->EnableWindow(FALSE);
                auxButton = (CButton *) GetDlgItem (IDC_P4);
                auxButton->EnableWindow(FALSE);
                auxButton = (CButton *) GetDlgItem (IDC_P5);
                auxButton->EnableWindow(FALSE);
                auxButton = (CButton *) GetDlgItem (IDC_P6);
                auxButton->EnableWindow(FALSE);
                auxButton = (CButton *) GetDlgItem (IDC_P7);
                auxButton->EnableWindow(FALSE);
                auxButton = (CButton *) GetDlgItem (IDC_P8);
                auxButton->EnableWindow(FALSE);

                /// Desactivar los grupos "restringidos", "historial",
"inicio", "fin", "Entradas"
                auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_RES);
                auxGroup->EnableWindow(FALSE);
                auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_HIS);
                auxGroup->EnableWindow(FALSE);
                auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_INI);
                auxGroup->EnableWindow(FALSE);
                auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_FIN);
                auxGroup->EnableWindow(FALSE);
                auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_ENT);
                auxGroup->EnableWindow(FALSE);

                /// Desactivar las etiquetas "Dia", "Mes", "Año"...
Static Text
                auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_INICIO);
                auxGroup->EnableWindow(FALSE);
                auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_FIN);
                auxGroup->EnableWindow(FALSE);
                auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_INICIO);
                auxGroup->EnableWindow(FALSE);
                auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_FIN);
                auxGroup->EnableWindow(FALSE);
                auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_INICIO);
                auxGroup->EnableWindow(FALSE);
                auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_FIN);
                auxGroup->EnableWindow(FALSE);
```

```
                    /// Desactivar las casillas de verificación
                    auxButton = (CButton *) GetDlgItem (IDC_FIN_SEMANA);
                    auxButton->EnableWindow(FALSE);
                    auxButton = (CButton *) GetDlgItem (IDC_AGOSTO);
                    auxButton->EnableWindow(FALSE);
                    auxButton = (CButton *) GetDlgItem (IDC_INTERVALO);
                    auxButton->EnableWindow(FALSE);

                    /// Desactivar los ComboBox "inicio" y "fin" de
restricción
                    auxEdit = (CEdit*)GetDlgItem (IDC_DIA_INICIO);
                    auxEdit->EnableWindow(FALSE);
                    auxEdit = (CEdit*)GetDlgItem (IDC_MES_INICIO);
                    auxEdit->EnableWindow(FALSE);
                    auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_INICIO);
                    auxEdit->EnableWindow(FALSE);
                    auxEdit = (CEdit*)GetDlgItem (IDC_DIA_FIN);
                    auxEdit->EnableWindow(FALSE);
                    auxEdit = (CEdit*)GetDlgItem (IDC_MES_FIN);
                    auxEdit->EnableWindow(FALSE);
                    auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_FIN);
                    auxEdit->EnableWindow(FALSE);

                    /// Desactivar los ComboBox Nombre, Apellido1 y
Apellido2
                    auxEdit = (CEdit*)GetDlgItem (IDC_NOMBRE_BOX);
                    auxEdit->EnableWindow(FALSE);
                    auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO1_BOX);
                    auxEdit->EnableWindow(FALSE);
                    auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO2_BOX);
                    auxEdit->EnableWindow(FALSE);

            }else{



                    SetDlgItemText(IDC_ADD_REG ,LPCTSTR("Añadir
Registro"));
                    SetDlgItemText(IDC_CHANGE_DATA ,LPCTSTR("Modificar
Datos de Registro"));

                    bGuardarCancelar = true;

            }

      }/// Fin de if(bGuardarCancelar)else{
}

void CRegistroControldeAccesoView::OnBnClickedEliminarHis()
{
      if( AfxMessageBox("Está usted seguro de querer eliminar el
historial de accesos de este usuario manera permanente",
MB_OKCANCEL|MB_ICONWARNING ) ==IDOK){
            ///MessageBox("Ha pulsado ok");


            CString csAux1;
            CString puerta;
            CString agosto;
            CString str;
```

```
            CButton * auxButton;
            CEdit * auxEdit;
            CWnd * auxGroup;
            int seguir=0;
            CString sd;
            CDaoDatabase *pDao = new CDaoDatabase;
            CDaoRecordset r(pDao);

            pDao->Open ("Alumnos.mdb");
            // TODO: Add your control notification handler code here
            pDao->Open ("Picadas.mdb"); /// Crear vinculo a tabla
"Picadas"

            sd = "SELECT * FROM Picadas WHERE dni =";//33 caracteres
            sd = sd+ stringDni ;
            try{
                  r.Open(dbOpenDynaset,LPCTSTR(sd),dbDenyWrite);

                  r.MoveFirst();
                  while(      !r.IsEOF () ) {
                        r.Delete();
                        r.MoveNext();

                  }
                  r.Close();
                  seguir=1;
                  pDao->Close();
            }catch(CDaoException* e ) {
                  long number =e->m_pErrorInfo->m_lErrorCode;
                  AfxMessageBox(e->m_pErrorInfo->m_strDescription);
                  e->Delete();
                  r.Close();
                  pDao->Close();
                  delete pDao;
                  if(number == 3021){
                        /// Significa que el registro no tiene "picadas"
en tabla
                        stringHistorial= _T("Este usuario no tiene
accesos registrados en su historial");
                  }
            }
            if(seguir==1){
                  stringHistorial= _T("Este usuario no tiene accesos
registrados en su historial");
                  UpdateData(false);
                  /// Activar los botones [Limpiar], [modificar datos],
[Eliminar]
                  auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_HIS);
                  auxButton->EnableWindow(FALSE);
                  auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_REG);
                  auxButton->EnableWindow(TRUE);

            }
      }
}
```

```
void CRegistroControldeAccesoView::OnBnClickedEliminarReg()
{
	/// Prueba de mesagebox

	if( AfxMessageBox("Está usted seguro de querer eliminar este
registro de manera permanente", MB_OKCANCEL|MB_ICONWARNING ) ==IDOK){
		///MessageBox("Ha pulsado ok");

		CButton * auxButton;
		CEdit * auxEdit;
		CWnd * auxGroup;

		CString sd;
		CDaoDatabase *pDao = new CDaoDatabase;
		CDaoRecordset r(pDao);


