UNIVERSIDAD DON BOSCO.
FACULTAD DE INGENIERIA.
ESCUELA DE ELECTRONICA.

TRABAJO DE GRADUACION:

"DISEÑO Y CONSTRUCCIÓN DE UN PAR DE TARJETAS DE CIFRADO/DESCIFRADO BLOWFISH PARA FINES DIDÁCTICOS."

ALUMNOS:

MARTÍNEZ TORRES TANIA DENISE.

TITULO A OTORGAR:

INGENIERO EN ELECTRONICA.

RIVERA ESCOBAR ERIC RIGOBERTO.

TITULO A OTORGAR:

INGENIERO EN TELECOMUNICACIONES.

CIUDADELA DON BOSCO, JULIO DE 2006.

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE ELECTRÓNICA

AUTORIDADES:

RECTOR
ING. FEDERICO HUGUET RIVERA

VICERRECTOR ACADÉMICO
PBRO. VÍCTOR BERMÚDEZ, sdb

SECRETARIO GENERAL
LIC. MARIO RAFAEL OLMOS

DECANO DE LA FACULTAD DE INGENIERÍA
ING. GODOFREDO GIRÓN

DIRECTOR DE ESCUELA DE ELECTRÓNICA
ING. OSCAR DURÁN VIZCARRA

ASESOR DEL TRABAJO DE GRADUACIÓN
ING. NESTOR LOZANO

JURADO EVALUADOR
ING. JUAN CARLOS CASTRO CHÀVEZ
ING. EDUARDO RIVERA
ING. NURY MARTÌNEZ

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE ELECTRÓNICA

JURADO EVALUADOR DEL TRABAJO DE GRADUACIÓN

Ing. Juan Carlos Castro JURADO	Ing. Eduardo Rivera JURADO
Ing. Nury Martìnez JURADO	Ing. Nestor Lozano ASESOR

AGRADECIMIENTOS

Queremos expresar nuestro agradecimiento:

A Dios todopoderoso por todo lo que me has dado en la vida.

A nuestros padres por entendernos y apoyarnos siempre en toda nuestra vida.

Al Ing. Oscar Durán Vizcarra e Ing. Erick Blanco por todo su apoyo en la elaboración de este proyecto.

Al asesor Ing. Néstor Lozano y al tutor Jorge López por sus observaciones, recomendaciones, correcciones y orientaciones en la creación de este proyecto.

Yo, Tania Martínez agradezco a Mario, Erika, José, Martín Campos, Jonathan y Norberto por la valiosa colaboración para la creación de los programas.

Y a Juan José Osiris, Carlos Mejía, Javier Rivero, Daniel Flores, Fabricio López, Melvin Orellana, Salvador Laínez, Mario Mejía, Ing. Orlando Cabrales e Ing. Julio Martínez por haberme ayudado en momentos claves de mi carrera.

Yo, Eric Rivera agradezco al Ing. Elliot Laínez y a los compañeros de trabajo que siempre me dieron apoyo y me entendieron en el tiempo de realización de este proyecto.

A todas aquellas personas que directa o indirectamente nos ayudaron y contribuyeron en la realización de este proyecto.

INDICE

- ★ Portada
- ★ Agradecimientos
- ★ Autoridades
- ★ Jurado Evaluador
- ★ Agradecimientos
- ★ Índice

INTRODUCCION

Desde hace ya varios años, el mundo ha venido pasando por diversos cambios tecnológicos en muchos campos. El campo de las telecomunicaciones y la informática han sido tal vez los que más han impactado a nivel mundial que comenzó con el fin de mantener a mucha gente en contacto de manera más fácil y eficiente. De entre ellas se pueden mencionar la telefonía celular (con todos los servicios integrados hasta ahora) y el uso de Internet.

Poco a poco, las tecnologías fueron mejorando hasta desarrollarse de tal manera que los usuarios tuvieran la necesidad de mejorar sus productos e incluso de los que no hacían uso de estas tecnologías se vieron casi en la necesidad de utilizarlas, ya que resultaba beneficioso de alguna forma para cada persona o empresa de cualquier tipo.

Debido a que la cantidad de usuarios que hace uso de estas tecnologías, a ciertos usuarios no les conviene trasportar cierto tipo de información a través de estos medios, ya que tal información se considera valiosa para ellos y no les conviene que otros que tienen acceso a la misma red de usuarios, logren obtenerla.

Dentro de estos servicios surgió desde sus inicios la política de privacidad y de protección de información. Una forma eficaz de estar seguro de que cierta información solo podrá ser manipulada y vista por usuarios permitidos previamente. Para ello, se tuvo que idear una forma de proteger datos y el método utilizado es el cifrado de datos o criptografía, que se ha usado previamente desde la era antigua.

Cifrando datos, se garantiza hasta cierto punto la seguridad y la privacidad de la información. Esto se hace por medio de algoritmos criptográficos que se encargan de cifrar información para que mientras se trasporta la información, no pueda ser entendida por usuarios o medios no autorizados. Estos tipos de algoritmos se usan bastante en el campo de telefonía y en Internet por razones muy obvias.

Teniendo en cuenta la necesidad del cifrado de datos, se tiene la necesidad de conocer este campo y por tal razón se propone una herramienta sencilla y entendible para comprender paso a paso el proceso de cifrado de información a través de uno de tantos algoritmos criptográficos existentes.

Se propone el algoritmo Blowfish que fue creado por Bruce Schnneier, un personaje actual que ha dado muchos aportes al campo de la criptografía y se ha elegido blowfish por su sencillez y efectividad a la hora de cifrar datos. Además, es uno de los algoritmos más básicos que existe y puede ser comprendido de manera fácil.

Por otra parte, se pretende crear una herramienta que no sólo involucre software, sino también hardware para demostrar que puede ser más efectivo cuando se trabaja con ambas cosas a la vez. Por ello se pretende desarrollar la tarjeta Blowfish cargando un software permanente para cifrar información y una interfaz gráfica para demostrar el proceso de cifrado Blowfish de forma interactiva y que interesados puedan comprender mejor o tener una idea básica del proceso de cifrado de la información.

Y como también se necesita descifrar la información porque de nada sirve ocultar una información, si el o los interesados no pueden acceder a su contenido. Para ello se propone también una tarjeta idéntica que sirva para descifrar lo que la tarjeta anterior cifró. Incluso con software idéntico, pero no igual. Se mostrará más adelante que los procesos tanto teórico como práctico para cifrar y descifrar la información en Blowfish, son similares, sólo con una variación inversa en los respectivos procesos. Incluso con similar interfaz gráfica, pero con una misma calidad de visualización para mejor comprensión.

Para ello es necesario explicar muchas cosas que son necesarias para entender el proceso y para desarrollar la herramienta.

Como ya se mencionó, se eligió el algoritmo Blowfish, cuya tarjeta se fabricará a partir de **microcontroladores,** en el que se construirá su circuito básico y se programará para que funcione como tarjeta de cifrado Blowfish y la Interfaz gráfica que ya se predeterminó que se hará en ambiente Visual Basic. Y el modelo de microcontrolador a utilizar que también ya estaba predeterminado, es el PIC 16F87.

Por tanto, en el presente documento se presentan nueve capítulos que explican lo necesario y lo básico de cada recurso utilizado. Previo a los capítulos se define el tema y se describe el proyecto con sus respectivos objetivos, alcances y limitantes surgidas.

En el capítulo I se tiene un tema básico de criptografía con conceptos básicos, un poco de historia y temas relacionados con este campo, con el fin de introducirnos al origen de los algoritmos criptograficos y enfocarnos en Blowfish con su explicación y diagramas de flujo de so operación.

El capítulo II se habla de los recursos básicos que utilizan las tarjetas de cifrado, que se detallan de forma general y enfocar la parte de hardware, específicamente es un tema general de microcontroladores que es la parte fundamental de las tarjetas y el capítulo III resalta un tema básico de software que es la programación, ya que se utiliza un lenguaje de programación y se presenta como tema básico. Y en el capítulo IV se habla ya de software específico que se ha utilizado para el desarrollo de las tarjetas como de la interfaz gráfica.

El capítulo V es un complemento del II, se detalla el método utilizado para grabar en los microcontroladores los programas de cifrado y descifrado. Y el capítulo VI es la descripción de un software que se utiliza para validar resultados del Blowfish comercial con el que se está desarrollando en este proyecto.

El capítulo VII es información específica del microcontrolador utilizado, es prácticamente un resumen de las hojas técnicas, sólo que más claro y más entendible.

En los capítulos VIII y IX se enfocan en la interfaz gráfica. El VIII es una descripción de los programas con que ha sido creada y el IX ya es de la interfaz en si, su funcionamiento, operación y una explicación de cómo utilizarla.

DEFINICIÓN DEL TEMA

Se diseñarán e implementarán dos tarjetas para cifrar y descifrar datos utilizando el algoritmo Blowfish. Estas tarjetas serán diseñadas con fines didácticos para poder enseñar el algoritmo a estudiantes en el futuro.

El diseño incluirá interfaces gráficas de software para la enseñanza del algoritmo en laboratorios, por lo que dichas interfaces deberán permitir observar la descripción del algoritmo, la forma en que funciona y además se podrá visualizar etapa por etapa la forma en que se encuentra la información cifrada/descifrada, esto último también podrá verse en las tarjetas.

DESCRIPCION.

Cada tarjeta es igual y posee las mismas funciones. Estas se encuentran conectadas a una computadora a través del puerto serial la cual interactúa con un software diseñado especialmente para su uso.

El software es una interfaz gráfica que permitirá visualizar el proceso de cifrado, además es de donde se ingresa el mensaje o archivo a cifrar o descifrar y también es el que se encarga de mandar los datos a la tarjeta. Entonces, existirá una interacción mutua entre tarjeta y computadora, ya que siempre se leerán los datos por cada proceso que realice la tarjeta.

Para iniciar el proceso, se ingresa el texto o archivo de texto a cifrar. Se selecciona el proceso de paso a paso o de corrido y la tarjeta recibe estos datos y comienza con el proceso de cifrado. Al terminar el proceso se le pide al usuario guardar el mensaje cifrado en un archivo de texto con un nombre cualquiera.

Para descifrar, como lo cifrado esta en un mensaje de texto, se puede abrir un archivo de texto y descifrarlo y le aplica el proceso inverso para obtener de nuevo el mensaje original.

La interfaz grafica permite ver el proceso de cifrado paso por paso y etapa por etapa y la forma de cómo van cambiando los datos a medida se va avanzando con el proceso.

Para el caso de que el proceso sea de enseñanza, se requiere simplemente un mensaje de texto de máximo 8 caracteres y se hará paso a paso para que los interesados vean mejor el proceso de cifrado.

El proceso total se dejará para cifrar archivos de texto de un máximo de 2040 caracteres.

OBJETIVOS.

Objetivo General:

Crear una herramienta que facilite a los estudiantes y a interesados, la comprensión del algoritmo Blowfish mediante el uso de software interactivo que a la vez interactúe con un hardware específico.

Objetivos Específicos:

- Elaborar un entorno gráfico que permita una buena apreciación del algoritmo Blowfish para el usuario y que a su vez responda a criterios didácticos para una mejor enseñanza del algoritmo.

- Diseñar un software de fácil manejo para el usuario que permita ver los procesos de cifrado paso por paso (etapa por etapa y vuelta por vuelta).

- Desarrollar un hardware poco complicado que maneje la mayor parte del proceso de cifrado y descifrado Blowfish y que a la vez permita verificar los resultados de software en la propia tarjeta por medio de indicadores.

ALCANCES.

- El software permitirá ver las etapas de cifrado y descifrado del algoritmo Blowfish, de una forma tal que se demuestre su funcionamiento real en base a las operaciones que realiza el Blowfish etapa por etapa y vuelta por vuelta.

- Que el sistema se pueda instalar en cualquier computadora con sistema operativo Windows.

- Los resultados podrán verse en pantalla y en las tarjetas.

- Que el proceso se pueda ejecutar paso a paso para bloques de texto, y de corrido cuando se desee cifrar archivos de texto.

- Seleccionar las mejores alternativas para la construcción de las tarjetas en base a simulaciones previas en software.

- Realizar una interfaz gráfica que permita una buena apreciación del algoritmo blowfish en base a criterios didácticos que se tomarán en cuenta en base a un estudio minucioso para presentar los datos de manera más amigable.

LIMITACIONES.

- El sistema solo se implementará para Windows y no para otros sistemas operativos, ya que eso involucra conocer o investigar a fondo sobre otros sistemas operativos en lo que respecta a lenguajes de programación para bajo y alto nivel y en desarrollo de aplicaciones.

- La velocidad de respuesta del diseño a implementar estará sujeta a los dispositivos seleccionados para la construcción de las tarjetas, así como la velocidad de respuesta del tipo y modelo de la computadora en que se estará ejecutando.

- Debido a que este proyecto es para fines didácticos se cifrará o descifrará solamente texto, en el caso del proceso paso a paso, bloques de texto de 8 caracteres.

- Para el caso de cifrado de archivos, el formato de estos debe ser de extensión txt y de un tamaño máximo de 2.040 caracteres (1,99 KB (2.040 bytes)).

CAPITULO I: CRIPTOGRAFIA.

1.1 INFORMACION PRELIMINAR.

La criptografía ha tenido una historia curiosa. Manejar información en secreto, ha sido un factor importante para su desarrollo y evolución. Su historia se remonta desde la época antigua y como se puede notar, se utilizó con fines militares o estrategias de guerra en las cuales a un pueblo o un bando, no le convenía que sus enemigos conocieran sus mensajes, ya que contenía importante información para tomar ventaja o, para ganar una batalla.

La criptografía se remonta hacia la edad antigua, donde imperios poderosos la utilizaban para comunicarse de manera secreta con los suyos (el griego, romano, egipcio, etc.).

En el siglo pasado podemos pensar en las Guerras Mundiales, más en la segunda, en el uso de criptografía. Aparecen maquinas codificadoras, es decir, un avance tecnológico en el campo de la criptografía por parte de los dos bandos dentro de estas guerras y también con su decodificador específico. Aunque no solamente se ha usado criptografía en guerras, este es sólo un campo de aplicación para los múltiples campos en donde se puede llevar a cabo. Por ejemplo tenemos para navegación tanto marítima como aérea, transmisiones de señales analógicas y digitales, códigos, incluso hasta el lenguaje de una nación podría considerarse como criptografía en groso modo, ya que solamente los que conocen ese lenguaje (que puede llamarse clave) lo entienden.

En fin, la criptografía es algo que se ha usado casi siempre a través de la historia y que ha servido como medio para proteger información valiosa para cierto usuario y que se debe proteger por lo menos hasta que llegue a su destino, evitando que pueda ser "leída o conocida" en el trayecto del envío y que solamente la puedan conocer los que conocen la clave para descifrarlo.

1.2 ASPECTOS INTRODUCTORIOS (TERMINOLOGIA) [1]

Criptografía.
Es la técnica, ciencia o arte de la escritura secreta. El principio básico de la criptografía es mantener la privacidad de la comunicación entre dos personas alterando el mensaje original de modo que sea incomprensible a toda persona distinta del destinatario.

La palabra criptografía proviene de las palabras griegas "kriptos" (oculto, secreto) y "graphein" (escritura). A la transformación del mensaje original en el mensaje cifrado (criptograma) le llamamos cifrado, y a la inversa, el paso del criptograma al mensaje original le llamamos descifrado.

[1] Resumen de [1],[2]. De aquí en adelante se se establecerán estas referencias bibliográficas para que el lector pueda verificar de donde proviene esta información. Los números de referencia se encuentran en la página de bibliografía y referencias en la página.

Emisor y Receptor.
Sus propias palabras dan a sobre entender su significado natural, pero en este caso, el emisor no solamente es el que envía el mensaje, sino también debe asegurarse de que el mensaje vaya seguro en el medio en que se envía, es decir, que un intruso fuera de esta emisión, pueda leer el mensaje, ya que sólo le interesa al emisor.

Mensaje y Cifrado.
Un mensaje es un texto sencillo. Si se le aplica un proceso para que deje de ser un texto sencillo y fácil de entender, a este proceso le llamamos cifrado[2]. Después que se aplica tal proceso, al texto sencillo le llamaremos texto cifrado. Y para recuperar el texto original, le aplicamos un proceso inverso al cual se le llama descifrado. Todo este proceso se muestra en la siguiente figura:

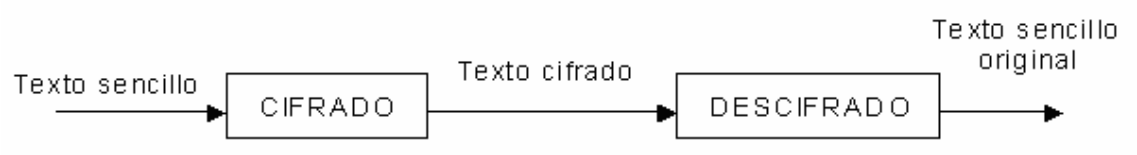


Fig. 1-1: Modelo sencillo de cifrado y descifrado.

Otra forma en que se puede representar un proceso de cifrado es de la siguiente manera: Denotemos el texto sencillo por M (de mensaje, que no se debe de entender como texto sino mas bien como mensaje), este puede ser también un archivo cualquiera (sonido, video u otro tipo de dato). El texto cifrado se denota por C. Algunas veces tanto M como C tienen el mismo tamaño, eso depende del tipo de mensaje que se maneje; pero en otras ocasiones uno es más grande que otro. Y se denota una función E, que corresponde al proceso de cifrado.

Matemáticamente:

$$E(M) = C \qquad \text{(Ecc. 1-1.)}$$

Y para el proceso inverso que sería descifrado, se denota la función D de descifrar:

$$D(C) = M \qquad \text{(Ecc. 1-2.)}$$

Que sería lo mismo que usar: $D(E(M)) = M.$ (Ecc. 1-3.)

Autenticación, Integridad y No Repudio.
Se entiende por autenticación como aquel proceso que garantiza que el mensaje recibido es realmente el que mandó el emisor, sin que algún intruso haya podido cambiar en el transcurso de emisor a receptor, diciendo que ese intruso es el emisor.

En cuanto a integridad, se refiere a que el mensaje llegue a su destino sin modificaciones causadas por algún intruso en el transcurso del envío. Lo que garantiza que el mensaje está tal cual y como fue enviado.

[2] Según la norma estándar ISO 7498-2, se usan los términos en inglés “encipher” y “decipher” que significan cifrado y descifrado. Algunas culturas usan “encrypt” y “decrypt” que en español algunos dirían encriptar y desencriptar, que aparentemente no existen en el lenguaje español”. Ref: BS, pg 1.

No repudio se refiere a que un emisor no niegue o falsee que él como emisor ha enviado un mensaje.

Los requerimientos anteriores son vitales para una interacción entre usuarios y especialmente entre computadores. En todo proceso que involucre seguridad, serán necesarias estas tres cosas: autenticación, integridad y no repudio.

Algoritmos y claves.
Un algoritmo criptográfico se le llama también cifrador. Es una función matemática usada para cifrar y descifrar información. (Por lo general, son dos funciones relacionadas: una para cifrar y la otra para descifrar). Si la seguridad del algoritmo está basada en mantener en secreto la forma en que trabaja se le denomina algoritmo restringido.

La criptografía moderna resuelve problemas con el uso de una CLAVE (KEY) y se denota por la letra K. Esta clave puede tomar un gran número de valores. El rango posible para valores de una clave se le llama KEYSPACE.
Tanto el cifrado como el descifrado utilizan una clave para sus respectivos procesos.
Matemáticamente se denota por:

$$E_K(M) = C$$
$$D_K(C) = M$$
$$D_K(E_K(M)) = M \qquad \text{(Ecc. 1-4.)}$$

La figura siguiente muestra el proceso de cifrado y descifrado utilizando clave.

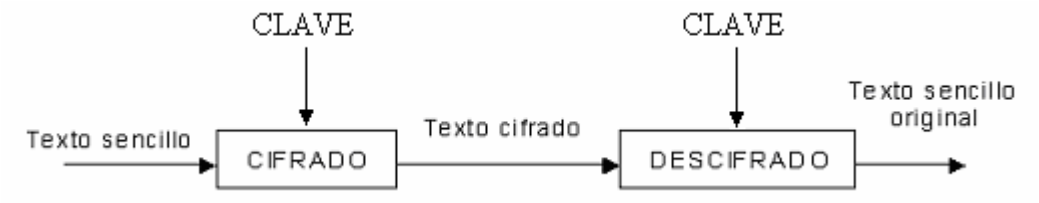


Fig. 1-2: Cifrado y descifrado con clave.

Otros algoritmos utilizan una clave diferente en el cifrado y en descifrado (ver figura siguiente). En este caso, la clave utilizada para el cifrado es K_1 y la utilizada para el descifrado es K_2. En este caso:

$$E_{K_1}(M) = C$$
$$D_{K_2}(C) = M \qquad \text{(Ecc. 1-5.)}$$
$$D_{K_2}(E_{K_1}(M)) = M$$

Fig. 1-3: Cifrado y descifrado con dos claves diferentes.

Toda la seguridad de los algoritmos está basada en la o las claves y no en la estructura o detalles del algoritmo. Esto indica que el algoritmo puede hacerse público para ser analizado, ya que lo importante es la clave o claves, que son predefinidas por el usuario.

Algoritmos simétricos.
Existen dos tipos de algoritmos según su clave: los de clave simétrica y los de clave pública.

Los Algoritmos Simétricos algunas veces son llamados algoritmos convencionales, ya que la clave utilizada para cifrar puede ser calculada en el proceso de descifrado o viceversa. Por lo general, en estos tipos de algoritmos, ambas claves para cifrar y descifrar son las mismas. También se les llama Algoritmos de Clave Secreta, donde la clave requerida para los procesos de cifrar y descifrar, debe ser conocida tanto por el emisor (el que cifra) como el receptor (el que la descifra). La seguridad de estos algoritmos, como se puede notar otra vez, reside en la clave.

Los Algoritmos de Clave Pública también son llamados algoritmos asimétricos y son diseñados de tal forma que la clave para cifrar es diferente de la clave para descifrar. Tampoco, el clave para cifrar no puede ser calculada en el proceso de descifrado. Y se le llama de clave pública porque *la clave para cifrar* puede ser conocida por cualquier extraño, pero solamente alguien específico conocerá la clave para descifrar la información. Se puede decir entonces que la clave para cifrar es una clave pública y que la clave para descifrar es una clave privada.

Criptoanálisis.
Es la ciencia que recobra el texto original de un mensaje sin acceder a la clave. Con éxito, el criptoanálisis puede recuperar tanto el texto original como la clave.
Otro punto importante del criptoanálisis es mantener en secreto el texto original, la clave, o ambos, para evitar que pueda ser detectada o leída por parte de intrusos ajenos al mensaje transmitido. Un atentado criptoanalítico se conoce como un ATAQUE[3].

Existen cuatro tipos de ataques criptoanalíticos, donde cada uno depende de que el criptoanálisis tiene el conocimiento completo del algoritmo de cifrado que se ha utilizado:

[3] No significa que sea un atentado en contra de la seguridad del mensaje, simplemente se utiliza esa palabra como referencia a que se intenta recuperar el mensaje original.

1. Ataque sólo al texto cifrado:
   El criptoanálisis tiene el texto cifrado de varios mensajes, todos ellos se han cifrado con el mismo algoritmo. Entonces el criptoanálisis debe recuperar el texto original de muchos mensajes como sea posible o mejor, deducir la claves (o claves) utilizada (s) para descifrar el o los mensajes con sus respectivas claves. Como se representa a continuación:

$$\begin{aligned} &Dando : C_1 = E_k(P_1), C_2 = E_k(P_2), ...C_i = E_k(P_i) \\ &Deducir\_Con : P_1, P_2, ...P_i; k; \end{aligned} \qquad \text{(Ecc. 1-6.)}$$

   O un algoritmo para referir P_{i+1} desde $C_{i+1} = E_k(P_{i+1})$

2. Ataque de texto original conocido:
   Se puede acceder no solamente al texto cifrado de los mensajes, sino también al texto original de esos mensajes. Este trabajo sirve para deducir la o las claves utilizadas para cifrar los mensajes y estas se pueden utilizar para descifrar nuevos mensajes cifrados con la misma clave.

$$Dando : P_1, C_1 = E_k(P_1); P_2, C_2 = E_k(P_2), ...P_i, C_i = E_k(P_i)$$

   Deducir aún k, o algoritmo para referir P_{i+1} desde $C_{i+1} = E_k(P_{i+1})$ (Ecc. 1-7.)

3. Ataque de texto original elegido:
   Se tiene acceso no sólo al texto cifrado y al texto original asociado para varios mensajes, pero también el criptoanalista elige el texto original que consigue cifrar. Es más fuerte que el anterior, ya que se puede escoger un bloque especifico de texto original para cifrar. Con esto se puede deducir la clave respectiva con que cada bloque fue cifrado.
   Se dan como en los casos anteriores solo que ahora se eligen las claves respectivas a cada mensaje y se asocian directamente.

4. Ataque de texto original elegido-adaptado:
   Este es un caso especial del anterior. Se puede hacer lo mismo que el caso anterior, pero además se puede modificar el bloque elegido basándose en los resultados obtenidos del descifrado de o de los mensajes.

Seguridad de algoritmos.

Los algoritmos tienen diferentes grados de seguridad dependiendo de que tan difícil son para romper. Si el costo requerido para romper un algoritmo es más grande que el valor de los datos cifrados, probablemente se está a salvo.

Ningún algoritmo es cien por ciento seguro. La seguridad de un algoritmo radica en el tiempo que un atacante se tarda en romperlo. El tiempo debe ser suficiente para descubrir al atacante y evitar que prosiga con su ataque y así poder bloquearlo.

El tiempo de rompimiento debe ser mucho mayor que el tiempo en que el mensaje cifrado debe permanecer en secreto. Lars Knudsen[4] clasifica los siguientes tipos de categorías de rompimiento de un algoritmo:

[4] Profesor del Departamento de Matemáticas de la Universidad Técnica de Dinamarca.(DTU por sus siglas en danés Danmarks Tekniske Universitet). Mayor información en : http://www2.mat.dtu.dk/people/Lars.R.Knudsen/.

1. **Rompimiento total.** Un atacante descubre la clave K y con ello tiene control sobre el mensaje cifrado.
2. **Deducción global.** El atacante encuentra un algoritmo alternativo A equivalente al algoritmo de interés, sin conocer la clave.
3. **Deducción local.** El atacante encuentra el texto original de un texto cifrado interceptado.

4. **Deducción de información.** El atacante obtiene información de algún texto original o clave, y puede extraer unos cuantos bits de la clave para usarlos para obtener más información sobre el texto original.

La complejidad a un ataque de un algoritmo se puede mejorar mediante diferentes maneras:

1. Complejidad en los datos. Se refiere a la cantidad de datos que se necesitan de entrada para el ataque.
2. Complejidad de procesamiento. El tiempo necesario para interceptar el ataque.
3. Requerimientos de almacenamiento. La cantidad de memoria necesaria para realizar el ataque.

EXOR SIMPLE.

Esta es una operación OR exclusiva que se denota por $\oplus$ en notación matemática, que consiste en las siguientes operaciones de bits (tabla de verdad):

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

Se le hace referencia a esta función ya que es una operación que se realiza en muchos algoritmos criptográficos y es además una función muy conocida de la que no hay mucho de que hablar. Solo recordando que también se pueden hacer operaciones con letras:

$$a \oplus a = 0$$
$$a \oplus b \oplus b = a$$

Firma Digital.

En el mundo real se han utilizado firmas que representan una identificación del individuo que la realiza. Una firma es auténtica, nadie más la puede hacer igual, inalterable y no puede ser repudiada.

Aunque esos argumentos no son del todo verdaderos, una firma es una identificación personal y privada. Por tal manera, una firma digital es una identificación que debería ser única y además tener las características de una firma normal, que permita garantizar la autenticidad de la información por parte del creador o del poseedor de tal información.

La firma digital no es una clave en si, pero se podría usar como tal. Por ejemplo, un mensaje cifrado no se puede descifrar si no cuenta con firma digital.

1.3 EVOLUCION HISTORICA[5]

Se puede decir que la criptografía es tan antigua como la civilización, cuestiones militares, religiosas o comerciales impulsaron desde tiempos remotos el uso de escrituras secretas; los antiguos egipcios usaron métodos criptográficos, mientras el pueblo utilizaba la lengua demótica los sacerdotes usaban la escritura hierática (jeroglífica) incomprensible para el resto.

Los antiguos babilonios también utilizaron métodos criptográficos en su escritura cuneiforme. El primer caso claro de uso de métodos criptográficos se dio durante la guerra entre Atenas y Esparta, el cifrado se basaba en la alteración del mensaje original mediante la inclusión de símbolos innecesarios que desaparecían al enrollar la lista en un rodillo llamado *escitala* , el mensaje quedaba claro cuando se enrollaba la tira de papel alrededor de un rodillo (escitala) de longitud y grosor adecuados. *Carlomagno* sustituía ya las letras por símbolos extraños. En la época de los romanos se utilizó el cifrado *César* que consistía en cambiar cada letra por la ocupaba tres lugares más adelante en el abecedario.

En la Edad Media, *San Bernardino* evitaba la regularidad de los signos (con lo que el criptoanálisis por el método de las frecuencias no era efectivo) sustituyendo letras por varios signos distintos, así tenía un símbolo para cada consonante, usaba tres signos distintos para cada una de las vocales y utilizaba signos sin ningún valor. El libro más antiguo del que se tiene constancia y que trata sobre criptografía es el *Liber Zifrorum* escrito por *Cicco Simoneta* en el siglo XIV. En el siglo XV destaca *León Battista Alberti* que es considerado por muchos el padre de la criptografía; crea la primera máquina de cifrar que consiste en dos discos concéntricos que giran independientes consiguiendo con cada giro un alfabeto de transposición. En el siglo XVI, *Girolamo Cardano* utilizó el método de la tarjeta con agujeros perforados, que se debía colocar sobre un texto para poder leer el mensaje cifrado ; en ese mismo siglo *Felipe II* utilizó una complicada clave que el francés *Viete* logró descifrar.

En ese mismo siglo, *Blaise de Vigenére* publica *Traicté des Chiffres* donde recoge los distintos métodos utilizados en su época, el método Vigenére es un método clásico de cifrado por sustitución que utiliza una clave. *Carlos I* de Inglaterra usó en el siglo XVII códigos de sustitución silábica. *Napoleón*, en sus campañas militares y en los escritos diplomáticos, usó los llamados métodos *Richelieu* y *Rossignol* y para evitar la regularidad de los símbolos asignaba números a grupos de una o más letras.