		/// 1º-> Comprobar si el registro tiene "Picadas" registradas
y en caso afirmativo eliminarlas
		///   2º-> Eliminar el registro del alumno con el DNI
seleccionado de la tabla "alumnos"
		pDao->Open ("Alumnos.mdb");
		pDao->Open ("Picadas.mdb"); /// Crear vinculo a tabla
"Picadas"
		sd = "SELECT * FROM Picadas WHERE dni =";//33 caracteres
		sd = sd+ stringDni ;
		try{
			r.Open(dbOpenDynaset,LPCTSTR(sd),dbDenyWrite);
			r.MoveFirst();
			while(	!r.IsEOF () ) {
				r.Delete();
				r.MoveNext();

			}
			r.Close();

		}catch(CDaoException* e ) {
			long number =e->m_pErrorInfo->m_lErrorCode;
			e->Delete();
			r.Close();
			if(number == 3021){
				/// Significa que el registro no tiene "picadas"
en tabla
			}
		}

		///   2º-> Eliminar el registro del alumno con el DNI
seleccionado de la tabla "alumnos"
		sd = "SELECT * FROM Alumnos WHERE dni =";//33 caracteres
		sd = sd+ stringDni ;
		try{
			//dbDenyWrite Otros usuarios no pueden modificar ni
agregar registros.
			r.Open(dbOpenDynaset,LPCTSTR(sd),dbDenyWrite);
			r.Delete();
			pDao->Close();
			delete pDao;
			/// Desactivar el boton [Eliminar Registro]
			auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_REG);
```

```
			auxButton->EnableWindow(FALSE);
		}catch(CDaoException* e ) {
			long number =e->m_pErrorInfo->m_lErrorCode;
			AfxMessageBox(e->m_pErrorInfo->m_strDescription);
			e->Delete();
			r.Close();
			pDao->Close();
			delete pDao;
			if(number == 3021){
				AfxMessageBox(e->m_pErrorInfo->m_strDescription);
			}
		}
		/// Limpiar todos los CEdit;  "grupo restringido",
"datospersonales" y "historial"
		StringYearInicio = _T(" ");   // Limpiar restringciones
		StringDiaInicio = _T(" ");
		StringMesInicio = _T(" ");
		StringDiaFin = _T(" ");
		StringMesFin = _T(" ");
		StringYearFin = _T(" ");

		stringNombre= _T("");
		stringApellido1= _T("");
		stringApellido2= _T("");
		stringHistorial= _T("");      // Limpiar Historial
		sd="El registro con DNI= ";
		sd=sd+stringDni;
		sd =sd+" ha sido eliminado permanentemente.";
		AfxMessageBox(sd);
		// Actualizar los CEdit;  view.cpp -> form
		stringDni = _T("");           //Limpiar datos personales
		/// Limpiar y desactivar las casillas de verificacion de
Puertas con acceso permitido
		auxButton = (CButton *) GetDlgItem (IDC_P1);
		auxButton-> SetCheck( BST_UNCHECKED );
		bP1=FALSE;
		auxButton->EnableWindow(FALSE);
		auxButton = (CButton *) GetDlgItem (IDC_P2);
		auxButton-> SetCheck( BST_UNCHECKED );
		bP2=FALSE;
		auxButton->EnableWindow(FALSE);
		auxButton = (CButton *) GetDlgItem (IDC_P3);
		auxButton-> SetCheck( BST_UNCHECKED );
		bP3=FALSE;
		auxButton->EnableWindow(FALSE);
		auxButton = (CButton *) GetDlgItem (IDC_P4);
		auxButton-> SetCheck( BST_UNCHECKED );
		bP4=FALSE;
		auxButton->EnableWindow(FALSE);
		auxButton = (CButton *) GetDlgItem (IDC_P5);
		auxButton-> SetCheck( BST_UNCHECKED );
		bP5=FALSE;
		auxButton->EnableWindow(FALSE);
		auxButton = (CButton *) GetDlgItem (IDC_P6);
		auxButton-> SetCheck( BST_UNCHECKED );
		bP6=FALSE;
		auxButton->EnableWindow(FALSE);
		auxButton = (CButton *) GetDlgItem (IDC_P7);
		auxButton-> SetCheck( BST_UNCHECKED );
		bP7=FALSE;
```

```
            auxButton->EnableWindow(FALSE);
            auxButton = (CButton *) GetDlgItem (IDC_P8);
            auxButton-> SetCheck( BST_UNCHECKED );
            bP8=FALSE;
            auxButton->EnableWindow(FALSE);
            /// Limpiar las casillas de verificacion del "Grupo
restringido"
            bCheckIntervalo = FALSE;
            bAgosto=FALSE;
            bFinSem=FALSE;
            auxButton = (CButton *) GetDlgItem (IDC_INTERVALO);
            auxButton-> SetCheck( BST_UNCHECKED );
            auxButton = (CButton *) GetDlgItem (IDC_FIN_SEMANA);
            auxButton-> SetCheck( BST_UNCHECKED );
            auxButton = (CButton *) GetDlgItem (IDC_AGOSTO);
            auxButton-> SetCheck( BST_UNCHECKED );
            /// Limpiar todos los CEdit;  "grupo restringido",
"datospersonales" y "historial"
      StringYearInicio = _T(" ");   // Limpiar restringciones
      StringDiaInicio = _T(" ");
      StringMesInicio = _T(" ");
      StringDiaFin = _T(" ");
      StringMesFin = _T(" ");
      StringYearFin = _T(" ");
      stringDni = _T("");           //Limpiar datos personales
      stringNombre= _T("");
      stringApellido1= _T("");
      stringApellido2= _T("");
      stringHistorial= _T("");      // Limpiar Historial

      // Actualizar los CEdit;  view.cpp -> form
      //UpdateData(false);

      /// El campo DNI se podrá escribir
      auxEdit = (CEdit*)GetDlgItem (IDC_DNI_BOX);
      auxEdit->SetReadOnly(FALSE);

      /// Limpiar las casillas de verificacion del "Grupo restringido"
      bCheckIntervalo = FALSE;
      auxButton = (CButton *) GetDlgItem (IDC_INTERVALO);
      auxButton-> SetCheck( BST_UNCHECKED );
      auxButton = (CButton *) GetDlgItem (IDC_FIN_SEMANA);
      auxButton-> SetCheck( BST_UNCHECKED );
      auxButton = (CButton *) GetDlgItem (IDC_AGOSTO);
      auxButton-> SetCheck( BST_UNCHECKED );