En el siglo XIX se utiliza ampliamente el método de transposición, consistente en la reordenación según distintos criterios de los símbolos del mensaje. *Kerckhoffs* escribe el libro La criptografía militar en las que da las reglas que debe cumplir un buen sistema criptográfico. En Primera Guerra Mundial los alemanes usaron el sistema denominado ADFGX en el que a cada combinación de dos letras del grupo ADFGX se le hace corresponder una letra del alfabeto y a la que posteriormente se le hacía una transposición en bloques de longitud 20. El presidente americano *Jefferson* diseño un cilindro formado por varios discos que se utilizaba como máquina criptográfica.

[5] Resumen de [6], [7] y [8].

El mayor desarrollo de la criptografía se dio en el periodo de entreguerras por la necesidad de establecer comunicaciones militares y diplomáticas seguras.

En 1940 se construyó la máquina *Hagelin C-48* consistente en seis volantes unidos por el eje y con distinto número de dientes. En la Segunda Guerra Mundial se construyó por parte alemana la máquina Enigma, que se basaba en un perfeccionamiento del cilindro de Jefferson, pero la máquina británica *Colossus* consiguió descifrar los mensajes cifrados con Enigma. Los americanos construyeron la máquina *Magic* utilizada para descifrar el código púrpura japonés; los americanos a su vez usaron a los indios navajos con su difícil lenguaje para la transmisión de mensajes.

Con el desarrollo de la informática en la segunda mitad de este siglo y con el uso cada vez más extendido de las redes informáticas y del almacenamiento masivo de información se ha dado paso a un gran salto en el estudio de sistemas criptográficos. En 1975 *Diffie y Hellman* establecieron las bases teóricas de los algoritmos de llave pública, hasta entonces no se concebía un sistema de cifrado que no fuese de clave secreta. En la actualidad se usan distintos métodos criptográficos, el DES (de llave secreta), método RSA, método de *Merkle y Hellman*, etc...

1.4 CIFRADO DE SUSTITUCION Y TRANSPOSICION. [6]

SUSTITUCION.
El método de **sustitución** consiste básicamente en sustituir los caracteres del mensaje inicial por otros; los nuevos caracteres pueden ser de cualquier tipo: letras, símbolos, dígitos, etc. Los caracteres iniciales siguen estando en el mismo orden pero salvo que se conozca la equivalencia entre los nuevos caracteres y los antiguos el mensaje es ilegible.

Podemos considerar dos tipos de sustitución:

1. Equivalencia entre alfabetos caracter a caracter. A cada letra del alfabeto ordinario se le hace corresponder un símbolo y el mensaje se cifra cambiando las letras iniciales por su equivalente, si a la letra A le asignamos el símbolo "@" en el mensaje cifrado tendremos *siempre* @ en lugar de A.
2. Utilización de cifra o clave. Distinto del anterior porque una vez establecida la correspondencia entre alfabetos (que en este caso pueden ser el mismo) la asignación de caracteres se realiza teniendo en cuenta la posición del caracter en el mensaje y el dígito que le corresponde según la clave, con un ejemplo quedará esto bastante más claro: Sea el mensaje "SECRETO" y la cifra "23" el mensaje cifrado se consigue (estamos utilizando el mismo alfabeto) adelantando 2 letras la primera que encontremos, 3 la segunda, 2 la tercera, 3 la cuarta y así sucesivamente.

[6] Tomado de [5]. Algunos bloques fueron tomados textualmente.

Para descifrar un mensaje cifrado debemos (en principio) conocer la correspondencia entre alfabetos y en su caso conocer también la clave. Algunos ejemplos se listan a continuación y son códigos que se usan en campos sencillos o se usaron alguna vez:

1. El lenguaje de las manos utilizado por los sordomudos.
2. Código Morse utilizado en telegrafía.
3. Alfabeto de señales con banderines, utilizado en la marina.
4. Lenguajes para ciegos, código Braille.
5. Código internacional de iniciales a-ALFA, b-BRAVO, c-CHARLIE...
6. Código Morse en el que se sustituye el punto por el 1 y la raya por el 2.
7. Símbolos astrológicos. Se sustituye cada símbolo por la inicial del signo o del planeta que representa.
8. Código ASCII en binario.
9. Código telefónico. A cada tecla del teléfono se le asignan tres letras en el siguiente orden 2-ABC 3-DEF 4-GHI 5-JKL 6-MNO 7-PRS 8-TUV 9-WXY 1-ÑQZ, cada letra se sustituye por el número al que está asignada + la posición que ocupa (que puede ser 1, 2 ó 3).
10. Código de telégrafo óptico.
11. Uno de los métodos propuestos por *Francis Bacon.* Las letras normales se transforman todas en 0, las negritas en 1, queda así una sucesión de 0 y 1 que se dividen en grupos de 5, y a partir de ahí se asignan al alfabeto en el siguiente orden 00000-A 00001-B 00010-C etc.
12. Clave César; es un método de cifra que utiliza un sólo dígito (el 3) por lo que equivale a una asignación directa, así A->D, B->E, etc.
13. Códigos de color en binario. A cada letra se le asigna su número en binario A-1 B-10 C-11 D-100, etc. después se le asigna un color a cada una de las posiciones (que para simplificar vamos a llamar unidades, decenas, centenas, etc...) así rojo-unidad, naranja-decena, amarillo-centena, etc. Así la letra E-101 la representamos por rojo-amarillo.

Para resolver los criptogramas anteriores es necesario conocer los códigos (algunos de ellos son de uso común, otros no tanto); ya que la longitud de los mensajes es demasiado corta como para hacer un análisis de frecuencias.

TRANSPOSICION.

El método de transposición consiste en una reordenación de los símbolos del mensaje original de modo que éste resulte ilegible. Si un mensaje consta de n letras se podrá transponer de n! (n factorial) formas. La reordenación se puede realizar desde un modo simple: escribiendo el mensaje letra a letra pero al revés, o utilizando complicados esquemas matriciales.

Los métodos básicos de sustitución y transposición se pueden combinar para formar métodos mixtos más seguros ante un ataque criptográfico. Los métodos clásicos han demostrado su ineficacia ante la potencia de cálculo de los modernos ordenadores, por lo que no se usan en aplicaciones que necesiten una mínima seguridad, sistemas de cifrado como los usados por algunos procesadores para cifrar textos han sido rápidamente criptoanalizados. El estudio de algoritmos seguros de cifrado es, hoy en día, un terreno provechoso para la investigación.

1.5 ALGORITMOS DE COMPUTADORA. [7]

Existen muchos algoritmos hoy en día. Los siguientes son por así decir los más comunes o tal vez los primeros implementados para una seguridad de datos mucho más significativa:

DES (Data Encryption Standard).
Es uno de los más populares algoritmos de cifrado y en Estados Unidos es un algoritmo estándar. No es más que un algoritmo simétrico y se utiliza la misma clave tanto para cifrado como para descifrado.

RSA (Iniciales de los creadores: Rivest, Shamir y Adleman).
Es popular entre los algoritmos de clave pública. Puede ser utilizado para cifrar información y también para realizar firmas digitales.

DSA (Digital Signature Algorithm).
Es un algoritmo de clave pública y que solamente puede ser usado para realizar firmas digitales y no para cifrado de datos.

Existen otros que se usan y que también se han dejado de usar. Algunos fueron base para el desarrollo de otros, debido a ello su importancia en el mundo de la criptografía. Otros que cabe mencionar son los de la siguiente lista[8]:

- Lucifer
- Madryga
- NewDES
- FEAL
- REDOC y REDOC III
- LOKI
- KHUFU y KHAFRE
- RC2
- IDEA
- MMB
- GOST
- CAST
- 3 WAY
- SAFER
- Blowfish

La descripción de estos algoritmos se puede encontrar de forma detallada en las referencias establecidas.

[7] Resumen y traducción de [1].
[8] La información referente a cada uno de estos algoritmos criptográficos se puede encontrar en el libro APPLIED CRIYPTOGRAPHY de Bruce Schneier PARTE III, DES en Pag. 265, desde 303-354. También se pueden encontrar otros algoritmos.

1.6 ALGORITMO BLOWFISH[9]

Blowfish es un algoritmo de cifrado creado por Bruce Schneier que cifra bloques de 64 bits con una clave de longitud variable. Los criterios de diseño tomados en cuenta fueron los siguientes:

- Rapidez, ya que encripta datos sobre microprocesadores de 32 bits a una velocidad de reloj de 26 ciclos por byte.
- Compacto. Puede correr en menos de 5K de memoria.
- Simple, ya que utiliza operaciones simples (suma, XOR y tablas de funciones de 32 bits).
- Seguridad variable en el sentido en que la clave es variable y puede tener un tamaño de hasta 448 bits.

El algoritmo consiste en dos partes:

1. La clave de expansión
2. El cifrado de datos.

La clave de expansión convierte una clave de hasta 448 bits en varios arreglos de subclaves haciendo un total de 4168 bytes.

El cifrado de datos consiste en una simple función de iteración de 16 vueltas, donde cada vuelta consiste de una permutación de clave dependiente y una sustitución de datos dependiente.

Todas las operaciones son SUMA y XOR lógicos en palabras de 32 bits. Hay una operación adicional que consiste en cuatro cajas de arreglos de datos indexados por vuelta. Para la operación SUMA, el acareo no se utiliza.

Para las subclaves, se tienen 18 arreglos P de 32 bits: P_1, P_2, P_3, P_4,……., P_{18}. Estos arreglos están predefinidos antes de generar las subclaves y antes de cifrar los datos. Estos arreglos (al igual que los de las cajas S) van cambiando en cada vuelta del algoritmo.

Los arreglos de datos son cuatro cajas S y tiene 256 entradas por caja:

$S_{1,0}$, $S_{1,1}$, ……., $S_{1,255}$
$S_{2,0}$, $S_{2,1}$, ……., $S_{2,255}$
$S_{3,0}$, $S_{3,1}$, ……., $S_{3,255}$
$S_{4,0}$, $S_{4,1}$, ……., $S_{4,255}$

[9] Traducción y resumen de [1],[4] y [5].

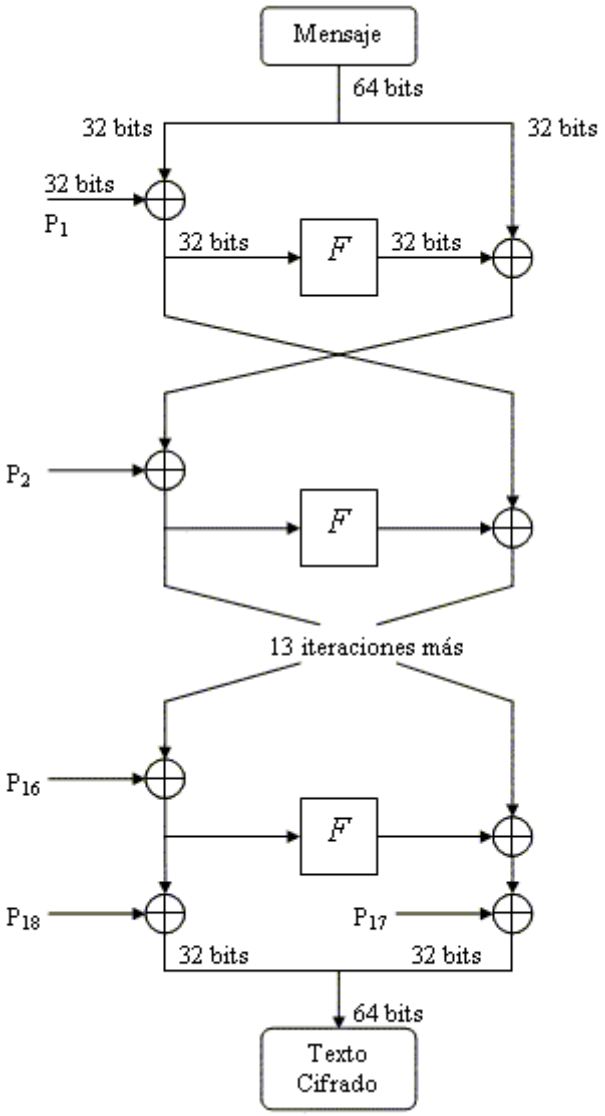


Fig. 1-4: Algoritmo Blowfish.

Proceso de Cifrado de Cifrado del Blowfish.

La entrada es un bloque de 64 bits que se parte en dos bloques de 32 bits cada uno. Uno pasa a ser la parte izquierda X_L y el otro la parte derecha X_R. Si la entrada es de más de 64 bits, se pasarán siempre por bloques de 64 bits y si un último bloque no alcanza los 64 bits, el resto son ceros (ceros a la izquierda).

Para este iniciar el proceso de cifrado, las subclaves P1 a P18 ya han sido calculadas primero (para mejor entendimiento del proceso Blowfish, se explica esta parte primero y después la generación de estas subclaves)

Para las vueltas i =1 hasta i =16:

$$X_L = X_L \oplus P_i$$
$$X_R = F(X_L) \oplus X_R$$

Después se intercambian X_L con X_R y en la última vuelta se realiza el siguiente proceso final de cifrado:

$$X_R = X_R \oplus P_{17}$$
$$X_L = X_L \oplus P_{18}$$

Por último se recombina X_L con X_R para terminar el proceso.

Las funciones F de las cajas S, se realizan de la siguiente manera:

Se divide XL en cuatro partes de 8 bits (a, b, c, d).

$$F(XL) = ((S_{1,a} + S_{2,b} \bmod 2^{32}) \oplus S_{3,c}) + S_{4,d} \bmod 2^{32} \qquad \text{(Ecc.1-7)}$$

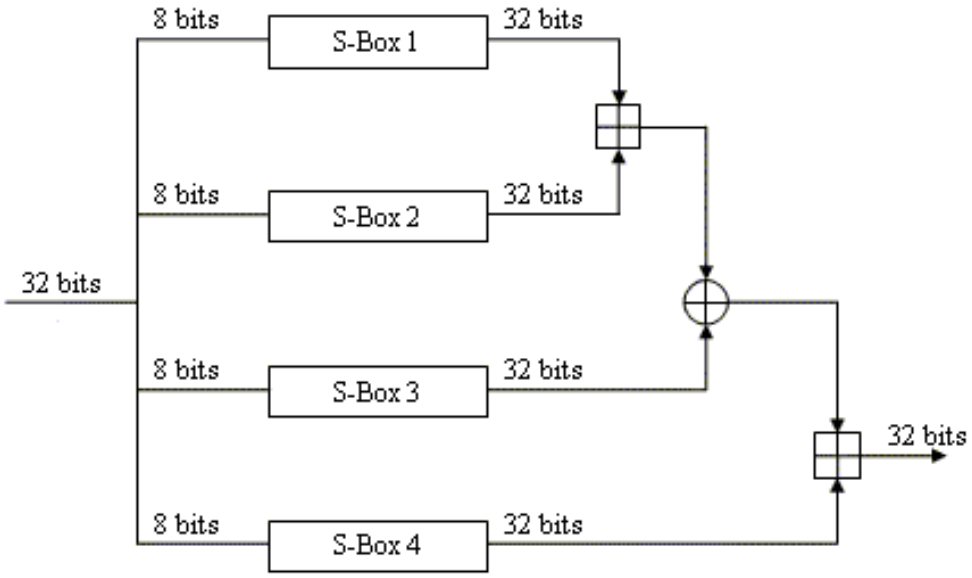


Fig. 1-5: Función F.

Recordemos que cada caja posee 256 valores hexadecimales de 32 bits cada uno. En la entrada de cada caja se tienen 8 bits y con ello, se pueden formar valores desde 0 hasta 255 (00000000_2 hasta 11111111_2) ó 00_{16} hasta FF_{16}, es decir, 256 valores. Si por ejemplo, en la entrada de una caja el valor es 31 (00001111_2), la salida de esta caja S será el número correspondiente a la posición 32 de dicha caja.

Generación de subclaves:

1. Se inicializa el primer arreglo P y después las cuatro cajas S en orden, como cadenas ordenadas. Estas cadenas constan de dígitos hexadecimales de π. Por ejemplo:

   P1 = 0x243f6a88
   P2 = 0x85a308d3
   P3 = 0x13198a2e
   P4 = 0x03707344

2. Se realiza una función XOR de P1 con la primera clave de 32 bits, XOR de P2 con la segunda clave y así sucesivamente hasta que se realiza la operación 14. Y si la clave no es lo suficientemente grande para abarcar todos los arreglos iniciales, la clave se va repitiendo hasta alcanzar los 56 caracteres.

3. Se cifran los valores de la clave, utilizando el algoritmo Blowfish, utilizando como P1 a P18, los valores generados del EXOR de 32 bits de la clave con cada arreglo inicial P.

4. Los valores finales obtenido de XL (parte izquierda) y XR(parte derecha), pasan a ser P1 y P2 respectivamente (primeras subclaves generadas).

5. Se cifra la salida del paso 3 con el algoritmo Blowfish usando las claves modificadas (P1 y P2).

6. Los valores finales obtenido de XL (parte izquierda) y XR(parte derecha), pasan a ser P3 y P4 respectivamente (segundas subclaves generadas).

7. Se reemplazan P3 y P4.

8. Se continua el proceso, reemplazando todas las entradas de los arreglos P y después todas las cajas S en orden con la salida que continuamente cambia con el algoritmo.

Habiendo calculado las subclaves, se procede al proceso de cifrado con los nuevos valores de P1 a P18.

Son 521 iteraciones en total para generar las subclaves requeridas.

Para descifrar el mensaje se sigue el mismo proceso, pero utilizando las subclaves en orden inverso, pero generadas en el mismo orden.

CAPITULO II: TARJETAS DE CIFRADO/DESCIFRADO BLOWFISH

2.1 LA TARJETA.

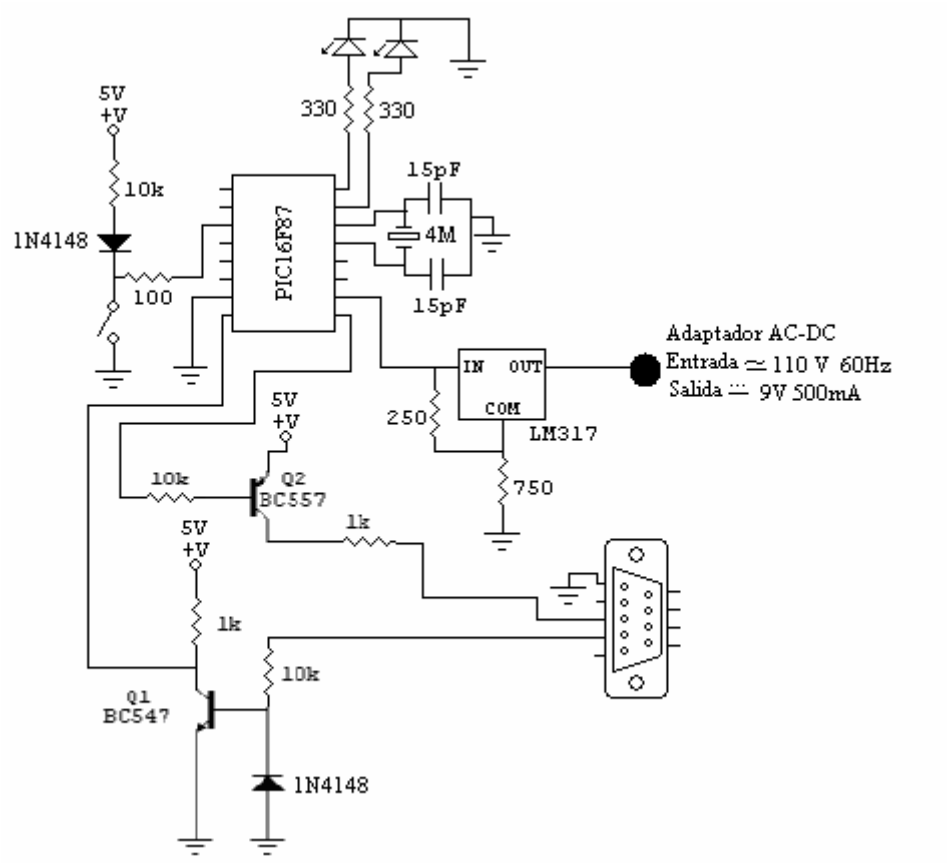


Fig. 2.1: Diagrama de la tarjeta.

La tarjeta de cifrado/descifrado[10], tiene un tamaño de 10 cms de ancho y 8 cms de largo, está compuesta de varios elementos, el principal es el microcontrolador PIC 16F87, que es el encargado de realizar los procesos de cifrado y descifrado y de comunicarse con la computadora, para alimentar el microcontrolador se utiliza un circuito compuesto por un adaptador AC-DC con una entrada de aproximadamente 120 V y 60Hz con una salida de 9V y 500 mA, conectado a un regulador de voltaje LM317 que con la combinación de las resistencias, se reduce el voltaje a 5 voltios, que es el nivel de voltaje que el PIC necesita.

Voltaje de salida = (1+ R2/R1)x1.25 , con R1 = 1KΩ y R2= 3KΩ
Voltaje de salida = 5 V

[10] Esquema de la tarjeta en anexos

Los otros elementos conectados al microcontrolador son:

Circuito oscilador, circuito de reset, circuito para convertir los niveles lógicos microcontrolador en los niveles lógicos del estándar RS-232 y viceversa, diez diodos led, ocho que sirven para mostrar los datos cuando se ejecuta un proceso paso a paso y dos que sirven como indicadores, estos circuitos se describen más adelante.

FUNCIONAMIENTO

Para que la tarjeta funcione correctamente se debe conectar a través del adaptador a un tomacorriente común (60Hz, 120 VAC), también debe conectarse a una computadora a través del cable serial al puerto com1.

Si la alimentación llega correctamente a la tarjeta, el diodo led conectado al pin 10 del microcontrolador se enciende, lo que indica que la tarjeta está lista para realizar cualquier proceso, cuando el usuario le da una orden a través del software este led se apaga y el otro led que está conectado al pin 12 del microcontrolador se pone intermitente lo que indica que el programa está procesando la información, si este led no está intermitente, es decir, se queda encendido o apagado es que ha ocurrido un error, en este caso o en el caso de que el usuario quiera interrumpir un proceso tiene la posibilidad de resetear el microcontrolador con el switch que está en la tarjeta, con esto se vuelve a encender el led que indica que la tarjeta está lista para realizar los procesos.

En todos los procesos el programa inicia calculando los nuevos arreglos P y S, al finalizar si es un proceso paso a paso el usuario podrá ver los datos tanto en el software como en la tarjeta, los datos se muestran de dos en dos en 8 leds que se encuentran en la tarjeta y en cajas de texto en pantalla, el software tiene un botón para ir mostrando los nuevos datos.

Si el proceso es cifrado o descifrado de archivos los datos no se pueden ver en la tarjeta solo en pantalla.

2.1.1 SELECCIÓN DEL MICROCONTROLADOR A UTILIZAR

Inicialmente se tenían contemplado utilizar uno de estos dos tipos de microcontrolador:

Microcontrolador serie PIC16F87 :
Procesador de 8 bits con un juego de 35 instrucciones, tiene 7 Kbytes de FLASH, 256 bytes de EEPROM y 368 bytes de RAM.

Microcontrolador serie 68HC12:
Procesador de 16 bits con un juego de instrucciones complejo, tiene 32 Kbytes de FLASH, 768 bytes de EEPROM, 1 Kbytes de RAM.

Si bien los dos microcontroladores poseen las características para poder ser utilizados en este proyecto, la selección se llevó a cabo, basándose en los aspectos que se muestran en la siguiente tabla:

Aspecto	Microcontrolador	
	PIC16F87	68HC12
Elementos para comunicarse con periféricos	X	x
Mejor precio	X	
Mayor facilidad de compra	X	
Mas información disponible	X	
Mayor experiencia en su programación		x
Mayor facilidad de programación		x
Mas capacidad de memoria		x
Mayor velocidad		x

Los aspectos están ordenados en orden de importancia por lo que aunque el microcontrolador 68HC12 cumple con más aspectos que el PIC 16F87, este último cumple con los aspectos más importantes, cuenta con los elementos para comunicarse con periféricos, tiene un precio mucho menor, tiene mayor facilidad de compra y hay más información disponible.

2.2 MICROCONTROLADORES[11]

Los microcontroladores son circuitos integrados programables que contienen todos los componentes de un computador. Se utilizan para controlar una tarea determinada y debido a su reducido tamaño, suele ir incorporado en el propio dispositivo que gobierna[12].

Las líneas de entrada/salida que poseen los microcontroladores soportan el conexionado de diversos elementos para llevar a cabo la tarea para la que fue destinado, estos elementos se describirán mas adelante.

El número de productos que funcionan con microcontroladores aumenta con el paso del tiempo y se espera que en unos cuantos años sean pocos los elementos que carezcan de microcontroladores.

2.2.1 Arquitectura interna

Como se mencionó en la sección anterior, los microcontroladores poseen todos los componentes de un computador, pero con características que no pueden alterarse.

Las partes principales de un microcontrolador son:

[11] Resumen de [3] y [9]
[12] Esta característica le confiere la denominación de “controlador incrustado”

a) El Procesador

Debido a la necesidad de conseguir un alto rendimiento en el procesamiento de las instrucciones, se ha generalizado el empleo de procesadores de arquitectura HARVARD frente a los tradicionales que seguían la arquitectura de VON NEUMANN, la cual se caracteriza porque el CPU se conecta a través de un sistema de buses con una memoria única, donde coexisten datos e instrucciones, como muestra la figura 2-1.

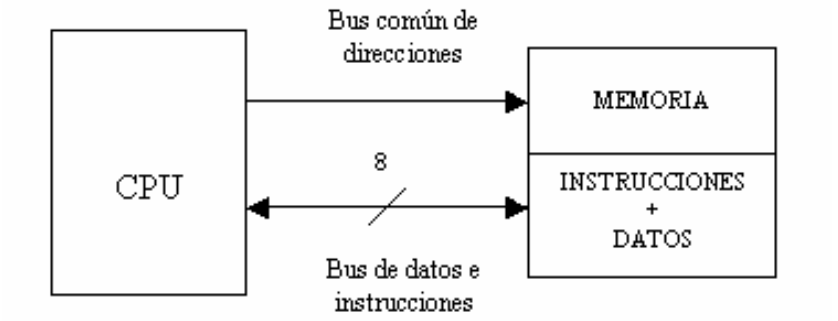


Fig. 2-2: Procesador de un microcontrolador

En la arquitectura HARVARD, la memoria de instrucciones y la memoria de datos son independientes y cada una dispone de su propio sistema de buses para el acceso. Ver figura 2-2.

El procesador de los microcontroladores modernos responde a la arquitectura **RISC** (Computadores de Juego de Instrucciones Reducido), poseen un conjunto de instrucciones de máquina pequeño y simple, de forma que la mayor parte de las instrucciones se ejecuta en un ciclo de instrucción.

Estas características aumentan el rendimiento del microcontrolador, al igual que la segmentación del procesador, el cual se descompone en etapas para poder procesar una instrucción diferente en cada una de ellas y trabajar con varias a la vez.

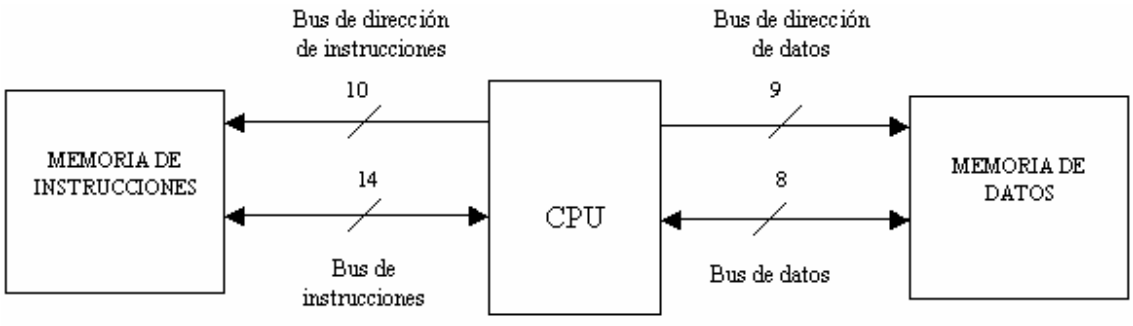


Fig. 2-3: Memoria de instrucciones y de datos conectadas al CPU.

b) Memoria de Datos

Los programas utilizan datos que varían continuamente, por lo que la memoria que los contiene debe ser de lectura y escritura, por esta razón se utiliza la memoria RAM (memoria de acceso aleatorio) estática (SRAM), la cual es más adecuada, aunque sea volátil, es decir si falla el suministro de alimentación los datos almacenados se pierden.
Existen microcontroladores que poseen como memoria de datos una de lectura y escritura del tipo EEPROM no volátil para evitar que los datos se pierdan por fallas de alimentación.

Tipos de memoria que existen en un microcontrolador son:

Memoria de Programa
Los microcontroladores están diseñados para que en su memoria de programa se almacenen todas las instrucciones del programa de control. No hay posibilidad de usar memorias externas de ampliación.
El programa a utilizar debe estar grabado de forma permanente en alguna memoria, esto depende del microcontrolador ya que algunos poseen ciertos tipos de memoria y carecen de otras, las memorias para grabar los programas son:

ROM con máscara
Memoria de solo lectura en la que el programa se graba en el chip durante el proceso de fabricación.

EPROM
Memoria de solo lectura borrable, para grabar esta memoria se utiliza un dispositivo físico gobernado desde un computador personal, llamado **grabador.** En la superficie de la cápsula del microcontrolador existe una ventana de cristal en la que se somete al chip de la memoria a rayos ultravioletas para producir su borrado y así poderla emplear nuevamente.

OTP (Programable una vez)
Este modelo de memoria sólo puede ser grabado una vez por el usuario, se utiliza el mismo procedimiento que con la memoria EPROM. No puede ser borrada después.

EEPROM
Memoria de solo lectura eléctricamente borrable, su grabación es similar a las memorias OTP y EPROM, pero su borrado es mucho más sencillo ya que se efectúa de la misma manera que el grabado, o sea, eléctricamente. Sobre el mismo zócalo del grabador puede ser programada y borrada tantas veces como se quiera, lo cual la hace ideal en la enseñanza y en la creación de nuevos proyectos

FLASH
Memoria no volátil, que al igual que la EEPROM se puede escribir y borrar en circuito, pero suelen disponer de mayor capacidad. El borrado sólo es posible con bloques completos y no se puede realizar sobre posiciones especificas.

c) Líneas de Entrada/Salida Para los Controladores Periféricos

Los microcontroladores cuentan con dos pines para su alimentación, otros dos para el cristal de cuarzo, el cual regula la frecuencia de trabajo del microcontrolador, y uno para el Reset, el resto de pines se utilizan para la comunicación con periféricos externos. Las líneas de E/S que se adaptan con los periféricos manejan información en paralelo y se agrupan en conjuntos de ocho, llamados puertos, también existen modelos con líneas que soportan la comunicación en serie.

d) Interrupciones

Como su nombre lo indica son una interrupción al flujo normal del programa, son originadas asincrónicamente por diversas fuentes, estas pueden ser sucesos externos al sistema, como la generación de un flanco o nivel activo en un pin del microcontrolador, o pueden ser internos, como el desbordamiento de un contador.

Al ocurrir una interrupción, el microcontrolador interrumpe su programa salva la dirección actual del contador del programa en la pila y luego carga el contador del programa con la dirección del vector de interrupción, ahí debe estar alguna rutina que deba realizarse en caso de ocurrir la interrupción, al finalizarla regresa de nuevo al programa justo donde se quedo cuando fue interrumpido.

e) Recursos Auxiliares

Existen diversos tipos de microcontroladores en el mercado, los cuales incorporan una diversidad de recursos que refuerzan la potencia y la flexibilidad del dispositivo. Entre los más comunes se encuentran:

- ✓ **Circuito Oscilador:** Encargado de generar los impulsos que sincronizan el funcionamiento de todo el sistema.
- ✓ **Temporizadores:** controla tiempos.
- ✓ **Perro Guardián (watchdog):** Provoca la reinicialización del sistema cuando el programa se bloquea.
- ✓ **Conversores AD y DA:** Se utilizan para enviar y recibir señales analógicas.
- ✓ **Comparadores analógicos:** Verifica el valor de una señal analógica.
- ✓ **Sistema de protección ante fallos de alimentación.**
- ✓ **Estado de Reposo:** El sistema queda congelado y el consumo de energía se reduce al mínimo.

2.3 LA FAMILIA DE MICROCONTROLADORES PIC. [13]

Un microcontrolador o un MCU es básicamente un microprocesador al que se le han integrado los bloques de memoria de datos, programa y los periféricos de entrada/salida, esto en cuanto al hardware y como principal característica del software, es que posee instrucciones orientadas al bit. Esto quiere decir que posee instrucciones en las cuales los operandos fuente y/o destino es una variable de simplemente un bit de longitud, a parte de contar con las clásicas instrucciones de byte por byte.

En la estructura del hardware del microcontrolador se pueden encontrar los siguientes sistemas (no siempre todos) dependiendo del fabricante y familia del microcontrolador.

- CPU
- Memoria de datos volátil: RAM.
- Memoria de datos no volátil: EEPROM.
- Memoria de programa: PROM, EPROM, EEPROM, FLASH.
- Entradas/Salidas discretas.
- Contadores y temporizadores.
- Conversores A/D y D/A.
- Comparadores analógicos.
- Gestión de interrupciones.
- Gestión de consumo de energía.
- Registros especiales.
- Reloj de tiempo real.
- Unidad de comunicaciones seriales: 12C, UART configurables.
- Interfase para programación en serie: SPI.

2.3.1 Reseña Histórica.

En 1965, la empresa GI creó una división de microelectrónica, GI Microelectronics Division, que comenzó fabricando memorias EPROM y EEPROM, que conformaban las familias AY3-XXXX y AY5-XXXX. A principios de los años 70's diseñó el microprocesador de 16 bits CP1600, razonablemente bueno, pero que no manejaba eficazmente las entradas y las salidas. Para solventar este problema, en 1975 diseñó un chip destinado a controlar E/S: el PIC (Peripheral Interfase Controller). Se trataba de un rápido pero limitado y con pocas instrucciones pues iba a trabajar en combinación con el CP1600.

La arquitectura del PIC, que se comercializó en 1975, era sustancialmente la misma que la de los actuales modelos PIC16C5X. En aquel momento se fabricaba con tecnología NMOS y el producto sólo se ofrecía con memoria ROM y con un pequeño pero robusto microcódigo.

La década de los 80's no fue buena para GI, que tuvo que estructurar sus negocios, concentrando sus actividades en los semiconductores de potencia.

[13] Resumen de [9].

La GI Microelectronics Division se convirtió en una empresa subsidiaria, llamada GI Microelectronics Inc. Finalmente en 1985, la empresa fue vendida a un grupo de un grupo de inversores de capital de riesgo, los cuales tras analizar la situación, rebautizaron a la empresa con el nombre de Arizona Microchips Technology y orientaron sus negocios a los PIC, las memorias EPROM paralelo y las EEPROM serie. Se comenzó rediseñando los PIC, que pasaron a fabricarse con tecnología CMOS, sugiriendo la familia de gama baja PIC16CSX, considerada como la clásica.

Una de las razones de éxito de los PIC se basa en su utilización. Cuando se aprende a manejar uno de ellos, conociendo su arquitectura y su repertorio de instrucciones, es muy fácil emplear otro modelo.

Microchip cuenta con su factoría principal en Chandler, Arizona, en donde se fabrican y prueban los chips con los más avanzados recursos técnicos. En 1993 construyó otra factoría de similares características en Tempe, Arizona. También cuenta con centros de ensamblaje y ensayos en Taiwan y Tailandia. Para tener una idea de su alta producción, se ha superado el millón de unidades por semana en productos CMOS de la familia PIC16CSX.

2.3.2 Las Tres Gamas de PIC.

Para resolver aplicaciones sencillas se precisan pocos recursos, en cambio, las aplicaciones grandes requieres numerosos y potentes recursos. Basándose en esta idea, Microchip construye diversos modelos de microcontroladores orientados a cubrir las necesidades de cada proyecto. Así, hay disponibilidad de microcontroladores sencillos y baratos para atender las aplicaciones simples y otros complejos y más costosos para las de mucha envergadura.

Entre los fabricantes de microcontroladores hay dos tendencias para resolver las demandas de los usuarios.

1. Microcontroladores de arquitectura cerrada.

Cada modelo se construye con un determinado CPU, cierta capacidad de memoria de datos, cierto tipo y capacidad de memoria de instrucciones, número de E/S y un conjunto de recursos auxiliares muy concreto. El modelo no admite variaciones ni ampliaciones.

La aplicación a la que se destina debe encontrar en su estructura todo lo que precisa y en caso contrario, hay que desecharlo. Microchip ha elegido principalmente este modelo de arquitectura.

2. Microcontroladores de arquitectura abierta.

Estos microcontroladores se caracterizan porque, además de disponer de una estructura interna determinada, pueden emplear sus líneas E/S para sacar al exterior los buses datos, direcciones y control, con los que se posibilita la ampliación de la memoria y las E/S con circuitos integrados externos. Microchip dispone de modelos PIC con arquitectura abierta, sin embargo, esta alternativa escapa de la idea de un microcontrolador incrustado y se asemeja a la solución que emplean los clásicos microprocesadores.

Los verdaderos microcontroladores responden a la arquitectura cerrada y permiten resolver una aplicación con un solo circuito integrado y a precio muy reducido.

La mayoría de los sistemas de control incrustados requieren CPU, memoria de datos, memoria de instrucciones, líneas de E/S y diversas funciones auxiliares como temporizadores, comunicación serie y otras. La capacidad y el tipo de las memorias, el número de líneas de E/S y el de temporizadores, así como circuitos auxiliares, son parámetros que dependen de la aplicación y varían mucho de unas situaciones a otras.

Para adaptarse de forma óptima a las necesidades de los usuarios, Microchip oferta tres gamas de microcontroladores de 8 bits.

Con las tres gamas de PIC se dispone de una gran variedad de modelos y encapsulados, permitiendo seleccionar el que mejor se acople a las necesidades de acuerdo con el tipo y capacidad e las memorias, el número de líneas de E/S y las funciones auxiliares precisas. Sin embargo, todas las versiones están construidas alrededor de una arquitectura común, un repertorio mínimo de instrucciones y un conjunto de opciones muy apreciadas, como el bajo consumo de potencia y el amplio margen de voltaje de alimentación.

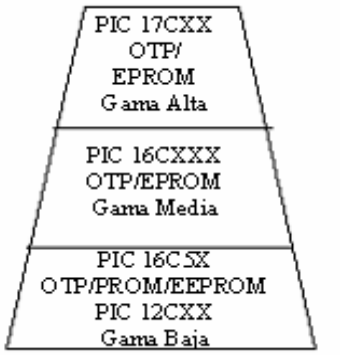


Fig. 2-4. Prestaciones de PIC.

En la figura 2-4 se aprecian las prestaciones de cada modelo PIC.

Junto con los microcontroladores, Microchip ha creado una serie de herramientas de ayuda al desarrollo del hardware y software de los proyectos de aplicación, que son válidas para la mayoría de sus modelos y que se citan a continuación:

- Ensamblador MPASM.
- Simulador Software MPSIM. No soporta modelos PIC 17CXX.
- Compiladote de lenguaje C, MP-C.
- Programador universal PRO MATE.
- Emulador universal PIC MASTER:
- Herramientas de desarrollo para Lógica Difusa FUZZY TECH-MP.
- Entorno de desarrollo integrado MPLAB[14]

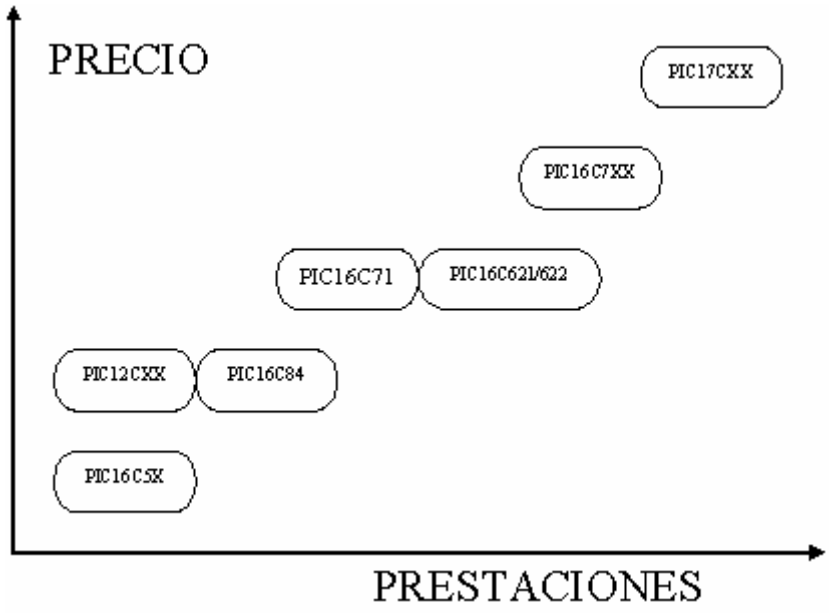


Figura 2-5: Gráfico Precio vrs. Prestaciones en PIC.

La figura 2-5 muestra un gráfico de muestra la relación "Precio vrs. Prestaciones" de los modelos de PIC.

2.3.2.1 Gama Baja.

La gama baja de PIC encuadra nueve modelos fundamentales en la actualidad, cuyas principales características aparecen en las figuras anteriores.

La memoria de programa puede contener 512, 1K y 2K palabras de 12 bits, y ser de tipo ROM, EPROM. También hay modelos con memoria OTP, que sólo puede ser grabada una vez por el usuario. La memoria de datos puede tener una capacidad comprendida entre 25 y 73 bytes. Sólo disponen de un temporizador (TMR0), un repertorio de 33 instrucciones y un número de pines para soportar las E/S comprendido entre 12 y 20. El voltaje de alimentación admite un valor muy flexible comprendido entre 2 y 6.25 V, lo cual posibilita el funcionamiento mediante pilas corrientes teniendo en cuanta su bajo consumo (menos de 2mA a 5V y 4MHZ).

[14] Entorno utilizado para el desarrollo del proyecto, pruebas y simulaciones.

Al igual que todos los miembros de la familia PIC 16/17, los componentes de la gama baja se caracterizan por poseer los siguientes recursos:

1. Sistema POR (POWER ON RESET).

Todos los PIC tienen la facultad de generar una autoreinicialización o auto reset al conectarles la alimentación.

2. Perro guardián (WATCHDOG).

Existe un temporizador que reduce un reset automáticamente si no es recargado antes que pase un tiempo prefijado. Así se evita que el sistema quede "colgado". Dado esa situación, el programa no recarga dicho temporizador y se genera un reset.

3. Código de protección.

Cuando se procede a realizar la grabación del programa, puede protegerse para evitar su lectura. También disponen, los PIC de posiciones reservadas para registrar números de serie, códigos de identificación, prueba, etc.

4. Líneas de E/S de alta corriente.

Las líneas de E/S de los PIC pueden proporcionar o absorber una corriente de salida comprendida entre 20 y 25mA, capaz de excitar directamente ciertos periféricos.

5. Modo de reposo (bajo consumo o SLEEP).

Ejecutando una instrucción SLEEP, el CPU y el oscilador principal se detienen y se reduce notablemente el consumo.

Por último, conviene nombrar dos restricciones importantes sobre los componentes de la gama baja:

- La pila o "stack" sólo dispone de dos niveles, lo que supone no poder encadenar más de dos subrutinas.
- Los microcontroladores de la gama baja no admiten interrupciones.

2.3.2.2 Gama Media.

En esta gama sus componentes añaden nuevas prestaciones a las que poseen los de la gama baja, haciéndolos más adecuados en las aplicaciones complejas.

Admiten interrupciones, poseen comparadores de magnitudes analógicas, convertidores A/D, puertos serie y diversos temporizadores.

Algunos modelos disponen de una memoria de instrucciones tipo OTP (One Time Programmable), que el usuario sólo la puede grabar una vez resulta más económico en la implementación de prototipo y pequeñas series.

Hay modelos de esta gama que disponen de una memoria de instrucciones tipo EEPROM, que debido a que se pueden borrar eléctricamente, son mucho más fáciles de reprogramar que las EPROM, que tienen que ser sometidas a rayos ultravioletas durante un tiempo determinado para realizar dicha operación.

Comercialmente el fabricante ofrece cuatro versiones de microcontroladores en prácticamente en todas las gamas.

1ª. Versión EPROM borrable con rayos ultravioleta. La cápsula dispone de una ventana de cristal en la superficie para permitir el borrado de la memoria de programa al someterla durante unos minutos a rayos ultravioleta mediante lámparas fluorescentes especiales.

2ª. Versión OTP. "Programable una sola vez". Son similares a la versión anterior, pero sin ventana y sin la posibilidad de borrar lo que se graba.

3ª. Versión QTP. Es el propio fabricante el que se encarga de grabar el código en todos los chips que configuran pedidos medianos y grandes.

4ª. Versión SQTP. El fabricante sólo graba unas pocas posiciones de código para labores de identificación, número de serie, palabra clave, checksum, etc.

El temporizador TMR1 que hay en esta gama tiene un circuito oscilador que puede trabajar de forma asíncrona y que puede incrementarse aunque el microcontrolador se halle en el modo de reposo (sleep), posibilitando la implementación de un reloj en tiempo real.

Las líneas de E/S del puerto B presentan una carga "pull-up" activada por software.

2.3.2.3 Gama Alta.

En la actualidad, esta gama está formulada por tres modelos cuyas prestaciones más representativas se mostraron anteriormente.

Los dispositivos PIC17C4X responden a microcontroladores de arquitectura abierta y además se pueden expandir en el exterior al poder sacar los buses de datos, direcciones y control. Así se pueden configurar sistemas similares a los que utilizan los microprocesadores convencionales, siendo capaces de ampliar la configuración interna del PIC añadiendo nuevos dispositivos de memoria y de E/S externas. Esta facultad obliga a estos componentes a tener un elevado número de pines comprendido entre 40 y 44. Además, admiten interrupciones, poseen puerto serie, varios temporizadores y mayores capacidades de memoria que alcanzan las 8K palabras en la memoria de interrupciones y 454 bytes en la memoria de datos.

2.4 Conexión entre la tarjeta y la computadora.

La tarjeta se conecta con la computadora a través del puerto serial, El puerto serial asíncrono es el principal dispositivo de comunicación de sistema a sistema. Asíncrono significa que no hay presente una señal de sincronización o de reloj, cada carácter está enmarcado entre señales de inicio y parada. Un solo bit 0, denominado bit de inicio, precede a cada carácter para indicar al sistema que los siguientes 8 bits constituyen un byte de datos. Uno o dos señales de alto siguen al carácter para señalar que dicho carácter ha sido enviado. [15]

Trama en una transmisión asíncrona.

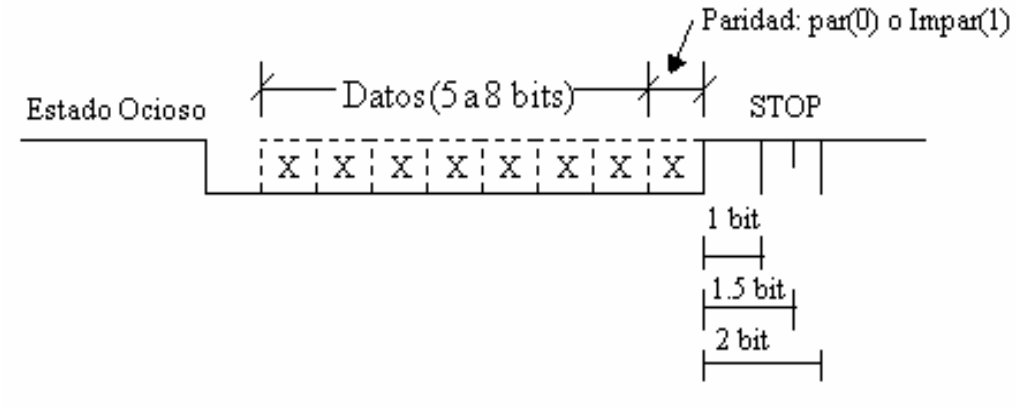


Fig. 2-6: Trama en una transición asíncrona.

El puerto serial que utilizan las computadoras es compatible con el estándar RS-232, cuyos voltajes para un nivel lógico alto están entre -3V y -15V y para un nivel lógico bajo los voltajes están entre +3V y +15V. Para convertir los niveles lógicos proporcionados por las líneas del microcontrolador con los exigidos por el puerto serial se utiliza el arreglo mostrado en la figura 2.5.

[15] Información tomada de [3].

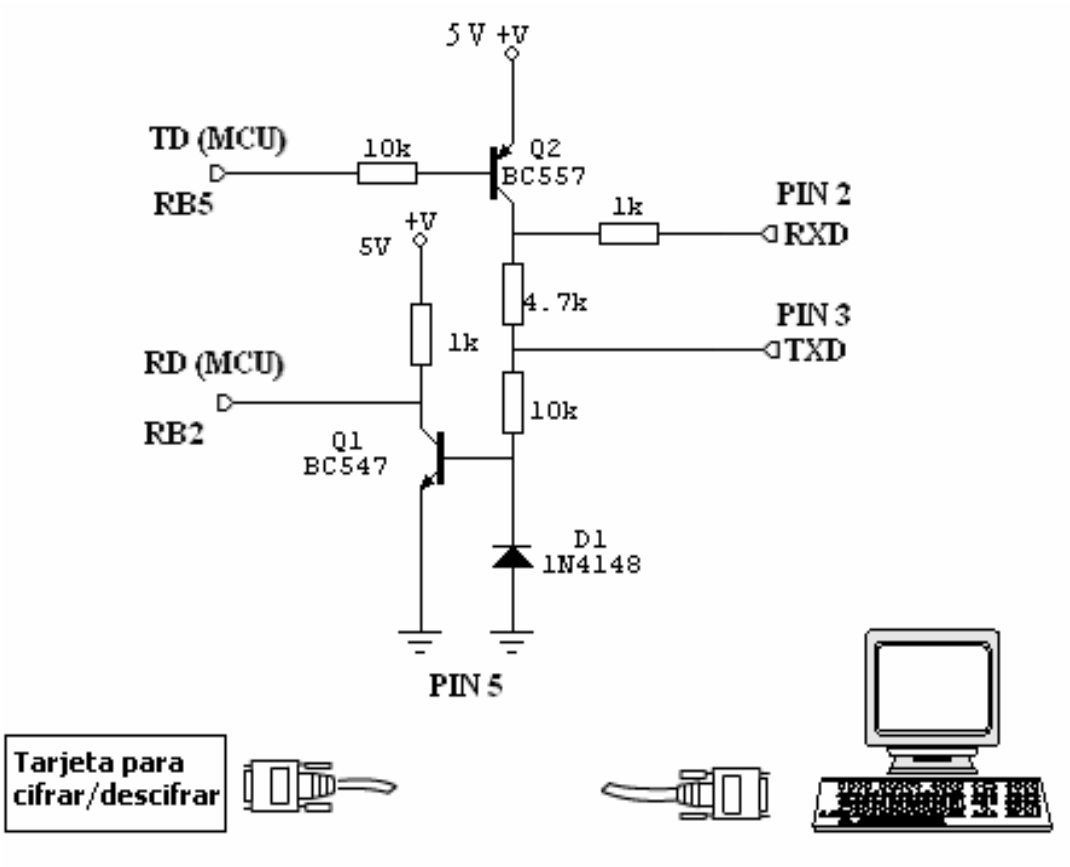


Fig. 2-7: Conversor RS-232 a TTL sin utilizar MAX-232.

Este circuito utiliza la propia corriente del puerto serial de la computadora para generar los símbolos del RS-232. Los pines marcados como TxD, RxD y Gnd corresponden al conector RS-232 de la computadora, mientras que los pines marcados como RD y TD van directamente al microcontrolador.
Este tipo de interfase es muy vista en ratones (MOUSE). Los puntos de alimentación son de 5V (los mismos que los del microcontrolador). [16]

[16] Mayor información en [11].

CAPITULO III: LENGUAJES DE PROGRAMACION.

Para este proyecto, se utilizara un lenguaje de programación para desarrollar gran parte de la aplicación. Lo que corresponde a la interfaz gráfica se creará utilizando programación orientada a objetos y lo que es la transferencia de datos entre las tarjetas y la interfaz. Para ello se presenta este capítulo en que el que habla de lo esencial de un lenguaje de programación específico que se ha elegido para el desarrollo de esta aplicación.

3.1 Lenguajes de programación.

Son lenguajes especiales que ayudan al usuario a comunicarse con la computadora. Establecen una comunicación entre el humano que utiliza palabras acordes con su forma de expresarse (lenguaje humano, sistema decimal, etc.) y la computadora (ceros y unos).

Los lenguajes de programación pueden clasificarse como:

Lenguaje de máquina:
Representa en lenguaje de más bajo nivel en el campo de la programación. Este lenguaje se refiere a números y códigos que solamente el microprocesador entiende. Este lenguaje es fácil de entender por la computadora, pero difícil para el usuario. El punto a favor es que un lenguaje escrito en lenguaje de máquina puede ejecutarse más rápido que otros programas escritos en otro tipo de lenguaje. Es el lenguaje original por la computadora el cual es generado por software y no por el usuario.

Lenguaje de bajo nivel.
Es un lenguaje de programación bien cercano al lenguaje de máquina. Es difícil de entender por las personas y requiere que los programadores codifiquen las instrucciones con muchos detalles. Se le conoce comúnmente como lenguaje ensamblador.

Lenguaje de alto nivel.
Es el que se asemeja más al lenguaje humano. Es más fácil de escribir, pero después se deben traducir por compiladores o por intérpretes a ensambladores y lenguaje de máquina para que la computadora los entienda.

3.2 Lenguajes de alto nivel. [17]

Los lenguajes de alto nivel más comunes son:

BASIC. (Beginners All-prupose Symbolic Instruction Code).
Fue el lenguaje de programación interactivo más popular de la década de los 70. Es un lenguaje de propósito general. Desarrollado por John Kemeny Thomas Kurtz en "Dartmouth Collage" en 1963. Existen numerosas versiones, algunas son compiladores y otras intérpretes.

[17] Resumen de [12].

COBOL (Common Business Oriented Language).
Es un lenguaje compilador diseñado para aplicaciones de negocios. Desarrollado en 1959 por el gobierno federa) de los Estados Unidos y fabricantes de computadoras bajo el liderazgo de Grace Hopper. Es el más utilizado por los "mainframe". COBOL está estructurado en cuatro divisiones; las cuales son:

- División de identificación - identifica el programa.
- División ambiental - identifica a las computadoras fuente y objeto.
- División de datos - identifica las memorias "buffer", constantes y áreas de trabajo.
- División de procedimiento - describe el procesamiento (la lógica del programa).

PASCAL.
Este programa recibió su nombre en honor a Blaise Paseal. Fue desarrollado por el científico suizo Niklaus Wirth en 1970 y diseñado para enseñar técnicas de programación estructurada. Es fácil de aprender y de usar y no utiliza línea sino ";" (punto y coma). Existen versiones de compilador, como de intérprete. Estas varían según la versión.

FORTRAN (FORmula TRANslator).
Es uno de los primeros lenguajes de alto nivel desarrollado en 1954 por John Backus y un grupo de programadores de IBM. Es un lenguaje compilador que se diseñó para expresar con facilidad las fórmulas matemáticas, resolver problemas científicos y de ingeniería.

ADA.
Es un lenguaje desarrollado como una norma del Departamento de Defensa de los Estados Unidos. Es un lenguaje basado en PASCAL, pero más amplio y específico. Fue diseñado tanto para aplicaciones comerciales como científicas. Es un lenguaje de multitareas que puede ser compilado por segmentos separados. Se llama ADA en honor de Augusta Ada Byron, condesa de Lovelace e hija del poeta inglés Lord Byron.

APL (A Programming Language).
Este programa fue desarrollado por Kenneth Inverson a mediados de la década de 1960 para resolver problemas matemáticos. Este lenguaje se caracteriza por su brevedad y por su capacidad de generación de matrices y se utiliza en el desarrollo de modelos matemáticos.

PL/1 (Programming Language 1).
Este programa fue desarrollado por IBM. Es un lenguaje de propósito general que incluye características de COBOL y de FORTRAN. Su principal utilidad es en los "mainframes".

RPG (Report Program Generator) Fue desarrollado por IBM en 1964 y diseñado para generar informes comerciales o de negocios.

Lenguaje C.
Fue desarrollado a principios de la década de los 70 en los laboratorios Bell por Brian Kernigham y Dennis Ritchie. Ellos necesitaban desarrollar un lenguaje que se pudiera integrar con UNIX, permitiendo a los usuarios hacer modificaciones y mejorías fácilmente. Fue derivado de otro lenguaje llamado BCPL.

Lenguaje C++.
Fue desarrollado por Bjarne Stroustrup en los laboratorios Bell a principios de la década de los '80. C++ introduce la programación orientada al objeto en C. Es un lenguaje extremadamente poderoso y eficiente. C++ es un súper conjunto de C, para aprender C++.

Visual BASIC.
Este programa fue creado por Microsoft. Es un programa moderno que da apoyo a las características y métodos orientados a objetos.

Programación orientada al objeto.
Las metas de la programación orientada al objeto es mejorar la productividad de los programadores haciendo más fácil de reutilizar y extender los programas y manejar sus complejidades. De esta forma, se reduce el costo de desarrollo y mantenimiento de los programas. En los lenguajes orientados al objetivo los dalos son considerados como objetos que a su vez pertenecen a alguna clase. A las operaciones que se definen sobre-(os objetos son llamados métodos. Ejemplo de programas orientados al objeto: Visual BASIC y C++.

El lenguaje de alto nivel que se ha elegido para este proyecto de tesis es Visual Basic, utilizando sobre todo la programación orientada a objetos que facilitará el desarrollo y reutilización del software. Estas cualidades de Visual Basic serán desarrolladas a continuación.

3.3 Lenguaje De Programación Visual Basic.[18]

Visual Basic 6.0 es uno de los lenguajes de programación que más entusiasmo despiertan entre los programadores de PCs, tanto expertos como novatos. En el caso de los programadores expertos por la facilidad con la que pueden desarrollar aplicaciones complejas en muy paco tiempo (comparado con lo que cuesta programar en Visual C++, por ejemplo). En el caso de los programadores novatos por el hecho de ver de lo que son capaces a los pocos minutos de empezar su aprendizaje. El precio que hay que pagar por utilizar Visual Basic 6.0 es una menor velocidad o eficiencia en las aplicaciones.

Visual Basic fue desarrollado por Microsoft, su programación se basa en uno de los lenguajes de programación más conocidos a nivel mundial, Basic. Lo que permite al programa, de manera conveniente, desarrollar aplicaciones para Microsoft Windows uno de los sistemas operativos de mayor uso a nivel mundial. Proporciona la clase de capacidad y desempeño muy altas necesarias para desarrollar sistemas para las

[18] Resumen y mayor información en [13],[14] y [15]. Información referente a 3.3 y subtìtulos.

empresas. Hace al lenguaje sumamente adecuado para aplicaciones en Internet, y dotado con características que las personas realmente necesitan tales como: cadenas de caracteres, gráficos, componentes con interfaces graficas, manejo de errores, multimedia (audio, imágenes, animación y video), procesamiento de archivos, procesamiento de bases de datos, redes de cliente/servidor basadas en Internet, buscadores para la World Wide Web, rnejoramiento de documentación de sitios en la World Wide Web con Visual Basic Script (VBScript), y componentes prediseñados. Hace al lenguaje extensible para que vendedores independientes puedan desarrollar componentes para un gran número de aplicaciones. Estos aspectos son precisamente lo que los negocios y organizaciones necesitan para cumplir con los requisitos del procesamiento de información de hoy en día Visual Basic 6.0 es un lenguaje de programación visual, también llamado lenguaje de 4^8 generaciones. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

3.3.1 Programación Orientada a Objetos.

La programación orientada a objetos (OOP: Object Oriented Programming) es una manera natural de pensar en el mundo y de escribir programas de cómputo. Al observar el mundo real que nos rodea, se nota que dondequiera que se vaya hay *objetos.* Gente, animales, plantas, automóviles, aviones, edificios, computadoras y demás. Los seres humanos pensamos en términos de objetos. Tenernos la maravillosa capacidad de la abstracción, que nos permite ver imágenes en la pantalla como objetos, tales como gente, aviones, árboles y montañas, en lugar de cómo puntos de colores (llamados píxeles). Podemos, si lo deseamos, pensar en términos de una playa, en lugar de granos de arena; de un bosque, en lugar de árboles; y de una casa, en lugar de ladrillos.