      /// Limpiar y desactivar las casillas de verificacion de Puertas
con acceso permitido
      auxButton = (CButton *) GetDlgItem (IDC_P1);
      auxButton-> SetCheck( BST_UNCHECKED );
      auxButton->EnableWindow(FALSE);
      auxButton = (CButton *) GetDlgItem (IDC_P2);
      auxButton-> SetCheck( BST_UNCHECKED );
      auxButton->EnableWindow(FALSE);
      auxButton = (CButton *) GetDlgItem (IDC_P3);
      auxButton-> SetCheck( BST_UNCHECKED );
      auxButton->EnableWindow(FALSE);
      auxButton = (CButton *) GetDlgItem (IDC_P4);
      auxButton-> SetCheck( BST_UNCHECKED );
      auxButton->EnableWindow(FALSE);
```

```
	auxButton = (CButton *) GetDlgItem (IDC_P5);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_P6);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_P7);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_P8);
	auxButton-> SetCheck( BST_UNCHECKED );
	auxButton->EnableWindow(FALSE);



	/// Desactivar los grupos "restringidos", "historial", "inicio",
"fin", "Entradas"
	auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_RES);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_HIS);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_INI);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_GROUP_FIN);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_GRUPO_ENT);
	auxGroup->EnableWindow(FALSE);

	/// Activar el grupo y EL CEdit "datos personales", "DNI";
Desactivar los CEdit nombre y apellidos
	auxGroup = (CWnd*)GetDlgItem (IDC_DATOS_PERSONALES);
	auxGroup->EnableWindow(TRUE);
	auxEdit = (CEdit*)GetDlgItem (IDC_DNI_BOX);
	auxEdit->EnableWindow(TRUE);
	auxEdit = (CEdit*)GetDlgItem (IDC_NOMBRE_BOX);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO1_BOX);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_APELLIDO2_BOX);
	auxEdit->EnableWindow(FALSE);

	/// Desactivar las casillas de verificación
	auxButton = (CButton *) GetDlgItem (IDC_FIN_SEMANA);
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_AGOSTO);
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_INTERVALO);
	auxButton->EnableWindow(FALSE);

	/// Desactivar los ComboBox "inicio" y "fin" de restricción
	auxEdit = (CEdit*)GetDlgItem (IDC_DIA_INICIO);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_MES_INICIO);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_INICIO);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_DIA_FIN);
	auxEdit->EnableWindow(FALSE);
	auxEdit = (CEdit*)GetDlgItem (IDC_MES_FIN);
	auxEdit->EnableWindow(FALSE);
```

```
	auxEdit = (CEdit*)GetDlgItem (IDC_YEAR_FIN);
	auxEdit->EnableWindow(FALSE);

	/// Desactivar las etiquetas "Dia", "Mes", "Año"...  Static Text
	auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_INICIO);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_E_DIA_FIN);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_INICIO);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_E_MES_FIN);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_INICIO);
	auxGroup->EnableWindow(FALSE);
	auxGroup = (CWnd*)GetDlgItem (IDC_E_YEAR_FIN);
	auxGroup->EnableWindow(FALSE);

	/// ACTIVAR/DESACTIVAR botones de la aplicacion
	/// Desactivar los botones  [Eliminar Historial],[Eliminar
Registro] y [Cambiar datos]
	auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_HIS);
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_ELIMINAR_REG);
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_CHANGE_DATA);
	auxButton->EnableWindow(FALSE);
	auxButton = (CButton *) GetDlgItem (IDC_LIMPIAR);
	auxButton->EnableWindow(TRUE);
	auxButton = (CButton *) GetDlgItem (IDC_BUSCAR_REG);
	auxButton->EnableWindow(TRUE);
	auxButton = (CButton *) GetDlgItem (IDC_ADD_REG);
	auxButton->EnableWindow(TRUE);



		UpdateData(false);

	}/// Fin el if del AFXMESSAGEBOX
}



/// Prueba de mesagebox
	/*
	if( AfxMessageBox("ESto es una Prueba", MB_OKCANCEL|MB_ICONSTOP )
==IDOK){
		MessageBox("Ha pulsado ok");
	}
	*/
```

stdafx.cpp: archivo de código fuente que contiene solo las inclusiones estándar

```
// Registro - Control de Acceso.pch será el encabezado precompilado
// stdafx.obj contendrá la información de tipos precompilada

#include "stdafx.h"
```

```
#include <afxdao.h>
```

MainFrm.h: interface de la clase CMainFrame

```
// MainFrm.h
//


#pragma once
class CMainFrame : public CFrameWnd
{

protected: // Crear sólo a partir de serialización
	CMainFrame();
	DECLARE_DYNCREATE(CMainFrame)

// Atributos
public:

// Operaciones
public:

// Reemplazos
public:
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

// Implementación
public:
	virtual ~CMainFrame();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // Miembros incrustados de la barra de control
	CStatusBar  m_wndStatusBar;
	CToolBar    m_wndToolBar;

// Funciones de asignación de mensajes generadas
protected:
	afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
	DECLARE_MESSAGE_MAP()
};
```

Registro-Control de Acceso.h: Es el fichero de cabecera principal de la aplicación. Incluye otros ficheros de cabecera necsarios para la aplicación, como resource.h y declara la clase aplicación CRegistroControldeAccesoApp.

```
// Registro - Control de Acceso.h: archivo de encabezado principal para
la aplicación Registro - Control de Acceso
//
#pragma once

#ifndef __AFXWIN_H__
	#error incluye 'stdafx.h' antes de incluir este archivo para PCH
#endif
```

```
#include "resource.h"       // Símbolos principales


// CRegistroControldeAccesoApp:
// Consulte la sección Registro - Control de Acceso.cpp para obtener
información sobre la implementación de esta clase
//

class CRegistroControldeAccesoApp : public CWinApp
{
public:
      CRegistroControldeAccesoApp();


// Reemplazos
public:
      virtual BOOL InitInstance();

// Implementación
      afx_msg void OnAppAbout();
      DECLARE_MESSAGE_MAP()
};

extern CRegistroControldeAccesoApp theApp;
```

Registro-Control_AccesoDoc.h: Declaracion de la clase CRegistroControldeAccesoDoc

```
// Registro - Control de AccesoDoc.h: interfaz de la clase
CRegistroControldeAccesoDoc
//


#pragma once

class CRegistroControldeAccesoDoc : public CDocument
{
protected: // Crear sólo a partir de serialización
      CRegistroControldeAccesoDoc();
      DECLARE_DYNCREATE(CRegistroControldeAccesoDoc)

// Atributos
public:

// Operaciones
public:

// Reemplazos
      public:
      virtual BOOL OnNewDocument();
      virtual void Serialize(CArchive& ar);

// Implementación
public:
      virtual ~CRegistroControldeAccesoDoc();
#ifdef _DEBUG
      virtual void AssertValid() const;
      virtual void Dump(CDumpContext& dc) const;
#endif
```

```
protected:

// Funciones de asignación de mensajes generadas
protected:
	DECLARE_MESSAGE_MAP()
};
```

Registro - Control de AccesoView.h: Contiene la declaracion de la clase CRegistroControldeAccesoView.