Se podría inclinar a dividir los objetos en dos categorías: objetos animados y objetos inanimados. Los objetos animados están "vivos" en algún sentido. Se mueven y hacen cosas. Los objetos inanimados, como las toallas, no hacen gran cosa. Sólo están ahí. Sin embargo, todos estos objetos tienen algo en común. Cuentan con atributos como tamaño, forma, color, peso, etc. Todos presentan comportamientos; por ejemplo, una pelota rueda, rebota, se infla y desinfla; un bebé llora, duerme, gatea, camina y parpadea; un automóvil acelera, frena, da vuelta; una toalla absorbe agua; etcétera.

Los seres humanos aprenden sobre los objetos estudiando sus atributos y observando sus comportamientos. Objetos diferentes pueden tener atributos similares y presentar comportamientos similares. Pueden hacerse comparaciones; por ejemplo, entre los bebés y los adultos, y entre los humanos y los chimpancés. Los automóviles, camiones, carritos y patinetas tienen mucho en común.

La OOP simula objetos reales con equivalentes de software. Aprovecha las relaciones de *clase* en las que los objetos de cierta *clase* (digamos una clase de vehículos) tienen las mismas características. Aprovecha las relaciones de *herencia* e incluso de herencia múltiple, en las que se derivan clases nuevas de objetos que heredan características de clases que ya existen y que, sin embargo, contienen características propias únicas.

Un objeto de la clase convertible tiene las características de la clase automóvil, pero de manera adicional: el techo del convertible sube y baja.

La OOP nos da una forma más natural e intuitiva de ver el proceso de programación, por medio de la simulación de objetos reales, sus atributos y sus comportamientos. La OOP también simula la comunicación entre objetos. Al igual que las personas se envían mensajes entre ellas (por ejemplo, el sargento que ordena a la tropa prestar atención), los objetos también se comunican por medio de mensajes.

La OOP encapsula datos (atributos) y funciones (comportamientos) en paquetes llamados objetos; los datos y las funciones de un objeto están íntimamente ligados. Los objetos tienen la propiedad de ocultamiento de información. Esto significa que, aunque los objetos tal vez sepan cómo comunicarse entre ellos a través de interfaces bien definidas, normalmente no se les permite saber la manera en que se implementan otras objetos; los detalles de implementación

Están ocultos en los objetos mismos. Con seguridad es posible manejar un automóvil sin que sea necesario saber cómo funcionan internamente el motor, la transmisión, y el sistema de escape. El ocultamiento de información es crucial para la buena ingeniería de software.

En la OOP la unidad de programación es *la clase, a* partir de *la cual los* objetos son creados en algún momento. Los sustantivos de la especificación de un sistema son los que ayudan al programador a determinar el conjunto de clases a partir del cual se crearán objetos que funcionarán juntos, implementando el sistema. *Los planos son para las casas lo que las clases son para los objetos.* Se pueden construir muchas casas a partir de un plano y podemos crear muchos objetos a partir de una clase.

Cuando se empaqueta el software en clases, éstas se vuelven componentes que pueden reutilizarse en otros sistemas de software. Muchos programadores de hoy en día consideran que con la tecnología de objetos se construirá la mayor parte del software del futuro, combinando partes "estandarizadas e intercambiables" llamadas clases.

La OOP requiere tres tecnologías básicas: *encupsulación, herencia y polimorfismo.* Visual Basic no permite herencia de la misma forma en que C++ y lava lo hacen, pero si permite la encapsulación y el polimorfismo. No obstante, Visual Basic permite a los programadores realizar muchos de los beneficios de la herencia a través del uso de *interfaces y* una técnica llamada *delegación.*

3.3.2 Manejo De Los Puertos Con Visual Basic.

Existen dos formas tradicionales para la comunicación de la computadora con algún periférico: a través del *puerto paralelo y* a través del *puerto serial.*

Con respecto a la comunicación a través del puerto paralelo, Visual Basic carece de comandos o componentes que le permitan escribir datos o leer datos del puerto. Sin embargo, esto se puede solucionar haciendo uso de algún archivo DLL (Dynamic Linked Library).

Como su nombre lo indica, los DLLs pueden permitir a Visual Basic enlazar (un paso antes de compilar) código (librerías que han sido desarrolladas en otros lenguajes como: Delphi, Borland C++ o Microsoft's Visual C++) en tiempo de ejecución (dinámicamente).

La comunicación por puerto paralelo no es una opción a utilizar en nuestro proyecto; por tal razón, no profundizaremos más en los detalles de este tema. Si el lector está interesado en la comunicación a través del puerto paralelo, puede consultar la bibliografía para encontrar información acerca del libro llamado "Parallel Port Complete". Este libro es una buena fuente de información para aquellos interesados en desarrollar programas en Visual Basic que se comuniquen con el puerto paralelo. O bien pueden consultar algunas de los sitios en Internet sugeridos en la bibliografía.

Visual Basic 2.0 y versiones superiores han sido equipadas con capacidades para la comunicación serial. El componente de software encargado de estas funciones en Visual Basic se llama MSCOMM. La versión 2.0 y 3.0 estaba equipada con PvSCOM.VBX y la versión 4.0 incluía tanto a MSCOMl6.OCX y MSCOMM32.OCX. La versión 5.0 y la más reciente la 6.0 son de 32 bit, por esta razón solo incluyen MSCOiVIM32.OCX. Las versiones VBX y OCX tienen ligeras diferencias en sus características.

MSCOMM encapsula las API (Application Programrning Interface) para comunicación de Windows, lo que permite funciones de comunicación tan simples como lectura y escritura de propiedades. Las API permiten notificación por medio de eventos que ocurren en la comunicación.

MSCOMM está oculto en tiempo de ejecución. Cuando se está ejecutando, MSCOMM no tiene interfase visual, todas las entradas y salidas deben realizarse por medio de código.

El control MSCOMM proporciona dos formas diferentes de tratamiento de las comunicaciones:

Las comunicaciones controladas por eventos son un método muy poderoso para el tratamiento de interacciones con el puerto serie. En muchas situaciones resulta útil que se notifique cuándo tiene lugar un evento; por ejemplo, cuándo llega un carácter o cuándo se produce un cambio en las líneas de Detección de portadora (CD) o solicitud de envío (RTS). En tales casos se utiliza el evento ()n(.'omm del control MSCOMM para interceptar y tratar estos eventos de comunicaciones.

El evento OnComm también detecta y trata los errores en las comunicaciones. En la propiedad CommEvent se puede ver una lista completa de todos los eventos y errores posibles en las comunicaciones.

También se puede sondear los eventos y errores si se comprueba el valor de la propiedad CommEvent después de cada función crítica de un programa. Esta alternativa es preferible si la aplicación es pequeña y autónoma. Por ejemplo, si se está escribiendo un marcador telefónico sencillo, no tiene sentido generar un evento después

de recibir cada carácter, ya que los únicos caracteres que piensa recibir son las respuestas de aceptación que envía el módem.

Cada uno de los controles MSCOMM que se usen corresponde a un puerto serie. Si se necesita tener acceso a más de un puerto serie en una aplicación, se debe usar más de un control MSCUMM. La dirección del puerto y la dirección de la interrupción pueden cambiarse desde el Panel de control de Windows.

Aunque el control MSCOMM tiene muchas propiedades importantes, hay algunas con las que un usuario debe familiarizarse primero.

3.3.3 Propiedades y Descripción.

CommPort: establece y devuelve el número del puerto de comunicaciones.

Settings: Establece y devuelve la velocidad en baudios, paridad, bits de datos y nit de parada en forma de cadena.

PortOpen: Establece y devuelve el estado de un puerto de comunicaciones. También abre y cierrra un puerto.

Input: Devuelve y quita caracteres del búfer de recepción.

Output: escribe una cadena de caracteres en el búfer de transmisión.

3.4 Microsoft Visual Basic 6.0.

Esta es la herramienta (software que permite realizar o prograar lo que se ha mencionado en de Visual Basic). Desde programación sencilla de alto nivel hasta la creación de objetos.

Primero fue GW-BASIC, luego se transformó en QuickBASIC y actualmente se le conoce como Visual Basic, Este programa permite crear ventanas, botones, menús y cualquier otro elemento de Windows de una forma fácil e intuitiva.

La versión que se utilizó en este proyecto fue Microsoft Visual Basic 6.0 que se incluye en el paquete Visual Studio 6 de Microsoft. Esta versión combina la sencillez del BASIC con un poderoso lenguaje de programación Visual que juntos permiten desarrollar robustos programas de 32 bits para Windows. Visual Basic 6.0 es una excelente herramienta de programación que permite crear aplicaciones propias para Windows 95/98, Windows NT o superior.

CAPITULO IV: PROGRAMACION DE PICS

4.1 SOFTWARE DE PROGRAMACION DE PICS

Un software gratuito muy utilizado por los programadores de pics es el MPLAB IDE, cuya versión utilizada se explica a continuación.

MPLAB IDE v7.21.

Es un entorno de desarrollo integrado (Integrated Development Environment, IDE) para la familia de microcontroladores PIC, desarrollado por Microchip y puede descargarse gratuitamente en su pagina Web[19].

MPLAB IDE tiene integrado:

- MPASMWIN: Ensamblador que convierte los programas creados en el editor de texto en código máquina (binario), para después cargarlo en la memoria del programa del PIC.

- MPLAB SIM: Programa simulador que reproduce por software el comportamiento del microcontrolador en la ejecución de un programa, presentado en la pantalla de la computadora el estado de todos los registros y recursos.

Para mayor información sobre como utilizar esta herramienta, consultar en [9].

4.2 EL GRABADOR DE PICS.

Es una tarjeta electrónica que soporta varios sócalos con diferentes números de pines en los que se introducen diversos tipos de microcontroladores, además posee circuitos auxiliares y de estabilización de la alimentación.

La grabación se controla mediante un programa de comunicación desde una computadora personal que se adapta al grabador por medio de algún puerto de entrada: SERIE, PARALELO, incluso hay modelos muy nuevos que se conectan por medio del puerto USB.

[19] http://www.microchip.com

4.2.1 TARJETA DEL GRABADOR.

La tarjeta de circuito impreso que se utiliza en el presente trabajo se denomina GRABADOR JDM[20] (http://www.jdm.homepage.dk/newpic.htm).

La figura siguiente muestra el diagrama de la tarjeta del grabador JDM:

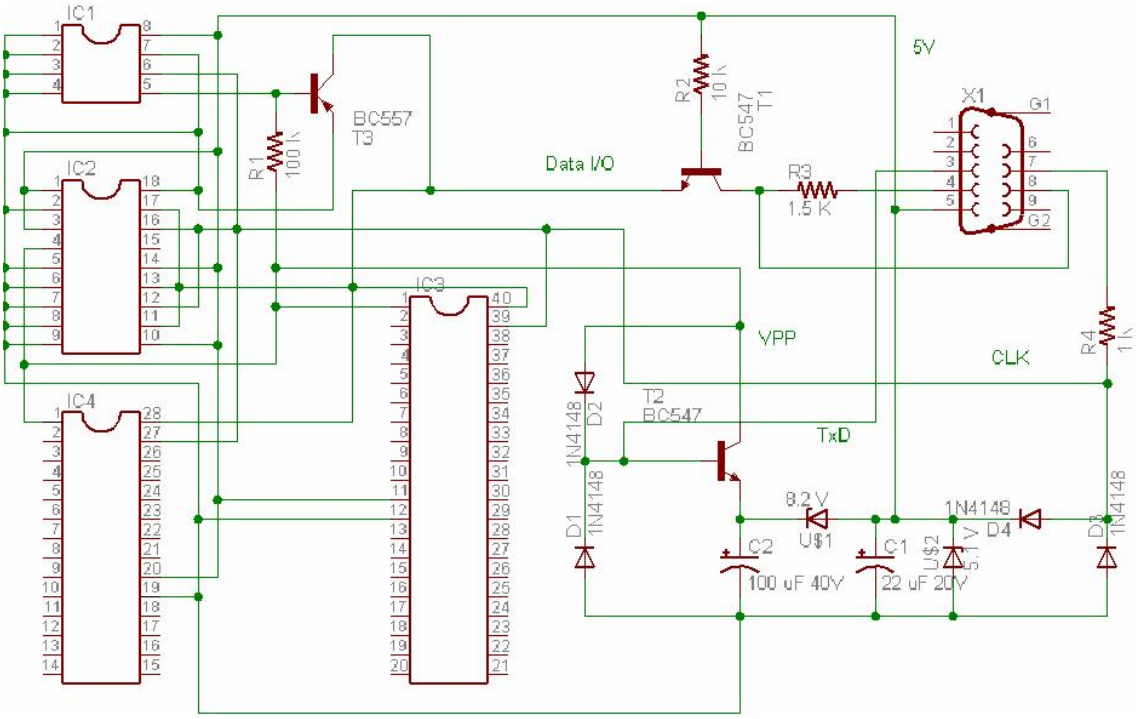


Fig. 5-1: Diagrama eléctrico del grabador de PIC.

Como se puede observar el circuito es sencillo y no posee componentes complicados en funcionamiento y además son fáciles de conseguir. Solamente podemos observar que consta de 3 transistores comunes, diodos generales, capacitores, resistencias y dos tipos de diodos zener. Las formas de circuitos integrados que aparecen son simplemente sócalos para IC de 4, 18, 28 y 40 pines, que son los tamaños de microcontroladores que existen. Se nota también que *el puerto de comunicación es SERIAL* y este grabador no necesita de alimentación externa, ya que trabaja solamente con un Tierra (GND) y con señales provenientes del puerto serie.

Los arreglos de diodos y zener sirven para regular los valores de las señales correspondientes que se necesitan para programar el pic.

NOTA: en la construcción de a tarjeta no es necesario adaptar todos los sócalos para IC, se pueden dejar solo los que se crea que van a utilizar, incluso uno solo, eso no afecta el proceso de grabación. En el caso de más de un sócalo, no se debe tener conectado dos o más pics en el grabador, esto no dañará gravemente a los pics, pero el grabador no será reconocido por el software de comunicación.

[20] Información completa y descargas en el siguiente enlace: http://www.jdm.homepage.dk/newpic.htm.

4.2.2 SOFTWARE DE COMUNICACIÓN WINPIC800.

Es el programa elegido para el proceso de grabación. Tiene la apariencia de la siguiente figura y es sencillo de usar y cuenta con diversas opciones que permiten un grabado, revisión y borrado de programas en un pic. Además acepta una gran variedad de tarjetas grabadoras disponibles gratuitamente en línea.

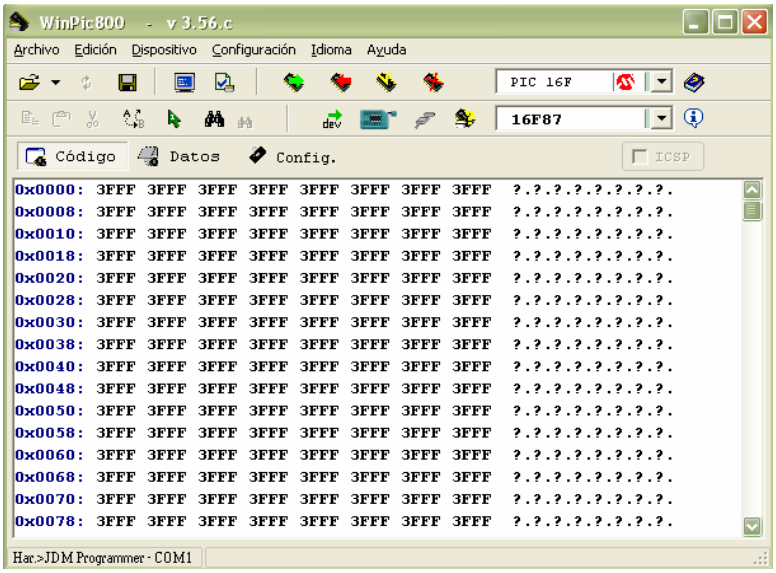


Fig. 5-2: Apariencia de la ventana principal del WinPic 800

Lo que se ve por debajo de los menús y botones, es el código HEX del archivo leído, ya sea en el pic o desde el disco duro, etc. Algunos botones de interés de este programa se explican a continuación:

CONFIGURACION SOFTWARE: opciones de configuración del software. Aparece el siguiente recuadro:

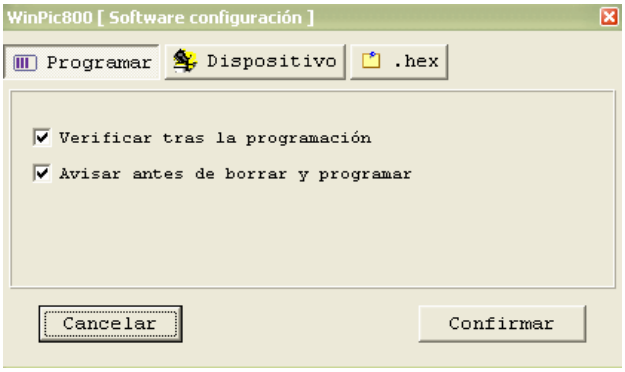


Fig. 5-3: Recuadro de Configuración de Software del WinPIc 800.

Solo son opciones de preferencia que permiten verificar los datos en la programación, activación de aviso antes y después de borrar, auto detección de dispositivo y actualización de archivo HEX antes de programar.

 CONFIGURACION HARDWARE: permite seleccionar el tipo de tarjeta electrónica utilizada. Además muestra información pertinente a cada tarjeta.

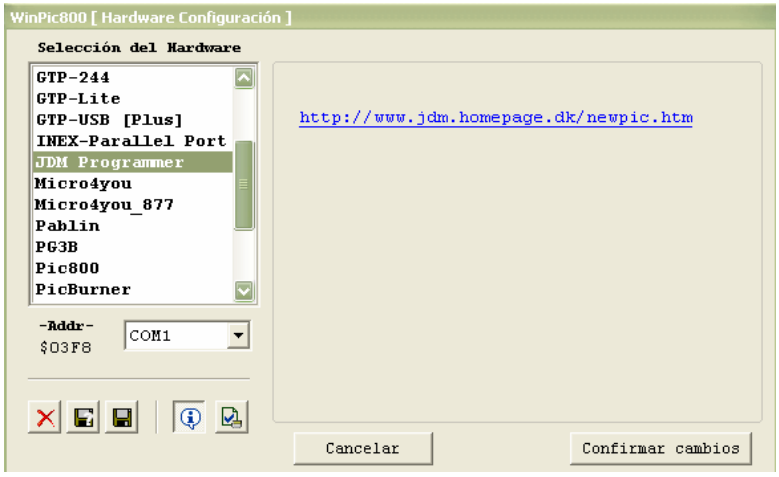


Fig. 5-4: Recuadro de Configuración del hardware (se refiere al grabador) del winPic 800.

En nuestro caso debemos seleccionar JDM Programmer y se puede notar que tiene el puerto serial por default.

 LEER TODO: Permite verificar en el pic el código, su ID, su configuración y los datos, verficando que esté en buen estado toda la información.

 PROGRAMAR TODO: sirve para programar en el pic el archivo HEX especificado.

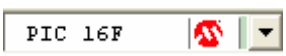
 VERIFICAR TODO: Permite verificar la información en el pic, si no tiene problemas o si ocurrió algún problema después de la programación.

 BORRAR TODO: elimina la información actual de un pic.

 LEER DEV ID-REV: Detecta pic y el ID de dicho dispositivo.

 TEST HARDWARE: Verifica el estado de funcionamiento de la tarjeta grabadora.

 DETECTAR PIC: Detecta el tipo de pic en cuanto a numeración se refiere cuando está conectado al grabador.



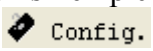
SELECCIÓN FAMILIA: selecciona una familia de pic para buscar su numeración exacta.

SELECCIÓN PIC: Selecciona el tipo de pic específico.

Una cosa importante es que este y la mayoría de los grabadores solamente aceptan archivos con extensión HEX, cuyo archivo resulta de la compilación de un archivo en formato ASM.

Para programar un PIC:

- Seleccionar la familia 16F y el pic 16F87 (para nuestro caso, usar el respectivo pic según se esté utilizando en otra ocasión).
- Tener conectada la tarjeta grabadora y también tener introducido el pic en el sócalo de la tarjeta de la forma correcta.
- En Configuración Hardware, seleccionar el quemador JDM y asegurarse que el puerto seleccionado sea el Serie (COM1).
- Dar clic en el botón Test Hardware para asegurarse que la PC reconoce el grabador.
- Darl clic en Detectar Pic para asegurarse que se está trabajndo con el pic correcto y que funciona correctamente junto con la tarjeta grabadora.
- Abrir un archivo de extensión Hex desde el botón abrir o desde el menú archivo/abrir (como se hace normalmente en windows).
- Una vez abierto el archivo Hex, dar clic en el botón Programar todo y esperar a que finalice el proceso de grabación.
- Si el pic tenía información almacenada, esta se borrará antes que se le programe la nueva.
- Para borrar información de un pic, dar clic en Borrar Todo una vez que se ha detectado el pic.
- Los procesos de Programar y Borrar, verifican los datos en el pic posterior o previamente sin necesidad de usar los botones de verificar. Puede hacerse después para estar seguro pero no es necesario.

NOTA IMPORTANTE: Es necesario especificar si el pic trabajará con oscilador interno o externo. Para ello dar clic en el botón 

y seleccionar el tipo de oscilador que el pic usará en un proceso posterior. Para nuestro caso, se debe seleccionará XTAL ya que nuestro oscilador constará de un cristal de 4MHz. Por conveniencia, seleccionar XTAL y solo activar la opción MCLRF y desactivar las demás.

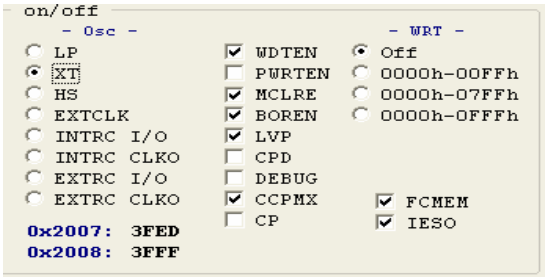


Fig. 5-5. Recuadro de Configuracón de grabación.

Como se puede observar, un grabador de pics no es tan complicado, son faciles de construir y podemos encontrar muchas versiones y utilizar las que creamos convenientes.

CAPITULO V: EL MICROCONTROLADOR PIC 16F87

5.1 Características

El PIC 16F87, es un microcontrolador con memoria de programa flash de 18/20/28 pines, pertenece a la gama media de la familia de los microcontroladores PIC, entre sus características principales están:

- ✓ Osciladores:
  Tres modos de oscilador a cristal :
  - LP(cristal de baja potencia).
  - XT (cristal/resonador).
  - HS(cristal/resonador de alta velocidad): hasta 20 MHz.

- ✓ Dos osciladores externos RC

- ✓ Oscilador interno RC con 8 frecuencias seleccionables, incluyendo: 31.25 kHz, 125 kHz, 250 kHz, 500 kHz, 1 MHz, 2 MHz, 4 MHz y 8 MHz.

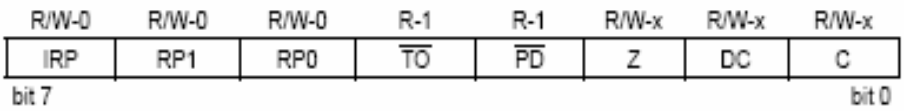
- ✓ Periféricos:
  - Módulo de captura y comparación, PWM (modulación por ancho de pulso)(CCP).
  - Convertidor analógico a digital de 7 canales de 10 bits.
  - Puerto serial síncrono en dos modalidades:

  -SPI (Interfase Periférica Serial)(Maestro/Esclavo).
  -I^2C (Circuito Inter-Integrado)(Esclavo).

  - Transmisor/Receptor Universal Síncrono y Asíncrono (AUSART), operación RS-232 usando oscilador interno (no se requiere un cristal externo), puede trabajar en tres modalidades:
    -Modo asíncrono (full-duplex)
    -Modo síncrono – maestro (half-duplex)
    -Modo síncrono – esclavo (half-duplex)
  - 16 líneas de entrada/salida (en dos puertos, puerto A (A0 a A7), puerto B (B0 a B7)).

- ✓ Memoria:
  - 7 Kbytes de memoria de programa Flash (4Kbytes para instrucciones de 14 bits).
  - 368 bytes de memoria de datos RAM.
    La memoria de datos está dividida en cuatro bancos, cada banco tiene reservado las posiciones mas bajas para los registros de funciones especiales. Bajo los registros de funciones especiales están los registros de propósito general, como se muestran en la figura 7-1.
    Para moverse entre bancos usando direccionamiento directo se utilizan los bits RP0 y RP1 del registro de estado y para moverse con direccionamiento indirecto se utiliza el bit IRP:

STATUS: Registro de estado



R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP**: Bit para seleccionar un banco (usado para direccionamiento indirecto)

1 = Banco 2, 3 (100h-1FFh)
0 = Banco 0, 1 (00h-FFh)

bit 6-5 **RP<1:0>:** Bits para seleccionar un banco (usado para direccionamiento directo)

11 = Banco 3 (180h-1FFh)
10 = Banco 2 (100h-17Fh)
01 = Banco 1 (80h-FFh)
00 = Banco 0 (00h-7Fh)

- 256 bytes de memoria de datos EEPROM.

Características especiales:

- 100,000 ciclos típicos de lectura y escritura de memoria de programa flash.
- 1,000,000 ciclos típicos de lectura y escritura de memoria de datos EEPROM.
- Retención de memoria de datos EEPROM por mas de 40 años.
- Temporizador Watchdog (perro guardián) extendido.
- Operación en un rango de voltaje amplio: 2.0V a 5.5V.
- Existen varias fuentes de interrupción.

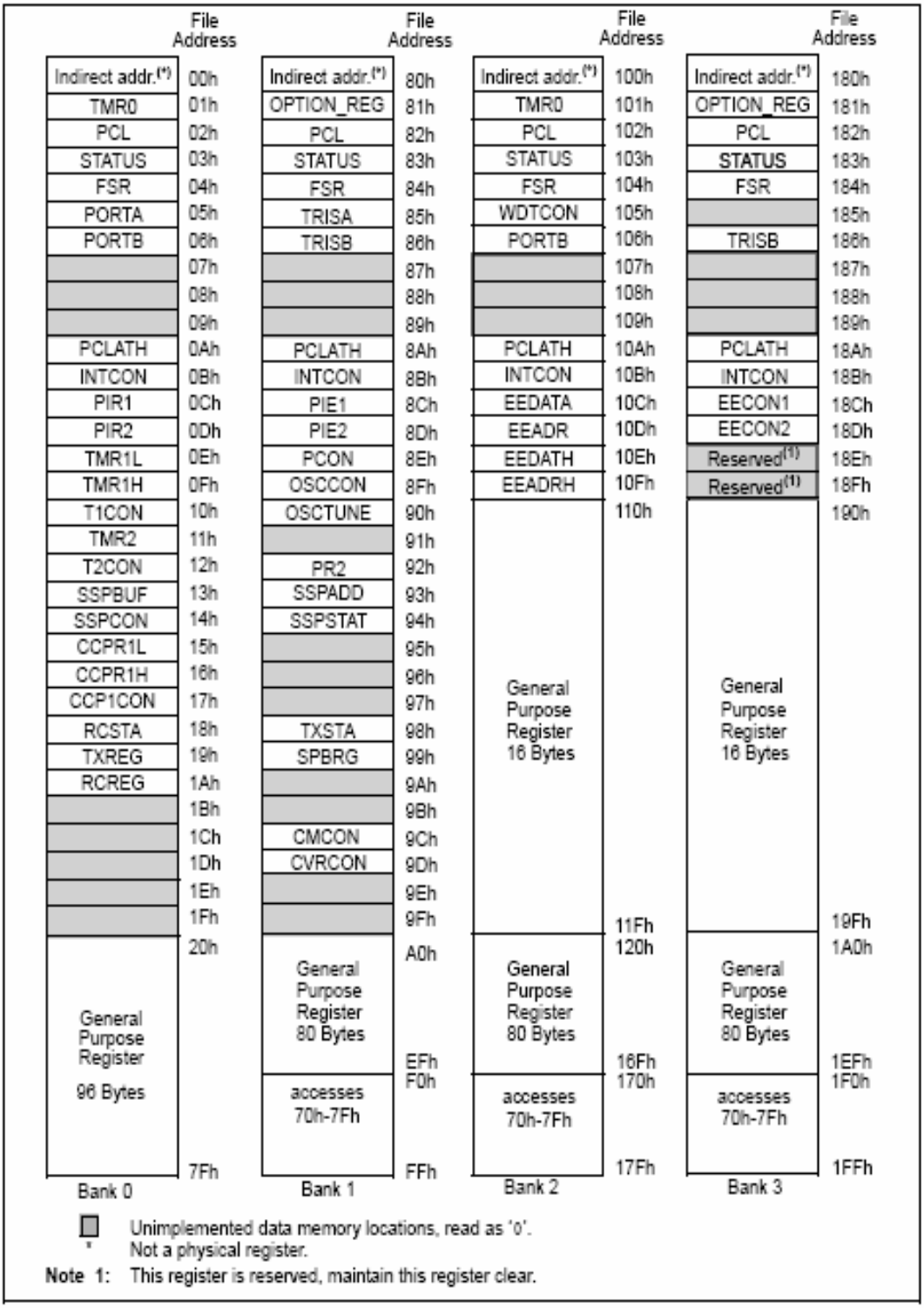


Fig. 5-1. Bancos de memoria del PIC 16F87.[21]

[21] Fuente: Hojas técnicas del PIC 16F87. (Enlace en las hojas de bibliografía y referencia).

5.2 Recursos Utilizados

5.2.1 Circuito Oscilador

Para este proyecto se eligió un oscilador a cristal tipo XT, la frecuencia del cristal es de 4MHz y el valor de los capacitores es de 15pF *que es el valor típico que sugiere la hoja técnica del microcontrolador.*

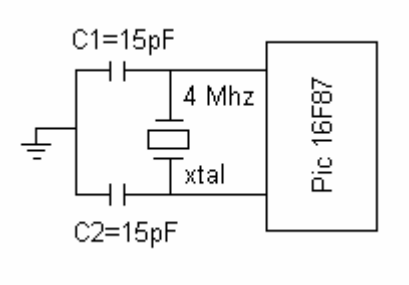


Fig. 5-2.

5.2.2 Transmisor/Receptor Universal Síncrono y Asíncrono AUSART

En este proyecto se utilizó esta interfase de comunicación serial para comunicarse con la computadora, en la modalidad full-duplex en modo asíncrono, los registros asociados a esta interfase son:

TXSTA: Registro de control y estado de transmision

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC**: Bit de la selección de la fuente del reloj
Modo asíncrono:
No importa.
Modo síncrono:
1 = Modo maestro (reloj generado internamente por BRG)
0 = Modo esclavo (reloj de una fuente externa)

bit 6 **TX9**: Bit para permitir la transmisión de 9-bit
1 = Seleccionar transmisión de 9-bit
0 = Seleccionar transmisión de 8-bit

bit 5 **TXEN**: Bit que permite la transmisión
1 = Transmisión permitida
0 = Transmisión no permitida
Nota: SREN/CREN sobrescribe TXEN en modo síncrono.

bit 4 **SYNC**: Bit de selección del modo de la AUSART

1 = Modo síncrono
0 = Modo asíncrono

bit 3 **No implementado se lee como '0'**

bit 2 **BRGH**: bit de selección de Baud Rate

Modo asíncrono:
1 = Alta velocidad
0 = Baja velocidad

Modo síncrono:
no usado en este modo.

bit 1 **TRMT**: Bit que muestra el estado del registro de cambio de transmisión
1 = TSR vacío
0 = TSR lleno

bit 0 **TX9D**: 9-bit de transmisión de datos, puede ser bit de paridad

RCSTA: Estado de recepcion y registro de control

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN**: Bit que activa el puerto serial

1 = Puerto serial activado(configura los pines RB2/SDO/RX/DT y RB5/SS/TX/CK como puerto serial)
0 = Puerto serial desactivado

bit 6 **RX9**: Bit que permite la recepción de 9-bit

1 = Selecciona recepción de 9-bit
0 = Selecciona recepción de 8-bit

bit 5 **SREN**: Bit que permite una única recepción

Modo asíncrono:
No importa.

Modo síncrono – Maestro:
1 = permite una única recepción
0 = desactiva una única recepción

Este bit es limpiado después de que una recepción ha sido completada

Modo síncrono-Esclavo:
No importa.

bit 4 **CREN**: Bit que permite una recepción continua

Modo asíncrono:
1 = Permita recepción continua
0 = No permite recepción continua

Modo síncrono:
1 = Permite la recepción continua hasta que el bit CREN es limpiado (CREN sobrescribe SREN)
0 = no permite la recepción continua

bit 3 **ADDEN**: bit que permite la detección de dirección

Modo asíncrono 9-bit (RX9 = 1):
1 = Permite la detección de dirección, permite interrupciones y carga el buffer de recepción cuando RSR<8> esta puesto

0 = Desactiva la detección de dirección, todos los bytes son recibidos y el noveno bit puede ser usado como bit de paridad

bit 2 **FERR**: bit de error de trama
1 = Error de trama (puede ser actualizado leyendo el registro RCREG y recibe el próximo byte válido)
0 = No hay error de trama

bit 1 **OERR**: bit de error por desborde
1 = Error por desborde (puede ser limpiado, limpiando el bit CREN)
0 = No ha habido error por desborde

bit 0 **RX9D**: 9-bit del dato recibido

REGISTRO SPBRG: Controla el Baud Rate, que es la velocidad de la comunicación, en este proyecto el Baud Rate seleccionado es de 9600 baudios

El valor que se debe poner en este registro, se obtiene con la siguiente fórmula: BaudRate = Fosc/16((x+1)) (esta fórmula es válida cuando el bit BRGH, del registro TXSTA es igual a uno).

x = valor que se carga en el registro SPBRG = (4MHz/(9600x16))-1 = 25

Los demás registros de la interfase AUSART quedan configurados así:

TXSTA: b'00100100' = h24
Transmisión de 8 bits, modo asíncrono, alta velocidad, transmisión activada.

RCSTA: b'10010000' = h90
Activa la interfase AUSART, configura los pines B2 y B5 como pines seriales, recepción continua activada.

5.3 Puertos

Puerto A: de este puerto, se han utilizado los pines del A0 al A4 donde se han conectado 5 de los 8 leds que muestran los datos del proceso de cifrado o descifrado cuando se ejecutan paso a paso.
En el pin A5, se conecta el circuito necesario para provocar un reset, el circuito se muestra a continuación:

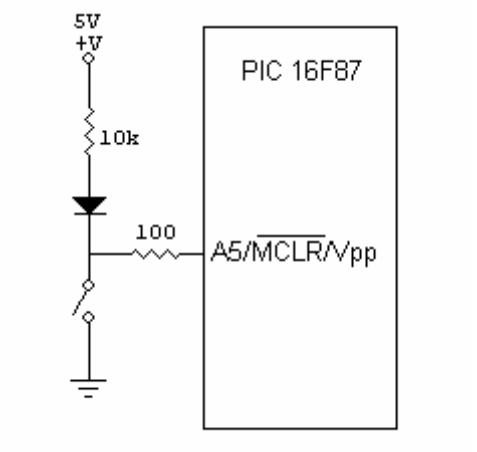


Fig. 5.3. Activación de interrupción del PIC 16F87

Puerto B: de este puerto, se han utilizado los pines del B0 al B6 , los pines B2 y B5 del puerto, son los pines de la interfase AUSART, el pin que corresponde a B2 es el pin de entrada de datos y el pin que corresponde a B5 es el pin de salida de datos, en los pines B0, B1 y B3 se han conectado 3 de los 8 leds que muestran los datos del proceso de cifrado o descifrado cuando se ejecutan paso a paso, en el pin B4 se conecta un led que sirve como indicador de que la tarjeta esta lista para realizar cualquier proceso y en el pin B6 se conecta otro led que sirve como indicador de que el programa se está ejecutando.

Interrupción

En este proyecto se utilizan dos tipos de interrupciones:

- Interrupción generada por RESET: se produce cuando el usuario resetea el microcontrolador a través del switch de la tarjeta o cuando el voltaje de alimentación del PIC baja del nivel permitido, al ocurrir un reset el programa se va a la dirección 0000h.
- Interrupción por recepción de datos: se produce al inicio de los procesos, el programa del PIC se encuentra en un lazo infinito hasta que el usuario le da una orden, este lazo se interrumpe y se va a realizar los procesos y luego vuelve al lazo a esperar una nueva orden, al ocurrir esta interrupción el programa se va a la dirección 0004h.

Los registros asociados a las interrupciones son:

INTCON: REGISTRO DE CONTROL DE INTERRUPCIONES

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

bit 7 **GIE**: bit que permite las interrupciones Globales.

1 = Permite todas las interrupciones no mascarables

0 = No permite ninguna interrupción

bit 6 **PEIE**: bit que permite interrupciones por periféricos

1 = Permite todas las interrupciones no mascarables por periféricos

0 = No permite ninguna interrupción por periféricos

bit 5 **TMR0IE**: bit que permite interrupción por desborde del TMR0

1 = Permite la interrupción por desborde del TMR0

0 = No permite la interrupción por desborde del TMR0

bit 4 **INT0IE**: bit que permite interrupción externa en RB0/INT

1 = Permite la interrupción externa en RB0/INT

0 = No permite la interrupción externa en RB0/INT

bit 3 **RBIE**: bit que permite la interrupción por cambio en el puerto B

1 = Permite la interrupción por cambio en el puerto B

0 = No permite la interrupción por cambio en el puerto B

bit 2 **TMR0IF**: bit bandera de la interrupción por desborde del TMR0

1 = El registro de TMR0 se ha desbordado (debe ser limpiada por software)

0 = El registro de TMR0 no se ha desbordado

bit 1 **INT0IF**: bit bandera de la interrupción externa en RB0/INT

1 = Ha ocurrido la interrupción externa en RB0/INT ha ocurrido (debe ser limpiada por software)

0 = No ha ocurrido la interrupción externa en RB0/INT

bit 0 **RBIF**: bit bandera de la interrupción por cambio en el puerto B

1 = Al menos uno de los pines RB7:RB4 ha cambiado de estado (debe ser limpiada por software)

0 = Ninguno de los pines RB7:RB4 ha cambiado de estado

PIE1: REGISTRO 1 DE ACTIVACION DE INTERRUPCIONES POR PERIFERICOS

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE(1)	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **No implementado**: se lee como cero

bit 6 **ADIE**: bit(1) que permite la interrupción del convertidor A/D

1 = Permitida

0 = No permitida

Nota 1: Este bit no esta implementado en el PIC16F87: se lee como cero

bit 5 **RCIE**: bit que permite la interrupción por recepción AUSART

1 = Permitida

0 = No permitida

bit 4 **TXIE:** bit que permite la interrupción por transmisión AUSART

1 = Permitida

0 = No permitida

bit 3 **SSPIE:** bit que permite la interrupción del puerto serial síncrono (SSP)

1 = Permitida

0 = No permitida

bit 2 **CCP1IE:** bit que permite la interrupción de CCP1

1 = Permitida

0 = No permitida

bit 1 **TMR2IE**: bit que permite la interrupción cuando TMR2 = PR2

1 = Permitida

0 = No permitida

bit 0 **TMR1IE:** bit que permite la interrupción por desborde del TMR1

1 = Permitida

0 = No permitida

Las interrupciones quedan configuradas así:

PIE1 = b'00100000' = h20

Interrupción por recepción permitida.

INTCON = b'11000000' = hC0

Interrupciones permitidas, interrupción por periféricos permitida.

CAPITULO VI: PROGRAMA DE LAS TARJETAS

6.1 Descripción General y Flujograma

El programa de las tarjetas de cifrado/descifrado Blowfish, inicia con la declaración de las variables a utilizar, luego el programa configura los registros correspondientes al puerto A, puerto B, la interfase AUSART y las interrupciones, después de esto enciende el led que indica que el programa está listo para ejecutar el proceso que se le indique y se queda esperando en un lazo infinito, hasta que reciba una orden por parte del usuario, cuando esto ocurre, se desactivan las interrupciones, se verifica si fue una interrupción por recepción, si no lo fue, se activan las interrupciones y se vuelve al lazo infinito a esperar una nueva interrupción, si fue una interrupción por recepción, se lee el dato recibido y se verifica que opción es, las opciones posibles son:

A = Cifrar paso a paso
B = Cifrar no paso a paso (cifrar archivos de texto)
C = Descifrar paso a paso
D = Descifrar no paso a paso (descifrar archivos de texto)

Si no es ninguna de estas opciones se activan las interrupciones y se vuelve al lazo infinito.

En cualquiera de las cuatro opciones el proceso inicial es la recepción de la clave, se reciben 72 caracteres aunque el número máximo de caracteres de la clave para el algoritmo Blowfish es 56 (448 bits), se necesitan 72 caracteres (576 bits), porque se necesita realizar la función XOR con los 18 arreglos P (576 bits), los valores iniciales de los arreglos P se muestran a continuación:

&H243F6A88, &H85A308D3, &H13198A2E, &H3707344,
&HA4093822, &H299F31D0, &H82EFA98, &HEC4E6C89,
&H452821E6, &H38D01377, &HBE5466CF, &H34E90C6C,
&HC0AC29B7, &HC97C50DD, &H3F84D5B5, &HB5470917,
&H9216D5D9, &H8979FB1B

Luego de realizar la función XOR y obtener otros arreglos P, se procede a calcular los arreglos finales P, este cálculo se realiza ejecutando el algoritmo Blowfish, usando inicialmente una cadena de ceros y los otros arreglos P.

Cabe destacar que el algoritmo Blowfish utiliza Cajas S, que son 4 bloques de 256 localidades de 32 bits, es decir 4096 bytes, lo que supera la capacidad de memoria de la mayoría de los microcontroladores por lo que se decidió guardar estos datos en el programa de Visual Basic de la interfaz gráfica, de tal forma que el programa del microcontrolador envía la parte de la caja S que se necesita y el programa de la interfaz gráfica busca el valor de la caja S que se requiere y se lo envía al microcontrolador.

Los resultados finales de la ejecución del algoritmo Blowfish son los nuevos valores P que van sustituyendo a los anteriores, hasta que se calculan los 18 valores de P, estos valores son enviados a la computadora.

Después de calcular todos los arreglos P, se procede a calcular los nuevos arreglos S, el proceso es el mismo, se ejecuta el algoritmo Blowfish hasta que se hayan calculado los 1024 arreglos S, que sustituyen a los anteriores[22], estos arreglos también son enviados a la computadora .

Si la opción seleccionada es descifrar (ya sea paso a paso o no) los arreglos P tienen que invertir su orden, si la opción fue cifrar (ya sea paso a paso o no) los arreglos P se dejan tal como están.

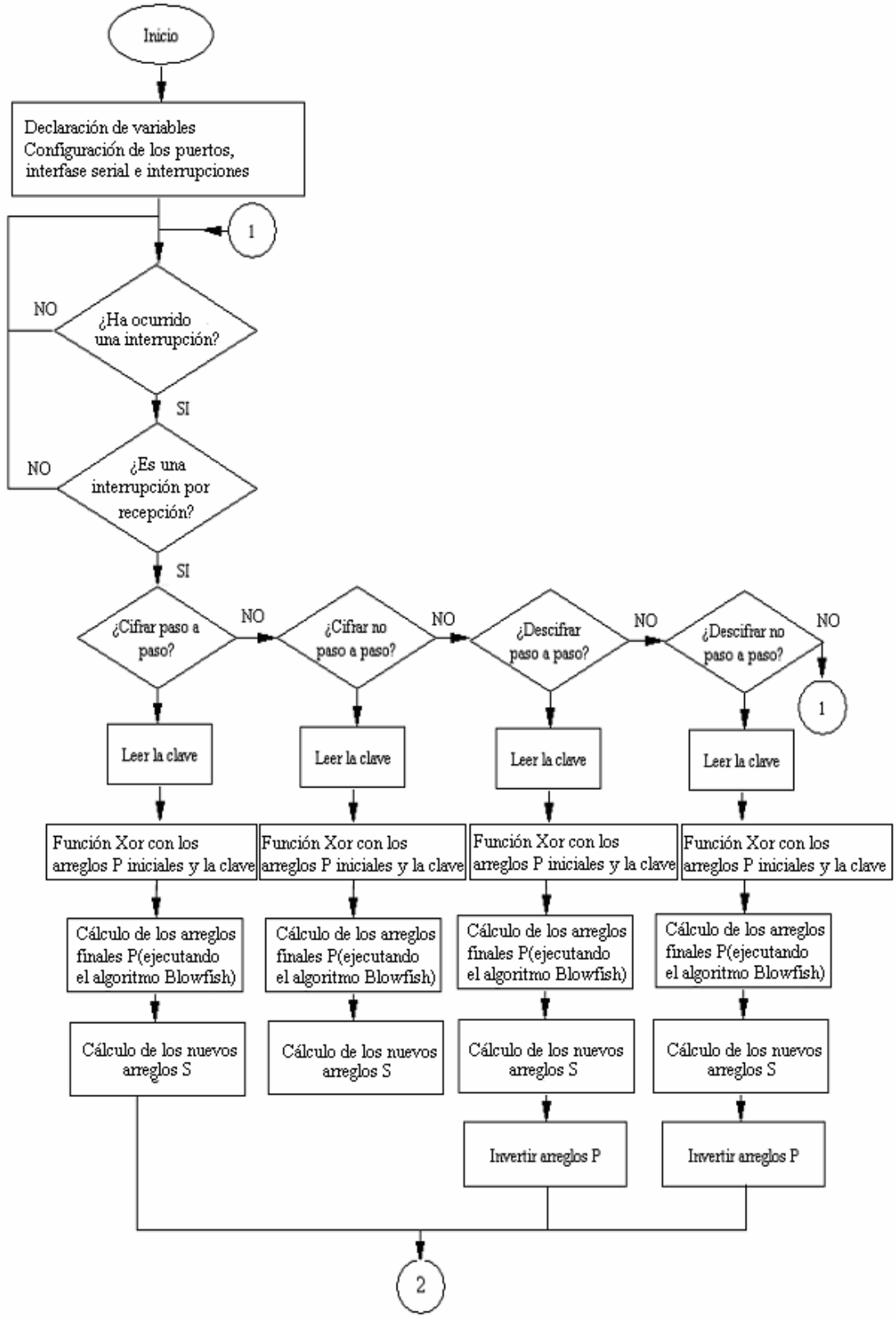
Después se recibe el número de veces que se debe ejecutar el algoritmo Blowfish para cifrar o descifrar datos, si es un proceso paso a paso este valor es uno y si es un proceso no paso a paso este valor puede estar entre 1 y 255, luego se recibe el texto a cifrar o descifrar y se ejecuta el algoritmo Blowfish el número de veces indicado, mientras el programa está procesando los datos está apagando y encendiendo un led, para indicar que el proceso se ejecuta correctamente.

si se está ejecutando un proceso paso a paso (cifrar o descifrar), el microcontrolador envía los datos de cada etapa del proceso a la computadora y también los saca por sus puertos paralelos para que se puedan ver los datos en ocho leds que están en las tarjetas, el usuario va dando la confirmación en el software de que se pueden ver nuevos datos en la pantalla y las tarjetas, ya que se necesita apreciar cada resultado, los datos se muestran de dos en dos tanto en las tarjetas como en pantalla.

Cuando se han cifrado o descifrado todos los datos, se activan las interrupciones y se vuelve al lazo infinito a esperar una nueva orden por parte del usuario[23].

[22] Los valores iniciales de los arreglos S son los que aparecen en el código del módulo de la interfaz grafica, en la sección de anexos.
[23] Para una mejor comprensión del programa ,ver en anexos el código del programa en ensamblador con comentarios

FLUJOGRAMA


Inicio
Declaración de variables
Configuración de los puertos,
interfase serial e interrupciones
1
NO
¿Ha ocurrido
una interrupción?
SI
NO
¿Es una
interrupción por
recepción?
SI
¿Cifrar paso a
paso?
NO
¿Cifrar no
paso a paso?
NO
¿Descifrar
paso a paso?
NO
¿Descifrar no
paso a paso?
NO
1
Leer la clave
Leer la clave
Leer la clave
Leer la clave
Función Xor con los
arreglos P iniciales y la clave
Función Xor con los
arreglos P iniciales y la clave
Función Xor con los
arreglos P iniciales y la clave
Función Xor con los
arreglos P iniciales y la clave
Cálculo de los arreglos
finales P(ejecutando
el algoritmo Blowfish)
Cálculo de los arreglos
finales P(ejecutando
el algoritmo Blowfish)
Cálculo de los arreglos
finales P(ejecutando
el algoritmo Blowfish)
Cálculo de los arreglos
finales P(ejecutando
el algoritmo Blowfish)
Cálculo de los nuevos
arreglos S
Cálculo de los nuevos
arreglos S
Cálculo de los nuevos
arreglos S
Cálculo de los nuevos
arreglos S
Invertir arreglos P
Invertir arreglos P
2

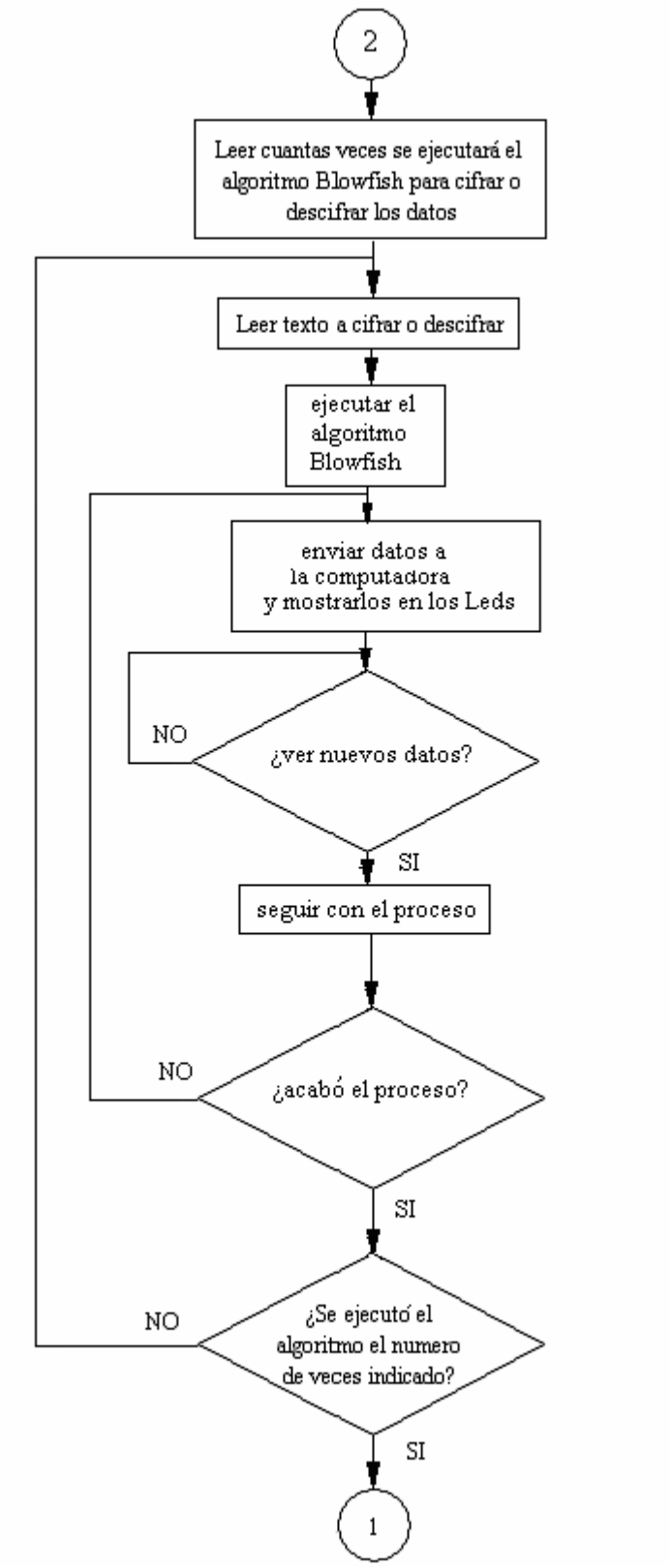


Fig. 6-1. Diagrama de flujo del programa de las tarjetas.

CAPITULO VII: INTERFAZ GRAFICA

7.1 Descripción General.

El programa de la interfaz gráfica está dividido en dos partes, la primera es para el cifrado o descifrado paso a paso y la segunda para el cifrado o descifrado no paso a paso.

Cifrado o descifrado paso a paso:
Esta parte es para que el usuario pueda ver paso a paso el proceso de cifrado o descifrado, el usuario inicia introduciendo un texto con un máximo de 8 caracteres (64 bits) (en el caso de que el usuario introdujera menos de estos caracteres, los caracteres faltantes se rellenan con el carácter ASCII correspondiente a 00), también debe introducirse una clave para cifrar o descifrar los datos con un máximo de 56 caracteres (448 bits), esta clave sea menor o igual a 56 caracteres se repite hasta lograr 72 caracteres, debido a que el programa del microcontrolador necesita recibir 576 bits (72 caracteres) para poder realizar la función XOR con 18 arreglos de 32 bits (576 bits). Como se mencionó en el programa del microcontrolador tanto esta parte de la interfaz gráfica como la parte de cifrado/descifrado no paso a paso, la computadora envía la parte de la Caja S que el microcontrolador le solicita cuando se ejecuta el algoritmo Blowfish.

Cuando el microcontrolador ha enviado los arreglos finales P y S, se procede al proceso de cifrado o descifrado, se presenta el formulario con tres esquemas el flujograma del algoritmo Blowfish, la función F y un cuadro donde se representan 13 iteraciones del algoritmo, el usuario puede ver los resultados paso a paso en las diferentes cajas de texto de estos esquemas, presionando un botón llamado "Continuar".

Cuando se finalice el proceso, el usuario podrá realizar cualquier otro proceso que desee.

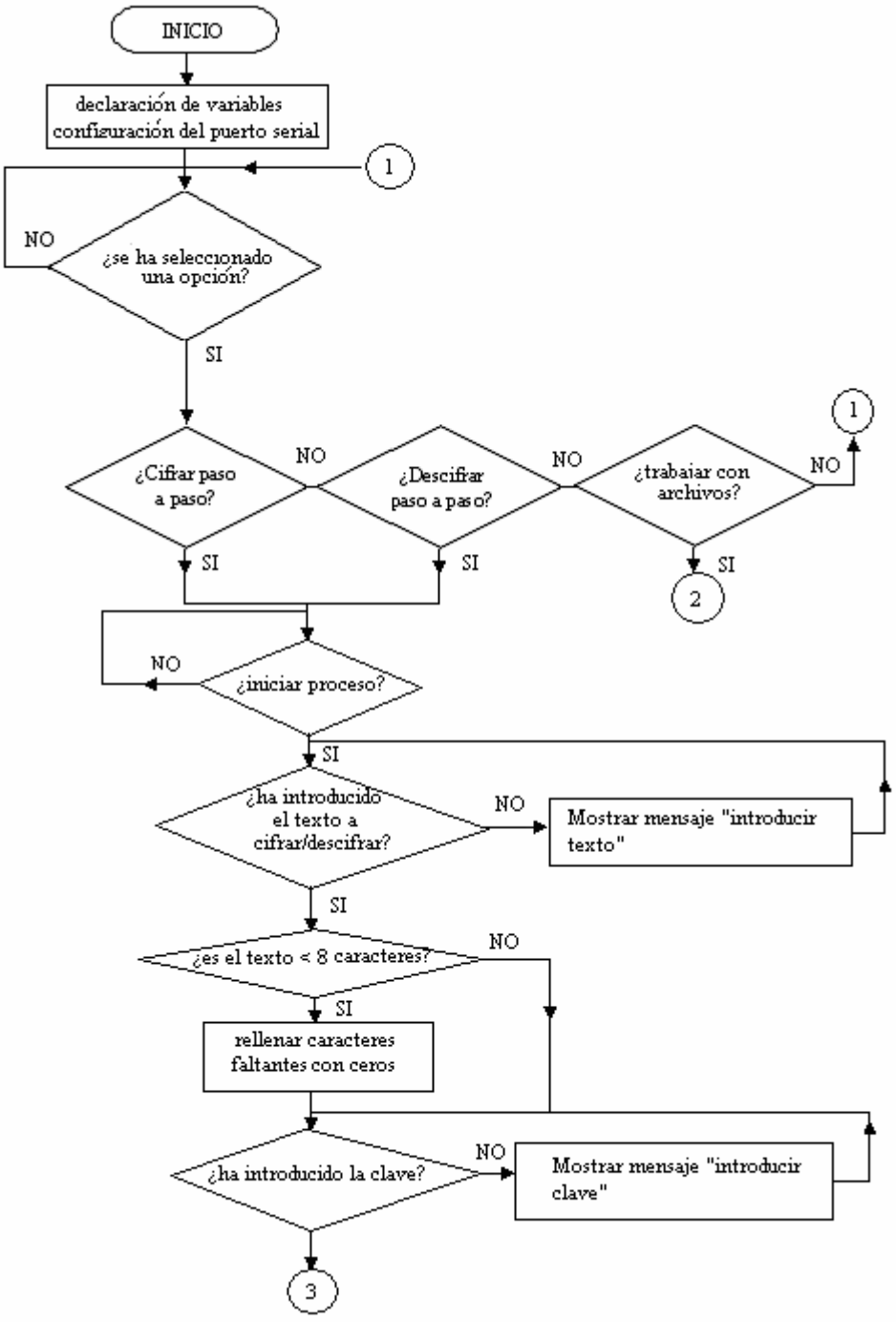
Cifrado o descifrado no paso a paso:
En esta parte lo que se manda a cifrar o descifrar es un archivo de texto, extensión .txt que puede tener desde 1 carácter hasta 2040 caracteres, también se debe introducir la clave para cifrar o descifrar que debe tener un máximo de 448 bits (56 caracteres), aunque al microcontrolador se le envían 72 caracteres como se explicó en el proceso de cifrado o descifrado paso a paso .

el programa calcula cuantas veces se debe ejecutar el algoritmo Blowfish para cifrar o descifrar todo el archivo (si el archivo tiene un número de caracteres que no es un múltiplo de 8 el número de caracteres faltantes se llena con el caracter ASCII correspondiente a 00), el programa va segmentando el texto en grupos de 8 caracteres.

Aquí no se muestran partes intermedias del programa, solo se muestra el resultado final del proceso de cifrado o descifrado, el cual el usuario puede guardar en un archivo de texto con extensión .txt, en esta parte del programa se muestra en pantalla el tiempo que tomo cifrar o descifrar un archivo[24].

[24] Para una mejor comprensión del programa, ver en anexos el código en Visual Basic del programa con comentarios, o revisar el manual de usuario.