```
// Registro - Control de AccesoView.h: interfaz de la clase
CRegistroControldeAccesoView
//


#pragma once
#include "afxwin.h"


class CRegistroControldeAccesoView : public CFormView
{
protected: // Crear sólo a partir de serialización
	CRegistroControldeAccesoView();
	DECLARE_DYNCREATE(CRegistroControldeAccesoView)

public:
	enum{ IDD = IDD_REGISTROCONTROLDEACCESO_FORM };

// Atributos
public:
	CRegistroControldeAccesoDoc* GetDocument() const;

// Operaciones
public:

// Reemplazos
	public:
virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:
	virtual void DoDataExchange(CDataExchange* pDX);    //
Compatibilidad con DDX/DDV
	virtual void OnInitialUpdate(); // Se llama la primera vez después
de la construcción

// Implementación
public:
	virtual ~CRegistroControldeAccesoView();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Funciones de asignación de mensajes generadas
protected:
	DECLARE_MESSAGE_MAP()
public:
```

```
	//afx_msg void OnStnClickedNombre();
	//afx_msg void OnStnClickedEDiaInicio();
	//afx_msg void OnCbnSelchangeDiaInicio();
	//afx_msg void OnBnClickedButton4();

	afx_msg void OnCbnSelchangeCombo1();

	afx_msg void OnCbnSelchangeCombo2();


protected:

public:


	CString StringYearInicio;
	CString StringDiaInicio;
	CString StringMesInicio;
	CString StringDiaFin;
	CString StringMesFin;
	CString StringYearFin;

	afx_msg void OnBnClickedIntervalo();
	BOOL bCheckIntervalo;
	afx_msg void OnBnClickedChangeData();

	afx_msg void OnBnClickedLimpiar();
	CString stringDni;
	CString stringNombre;
	CString stringApellido1;
	CString stringApellido2;
	CString stringHistorial;

	///CString stringPuerta;
	///CString StringAgosto;


	afx_msg void OnBnClickedBuscarReg();
	afx_msg void OnBnClickedButton5();
	afx_msg void OnBnClickedAddReg();

	BOOL bAgosto;
	BOOL bFinSem;
	bool bGuardarCancelar;
	BOOL bP1;
	BOOL bP2;
	BOOL bP3;
	BOOL bP4;
	BOOL bP5;
	BOOL bP6;
	BOOL bP7;
	BOOL bP8;
	afx_msg void OnBnClickedEliminarHis();
	afx_msg void OnBnClickedEliminarReg();
};

#ifndef _DEBUG  // Versión de depuración en Registro - Control de
AccesoView.cpp
```

```
inline CRegistroControldeAccesoDoc*
CRegistroControldeAccesoView::GetDocument() const
   { return reinterpret_cast<CRegistroControldeAccesoDoc*>(m_pDocument);
}
#endif
```

Resource.h: Este es el fichero de cabecera estándar que define los identificadores de los recursos. El entorno de desarrollo de Visual C++ se encarga de leer y actualizar estos este fichero.

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Registro - Control de Acceso.rc
//
#define IDD_ABOUTBOX                    100
#define IDP_OLE_INIT_FAILED             100
#define IDD_REGISTROCONTROLDEACCESO_FORM 101
#define IDR_MAINFRAME                   128
#define IDR_MFC_ICON                    128
#define IDR_RegistroControlTYPE         129
#define IDC_APELLIDO1                   1000
#define IDC_NOMBRE                      1001
#define IDC_EDIT1                       1002
#define IDC_NOMBRE_BOX                  1002
#define IDC_APELLIDO2                   1003
#define IDC_DNI                         1004
#define IDC_EDIT2                       1005
#define IDC_DNI_BOX                     1005
#define IDC_EDIT3                       1006
#define IDC_APELLIDO1_BOX               1006
#define IDC_EDIT4                       1007
#define IDC_APELLIDO2_BOX               1007
#define IDC_DATOS_PERSONALES            1008
#define IDC_CHECK1                      1009
#define IDC_FIN_SEMANA                  1009
#define IDC_CHECK2                      1010
#define IDC_AGOSTO                      1010
#define IDC_CHECK3                      1011
#define IDC_INTERVALO                   1011
#define IDC_YEAR_INICIO                 1016
#define IDC_E_DIA_INICIO                1017
#define IDC_E_MES_INICIO                1018
#define IDC_E_YEAR_INICIO               1019
#define IDC_DIA_FIN                     1021
#define IDC_MES_FIN                     1022
#define IDC_COMBO6                      1023
#define IDC_YEAR_FIN                    1023
#define IDC_E_DIA_FIN                   1024
#define IDC_COMBO1                      1025
#define IDC_DIA_INICIO                  1025
#define IDC_COMBO2                      1026
#define IDC_MES_INICIO                  1026
#define IDC_BUTTON1                     1027
#define IDC_LIMPIAR                     1027
#define IDC_BUTTON2                     1028
#define IDC_BUSCAR_REG                  1028
#define IDC_BUTTON3                     1029
#define IDC_ELIMINAR_HIS                1029
#define IDC_BUTTON4                     1030
#define IDC_ELIMINAR_REG                1030
```

```
#define IDC_BUTTON5                     1031
#define IDC_ADD_REG                     1031
#define IDC_BUTTON6                     1032
#define IDC_CHANGE_DATA                 1032
#define IDC_EDIT5                       1033
#define IDC_HISTORIAL                   1033
#define IDC_GRUPO_RES                   1034
#define IDC_GROUP_INI                   1035
#define IDC_GROUP_FIN                   1036
#define IDC_GRUPO_HIS                   1037
#define IDC_E_YEAR_FIN                  1038
#define IDC_E_MES_FIN                   1039
#define IDC_P1                          1048
#define IDC_P2                          1049
#define IDC_P3                          1050
#define IDC_P4                          1051
#define IDC_P5                          1052
#define IDC_P6                          1053
#define IDC_P7                          1054
#define IDC_CHECK11                     1055
#define IDC_P8                          1055
#define IDC_GRUPO_ENT                   1056

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        131
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         1057
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Stdafx.h: Este fichero junto con stdafx.cpp es el encargado de construir el fichero de cabecera precompiladdo denominado stdafx.psh y un fichero precompilado de tipos denominado stdafx.obj.

```
// stdafx.h: archivo de inclusión para archivos de inclusión estándar del
sistema,
// o archivos de inclusión específicos del proyecto utilizados
frecuentemente,
// pero cambiados rara vez

#pragma once

#include <afxdao.h>
#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN         // Excluir material rara vez utilizado de
encabezados de Windows
#endif

// Modificar las siguientes secciones define si su objetivo es una
plataforma distinta a las especificadas a continuación.
// Consulte la referencia MSDN para obtener la información más reciente
sobre los valores correspondientes a diferentes plataformas.
#ifndef WINVER                       // Permitir el uso de características
específicas de Windows 95 y Windows NT 4 o posterior.
#define WINVER 0x0400        // Cambiar para establecer el valor
apropiado para Windows 98 y Windows 2000 o posterior.
```

```
#endif

#ifndef _WIN32_WINNT          // Permitir el uso de características
específicas de Windows NT 4 o posterior.
#define _WIN32_WINNT 0x0400         // Cambiar para establecer el valor
apropiado para Windows 98 y Windows 2000 o posterior.
#endif

#ifndef _WIN32_WINDOWS        // Permitir el uso de características
específicas de Windows 98 o posterior.
#define _WIN32_WINDOWS 0x0410 // Cambiar para establecer el valor
apropiado para Windows Me o posterior.
#endif

#ifndef _WIN32_IE             // Permitir el uso de características
específicas de Internet Explorer 4.0 o posterior.
#define _WIN32_IE 0x0400      // Cambiar para establecer el valor
apropiado para IE 5.0 o posterior.
#endif

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS     // Algunos constructores
CString serán explícitos

// Desactiva la ocultación de MFC para algunos mensajes de advertencia
comunes y, muchas veces, omitidos de forma consciente
#define _AFX_ALL_WARNINGS

#include <afxwin.h>          // Componentes principales y estándar de MFC
#include <afxext.h>          // Extensiones de MFC
#include <afxdisp.h>         // Clases de automatización de MFC

///Esta LA AÑADO YO
#include <windows.h>

#include <afxdtctl.h>         // Compatibilidad MFC para controles
comunes de Internet Explorer 4
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                 // Compatibilidad MFC para controles
comunes de Windows
#endif // _AFX_NO_AFXCMN_SUPPORT



#include <afxdb.h>         // ODBC
#include <atlbase.h>
#include <afxoledb.h>
#include <atlplus.h>
#include <afxdao.h>
/// Las siguientes las añado yo
```

9.3 Codigo del Microcontrolador.

```
; RA1 entrada
; El resto salidas
; El Puerto B será el encargado de mostrar la puerta a abrir 1-8

; Configuración del puerto COM:
; Bits por segundo = 4800 baudios
; Bits de datos = 8
; Paridad: ninguna
; Bits de parada = 2
; Control de flujo: ninguno
; Tiempo por bit:
; 4800 baudios -> 1/4800 seg -> 208,3 microsegundos


LIST P=PIC16F84A  ; Pic a usar

#INCLUDE <P16F84A.INC>

  __CONFIG _CP_OFF&_PWRTE_ON&_WDT_OFF&_HS_OSC ;


nBITS_RS232 EQU   0x0C        ; Puntero del mensaje
DatoRecibido      EQU   0x0D        ; Registro de transmisión
BITS        EQU   0x0E        ; Número de bits de datos
CNTMSEC           EQU   0x0F        ; Número de milisegundos de retardo
W_Temp            equ   0x10  ; Registro para guardar temporalmente W.-
STATUS_Temp equ   0x11  ; Registro para guardar temporalmente STATUS
c3s         equ   0x12  ; Contador para temporizar 3 segundos



      org   0x00  ;
      goto  inicio;

      org   0x05  ;
inicio


      BSF   STATUS,RP0  ; Banco de memoria 1
      movlw B'00010'    ;
      MOVWF TRISA       ; Ra1 entrada
      movlw B'00000000' ;
      MOVWF TRISB       ; PuertoB=Salida

      BCF   STATUS,RP0  ; Activa el banco de memoria 0.

;-------------  A partir de aqui empieza la programacion

;-- Permanecer testeando la entrada RA1 y cuando su estado cambie y
tengamos un "0" este será el bit de inicio de trama

testearRA1
      btfsc PORTA,1                 ; Comprobar RA1, Salta si RA1=0.
      goto  testearRA1        ;

      movlw d'8'              ; Número de bits a recibir.
      movwf nBITS_RS232       ;
```

```
      call  Retardo_200micros ; Retardo de 1,5 periodo de bits,
312microsegundos
      call  Retardo_100micros ;
      call  Retardo_5micros         ;
      call  Retardo_5micros         ;

RS232_LeeBit
      bcf   STATUS,C          ; Ahora lee el pin. En principio supone que
es 0.
      btfsc PORTA,1                 ; ¿Realmente es cero?
      bsf   STATUS,C          ; No, pues cambia a "1".
      rrf   DatoRecibido,F          ; Introduce el bit en el registro de
lectura.
      call  Retardo_5micros         ;
      call  Retardo_200micros ;
      decfsz      nBITS_RS232,F           ; Comprueba que es el último
bit.
      goto  RS232_LeeBit            ; Si no es el último bit pasa a leer
el siguiente.
      call  Retardo_200micros ;
      call  Retardo_200micros ;
      call  Retardo_5micros         ; Espera un tiempo igual al los 2
bits de Stop.
      call  Retardo_5micros         ;
      call  Retardo_5micros         ;
      movf  DatoRecibido,W          ; El resultado en el registro W.
      call  puerta                  ;
      movwf PORTB                   ;


      movlw B'10010110'       ; 150
      movwf c3s               ; 20 milisegundos x50 = 1 segundo
temporizacion3s
      clrwdt                        ; x150 =3 segundos
      call  Retardo_20ms            ;
      decfsz      c3s,1             ;
      goto  temporizacion3s         ;
      movlw B'00000000'             ;
      movwf PORTB                   ;
      goto  testearRA1              ;

      puerta
      addwf PCL, 1      ; sumamos offset al PCL
      retlw   B'00000000'     ;  0
      retlw   B'00000001'     ; 1
      retlw   B'00000010'     ; 2
      retlw   B'00000100'     ; 3
      retlw   B'00001000'     ; 4
      retlw   B'00010000'     ; 5
      retlw   B'00100000'     ; 6
      retlw   B'01000000'     ; 7
      retlw B'10000000' ; 8

      INCLUDE  <RETARDOS.INC>       ;
      end                           ;
```

9.4 Código de la API PC/SC para Windows, winscard.h

WinSCard.h: Código de la API para Windows de PC/SC.