FLUJOGRAMA


INICIO
declaración de variables
confiuración del puerto serial
1
NO
¿se ha seleccionado una opción?
SI
¿Cifrar paso a paso?
NO
¿Descifrar paso a paso?
NO
¿trabaiar con archivos?
NO
1
SI
SI
SI
2
NO
¿iniciar proceso?
SI
¿ha introducido el texto a cifrar/descifrar?
NO
Mostrar mensaje "introducir texto"
SI
¿es el texto < 8 caracteres?
NO
SI
rellenar caracteres faltantes con ceros
¿ha introducido la clave?
NO
Mostrar mensaje "introducir clave"
3

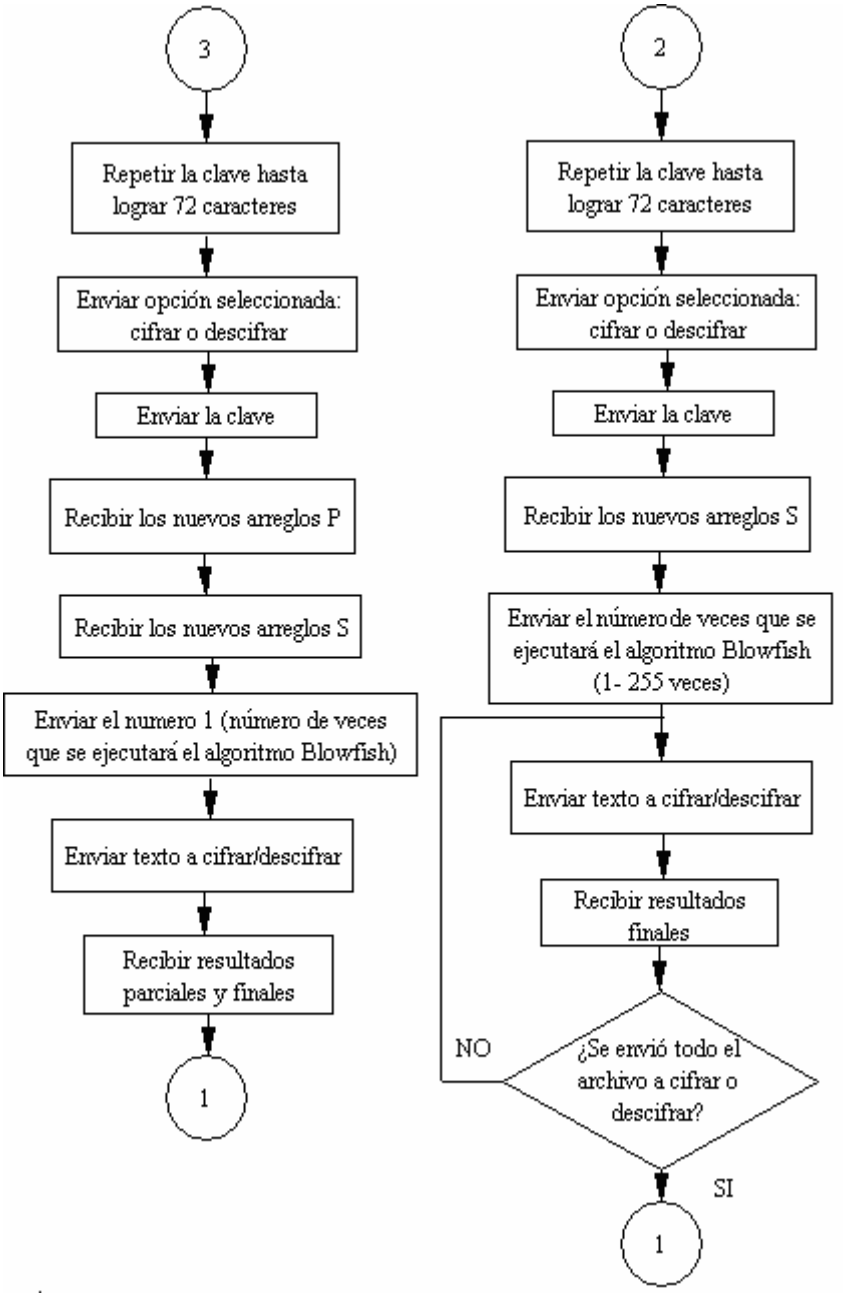


Fig. 7-2. Diagrama de flujo del programa de la interfaz gráfica.

CAPITULO VIII: MANUAL DE USUARIO.

Esta parte explica detalladamente como operar el proyecto. Los requerimientos que se necesitan o que se necesita tener y/o hacer inicialmente antes de comenzar a relizar alguna operación.

También se muestras en imágenes las formas que se verán en pantalla y que se debe hacer en cada etapa de algún proceso y los resultados que se mostrarán (dependiendo de los datos iniciales).

8.1 REQUERIMIENTOS Y ELEMENTOS NECESARIOS

REQUERIMIENTOS DEL SISTEMA

Sistema Operativo	**Windows** (probado en: Windows 98, Windows 2000 y Windows XP)
Interfase	**Serial (COM1)**
Espacio en Disco duro	**(Blow2006.exe) 140KB**

ELEMENTOS NECESARIOS

- Adaptador AC-DC Entrada: 120 VAC, 60Hz Salida: 9VDC, 300-500mA.

- Tarjeta para cifrar/descifrar mediante el algoritmo Blowfish. (desarrolada en este proyecto).
- Cable serial DB9 Hembra - DB9 Hembra.
- Computadora con puerto serial (Com1).
- Programa “Blow2006”. (desarrollado en este proyecto).

FORMA DE CONEXION

1. Conecte la tarjeta para cifrar/descifrar datos mediante el algoritmo Blowfish al puerto serial de la computadora (Com1).

2. Conecte la tarjeta a un toma corriente común (120 VAC, 60Hz) mediante el adaptador AC-DC, si la alimentación esta bien, se enciende el led amarillo (D11[25]), indicando que la tarjeta está lista para realizar los procesos. En ciertas ocasiones, otros led (verdes o naranjas), pueden encenderse debido a que inicialmente las líneas de algunos pines del pic están al aire, esto produce que las salidas de algunos puertos estén en 1 mientras no se realiza ningún proceso. Esto no altera resultados y además, siempre que se realice un proceso, estos led permanecerán apagados y no se deberán encender. En caso contrario, será una falla de tarjeta.

La forma de conexión se muestra a continuación:

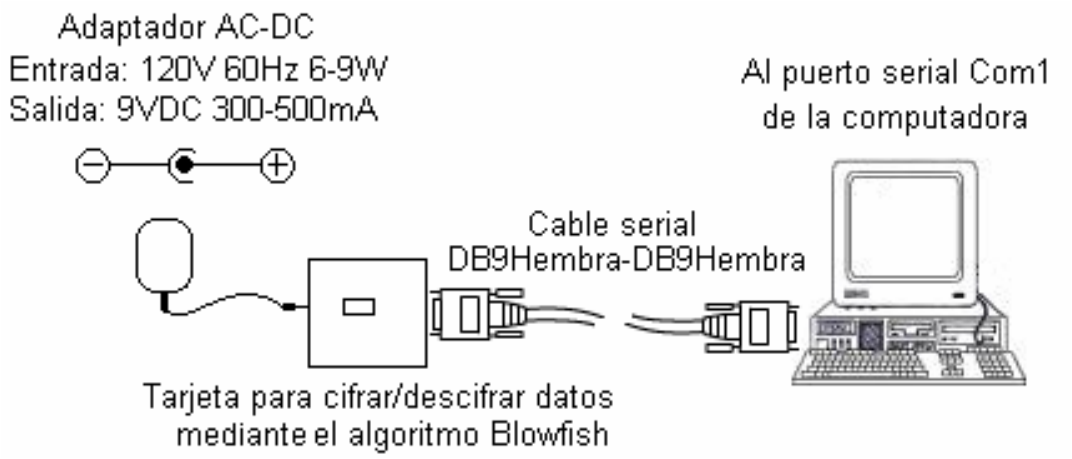


Fig. 8.1: Forma de conexión entre tarjeta y PC.

[25] Ver en el diagrama esquemático de la tarjeta en anexos

8.2 FUNCIONAMIENTO DE LAS TARJETAS

Cuando se halla conectado la tarjeta se debe cargar el programa de la interfaz gráfica "Blow2006", este programa está en formato ejecutable (.exe) para que el usuario pueda acceder a él dando doble clic, y en formato .vbp para que los usuarios que disponen de Visual Basic puedan verificar la codificación del mismo.

Al cargar el programa aparece la ventana que se muestra en la figura 8.1. Inicialmente los botones "Iniciar Proceso" y "cifrar/descifrar" están inactivos y las cajas de texto están bloqueadas hasta que el usuario seleccione una de las tres opciones disponibles:

- Cifrar paso a paso
- Descifrar paso a paso
- Trabajar con archivos de texto

Al seleccionar una de esas tres opciones se activa el botón "Iniciar Proceso", al presionarlo, si las opciones fueron cifrar o descifrar paso a paso, las cajas de texto ya permiten la introducción de caracteres, y se activa el botón "cifrar/descifrar" con el nombre del proceso que se quiere realizar, si este botón se presiona y hay un campo en blanco ya sea el de la clave o el texto a cifrar/descifrar, el programa envía un mensaje de advertencia, tal como se muestra en la figura 8.2.

Si se ha escrito tanto la clave como el texto a cifrar o descifrar aparece un mensaje de espera en la ventana como se muestra en la figura 8.3 y la tarjeta debe apagar el led indicador (amarillo) de que está lista para un proceso (D11) y tener intermitente otro led (D12[26] y de color rojo) indicando que se está realizando un proceso, si este led no está intermitente sino apagado o encendido quiere decir que ha ocurrido un error que puede deberse a una mala conexión del cable serial o una falla en la alimentación, de ser así debe resetearse la tarjeta, verificar la conexión serial con la computadora y volver a cargar el programa.

[26] Ver en el diagrama esquemático de la tarjeta en anexos

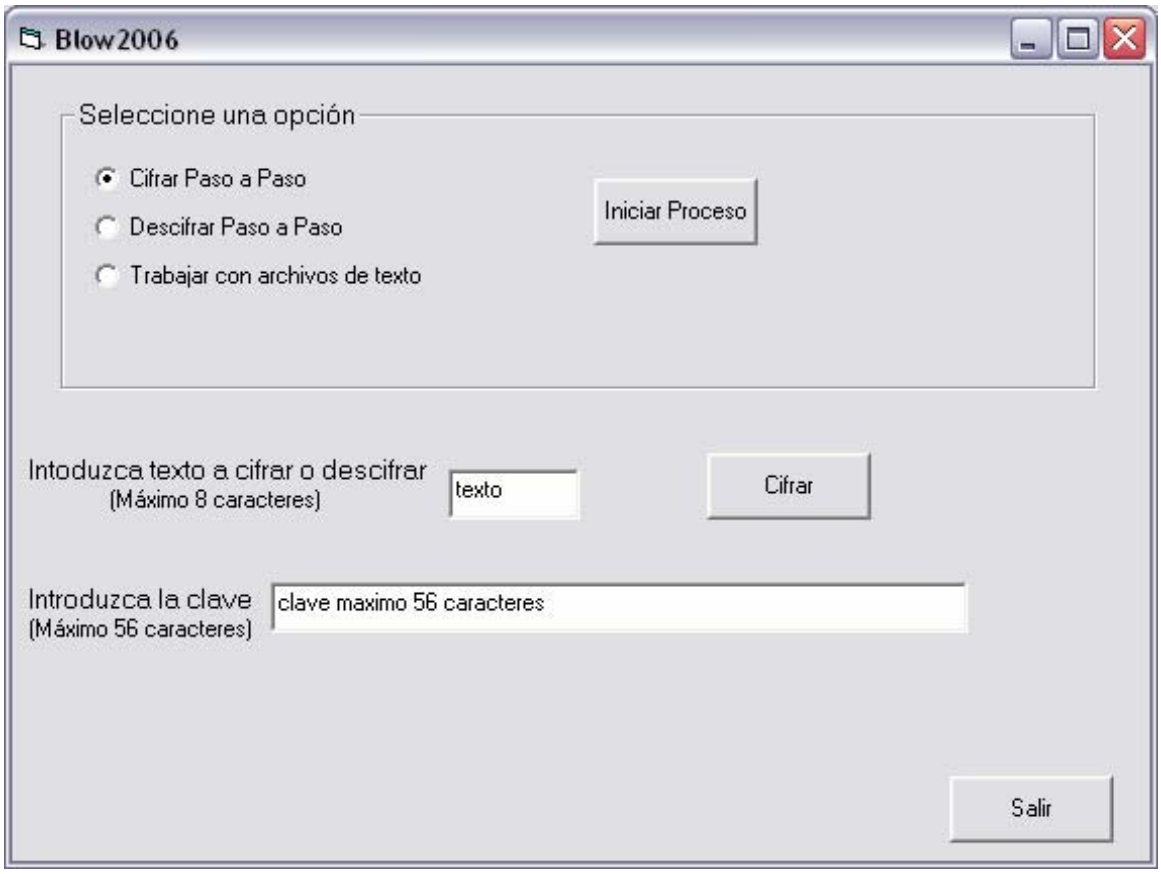


Fig. 8.2: Forma 1 del programa Blow2006.

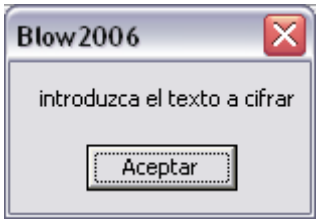


Fig. 8.2. Caja de advertencia que aparece cuando no se ha introducido texto.

En el caso de que no haya ocurrido ningún error el led seguirá intermitente mientras se calculan los arreglos P y S, esto lleva un tiempo de aproximadamente cuatro minutos.

Cuando termina el cálculo aparece la ventana con el flujograma del algoritmo Blowfish como se muestra en la figura 8.4, aquí el usuario puede ver de una forma más amigable cómo se lleva a cabo el proceso de cifrado o descifrado.

El usuario va dando clic al botón que inicialmente se llama “Iniciar” y luego cambia su nombre a “continuar” para ir viendo los datos tanto en pantalla como en la tarjeta, en la pantalla los datos aparecen tanto en ascii como en hexadecimal y en la tarjeta los datos se muestran en hexadecimal a través de ocho leds, los datos se van mostrando de dos en dos en ambas partes, es decir, en pasos de 8 bits.

Cuando se ha completado el proceso se desactiva el botón “continuar”, el usuario debe dar clic al botón “regresar” para regresar a la ventana principal, este botón “regresar” está disponible desde el principio por si el usuario quiere interrumpir el proceso y volver a la ventana principal.

Si el usuario quiere realizar el proceso de descifrado paso a paso, y acaba de realizar un cifrado paso a paso, al hacer clic en el botón “iniciar proceso” aparece en el campo texto lo que acaba de cifrar por si se quiere descifrar esto, pero si no se quiere descifrar ese texto se puede borrar y colocar el texto que el usuario desee, el proceso de descifrado paso a paso trabaja de la misma forma que el cifrado paso a paso, ver en la figura 8.5 el resultado de un proceso de descifrado paso a paso.

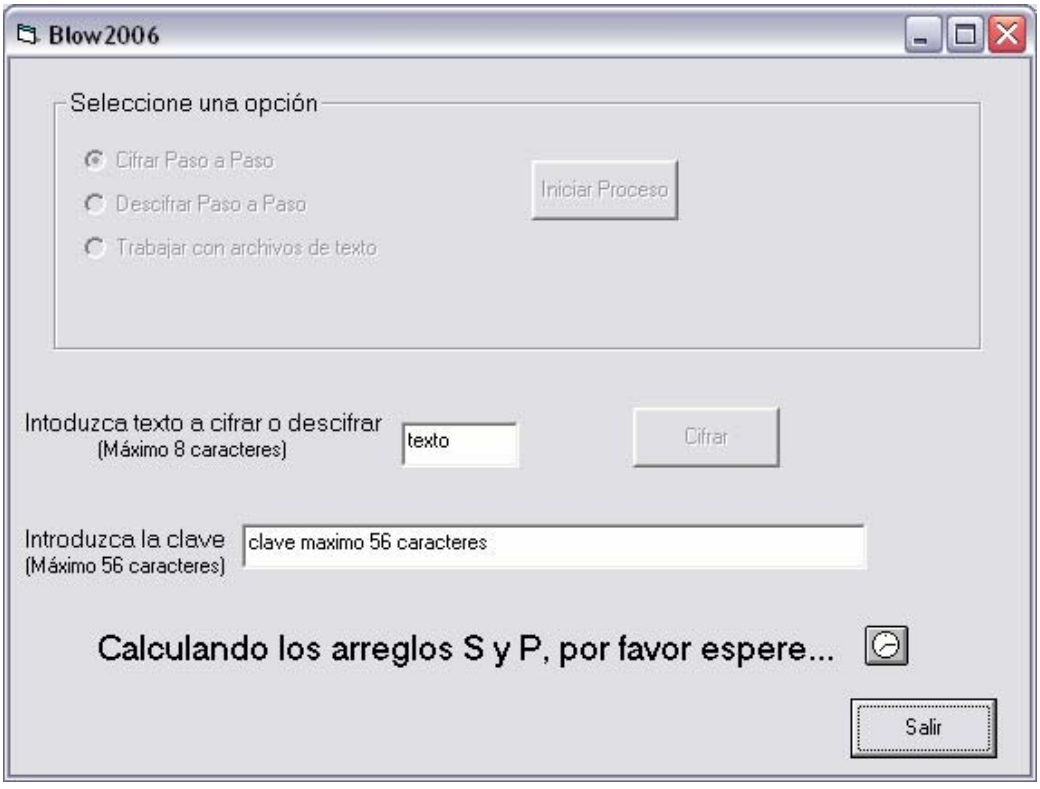


Fig. 8.3: Forma 2 del programa Blow2006.

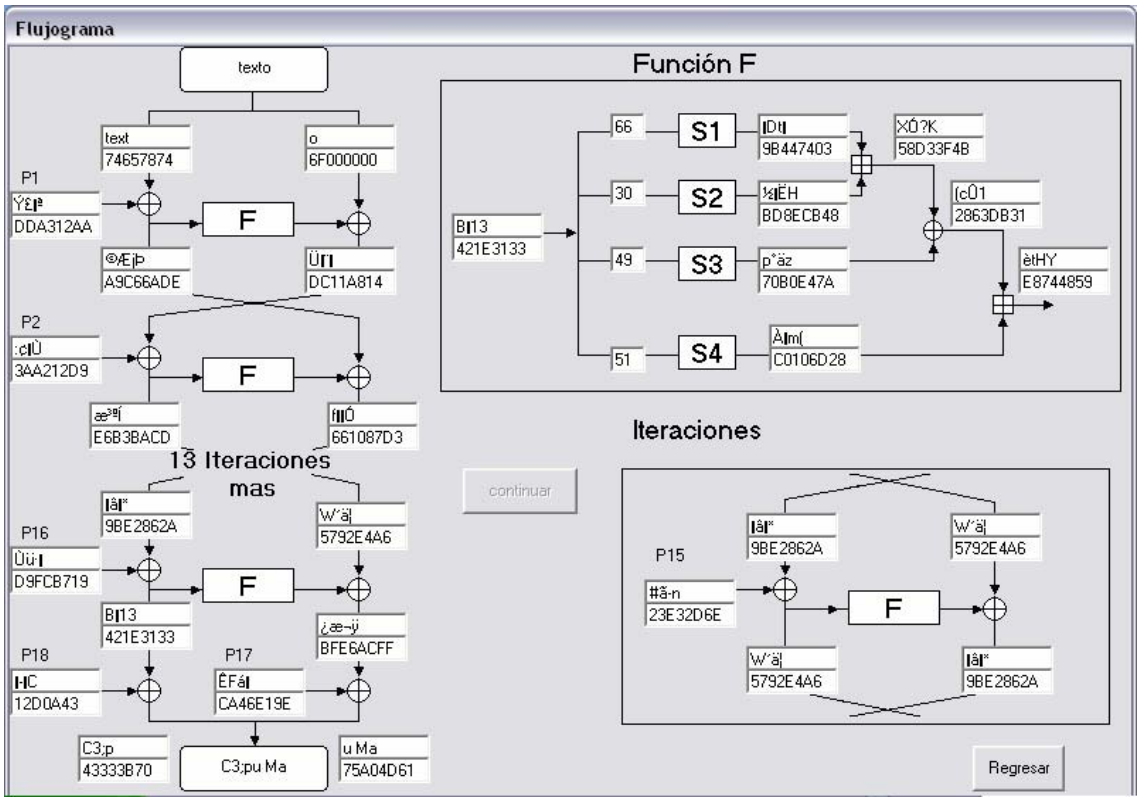


Fig. 8.4: Forma 3 del programa Blow2006 que muetra el proceso de cifrado/descifrado del algoritmo Blowfish. .

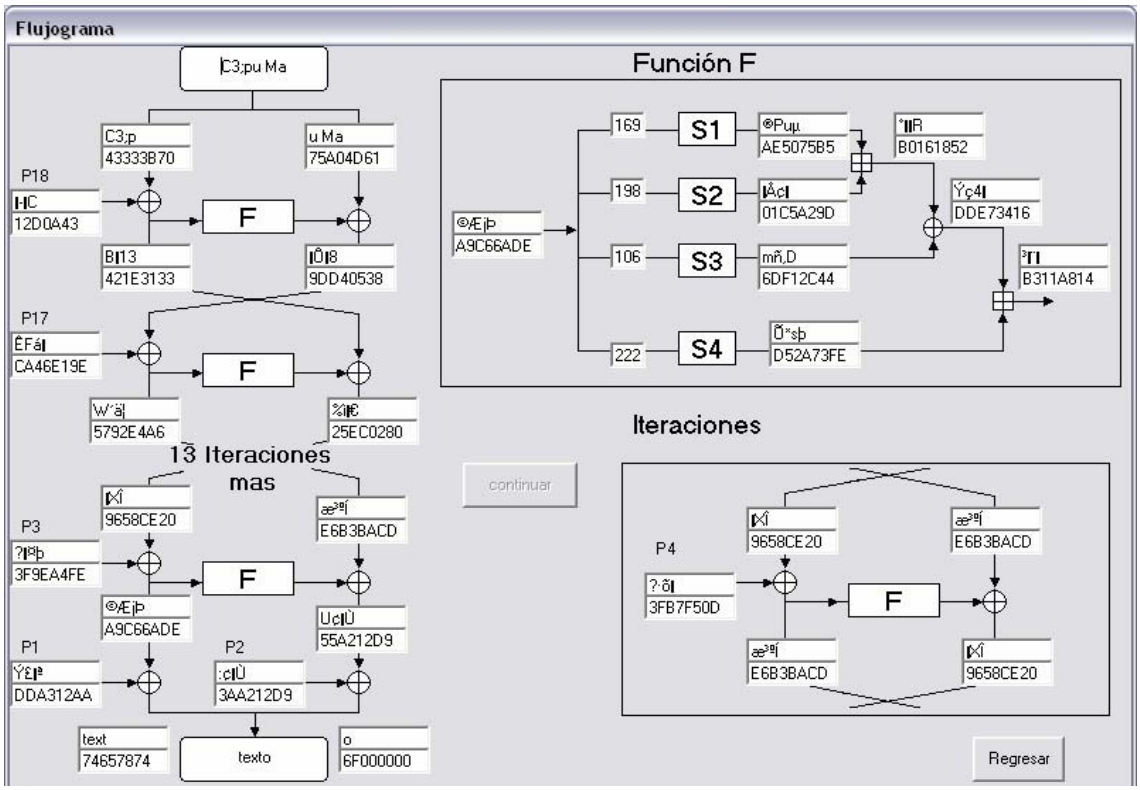


Fig. 8.5: Proceso avanzado del Blowfish cuando se está iterando con P5. En el cifrado irá aumentando y en descifrado ira disminuyendo.

Si se selecciona “Trabajar con archivos de texto” al hacer clic en el botón “Iniciar proceso” aparece una nueva ventana con dos opciones “Cifrar Archivos” y “Descifrar Archivos”, ver figura 8.6.

Al seleccionar cualquiera de las dos opciones se activa el botón “Abrir archivo”, al darle clic aparece el cuadro de diálogo Abrir, restringido para abrir solo archivos con extensión .txt, si se le da cancelar, es decir no se abre ningún archivo el botón “cifrar/descifrar” no se activa, este botón se activa hasta que se abra un archivo de texto, este archivo de texto no debe estar en blanco ni sobrepasar los 2040 caracteres (1,99KB), en cualquiera de los dos casos aparece el mensaje de advertencia correspondiente y los archivos no se abren, si el archivo es válido, su contenido se muestra en la caja de texto superior, debe introducirse la clave y dar clic sobre el botón “cifrar/descifrar”, aparece el mensaje “Procesando por favor espere..” en la ventana, se apaga el led indicador y se pone intermitente el otro led indicando que se está realizando un proceso, el tiempo que lleva el proceso depende de la velocidad de la computadora y del tamaño del archivo, ver tabla de comparación de tamaños de archivo y tiempo de procesamiento en la figura 8.7, en los leds de la tarjeta no se muestra ningún dato y en pantalla solo los datos finales.

Cuando el proceso finaliza el led intermitente se apaga, y se enciende el led que indica que la tarjeta ya está lista para otro proceso, en pantalla aparece el cuadro de diálogo Guardar, por si el usuario quiere guardar el archivo que se acaba de cifrar/descifrar, en la caja de texto inferior aparece el contenido del archivo (cifrado o descifrado).

En pantalla también aparece el tiempo que se llevó el proceso en minutos y segundos ver ver figura 8.8.

En esta ventana también aparece el botón “regresar” para volver a la ventana principal en el momento que se necesite.

Para salir o cerrar el programa “Blow2006”, el usuario puede hacer clic en el botón “Salir” de la ventana principal o en el botón “cerrar” que esta en la parte superior de la ventana principal.

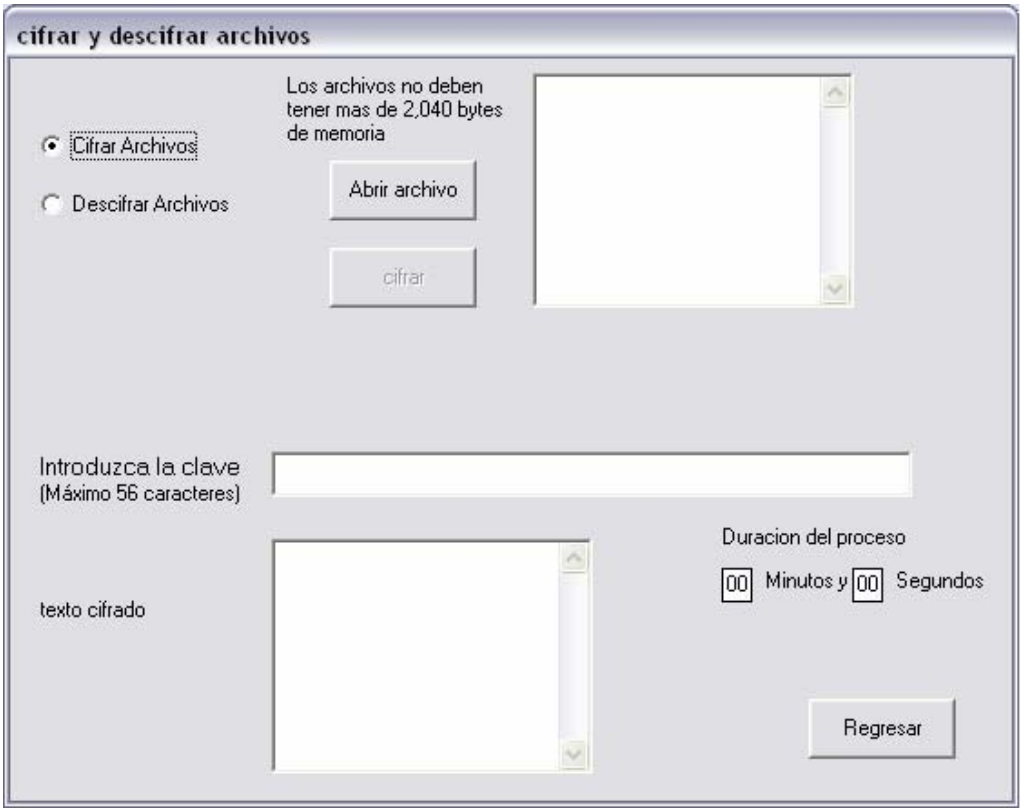


Fig. 8.6: Forma de cifrar o descifrar archivos.

8.3 TIEMPO DE EJECUCION DE ARCHIVOS DE TEXTO

Se refiere al tiempo que le toma al conjunto, tarjeta, computadora y software en terminar un proceso BlowFish. Ya se mencionó que para proceso paso a paso, se tarda alrededor de 4 minutos y puede variar en más o menos 5 segundos.

En el caso de cifrado o descrifrado de archivos de texto, el tiempo que se toma en realizar todo un proceso, dependerá del tamaño del archivo sometido al proceso (en otras palabras, de la cantidad de caracteres que contiene el archivo) que no deberá exceder los 2,040 bytes.

A continuación se presenta un tabla de comparación de tmaño de archivo contra tiempo de cifrado o descifrado. Hay que tener en cuenta que estos datos pueden variar dependiendo del tipo de computadora que se utilice. El tiempo se puede reducir para un archivo de cierto tamaño si el procesador es más rápido y se puede reducir si el procesador es más lento. El tiempo puede variar, pero no en gran medida.

TABLA DE COMPARACION

Tamaño de archivo	Tiempo de cifrado/descifrado
1,99KB (2,040 bytes)	5 Minutos 48 Segundos
1,28KB (1,312 bytes)	5 Minutos 7 Segundos
1KB (1,024 bytes)	4 Minutos 52 Segundos
550 bytes	4 Minutos 24 Segundos
200 bytes	4 Minutos 4 Segundos
8 bytes	3 Minutos 53 Segundos

Fig. 7: Tabla comparativa de tamaño de archivo contra tiempo de ejecuación.

Estos datos fueron tomados con una computadora con un procesador Pentium III (935MHz) y sistema operativo Windows XP

Fig. 8.8: Forma que aparece en el proceso de cifrado/descifrado de archivos de texto.

LISTA DE ELEMENTOS Y COSTOS.

CANTIDAD	DESCRIPCION	PRECIO UNITARIO $	TOTAL $
2	Microcontrolador PIC16F87	3,30	6,60
8	Diodo Led verde 5mm	0,20	1,60
8	Diodo Led anaranjado 5mm	0,20	1,60
2	Diodo Led rojo 5mm	0,20	0,40
2	Diodo Led amarillo 5mm	0,20	0,40
20	Resistencia 330 1/4W	0,25	5,00
6	Resistencia 10K 1/4W	0,25	1,50
4	Resistencia 1K 1/4W	0,25	1,00
2	Resistencia 100 1/4W	0,25	0,50
2	Resistencia 510 1/4W	0,25	0,50
2	Resistencia 1,5K 1/4W	0,25	0,50
4	Diodo proposito general 1N4148	0,20	0,80
4	Capacitor 15pF	0,12	0,48
2	Cristal 4MHz	4,00	8,00
2	Switch	0,30	0,60
2	Transistor BC547	0,45	0,90
2	Transistor BC558	0,45	0,90
2	Regulador de voltaje LM317	0,50	1,00
2	Conector serial DB9M	3,00	6,00
2	Conector para adaptador	0,34	0,68
2	Adaptador AC-DC. IN=110V, 60Hz O=9V,500mA	2,30	4,60
2	Base para integrado (18 pines)	0,17	0,34
2	Tarjeta para hacer circuito impreso	6,00	12,00
1	Cable serial DB9F*DB9F	2,44	2,44
		TOTAL	58,34

CONCLUSIONES

- El elemento central de la tarjeta para cifrar/descifrar datos mediante el algoritmo Blowfish es el microcontrolador PIC16F87, que se encarga de llevar a cabo los procesos de cifrado y descifrado, de comunicarse con la computadora y de mostrar los datos en los leds, este microcontrolador posee una capacidad de memoria muy buena, tiene 7 KB de memoria Flash (memoria de programa) y 368 bytes de memoria RAM (memoria de datos) y el programa de las tarjetas solamente utiliza 1.4 KB de memoria de programa y 214 bytes de memoria de datos. Este microcontrolador posee características especiales como: un amplio rango de voltaje de operación (2 a 5.5 Voltios), operación del microcontrolador en ocho diferentes modos de oscilador lo que permite un amplio rango de velocidades en que el microcontrolador puede operar (en este caso hemos utilizado el oscilador a cristal XT con un cristal de 4MHz) y la comunicación RS232 a diferentes velocidades sin requerir un oscilador externo, todo esto hace que el microcontrolador PIC16F87 sea la opción ideal para la tarjeta ya que es un elemento eficiente que cumple con las funciones requeridas de memoria y de comunicación, tiene un bajo costo y no disipa casi nada de potencia, es decir, no se calienta como otros microcontroladores, por lo tanto, para este proyecto, no es necesario utilizar un sistema de aire para evitar que el pic se caliente.