```
 *
 * SmartCard API
 *
 * THIS SOFTWARE IS NOT COPYRIGHTED
 *
 * This source code is offered for use in the public domain. You may
 * use, modify or distribute it freely.
 *
 * This code is distributed in the hope that it will be useful but
 * WITHOUT ANY WARRANTY. ALL WARRANTIES, EXPRESS OR IMPLIED ARE
HEREBY
 * DISCLAIMED. This includes but is not limited to warranties of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 */

#ifndef __WINSCARD_H
#define __WINSCARD_H
#if __GNUC__ >=3
#pragma GCC system_header
#endif

#include <WinSmCrd.h>

#ifdef __cplusplus
extern "C" {
#endif

#define SCARD_S_SUCCESS               NO_ERROR

#define SCARD_F_INTERNAL_ERROR        ((DWORD)0x80100001)
#define SCARD_E_CANCELLED             ((DWORD)0x80100002)
#define SCARD_E_INVALID_HANDLE        ((DWORD)0x80100003)
#define SCARD_E_INVALID_PARAMETER     ((DWORD)0x80100004)
#define SCARD_E_INVALID_TARGET        ((DWORD)0x80100005)
#define SCARD_E_NO_MEMORY             ((DWORD)0x80100006)
#define SCARD_F_WAITED_TOO_LONG       ((DWORD)0x80100007)
#define SCARD_E_INSUFFICIENT_BUFFER   ((DWORD)0x80100008)
#define SCARD_E_UNKNOWN_READER        ((DWORD)0x80100009)
#define SCARD_E_TIMEOUT               ((DWORD)0x8010000A)
#define SCARD_E_SHARING_VIOLATION     ((DWORD)0x8010000B)
#define SCARD_E_NO_SMARTCARD          ((DWORD)0x8010000C)
#define SCARD_E_UNKNOWN_CARD          ((DWORD)0x8010000D)
#define SCARD_E_CANT_DISPOSE          ((DWORD)0x8010000E)
#define SCARD_E_PROTO_MISMATCH        ((DWORD)0x8010000F)
```

```
#define SCARD_E_NOT_READY              ((DWORD)0x80100010)
#define SCARD_E_INVALID_VALUE          ((DWORD)0x80100011)
#define SCARD_E_SYSTEM_CANCELLED       ((DWORD)0x80100012)
#define SCARD_F_COMM_ERROR             ((DWORD)0x80100013)
#define SCARD_F_UNKNOWN_ERROR          ((DWORD)0x80100014)
#define SCARD_E_INVALID_ATR            ((DWORD)0x80100015)
#define SCARD_E_NOT_TRANSACTED         ((DWORD)0x80100016)
#define SCARD_E_READER_UNAVAILABLE     ((DWORD)0x80100017)
#define SCARD_P_SHUTDOWN               ((DWORD)0x80100018)
#define SCARD_E_PCI_TOO_SMALL          ((DWORD)0x80100019)
#define SCARD_E_READER_UNSUPPORTED     ((DWORD)0x8010001A)
#define SCARD_E_DUPLICATE_READER       ((DWORD)0x8010001B)
#define SCARD_E_CARD_UNSUPPORTED       ((DWORD)0x8010001C)
#define SCARD_E_NO_SERVICE             ((DWORD)0x8010001D)
#define SCARD_E_SERVICE_STOPPED        ((DWORD)0x8010001E)
#define SCARD_E_UNEXPECTED             ((DWORD)0x8010001F)
#define SCARD_E_ICC_INSTALLATION       ((DWORD)0x80100020)
#define SCARD_E_ICC_CREATEORDER        ((DWORD)0x80100021)
#define SCARD_E_UNSUPPORTED_FEATURE    ((DWORD)0x80100022)
#define SCARD_E_DIR_NOT_FOUND          ((DWORD)0x80100023)
#define SCARD_E_FILE_NOT_FOUND         ((DWORD)0x80100024)
#define SCARD_E_NO_DIR                 ((DWORD)0x80100025)
#define SCARD_E_NO_FILE                ((DWORD)0x80100026)
#define SCARD_E_NO_ACCESS              ((DWORD)0x80100027)
#define SCARD_E_WRITE_TOO_MANY         ((DWORD)0x80100028)
#define SCARD_E_BAD_SEEK               ((DWORD)0x80100029)
#define SCARD_E_INVALID_CHV            ((DWORD)0x8010002A)
#define SCARD_E_UNKNOWN_RES_MNG        ((DWORD)0x8010002B)
#define SCARD_E_NO_SUCH_CERTIFICATE    ((DWORD)0x8010002C)
#define SCARD_E_CERTIFICATE_UNAVAILABLE ((DWORD)0x8010002D)
#define SCARD_E_NO_READERS_AVAILABLE   ((DWORD)0x8010002E)
#define SCARD_E_COMM_DATA_LOST         ((DWORD)0x8010002F)
#define SCARD_E_NO_KEY_CONTAINER       ((DWORD)0x80100030)
#define SCARD_W_UNSUPPORTED_CARD       ((DWORD)0x80100065)
#define SCARD_W_UNRESPONSIVE_CARD      ((DWORD)0x80100066)
#define SCARD_W_UNPOWERED_CARD         ((DWORD)0x80100067)
#define SCARD_W_RESET_CARD             ((DWORD)0x80100068)
#define SCARD_W_REMOVED_CARD           ((DWORD)0x80100069)
#define SCARD_W_SECURITY_VIOLATION     ((DWORD)0x8010006A)
#define SCARD_W_WRONG_CHV              ((DWORD)0x8010006B)
#define SCARD_W_CHV_BLOCKED            ((DWORD)0x8010006C)
#define SCARD_W_EOF                    ((DWORD)0x8010006D)
#define SCARD_W_CANCELLED_BY_USER      ((DWORD)0x8010006E)
#define SCARD_W_CARD_NOT_AUTHENTICATED ((DWORD)0x8010006F)

#define SCARD_SHARE_EXCLUSIVE (0x1)
#define SCARD_SHARE_SHARED    (0x2)
#define SCARD_SHARE_DIRECT    (0x3)
```

```
#define SCARD_LEAVE_CARD   (0x0)
#define SCARD_RESET_CARD   (0x1)
#define SCARD_UNPOWER_CARD (0x2)
#define SCARD_EJECT_CARD   (0x3)