- En la comunicación serial entre la tarjeta y la computadora, se puede utilizar un circuito que convierte los voltajes que acepta el microcontrolador a los voltajes que acepta la computadora y viceversa, en lugar del circuito integrado que se usa generalmente el MAX232, pues es más económico construir este circuito que la compra de este integrado y los resultados son buenos, ya que no se generan errores de transmisión ni de niveles TTL, aunque cabe resaltar, que en otras aplicaciones de mayor importancia y de mayores recursos, no convendría utilizar este circuito equivalente, ya que la velocidad puede reducirse y se pueden dar malas transmisiones de datos. Por tanto, este circuito equivalente es bueno utilizarlo en aplicaciones donde no se requieran tasas de bits grandes y que además la transmisión serial no incluya paquetes de bits grandes, por ejemplo

superior a 128 bits. En caso contrario, resulta más conveniente utilizar este circuito equivalente, ya que solo está compuesto por un par de transistores, un diodo y cinco resistencias.

- El uso del puerto serial hace más fácil la conexión con la computadora ya que no es necesario abrir el case de la computadora como es el caso con las tarjetas basadas en el bus ISA o PCI. De esta forma, se podrá conectar y desconectar la tarjeta a cualquier PC sin problemas. Solo se tiene una limitante para el uso de este puerto y es el cable para conectar, ya que según nuestro diseño se debe utilizar un cable hembra-hembra, aunque si no se consigue se puede construir uno artesanal con el fin de solo realizar una pequeña prueba.

- Los datos de los procesos paso a paso puedan ser apreciados, tanto en la tarjeta como en la interfaz gráfica de la computadora, con esto se hace más fácil la comprensión del algoritmo Blowfish y puede hacerse al ritmo que el usuario decida.

- En la interfaz gráfica los datos se presentan de tal manera que es más fácil para el usuario apreciar como funciona el algoritmo Blowfish.

- En caso de que hardware o software presenten una falla mientras se realiza el proceso de cifrado o descifrado, la tarjeta posee un sistema para reiniciarla o también si el usuario quiere interrumpir un proceso y de esta manera se evite que se tenga que dar un reinicio de energía o desconexión manual de la transmisión para evitar una espera mientras se termina un proceso y así se evitan averías por malos usos.

- En la interfaz gráfica, la presentación de los datos tanto en ascii como en hexadecimal, permiten no solo mejor apreciación del proceso, sino también pueden servir para comprobar manualmente de que el proceso que se está realizando es el correcto, ya que para un usuario es más fácil realizar una operación lógica en hexadecimal que en ASCII.

BIBLIOGRAFIA Y ENLACES.

LIBROS DE TEXTO:

- 🕮 **[1]Applied Cryptography : protocols, algorithms, and source code in C.**
  Schneier, Bruce.
  Editorial Wiley.
  SECOND EDITION.

- 🕮 **[2]Modern Cryptography: Theory & Practice**
  Wenbo Mao
  Prentice Hall

- 🕮 **[3]Microcontroladores PIC: Diseño Práctico de Aplicaciones**
  José Ma. Angulo Usategui e
  Ignacio Angulo Martínez
  2a. Edición

ENLACES:

SITIOS QUE CONTIENEN INFORMACION SOBRE CRIPTOGRAFIA Y EL ALGORITMO BLOWFISH:

- [4]http://www.schneier.com/blowfish.html
- [5]http://www.finecrypt.net/blowfish.html
- [6]http://leo.worldonline.es/jlquijad/histo.htm
- [7]http://es.wikipedia.org/wiki/Criptograf%C3%ADa
- [8]http://rinconquevedo.iespana.es/rinconquevedo/criptografia/criptografia.htm
- www.di-mgt.com.au/crypto.html (descarga del VB6)

SITIOS QUE CONTIENEN TEMAS Y HERRAMIENTAS UTILIZADAS RELACIONADAS CON LOS PIC

- [9] http://www.todopic.com.ar

  (De esta página se descargón un documento que contiene amplia información general sobre los PICs.) El enlace de descarga es:
  http://www.todopic.com.ar/apuntes/manula_pic_man_pic.zip

- [10]http://www.jdm.homepage.dk/newpic.htm (Grabador)

- http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1475&category=devSoftware

(De esta pagina se descargo el simulador y ensamblador MPLAB IDE v7.21)

- [11]http://www.pablin.com.ar/electron/circuito/mc/ttl232/

  (Página donde esta desarrollado un circuito para no utilizar el max232)

- [12]http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n

SITIOS RELACIONADOS CON VISUAL BASIC.

- [13]http://www.monografias.com/trabajos10/visual/visual.shtml
- [14]http://www.telecable.es/personales/jrubi/index.htm?curso.htm
- http://www.abcdatos.com/tutoriales/tutorial/g205.html

  (curso de manejo puerto serial)

OTROS SITIOS.

- http://www.jimprice.com/jim-asc.htm (Tabla ascii para windows)

HOJAS TECNICAS DEL PIC16F87

- http://ww1.microchip.com/downloads/en/devicedoc/30487b.pdf

ANEXOS

Anexo 1: Códigos ASM de las tarjetas

```
;-------------------------------------------------------------------------------------------------
;Programa de las tarjetas de cifrado/descifrado Blowfish para fines didácticos
;trabaja con el puerto serial: 9600 baudios, 8 bits de datos, no paridad
;Frecuencia del cristal = 4 MHz
;-------------------------------------------------------------------------------------------------

                LIST   p=16F87     ;indica el modelo del pic

                #include "P16F87.INC"  ;se incluye la definición de los registros
internos

;***********  Variables a utilizar en la rutina de cifrado**********************

W               EQU 0x00 ;registro destino W
F               EQU 0x01 ;registro destino F
AUXI            EQU 0x68 ;variable auxiliar
DIREC           EQU 0x69 ;variable temporal donde se almacenan direcciones
DIREC1          EQU 0x6A ;variable temporal donde se almacenan direcciones
DIREC2          EQU 0x6B;variable temporal donde se almacenan direcciones
DIREC3          EQU 0x6C ;variable temporal donde se almacenan direcciones
CONTA1          EQU 0x6D ;contador que asegura que sean 32 bits los que se operen,
;también se usa para contar datos recibidos
CONTA2          EQU 0x6E ;cuenta las vueltas del algoritmo Blowfish
VARI            EQU 0x6F ;variable que indica si se utilizara banco 0-1 o banco 2
CONTA3          EQU 0x70 ;contador que cuenta las vueltas necesarias para generar
                         ;los arreglos finales de P, también se utiliza para invertir las P
                         ;para el proceso de descifrar
DIREC4          EQU 0x71 ;variable temporal donde se almacenan direcciones
DIREC5          EQU 0x72 ;variable temporal donde se almacenan direcciones
CONTA4          EQU 0x73 ;contador que indica cuantos datos de 64 bits se deben
                         ;cifrar o descifrar
CONTA5          EQU 0xE8 ;contador de cajas S
CONTA6          EQU 0x7D ;contador de los arreglos S
ACARREO         EQU 0x7E ;localidad donde se guarda el acarreo
BANDERA         EQU 0x7F ;bandera auxiliar de acarreo
INDIC           EQU 0x7C ;variable que indica si se van enviar los resultados a la
                         ;computadora
HARDW           EQU 0xE9 ;variable que guarda la dirección de los datos que se
                         ;mostraran en la tarjeta

;Los 64 bits del texto se almacenaran de la 74 a la 7B
;la Clave esta almacenada a partir de la A0 hasta la E7
;los P Xor con la clave están a partir de la 20-67
;XL se guarda de la 110 a la 113 y otras veces se guardara
;de la 74 a la 77
;XR se guarda de la 114 a la 117 y otras veces se guardara
;de la 78-7B
```

```
;el texto cifrado queda de la 74-7B

        ORG 0

        goto INITVAR

        ORG 4

        goto INTER

         ORG 5
;-------------------------------------------------------------------------------
;Esta rutina configura los registros del puerto A, de la interfase AUSART
;y las interrupciones
;-------------------------------------------------------------------------------

INITVAR bcf PORTB,0 ;limpia la salida
bcf PORTB,1 ;limpia la salida
bcf PORTB,3 ;limpia la salida
bcf PORTB,4 ;limpia la salida
bcf PORTB,6 ;limpia la salida
clrf PORTA ;limpia la salida
bsf STATUS,5 ;cambia al banco 1
bcf TRISA,0 ;configura la PIN 0 del puerto A (A0) como salida
bcf TRISA,1 ;configura la PIN 1 del puerto A (A1) como salida
bcf TRISA,2 ;configura la PIN 2 del puerto A (A2) como salida
bcf TRISA,3 ;configura la PIN 3 del puerto A (A3) como salida
bcf TRISA,4 ;configura la PIN 4 del puerto A (A4) como salida
bcf TRISB,0 ;configura al PIN 0 del puerto B (B0)como salida
bcf TRISB,1 ;configura al PIN 1 del puerto B (B1)como salida
bcf TRISB,3 ;configura al PIN 3 del puerto B (B3)como salida
bcf TRISB,4 ;configura al PIN 4 del puerto B (B4)como salida
bcf TRISB,6 ;configura al PIN 6 del puerto B (B6)como salida
bcf STATUS,5;cambia al banco 0
bsf RCSTA,7 ;activa la AUSART, los pines B2 y B5 son pines seriales
bcf STATUS,6 ;cambia al banco 1
bsf STATUS,5 ;cambia al banco 1
movlw 0x24  ;configura la USART: transmisión de 8 bits, modo asíncrono, alta
;velocidad, activa transmisión
movwf TXSTA
movlw .25  ;Baud Rate = Fosc/(16(X+1))= 9600 baudios, X=(Fosc/((Baud rate)x16))-1
movwf SPBRG ; X=(4MHz/(9600x16))-1 = 25
bsf PIE1,5 ;habilita la interrupción por recepción
bcf STATUS,6 ;cambia al banco 0
bcf STATUS,5 ;cambia al banco 0
bsf RCSTA,4 ;configura la USART para recepción continua
movlw 0xC0 ; activa las interrupciones: por periféricos
movwf INTCON
ESPERA  bsf PORTB,4 ;enciende led que indica que el micro esta listo
goto ESPERA
```

```
;-------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------
;Esta es la rutina de la interrupción
;Ocurre cuando la computadora envía una solicitud (por el puerto serial)
;en AUXI se guarda la alternativa que el usuario selecciono
;41(A)=cifrar paso a paso, 42(B)=cifrar no paso a paso 43(C)=descifrar paso a paso
;44(D)=descifrar no paso a paso
;--------------------------------------------------------------------------------------
INTER   bcf INTCON,7 ;deshabilita todas las interrupciones
bcf PORTB,0 ;limpia la salida
bcf PORTB,1 ;limpia la salida
bcf PORTB,3 ;limpia la salida
bcf PORTB,4 ;limpia la salida
bcf PORTB,6 ;limpia la salida
bcf PORTA,0
bcf PORTA,1
bcf PORTA,2
bcf PORTA,3
bcf PORTA,4
btfss PIR1,5 ;verifica si fue una interrupción por recepción
goto FINTER  ;si no fue se sale de la interrupción
call RECIBE  ;si fue por recepción se va a RECIBE a leer el dato recibido
STAR  movlw 0x40
subwf AUXI,F
decfsz AUXI,F ;si fue la opción 41(A) se va a CPAP(cifrar paso a paso)
goto PART1    ;si no se va a verificar a PART1
goto CPAP
PART1 decfsz AUXI,F;si se selecciono la opción 42(B) se va a CNPAP (cifrar no paso
a paso)
goto PART2   ;si no se va a verificar a PART2
goto CNPAP
PART2 decfsz AUXI,F ;si se selecciono la opción 43(C) se va a DESPAP (descifrar
;paso a paso)
goto PART3    ;si no se va a verificar a PART3
goto DESPAP
PART3 decfsz AUXI,F;si se selecciono la opción 44(D) se va a DESNPAP (descifrar
;no paso a paso)
goto FINTER  ;si no se sale de la interrupción
goto DESNPAP
;""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

CPAP  bcf INDIC,0 ;borra el bit 0 de INDIC, para indicar que no se envíen a la PC
                  ;resultados
call INICIO ;intermedios
call CAJAS
bsf INDIC,0 ;pone a 1 el bit 0 de INDIC, para indicar se envíen a la PC resultados
call TRABAJO
goto FINTER
;""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
```

```
CNPAP    bcf INDIC,0 ;borra el bit 0 de INDIC, para indicar que no se envíen a la PC
;resultados
call INICIO
call CAJAS
call TRABAJO
goto FINTER
;""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

DESPAP bcf INDIC,0 ;borra el bit 0 de INDIC, para indicar que no se envíen a la PC
;resultados
call INICIO
call CAJAS
call INVP
bsf INDIC,0 ;pone a 1 el bit 0 de INDIC, para indicar se envíen a la PC resultados
call TRABAJO
goto FINTER
;""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

DESNPAP bcf INDIC,0 ;borra el bit 0 de INDIC, para indicar que no se envíen a la PC
;resultados
call INICIO
call CAJAS
call INVP
call TRABAJO
goto FINTER
;""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

FINTER
bsf INTCON,7 ;activa las interrupciones
retfie       ;sale de la rutina de interrupción
;------------------------------------------------------------------------------------------

INICIO
;`````````````````````````````````````````````````````````````````````````````````````````
movlw 0x48     ;Para iniciar se lee la clave, son 448 bits o sea 56 caracteres, pero
movwf CONTA1   ;se necesita recibir 16 caracteres del inicio de la clave para hacer
movlw 0xA0     ;la funcion xor con los 18 arreglos P, por lo que se leen 56+16=72=h48
;datos
movwf FSR      ;los datos se deben guardar desde la A0 en adelante
ETQU call RECIBE
movf AUXI,W
movwf INDF
incf FSR,F
decfsz CONTA1,F ; verifica si ya se guardaron los 72 datos de 8 bits
goto ETQU ; si no sigue recibiendo y si ya lo hizo se va a PCLAV
;`````````````````````````````````````````````````````````````````````````````````````````
```

```
call PCLAV
movlw 0x09 ;luego se generan las P finales son 9 pares de P que se deben generar (P1-
;P18)
movwf CONTA3
movlw 0x20 ;las P inician desde la dirección 0x20
movwf DIREC5
call LIMPIE ;limpia donde esta el texto a cifrar
CONTINUE
call BLOWFISH
movlw 0x34 ;guarda los arreglos finales P en donde estaban los anteriores
movwf DIREC4
clrf CONTA1
VOLVER
movf DIREC4,W
movwf FSR
bsf STATUS,7
movf INDF,W
bcf STATUS,7
movwf AUXI
movf DIREC5,W
movwf FSR
movf AUXI,W
movwf INDF
incf DIREC4,F
incf DIREC5,F
incf CONTA1,F
btfss CONTA1,3 ;verifica si ya se guardaron 64 bits
goto VOLVER
decfsz CONTA3,F ;verifica si ya se calcularon todos los arreglos P si no continua
goto CONTINUE   ;si ya se calcularon los P, se va ya a CIFRAR los datos
return
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
TRABAJO
;``````````````````````````````````````````````````````````````````````````````
call RECIBE    ;se recibe la cantidad de arreglos de 64 bits a cifrar o descifrar
movf AUXI,W
movwf CONTA4   ;esto se guarda en CONTA4
ETQU1 clrf CONTA1    ;se leen los 64 bits que se van a cifrar o descifrar
movlw 0x74     ;los datos se almacenaran a partir de la 74
movwf FSR
OTRO call RECIBE   ;recibe los datos
movf AUXI,W
movwf INDF
incf FSR,F
incf CONTA1,F
btfss CONTA1,3 ;en total son 8 datos de 8 bits, verifica si ya se recibieron los 8
goto OTRO      ;sino sigue recibiendo
call BLOWFISH  ;si ya se recibieron los 64 bits se va a BLOWFISH
decfsz CONTA4,F ;verifica si ya se cifraron o descifraron todos los arreglos de 64 bits
```

```
goto ETQU1
return
;-------------------------------------------------------------------------------
CAJAS
movlw 0x04    ;rutina para calcular los nuevos arreglos S, son 4 cajas y 256
bsf STATUS,5 ;cambia al banco 1
movwf CONTA5  ;arreglos en cada una
OTRAS
bcf STATUS,5 ;cambia al banco 0
movlw 0x80   ;como el algoritmo Blowfish da dos arreglos S solo se llama
256/2=128=h80
movwf CONTA6 ;
CONTS
call BLOWFISH
decfsz CONTA6,F
goto CONTS
DECCONTA
bsf STATUS,5 ;cambia al banco 1
decfsz CONTA5,F
goto OTRAS
bcf STATUS,5 ;cambia al banco 0
return
;-------------------------------------------------------------------------------
;rutina que hace XOR la clave con los valores iniciales de P
;-------------------------------------------------------------------------------
PCLAV
movlw 0xA0 ;carga a FSR con la dirección A0 que es donde esta la clave
movwf FSR
movf INDF,W ;carga el valor que esta donde apunta el registro FSR en el primer
;caso el valor que este almacenado en la dirección A0
xorlw 0x24  ;hace una operación Xor con los primeros 8 bits de P1, el resultado
call RUTINA ;se almacena en W, luego llama a RUTINA
xorlw 0x3F  ;en w trae los siguientes 8 bits de la clave, para hacer Xor con
call RUTINA ;los otros 8 bits de P
xorlw 0x6A  ;el proceso se repite hasta llegar a P14
call RUTINA
xorlw 0x88
call RUTINA
xorlw 0x85
call RUTINA
xorlw 0xA3
call RUTINA
xorlw 0x08
call RUTINA
xorlw 0xD3
call RUTINA
xorlw 0x13
call RUTINA
xorlw 0x19
call RUTINA
```

```
xorlw 0x8A
call RUTINA
xorlw 0x2E
call RUTINA
xorlw 0x03
call RUTINA
xorlw 0x70
call RUTINA
xorlw 0x73
call RUTINA
xorlw 0x44
call RUTINA
xorlw 0xA4
call RUTINA
xorlw 0x09
call RUTINA
xorlw 0x38
call RUTINA
xorlw 0x22
call RUTINA
xorlw 0x29
call RUTINA
xorlw 0x9F
call RUTINA
xorlw 0x31
call RUTINA
xorlw 0xD0
call RUTINA
xorlw 0x08
call RUTINA
xorlw 0x2E
call RUTINA
xorlw 0xFA
call RUTINA
xorlw 0x98
call RUTINA
xorlw 0xEC
call RUTINA
xorlw 0x4E
call RUTINA
xorlw 0x6C
call RUTINA
xorlw 0x89
call RUTINA
xorlw 0x45
call RUTINA
xorlw 0x28
call RUTINA
xorlw 0x21
call RUTINA
```

```
xorlw 0xE6
call RUTINA
xorlw 0x38
call RUTINA
xorlw 0xD0
call RUTINA
xorlw 0x13
call RUTINA
xorlw 0x77
call RUTINA
xorlw 0xBE
call RUTINA
xorlw 0x54
call RUTINA
xorlw 0x66
call RUTINA
xorlw 0xCF
call RUTINA
xorlw 0x34
call RUTINA
xorlw 0xE9
call RUTINA
xorlw 0x0C
call RUTINA
xorlw 0x6C
call RUTINA
xorlw 0xC0
call RUTINA
xorlw 0xAC
call RUTINA
xorlw 0x29
call RUTINA
xorlw 0xB7
call RUTINA
xorlw 0xC9
call RUTINA
xorlw 0x7C
call RUTINA
xorlw 0x50
call RUTINA
xorlw 0xDD
call RUTINA
xorlw 0x3F
call RUTINA
xorlw  0x84
call RUTINA
xorlw  0xD5
call RUTINA
xorlw  0xB5
call RUTINA
```

```
xorlw  0xB5
call RUTINA
xorlw  0x47
call RUTINA
xorlw  0x09
call RUTINA
xorlw  0x17
call RUTINA
xorlw  0x92
call RUTINA
xorlw  0x16
call RUTINA
xorlw  0xD5
call RUTINA
xorlw  0xD9
call RUTINA
xorlw  0x89
call RUTINA
xorlw  0x79
call RUTINA
xorlw  0xFB
call RUTINA
xorlw  0x1B
call RUTINA
return

;--------------------------------------------------------------------------
;--------------------------------------------------------------------------
;rutina que almacena el resultado de la función XOR, es necesaria porque se
;trabaja con diferentes bancos
;--------------------------------------------------------------------------
RUTINA
movwf AUXI  ;salva el valor de W en AUXI para no perderlo
movlw 0x80  ; carga w con 80
subwf FSR,F ; al registro FSR le resta 80 para acceder a las direcciones
movf AUXI,W ;del banco 0, es decir la primera vez apunta a A0 al restarle
movwf INDF  ;80 queda en 20 que es donde se almacenan los nuevos arreglos P
movlw 0x80  ;cargo AUXI en W y lo guardo a partir de la dirección 20
addwf FSR,F ;le sumo 80 a FSR para volver al rango de direcciones de A0
incf FSR,F  ;en adelante, incremento FSR para seguir con los siguientes
movf INDF,W ;8 bits de la clave y finalmente guardo el valor donde apunta
return      ;ahora FSR en w y retorna
;--------------------------------------------------------------------------
;--------------------------------------------------------------------------
;Rutina para enviar datos a través del puerto serial, se le tiene que dar en auxi
;el dato a enviar
;--------------------------------------------------------------------------
```

```
ENVIAR        bsf STATUS,5 ;cambia al banco 1
movf HARDW,W
bcf STATUS,5 ;cambia al banco 0
movwf TXREG
bsf STATUS,5 ;cambia al banco 1
COMPTX       btfss TXSTA,1 ;comprueba si acabo de transmitir
goto COMPTX
bcf STATUS,5 ;cambia al banco 0
return
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
;Rutina para recibir datos a través del puerto serial, se le tiene que dar
;-------------------------------------------------------------------------
RECIBE
COMPRX  btfss PIR1,5
goto COMPRX
movf RCREG,W
movwf AUXI
return
;-------------------------------------------------------------------------
;rutina que limpia la dirección donde se pone el texto a cifrar ya que para generar
;las P finales se necesita que estén en cero
;-------------------------------------------------------------------------
LIMPIE
clrf CONTA1
movlw 0x74
movwf FSR
CLEAR
movlw 0x00
movwf INDF
incf FSR,F
incf CONTA1,F
btfss CONTA1,3 ;verifica si el bit 3 de conta 1 vale 0, es decir  si CONTA1
goto CLEAR     ;vale 8(01000), si es así termina porque ya se limpiaron los
return


;-------------------------------------------------------------------------
; Desde aqui comienza el algoritmo Blowfish
;-------------------------------------------------------------------------

BLOWFISH
clrf CONTA2 ;limpia el contador de las vueltas
movlw 0x20  ;guarda en DIREC3 la dirección donde están almacenados los arreglos P
movwf DIREC3
VUELTA
movlw 0x74  ;DIREC se carga con la dirección donde esta el texto
movwf DIREC
bcf VARI,0 ;pone a 0 el bit 0 de VARI, indica que DIREC estará en banco 0
```

```
movf DIREC3,W ;guarda DIREC 3 en DIREC1 que es donde esta las direcciones de os
;arreglos P
movwf DIREC1
bcf VARI,1 ;pone a 0 el bit 1 de VARI, indica que DIREC1 estará en banco 0
movlw 0x10 ;DIREC2 se carga con la dirección donde se guardara XL,(es 110)
movwf DIREC2
bsf VARI,2 ;pone a 1 el bit 2 de VARI, indica que DIREC2 estará en el banco 2
call EXOR  ;llama a EXOR
;````````````````````````````````````````````````````````````````````````````
;
call PAUSA
clrf CONTA1
movlw 0x10
movwf FSR
ENVR bsf STATUS,7 ;cambia al banco 2
movf INDF,W
bcf STATUS,7 ;cambia al banco 0
bsf STATUS,5 ;cambia al banco 1
movwf HARDW
bcf STATUS,5 ;cambia al banco 0
btfss INDIC,0 ;verifica si quiere que se envíe este resultado parcial
goto HOP0
call PAUSA
call TARJETA
HOP0 call ENVIAR
incf FSR,F
incf CONTA1,F
btfss CONTA1,2 ;Verifica si ya se enviaron los 32 bits
goto ENVR
;````````````````````````````````````````````````````````````````````````````
;
;````````````````````````````````````````````````````````````````````````````
;
bcf PORTB,6 ;indicador para ver que pasa las etapas
clrf CONTA1
movlw 0x18
movwf FSR
RCBR call RECIBE
movf AUXI,W
bsf STATUS,7 ;cambia al banco 2
movwf INDF
bcf STATUS,7 ;cambia al banco 0
incf FSR,F
incf CONTA1,F
btfss CONTA1,4 ;Verifica si ya se recibieron los 32x4 = 128 bits o sea 16 datos de 8
;bits
goto RCBR       ;que son los 4 arreglos S
;````````````````````````````````````````````````````````````````````````````
;
```

```
call FUN   ;llama a la rutina de la función F
movlw 0x78
movwf DIREC
bcf VARI,0
movlw 0x30
movwf DIREC1
bsf VARI,1
movlw 0x14
movwf DIREC2
bsf VARI,2
call EXOR
;````````````````````````````````````````````````````````````````````````````````
clrf CONTA1
movlw 0x14
movwf FSR
ENVR1 bsf STATUS,7 ;cambia al banco 2
movf INDF,W
bcf STATUS,7 ;cambia al banco 0
bsf STATUS,5 ;cambia al banco 1
movwf HARDW
bcf STATUS,5 ;cambia al banco 0
btfss INDIC,0 ;verifica si quiere que se envíe este resultado parcial
goto HOP4
call PAUSA
call TARJETA
call ENVIAR
incf FSR,F
incf CONTA1,F
btfss CONTA1,2 ;Verifica si ya se enviaron los 32 bits
goto ENVR1

HOP4 bcf PORTB,6 ;indicador para ver que pasa las etapas

movlw 0x14
movwf DIREC
bsf VARI,0
movf DIREC3,W ;vuelve a poner en DIREC1 el valor de DIREC3 (arreglos P)
movwf DIREC1
bcf VARI,1
movlw 0x78
movwf DIREC2
bcf VARI,2
call EXOR
;````````````````````````````````````````````````````````````````````````````````
call PAUSA
clrf CONTA1
movlw 0x78
movwf FSR
ENVR2 movf INDF,W
bsf STATUS,5 ;cambia al banco 1
```

```
movwf HARDW
bcf STATUS,5 ;cambia al banco 0
btfss INDIC,0 ;verifica si quiere que se envíe este resultado parcial
goto HOP5
call PAUSA
call TARJETA
HOP5 call ENVIAR
incf FSR,F
incf CONTA1,F
btfss CONTA1,2 ;Verifica si ya se enviaron los 32 bits
goto ENVR2

bsf PORTB,6 ;indicador para ver que pasa las etapas
;````````````````````````````````````````````````````````````````````````````
;````````````````````````````````````````````````````````````````````````````
clrf CONTA1
movlw 0x18
movwf FSR
RCBR1 call RECIBE
movf AUXI,W
bsf STATUS,7 ;cambia al banco 2
movwf INDF
bcf STATUS,7 ;cambia al banco 0
incf FSR,F
incf CONTA1,F
btfss CONTA1,4 ;Verifica si ya se recibieron los 32x4 = 128 bits o sea 16 datos de 8
;bits
goto RCBR1      ;que son los 4 arreglos S
;````````````````````````````````````````````````````````````````````````````
call FUN
movlw 0x10
movwf DIREC
bsf VARI,0
movlw 0x30
movwf DIREC1
bsf VARI,1
movlw 0x74
movwf DIREC2
bcf VARI,2
call EXOR
;````````````````````````````````````````````````````````````````````````````
clrf CONTA1
movlw 0x74
movwf FSR
ENVR3 movf INDF,W
bsf STATUS,5 ;cambia al banco 1
movwf HARDW
bcf STATUS,5 ;cambia al banco 0
btfss INDIC,0 ;verifica si quiere que se envíe este resultado parcial
goto HOP6
```

```
call PAUSA
call TARJETA
call ENVIAR
incf FSR,F
incf CONTA1,F
btfss CONTA1,2 ;Verifica si ya se enviaron los 32 bits
goto ENVR3
HOP6 bcf PORTB,6 ;indicador para ver que pasa las etapas

;``````````````````````````````````````````````````````````````````````````````
;
incf CONTA2,F
btfss CONTA2,3
goto VUELTA
movlw 0x74
movwf DIREC
bcf VARI,0
movf DIREC3,W
movwf DIREC1
bcf VARI,1
movlw 0x38
movwf DIREC2
bsf VARI,2
call EXOR
movlw 0x78
movwf DIREC
movlw 0x34
movwf DIREC2
bsf VARI,2
call EXOR
;''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
;
call PAUSA
clrf CONTA1
movlw 0x34
movwf FSR
ENVR5 bsf STATUS,7 ;cambia al banco 2
movf INDF,W
bcf STATUS,7 ;cambia al banco 0
bsf STATUS,5 ;cambia al banco 1
movwf HARDW
bcf STATUS,5 ;cambia al banco 0
btfss INDIC,0 ;verifica si quiere que se envíe este resultado parcial
goto HOP7
call PAUSA
call TARJETA
HOP7 call ENVIAR
incf FSR,F
incf CONTA1,F
btfss CONTA1,2 ;Verifica si ya se enviaron los 32 bits
goto ENVR5
bsf PORTB,6 ;indicador para ver que pasa las etapas
```

```
;````````````````````````````````````````````````````````````````````````````
;````````````````````````````````````````````````````````````````````````````
call PAUSA
clrf CONTA1
movlw 0x38
movwf FSR
ENVR4 bsf STATUS,7 ;cambia al banco 2
movf INDF,W
bcf STATUS,7 ;cambia al banco 0
bsf STATUS,5 ;cambia al banco 1
movwf HARDW
bcf STATUS,5 ;cambia al banco 0
btfss INDIC,0 ;verifica si quiere que se envíe este resultado parcial
goto HOP8
call PAUSA
call TARJETA
HOP8 call ENVIAR
incf FSR,F
incf CONTA1,F
btfss CONTA1,2 ;Verifica si ya se enviaron los 32 bits
goto ENVR4
bcf PORTB,6 ;indicador para ver que pasa las etapas
;````````````````````````````````````````````````````````````````````````````
clrf CONTA1 ;esta parte es necesaria porque cuando se están calculando los arreglos P
movlw 0x34  ; y los arreglos S se usa como texto a cifrar el ultimo resultado
movwf FSR   ;este queda desde la 134 pero el algoritmo cifra lo que esta desde la 74
HOPA bsf STATUS,7 ; por eso se pasa lo que esta desde la 134 a la 74
movf INDF,W
bcf STATUS,7
movwf AUXI
movlw 0x40  ; carga w con 40
addwf FSR,F ; al registro FSR le suma 40 para acceder a las direcciones
movf AUXI,W ;del banco 2, es decir la primera vez apunta a 34 al sumarle
movwf INDF  ;40 queda en 74 que es donde se almacenaran los nuevos arreglos P
movlw 0x40  ;cargo AUXI en W y lo guardo a partir de la dirección 74
subwf FSR,F ;le resto 40 a FSR para volver al rango de direcciones de 34(134)
incf FSR,F  ;en adelante, incremento FSR para seguir con los siguientes
movf INDF,W ;bits de los arreglos P y finalmente guardo el valor donde apunta
incf CONTA1,F
btfss CONTA1,3
goto HOPA
;-------------------------------------------------------------------------------
return
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
;Rutina que hace la función XOR con lo que tiene en DIREC y DIREC1 y lo guarda en
;DIREC2
;-------------------------------------------------------------------------------
```

```
EXOR clrf CONTA1  ;limpia conta1
SEGUIR        movf DIREC,W ;guarda en w el valor de DIREC
movwf FSR ;y luego w en FSR
btfss VARI,0 ;salta si el bit 0 de VARI vale 1
goto ETIQ    ; si el bit 0 de VARI es 0 salta a ETIQ
bsf STATUS,7 ;como el bit 0 de VARI vale 1, cambia al banco 2
ETIQ   movf  INDF,W ;pasa a w el valor que esta en la dirección donde
;apunta FSR
bcf STATUS,7 ;cambia al banco 0
movwf AUXI   ;guarda w en AUXI
movf DIREC1,W ;guarda en w el valor de DIREC1
movwf FSR     ;y luego w en FSR
btfss VARI,1 ;salta si el bit 1 de VARI vale 1
goto ETIQ1   ; si el bit 0 de VARI es 0 salta a ETIQ1
bsf STATUS,7 ;como el bit 0 de VARI vale 1, cambia al banco 2
ETIQ1 movf INDF,W  ;pasa a w el valor que esta en la dirección donde
;apunta FSR
bcf STATUS,7 ;cambia al banco 0
xorwf AUXI,F ;hace la función xor con w y con auxi el resultado lo
;guarda en auxi
movf DIREC2,W ;guarda DIREC2 en w
movwf FSR     ;y luego w en FSR
movf AUXI,W   ;pasa AUXI a W
btfss VARI,2  ;salta si el bit 2 de VARI vale 1
goto ETIQ2
bsf STATUS,7  ;pasa al banco 2
ETIQ2 movwf INDF    ;guarda w (que es el resultado de XOR)en la dirección donde
;apunta FSR
bcf STATUS,7  ;vuelve al banco 0
incf DIREC,F  ;incrementa las direcciones y el contador
incf DIREC1,F
incf DIREC2,F
incf CONTA1,F
btfss CONTA1,2;si el contador vale 4 (0100) o sea el bit 2 es igual a 1 retorna
goto SEGUIR   ;si no es asi sigue haciendo XOR
return
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
;Funcion F
;-------------------------------------------------------------------------------
FUN    movf DIREC1,W ;salva lo que tiene DIREC1 en DIREC3, son los arreglos P
movwf DIREC3
movlw 0x18
movwf DIREC  ;carga en DIREC la dirección de la primera caja S
movlw 0x1C
movwf DIREC1 ;carga en DIREC1 la dirección de la segunda caja S
movlw 0x28
movwf DIREC2;carga en DIREC2 la dirección donde se guarda el resultado
;de la funcion OR
```

```
call FUNADD  ;llama a la función FUNADD
clrf CONTA1
movlw 0x28
movwf FSR
ENVR6          bsf STATUS,7 ;cambia al banco 2
movf INDF,W
bcf STATUS,7 ;cambia al banco 0
bsf STATUS,5 ;cambia al banco 1
movwf HARDW
bcf STATUS,5 ;cambia al banco 0
btfss INDIC,0 ;verifica si quiere que se envíe este resultado parcial
goto HOP1
call PAUSA
call TARJETA
call ENVIAR
incf FSR,F
incf CONTA1,F
btfss CONTA1,2 ;Verifica si ya se enviaron los 32 bits
goto ENVR6

HOP1    bsf PORTB,6 ;indicador para ver que pasa las etapas
movlw 0x28
movwf DIREC
bsf VARI,0 ;se trabajara en banco 2
movlw 0x20
movwf DIREC1
bsf VARI,1 ;se trabajara en banco 2
movlw 0x2C
movwf DIREC2
bsf VARI,2 ;se trabajara en banco 2
call EXOR

clrf CONTA1
movlw 0x2C
movwf FSR
ENVR7          bsf STATUS,7 ;cambia al banco 2
movf INDF,W
bcf STATUS,7 ;cambia al banco 0
bsf STATUS,5 ;cambia al banco 1
movwf HARDW
bcf STATUS,5 ;cambia al banco 0
btfss INDIC,0 ;verifica si quiere que se envíe este resultado parcial
goto HOP2
call PAUSA
call TARJETA
call ENVIAR
incf FSR,F
incf CONTA1,F
btfss CONTA1,2 ;Verifica si ya se enviaron los 32 bits
goto ENVR7
```

```
HOP2    bcf PORTB,6 ;indicador para ver que pasa las etapas
movlw 0x2C
movwf DIREC
movlw 0x24
movwf DIREC1
movlw 0x30
movwf DIREC2
call FUNADD
;``````````````````````````````````````````````````````````````````````
;

clrf CONTA1
movlw 0x30
movwf FSR
ORAS1   bsf STATUS,7 ;cambia al banco 2
movf INDF,W
bcf STATUS,7 ;cambia al banco 0
bsf STATUS,5 ;cambia al banco 1
movwf HARDW
bcf STATUS,5 ;cambia al banco 0
btfss INDIC,0 ;verifica si quiere que se envíe este resultado parcial
goto HOP3
call PAUSA
call TARJETA
call ENVIAR
incf FSR,F
incf CONTA1,F
btfss CONTA1,2 ;verifica si se enviaron los 32 bits
goto ORAS1
HOP3    bsf PORTB,6 ;indicador para ver que pasa las etapas
return
;-------------------------------------------------------------------
;-------------------------------------------------------------------
;Funcion add, hace una suma con lo que tiene DIREC y DIREC1 y lo guarda en
DIREC2
;no toma en cuenta el acarreo en el ultimo digito
;-------------------------------------------------------------------
FUNADD      clrf CONTA1
movlw 0x03
addwf DIREC,F
addwf DIREC1,F
addwf DIREC2,F
clrf BANDERA
clrf ACARREO
CONT    btfss BANDERA,0
goto NOAC
movlw 0x01
movwf ACARREO
bcf BANDERA,0
NOAC    movf DIREC,W
```

```
movwf FSR
bsf STATUS,7
movf INDF,W
bcf STATUS,7
movwf AUXI
movf DIREC1,W
movwf FSR
bsf STATUS,7
movf INDF,W
bcf STATUS,7
bcf STATUS,0  ;borra la bandera de acarreo
addwf AUXI,F  ;realiza la suma
btfsc STATUS,0 ;verifica si hubo acarreo
bsf BANDERA,0
VEROTRO movf ACARREO,W
bcf STATUS,0
addwf AUXI,F    ;suma el acarreo
clrf ACARREO
btfsc STATUS,0 ;Verifica si hubo acarreo
bsf BANDERA,0
movf DIREC2,W
movwf FSR
movf AUXI,W
bsf STATUS,7
movwf INDF
bcf STATUS,7
decf DIREC,F
decf DIREC2,F
decf DIREC1,F
incf CONTA1,F
btfss CONTA1,2
goto CONT
return
;--------------------------------------------------------------------------
;--------------------------------------------------------------------------
;Rutina que hace una pausa antes de enviar los datos para que el usuario
;pueda apreciar los resultados tanto en Hardware como en software
;la pausa termina hasta que recibe el OK de la computadora que en este caso
;es el envio de una "E"
;--------------------------------------------------------------------------
PAUSA
BUCLE1 call RECIBE
movlw 0x45
bcf STATUS,2
xorwf AUXI,W
btfss STATUS,2
goto BUCLE1
return
;--------------------------------------------------------------------------
;Rutina que invierte los arreglos P para los procesos de descifrado
```

```
;-------------------------------------------------------------------------
INVP
movlw 0x20
movwf DIREC4
movlw 0xA0
movwf DIREC5
movlw 0x48    ;numero de arreglos P
movwf CONTA3
MASP
call INVERT
decfsz CONTA3,F
goto MASP
movlw 0x12
movwf CONTA1
movlw 0xA0
movwf DIREC4
movlw 0X6C
movwf DIREC5
MASP1
decf DIREC5,F
movlw 0x07    ;a DIREC5 se le resta 8 porque esa es la longitud de cada arreglo P
subwf DIREC5,F ;los arreglos se van guardando desde la dirección mas baja a la mas
;alta
movlw 0x04
movwf CONTA3
IVTR
call INVERT
decfsz CONTA3,F
goto IVTR
decfsz CONTA1,F
goto MASP1
return
;-----------------------------------------------------------------------------------
;.....................................................................................
INVERT
movf DIREC4,W ;Rutina que pasa lo que tiene una dirección a otra
movwf FSR
movf INDF,W
movwf AUXI
movf DIREC5,W
movwf FSR
movf AUXI,W
movwf INDF
incf DIREC4,F
incf DIREC5,F
return
;.....................................................................................
;-----------------------------------------------------------------------------------
;rutina que muestra los datos en hardware, primero verifica si el usuario presiona el
;botón de la interfaz grafica, si es así quiere decir que no quiere ver los resultados
```

```
;anteriores en hardware, si presiona el botón de la tarjeta muestra los resultados en
;Hardware
;--------------------------------------------------------------------------------
TARJETA
bsf STATUS,5 ;cambia al banco 1
movf HARDW,W  ;guardamos el dato que esta en HARDW en AUXI para no perder el
;dato
bcf STATUS,5 ;cambia al banco 0
movwf AUXI
bcf PORTA,0
bcf PORTA,1
bcf PORTA,2
bcf PORTA,3 ;AUXI tiene al dato que se va a mostrar
btfsc AUXI,0 ;si el bit 0 de AUXI es cero, sigue verificando los demás bits
bsf PORTA,0 ;si es 1 entonces se saca un 1 en el pin 0 del puerto A, si es cero no se
;hace nada
btfsc AUXI,1;porque arriba se pusieron a cero estos bits
bsf PORTA,1 ;lo mismo para los demás bits
btfsc AUXI,2
bsf PORTA,2
btfsc AUXI,3
bsf PORTA,3
bcf PORTB,0   ;limpia los pines del puerto B donde se muestra la parte mas
;significativa
bcf PORTB,1   ;del dato que se quiere ver
bcf PORTB,3
bcf PORTA,4
btfsc AUXI,4 ;si el bit 4 de AUXI es cero, sigue verificando los demás bits
bsf PORTA,4 ;si es 1 entonces se saca un 1 en el pin 4 del puerto A, si es cero no se
;hace nada
btfsc AUXI,5;porque arriba se pusieron a cero estos bits
bsf PORTB,0 ;lo mismo para los demás bits
btfsc AUXI,6
bsf PORTB,1
btfsc AUXI,7
bsf PORTB,3
return
;--------------------------------------------------------------------------------------------

END
```