#define SCARD_AUTOALLOCATE   ((DWORD)-1)
#define SCARD_SCOPE_USER     (0x0)
#define SCARD_SCOPE_TERMINAL (0x1)
#define SCARD_SCOPE_SYSTEM   (0x2)

#define SCARD_PROVIDER_PRIMARY (0x1)
#define SCARD_PROVIDER_CSP     (0x2)

typedef ULONG_PTR SCARDCONTEXT, *PSCARDCONTEXT,
*LPSCARDCONTEXT;
typedef ULONG_PTR SCARDHANDLE, *PSCARDHANDLE, *LPSCARDHANDLE;
typedef const BYTE *LPCBYTE;

typedef struct _SCARD_READERSTATEA
{
  LPCSTR szReader;
  LPVOID pvUserData;
  DWORD dwCurrentState;
  DWORD dwEventState;
  DWORD cbAtr;
  BYTE rgbAtr[36];
} SCARD_READERSTATEA, *PSCARD_READERSTATEA,
*LPSCARD_READERSTATEA;

typedef struct _SCARD_READERSTATEW
{
  LPCWSTR szReader;
  LPVOID pvUserData;
  DWORD dwCurrentState;
  DWORD dwEventState;
  DWORD cbAtr;
  BYTE rgbAtr[36];
} SCARD_READERSTATEW, *PSCARD_READERSTATEW,
*LPSCARD_READERSTATEW;

typedef struct _SCARD_ATRMASK
{
  DWORD cbAtr;
  BYTE rgbAtr[36];
  BYTE rgbMask[36];
} SCARD_ATRMASK, *PSCARD_ATRMASK, *LPSCARD_ATRMASK;

HANDLE STDCALL SCardAccessStartedEvent(VOID);
LONG STDCALL SCardAddReaderToGroupA(SCARDCONTEXT, LPCSTR, LPCSTR);
```

LONG STDCALL SCardAddReaderToGroupW(SCARDCONTEXT, LPCWSTR, LPCWSTR);
LONG STDCALL SCardBeginTransaction(SCARDHANDLE);
LONG STDCALL SCardCancel(SCARDCONTEXT);
LONG STDCALL SCardConnectA(SCARDCONTEXT, LPCSTR, DWORD, DWORD, LPSCARDHANDLE, LPDWORD);
LONG STDCALL SCardConnectW(SCARDCONTEXT, LPCWSTR, DWORD, DWORD, LPSCARDHANDLE, LPDWORD);
LONG STDCALL SCardControl(SCARDHANDLE, DWORD, LPCVOID, DWORD, LPVOID, DWORD, LPDWORD);
LONG STDCALL SCardDisconnect(SCARDHANDLE, DWORD);
LONG STDCALL SCardEndTransaction(SCARDHANDLE, DWORD);
LONG STDCALL SCardEstablishContext(DWORD, LPCVOID, LPCVOID, LPSCARDCONTEXT);
LONG STDCALL SCardForgetCardTypeA(SCARDCONTEXT, LPCSTR);
LONG STDCALL SCardForgetCardTypeW(SCARDCONTEXT, LPCWSTR);
LONG STDCALL SCardForgetReaderA(SCARDCONTEXT, LPCSTR);
LONG STDCALL SCardForgetReaderW(SCARDCONTEXT, LPCWSTR);
LONG STDCALL SCardForgetReaderGroupA(SCARDCONTEXT, LPCSTR);
LONG STDCALL SCardForgetReaderGroupW(SCARDCONTEXT, LPCWSTR);
LONG STDCALL SCardFreeMemory(SCARDCONTEXT, LPCVOID);
LONG STDCALL SCardGetAttrib(SCARDHANDLE, DWORD, LPBYTE, LPDWORD);
LONG STDCALL SCardGetCardTypeProviderNameA(SCARDCONTEXT, LPCSTR, DWORD, LPSTR, LPDWORD);
LONG STDCALL SCardGetCardTypeProviderNameW(SCARDCONTEXT, LPCWSTR, DWORD, LPWSTR, LPDWORD);
LONG STDCALL SCardGetProviderIdA(SCARDCONTEXT, LPCSTR, LPGUID);
LONG STDCALL SCardGetProviderIdW(SCARDCONTEXT, LPCWSTR, LPGUID);
LONG STDCALL SCardGetStatusChangeA(SCARDCONTEXT, DWORD, LPSCARD_READERSTATEA, DWORD);
LONG STDCALL SCardGetStatusChangeW(SCARDCONTEXT, DWORD, LPSCARD_READERSTATEW, DWORD);
LONG STDCALL SCardIntroduceCardTypeA(SCARDCONTEXT, LPCSTR, LPCGUID, LPCGUID, DWORD, LPCBYTE, LPCBYTE, DWORD);
LONG STDCALL SCardIntroduceCardTypeW(SCARDCONTEXT, LPCWSTR, LPCGUID, LPCGUID, DWORD, LPCBYTE, LPCBYTE, DWORD);
LONG STDCALL SCardIntroduceReaderA(SCARDCONTEXT, LPCSTR, LPCSTR);
LONG STDCALL SCardIntroduceReaderW(SCARDCONTEXT, LPCWSTR, LPCWSTR);
LONG STDCALL SCardIntroduceReaderGroupA(SCARDCONTEXT, LPCSTR);
LONG STDCALL SCardIntroduceReaderGroupW(SCARDCONTEXT, LPCWSTR);
LONG STDCALL SCardIsValidContext(SCARDCONTEXT);
LONG STDCALL SCardListCardsA(SCARDCONTEXT, LPCBYTE, LPCGUID, DWORD, LPCSTR, LPDWORD);
LONG STDCALL SCardListCardsW(SCARDCONTEXT, LPCBYTE, LPCGUID, DWORD, LPCWSTR, LPDWORD);
LONG STDCALL SCardListInterfacesA(SCARDCONTEXT, LPCSTR, LPGUID, LPDWORD);
LONG STDCALL SCardListInterfacesW(SCARDCONTEXT, LPCWSTR, LPGUID,

LPDWORD);
LONG STDCALL SCardListReaderGroupsA(SCARDCONTEXT, LPSTR, LPDWORD);
LONG STDCALL SCardListReaderGroupsW(SCARDCONTEXT, LPWSTR, LPDWORD);
LONG STDCALL SCardListReadersA(SCARDCONTEXT, LPCSTR, LPSTR, LPDWORD);
LONG STDCALL SCardListReadersW(SCARDCONTEXT, LPCWSTR, LPWSTR, LPDWORD);
LONG STDCALL SCardLocateCardsA(SCARDCONTEXT, LPCSTR, LPSCARD_READERSTATEA, DWORD);
LONG STDCALL SCardLocateCardsW(SCARDCONTEXT, LPCWSTR, LPSCARD_READERSTATEW, DWORD);
LONG STDCALL SCardLocateCardsByATRA(SCARDCONTEXT, LPSCARD_ATRMASK, DWORD, LPSCARD_READERSTATEA, DWORD);
LONG STDCALL SCardLocateCardsByATRW(SCARDCONTEXT, LPSCARD_ATRMASK, DWORD, LPSCARD_READERSTATEW, DWORD);
LONG STDCALL SCardReconnect(SCARDHANDLE, DWORD, DWORD, DWORD, LPDWORD);
LONG STDCALL SCardReleaseContext(SCARDCONTEXT);
VOID STDCALL SCardReleaseStartedEvent(HANDLE);
LONG STDCALL SCardRemoveReaderFromGroupA(SCARDCONTEXT, LPCSTR, LPCSTR);
LONG STDCALL SCardRemoveReaderFromGroupW(SCARDCONTEXT, LPCWSTR, LPCWSTR);
LONG STDCALL SCardSetAttrib(SCARDHANDLE, DWORD, LPCBYTE, DWORD);
LONG STDCALL SCardSetCardTypeProviderNameA(SCARDCONTEXT, LPCSTR, DWORD, LPCSTR);
LONG STDCALL SCardSetCardTypeProviderNameW(SCARDCONTEXT, LPCWSTR, DWORD, LPCWSTR);
LONG STDCALL SCardState(SCARDHANDLE, LPDWORD, LPDWORD, LPBYTE, LPDWORD);
LONG STDCALL SCardStatusA(SCARDHANDLE, LPSTR, LPDWORD, LPDWORD, LPDWORD, LPBYTE, LPDWORD);
LONG STDCALL SCardStatusW(SCARDHANDLE, LPWSTR, LPDWORD, LPDWORD, LPDWORD, LPBYTE, LPDWORD);
LONG STDCALL SCardTransmit(SCARDHANDLE, LPCSCARD_IO_REQUEST, LPCBYTE, DWORD, LPSCARD_IO_REQUEST, LPBYTE, LPDWORD);

#ifndef _DISABLE_TIDENTS

#ifdef UNICODE
typedef struct SCARD_READERSTATEW SCARD_READERSTATE, *PSCARD_READERSTATE, *LPSCARD_READERSTATE;
#define SCardAddReaderToGroup SCardAddReaderToGroupW
#define SCardConnect SCardConnectW
#define SCardForgetCardType SCardForgetCardTypeW
#define SCardForgetReader SCardForgetReaderW
#define SCardForgetReaderGroup SCardForgetReaderGroupW
#define SCardGetCardTypeProviderName SCardGetCardTypeProviderNameW

```
#define SCardGetProviderId SCardGetProviderIdW
#define SCardGetStatusChange SCardGetStatusChangeW
#define SCardIntroduceCardType SCardIntroduceCardTypeW
#define SCardIntroduceReader SCardIntroduceReaderW
#define SCardIntroduceReaderGroup SCardIntroduceReaderGroupW
#define SCardListCards SCardListCardsW
#define SCardListInterfaces SCardListInterfacesW
#define SCardListReaderGroups SCardListReaderGroupsW
#define SCardListReaders SCardListReadersW
#define SCardLocateCards SCardLocateCardsW
#define SCardLocateCardsByATR SCardLocateCardsByATRW
#define SCardRemoveReaderFromGroup SCardRemoveReaderFromGroupW
#define SCardSetCardTypeProviderName SCardSetCardTypeProviderNameW
#define SCardStatus SCardStatusW
#else /* !UNICODE */
typedef struct SCARD_READERSTATEA SCARD_READERSTATE,
*PSCARD_READERSTATE, *LPSCARD_READERSTATE;
#define SCardAddReaderToGroup SCardAddReaderToGroupA
#define SCardConnect SCardConnectA
#define SCardForgetCardType SCardForgetCardTypeA
#define SCardForgetReader SCardForgetReaderA
#define SCardForgetReaderGroup SCardForgetReaderGroupA
#define SCardGetCardTypeProviderName SCardGetCardTypeProviderNameA
#define SCardGetProviderId SCardGetProviderIdA
#define SCardGetStatusChange SCardGetStatusChangeA
#define SCardIntroduceCardType SCardIntroduceCardTypeA
#define SCardIntroduceReader SCardIntroduceReaderA
#define SCardIntroduceReaderGroup SCardIntroduceReaderGroupA
#define SCardListCards SCardListCardsA
#define SCardListInterfaces SCardListInterfacesA
#define SCardListReaderGroups SCardListReaderGroupsA
#define SCardListReaders SCardListReadersA
#define SCardLocateCards SCardLocateCardsA
#define SCardLocateCardsByATR SCardLocateCardsByATRA
#define SCardRemoveReaderFromGroup SCardRemoveReaderFromGroupA
#define SCardSetCardTypeProviderName SCardSetCardTypeProviderNameA
#define SCardStatus SCardStatusA
#endif /* UNICODE */

#endif /* _DISABLE_TIDENTS */

extern const SCARD_IO_REQUEST g_rgSCardT0Pci;
extern const SCARD_IO_REQUEST g_rgSCardT1Pci;
extern const SCARD_IO_REQUEST g_rgSCardRawPci;

#ifdef __cplusplus
}
#endif
#endif /* __WINSCARD_H */
```

/* EOF *

Bibliografía.

OCF Una metodología de programación para el desarrollo de aplicaciones basadas en tarjetas inteligentes.

Juan Antonio Sanchez. PFC: OCF Una metodología de programación para el desarrollo de aplicaciones basadas en tarjetas inteligentes. Universidad de Murcia.

Fco. Javier Ceballos. Visual C++ Aplicaciones para Win32. Ra-Ma, 1999.

Jon Bates y Tim Tomkins. Descubre Microsoft Visual C++. Prentice Hall

Manual de usuario PIC School. Disponible en internet:
http://www.msebilbao.com/notas/downloads/Manual%20de%20usuario%20PIC%20School.pdf

Datasheet del microcontrolador PIC16f84A. Disponible en Internet:
http://www.microchip.com/downloads/

Características de Microsoft Visual Basic, Visual C++ y Visual C#. Disponible en Internet: http://www.microsoft.com.

Entorno de desarrollo MPLAB de Microchip. Disponible en Internet:
http://www.microchip.com.

Winpic800 disponible en Internet: www.winpic800.com/

Foro sobre Microcontroladores PIC :
http://todopic.mforos.com/