Anexo 2: Códigos en Visual Basic de la interfaz gráfica

FORMA 1

```
Option Explicit
Public selecta As String
Public texto As String
Public key As String
Public opcion As String
Public s1 As Variant
Public s2 As Variant
Public s3 As Variant
Public s4 As Variant
Public s11 As Variant
Public s22 As Variant
Public s33 As Variant
Public s44 As Variant
Public cs1 As Variant
Public cs2 As Variant
Public cs3 As Variant
Public cs4 As Variant
Public cajas As Variant
Public buferentrada As String
Public conta As Integer
Public auxi As String
Public archivos As String
Public v5 As String
Public v6 As Double
Public temp As String
Public tempor As String
Public contar As Integer
Private Sub Form_Load()
With MSComm1
.CommPort = 1
.Settings = "9600,N,8,1" 'configuración del puerto serial 9600 baudios, no paridad
                         '8 bits de datos, 1 bit de paro

.InputMode = comInputModeText 'recepcion en modo texto
.InputLen = 0 'el control MSComm leera todo el bufer de recepcion
.PortOpen = True 'abre el puerto serial
End With
End Sub
```

```
Private Sub Optcifpap_Click() 'botón de opción
'guarda en selecta la opción seleccionada
selecta = "cifpap"
Cmdiniciarproc.Enabled = True 'activa el botón "Iniciar proceso"
Cmdcifrardescifrar.Enabled = False 'desactiva el botón cifrar/descifrar
Text1.Locked = True 'bloquea las cajas de texto de la clave y el texto a cifrar
Text2.Locked = True
End Sub
Private Sub Optdescifpap_Click() 'igual que en el otro botón de opción
selecta = "descifpap"
Cmdiniciarproc.Enabled = True
Cmdcifrardescifrar.Enabled = False
Text1.Locked = True
Text2.Locked = True
End Sub
Private Sub Optarchivos_Click() 'igual que en el otro botón de opción
selecta = "archivos"
Cmdiniciarproc.Enabled = True
Cmdcifrardescifrar.Enabled = False
Text1.Locked = True
Text2.Locked = True
End Sub
Private Sub Cmdiniciarproc_Click()
Text1.Text = "" 'limpia las caja de texto
Text2.Text = ""
Select Case selecta 'segun la opción que se haya guardado en "selecta"
'asi llama a su respectiva rutina
   Case "cifpap"
      Call cifpap
   Case "descifpap"
      Call descifpap
   Case "archivos"
      Call descifnopap
End Select
End Sub
Private Sub cifpap() 'si se escogió cifrar paso a paso
Text1.Locked = False 'permite ya escribir el texto y la clave
Text2.Locked = False
Cmdcifrardescifrar.Caption = "Cifrar" 'indica en la pantalla que es un proceso de
cifrado
Cmdcifrardescifrar.Enabled = True 'activa el botón cifrar/descifrar
End Sub
Private Sub descifpap() 'si se escogió descifrar paso a paso
Text1.Text = archivos
texto = archivos
Text1.Locked = False    'permite ya escribir el texto y la clave
Text2.Locked = False
Cmdcifrardescifrar.Caption = "Descifrar" 'indica en la pantalla que es un proceso
'de descifrado
Cmdcifrardescifrar.Enabled = True 'activa el botón cifrar/descifrar
```

```
End Sub
Private Sub descifnopap() 'si se escogió trabajar con archivos
Form2.Show 'muestra la forma del flujograma
Unload Me  'descarga y oculta este forma
End Sub
Private Sub Cmdcifrardescifrar_Click()
Dim M As String
If Text1.Text = "" Then 'si el texto o la clave no han sido introducidas
MsgBox "introduzca el texto a cifrar" 'muestra un mensaje de advertencia
Exit Sub 'y se sale de la rutina hasta que se introduzcan la clave y el texto
End If

If Text2.Text = "" Then
MsgBox "introduzca la clave a cifrar"
Exit Sub
End If
Text1.Locked = True
Text2.Locked = True
auxi = "00"
auxi = Chr(hexascii(auxi))
If Len(texto) <> 8 Then 'si la longitud del texto es menor a 8
'los caracteres faltantes se rellenan con el caracter ascii equivalente a 00
texto = texto & String(8 - Len(texto), auxi)
End If

key = Text2.Text 'el usuario puede introducir un maximo de 56 caracteres
M = key & key  'la clave debe ser de 72 caracteres por lo cual iniciamos duplicándola
Do While Len(M) < 72 'si esto no basto seguimos añadiendo la clave hasta lograr un
 M = M & key 'numero >= a 72, aunque el algoritmo Blowfish requiere 56 caracteres
'de la clave como maximo, se le agregan mas caracters porque asi lo requiere
'el programa del pic para poder hacer la funcion xor con los 18 arreglos P
'ya que con 56 carcteres solo llega a P14

Loop
   If Len(M) > 72 Then 'si nos pasamos de 72 caracteres, quitamos los caracteres
'sobrantes
      M = Mid(M, 1, 72)
    End If
key = M

Cmdiniciarproc.Enabled = False 'desactiva el botón "Iniciar proceso"
Optcifpap.Enabled = False      'desactiva los botones de opción
Optdescifpap.Enabled = False
Optarchivos.Enabled = False
Cmdcifrardescifrar.Enabled = False 'desactiva este botón "cifrar/descifrar"
Call Blowfish  'llama a la rutina blowfish
Optcifpap.Enabled = True       'activa los botones de opción
Optdescifpap.Enabled = True
Optarchivos.Enabled = True
Cmdiniciarproc.Enabled = True 'activa el botón "Iniciar proceso"
```

```
End Sub

Private Sub Blowfish()
If selecta = "cifpap" Then 'segun la opción seleccionada se envia A o B al
'microcontrolador
opcion = "A"
End If

If selecta = "descifpap" Then
opcion = "C"
End If
Call enviar 'envia la opción seleccionada por el usuario
opcion = key
Call enviar 'envia la clave
Call cajasS 'inicializa los valores de Cajas S
Call envrecib 'llama a la rutina envrecib
End Sub

Private Sub envrecib()
Dim i As Integer
Dim j As Integer
Label6.Visible = True
Image1.Visible = True
If selecta = "cifpap" Then
   i = 0
   For conta = 1 To 9 Step 1 'rutina para mostrar los arreglos P son 18
   '(9 pares)
   Call algoritmo
   pboxes(i) = temp
   'se guardan los nuevos arreglos P
   pboxes(i + 1) = tempor
   i = i + 2
   Next
Else
   i = 17
   For conta = 1 To 9 Step 1 'rutina para mostrar los arreglos P son 18
   '(9 pares)
   Call algoritmo
   pboxes(i) = temp
   'se guardan los nuevos arreglos P
   pboxes(i - 1) = tempor
   i = i - 2
   Next
End If

For i = 0 To 3 Step 1
   For j = 0 To 255 Step 2 'rutina para obtener los nuevos valores de cajas S
'son 4 cajas, con 256 arreglos
      Call algoritmo
      v5 = temp           'los arreglo S se pasan a decimal
```

```
      Call asciidec
      sboxes(i, j) = v6    'se guardan los nuevos arreglos S
      v5 = tempor
      Call asciidec
      sboxes(i, j + 1) = v6
   Next j
Next i
Label6.Visible = False
Image1.Visible = False
opcion = Chr(hexascii("01")) 'envia el caracter ascii equivalente a 01, que es el
'numero de veces que se ejecutará el algoritmo Blowfish
Call enviar '(se hace asi porque el puerto envia y recibe siempre en ascii)
opcion = texto 'envia el texto a cifrar
Call enviar
Hide
Form3.Show
End Sub
Public Sub enviar()
MSComm1.Output = opcion
Do While Form1.MSComm1.OutBufferCount > 0 'rutina de envio, espera hasta que
'se terminen de enviar todos los caracteres
Loop
End Sub
Public Sub recibir()
Do While MSComm1.InBufferCount < 4 'rutina de recepcion, espera hasta recibir
'4 caracteres (32 bits)
DoEvents 'sentencia para poder interrumpir este lazo
Loop
buferentrada = MSComm1.Input 'lee el bufer de entrada
End Sub
Private Sub algoritmo()    'algoritmo Blowfish que no muestra resultados parciales
For contar = 1 To 8 Step 1
      Call aceptar    'las rutinas "aceptar" son confirmaciones son automaticas
      Call recibir
      Call recibirS   'en este caso recibe que valores de caja S necesita y la
      Call enviarS    'interfaz los busca y luego se los envia
      Call aceptar
      Call recibir
      Call recibirS
      Call enviarS
Next
      Call aceptar
      Call recibir
      temp = buferentrada
      Call aceptar
      Call recibir
      tempor = buferentrada
End Sub

Public Sub recibirS() 'rutina para encontrar los valores de caja S requerido
```

```
'se reciben los cuatro caracteres unidos y se deben separar
s1 = Mid(buferentrada, 1, 1) 'separa el primer caracter de la cadena
s2 = Mid(buferentrada, 2, 1) 'separa el segundo caracter de la cadena
s3 = Mid(buferentrada, 3, 1) 'separa el tercer caracter de la cadena
s4 = Mid(buferentrada, 4, 1) 'separa el cuarto caracter de la cadena

cs1 = Asc(s1) 'convierte a decimal estos valores
cs2 = Asc(s2)
cs3 = Asc(s3)
cs4 = Asc(s4)

s1 = sboxes(0, cs1) 'selecciona de las cajas S el valor requerido
s2 = sboxes(1, cs2)
s3 = sboxes(2, cs3)
s4 = sboxes(3, cs4)

s11 = Hex(s1) 'convierte ese valor a Hexadecimal
s22 = Hex(s2)
s33 = Hex(s3)
s44 = Hex(s4)

cajas = s11   'luego lo convierte a ASCII
Call convertirha
s1 = cajas   'guarda el nuevo valor que será el que se debe enviar

cajas = s22
Call convertirha
s2 = cajas

cajas = s33
Call convertirha
s3 = cajas

cajas = s44
Call convertirha
s4 = cajas
End Sub
Public Sub enviarS()
opcion = s1  'envia los valores de caja S requerido
Call enviar
opcion = s2
Call enviar
opcion = s3
Call enviar
opcion = s4
Call enviar
End Sub
Public Sub convertirha() 'rutina que agrega ceros, si algun valor de caja S tiene
'una longitud menor a 8
Dim msg As Variant
```

```
Dim i As Integer
Dim longt As Integer
longt = Len(cajas)
If longt < 8 Then
cajas = String(8 - Len(cajas), "0") & cajas 'convierte a ascii todo
'el valor de caja S
End If
msg = Empty
For i = 1 To 8 Step 2
msg = msg & Chr(hexascii(Mid(cajas, i, 2)))
Next i
cajas = msg
End Sub
Public Function hexascii(ByVal sNumero As String) As Double
Dim inicio As Integer  'rutina que convierte un valor hexadecimal a ASCII
Dim suma As Integer
Dim digito As Variant
Dim valor As Variant
Dim monto As Variant
Dim i As Integer
Dim potencia As Integer
Dim longt As Integer
inicio = 2
suma = 0
For i = 1 To 2
digito = Mid(sNumero, i, 1)
If IsNumeric(digito) Then 'verifica si es un numero o una letra
valor = CInt(digito)      'si es un numero, lo pasa a entero
Else
valor = Asc(digito) - 55 'si es una letra la pasa a decimal y le resta 55
End If
potencia = inicio - 1
monto = valor * 16 ^ potencia
suma = suma + monto
inicio = inicio - 1
Next i
hexascii = suma
End Function
Public Sub asciidec() 'subrutina para convertir de ascii a decimal
Dim v1 As Byte
Dim v2 As Byte
Dim v3 As Byte
Dim v4 As Byte
v1 = Asc(Mid(v5, 1, 1)) 'se convierten a decimal los valores ascii
v2 = Asc(Mid(v5, 2, 1))
v3 = Asc(Mid(v5, 3, 1))
v4 = Asc(Mid(v5, 4, 1))
'rutina que finalmente encuentra el valor decimal de todo el arreglo ascii
v6 = ((v1 And &H7F) * &H1000000) Or (v2 * &H10000) Or (CLng(v3) * &H100) Or
v4
```

```
  If v1 And &H80 Then
      v6 = v6 Or &H80000000
  End If
End Sub
Public Sub aceptar()
opcion = "E"  'se envia una E al microcontrolador para indicarle
'que ya puede enviar los nuevos datos

Call enviar
opcion = ""
End Sub
Private Sub cmdsalir_Click() 'botón salir, cierra el puerto y finaliza el programa
MSComm1.PortOpen = False
End
End Sub
Private Sub Text1_Change()
texto = Text1.Text
End Sub
```

FORMA 2

```
Option Explicit
Public file As String
Public fichero As Integer
Public ini As Long
Public lsTexto As String
Public lnpos As Long
Public longtext As Long
Public key As String
Public doctext As String
Public ubicacion As String
Public num As Variant
Public tama As Long
Public contador As Integer
Public contad As Integer
Public contando As Integer
Public contado As Variant
Public xau As String
Public c As Long
Public d As String
Public temp As String
Public tempor As String
Public gstrMsg As String
Public gintResponse As String
Public tinic As Double
Public tfin As Double
Public minutos As Double
Public segundos As Double

Private Sub Optcifarch_Click() 'boton de opcion
```

```
Cmdabrir.Enabled = True 'activa el boton "abrir archivo"
Form1.selecta = "cifra" 'guarda en "selecta" la opcion seleccionada
Cmdcd.Caption = "cifrar" 'indica que proceso se esta realizando
Label1.Caption = "texto cifrado"
End Sub
Private Sub Optdescifarch_Click() 'igual que el otro boton de opcion
Cmdabrir.Enabled = True
Form1.selecta = "descifra"
Cmdcd.Caption = "descifrar"
Label1.Caption = "texto descifrado"
End Sub
Private Sub Cmdabrir_Click()
file = ""
'si el usuario presiona el boton cancelar y no abre ningun archivo se sale
'de la rutina
On Error GoTo Cancelar
'Coloca el titulo del cuadro de dialogo
CommonDialog1.DialogTitle = "Abrir"
'Establece los filtros, solo archivos *.txt
CommonDialog1.Filter = "Archivos de texto (*.txt) |*.txt|"
'Presenta el cuadro de dialogo Abrir
CommonDialog1.ShowOpen
'abre el fichero para leer
fichero = FreeFile 'selecciona el numero de fichero que este disponible
Open CommonDialog1.FileName For Binary Access Read As #fichero
' Leer la información del fichero
file = Input(LOF(fichero), #fichero) 'en la variable file se guardan
'los datos del archivo de texto
Text4.Text = file 'estos datos se muestran en pantalla
tama = Len(file) 'se calcula la longitud de este archivo
If tama > 2040 Then 'si es mayor a 2040 bytes muestra el siguiente mensaje
'y se sale de la rutina
MsgBox "el archivo sobrepasa los 2,040 bytes"
Exit Sub
End If
If tama = 0 Then 'si el archivo esta en blanco muestra mensaje de advertencia
'y se sale de la rutina
MsgBox "archivo en blanco"
Exit Sub
End If
Cmdcd.Enabled = True 'activa el boton cifrar/descifrar
Text1.Text = "" 'limpia el campo de la clave
Text1.Locked = False 'permite que ya se pueda escribir en el campo de la clave
Cancelar:
Close #fichero
End Sub


Private Sub Cmdcd_Click() 'boton cifrar/descifrar
Dim i As Integer
```

```
Dim j As Integer
Dim M As String
Text2.Text = "00"
Text5.Text = "00"
tinic = Timer   ‘gurda el tiempo de inicio del proceso
doctext = "" 'limpia la variable doctext
   If Text1.Text = "" Then 'manda mensaje de advertencia si la clave no ha sido
   'introducida
      MsgBox "Introduzca la clave"
      Exit Sub            ' y se sale de la rutina
   End If
   key = Text1.Text
M = key & key  'la clave debe ser de 72 caracteres por lo cual iniciamos duplicándola
Do While Len(M) < 72 'si esto no basto seguimos añadiendo la clave hasta lograr un
 M = M & key 'numero >= a 72, aunque el algoritmo Blowfish requiere 56 caracteres
'de la clave como maximo, se le agregan mas caracters porque asi lo requiere
'el programa del pic para poder hacer la funcion xor con los 18 arreglos P
'ya que con 56 carcteres solo llega a P14
Loop
  If Len(M) > 72 Then 'si nos pasamos de 72 caracteres, quitamos los caracteres
'sobrantes
     M = Mid(M, 1, 72)
   End If
key = M

   Label5.Visible = True 'aparece el mensaje "procesando, por favor espere"
   Image1.Visible = True ' y la imagen del reloj
   Cmdabrir.Enabled = False 'desactiva los botones abrir y cifra/descifrar
   Optcifarch.Enabled = False
   Optdescifarch.Enabled = False
   Cmdcd.Enabled = False 'desactiva el boton cifrar/descifrar

If Form1.selecta = "cifra" Then  'si se selecciono cifrar sigue esta rutina
       Call cifraarchivo  'llama a la rutina cifraarchivo
       GoTo rutinab       'luego salta a rutinab
Else
       Call descifraarchivo
End If

rutinab:
       For contad = 1 To 9 Step 1
          Call algoritmo 'se calculan los arreglos P
       Next
For i = 0 To 3 Step 1
   For j = 0 To 255 Step 2 'rutina para obtener los nuevos valores de cajas S
'son 4, con 256 arreglos
      Call algoritmo
      Form1.v5 = temp
      Call Form1.asciidec        'los arreglos S se pasan a decimal
      sboxes(i, j) = Form1.v6    'se guardan los nuevos arreglos S
```

```
      Form1.v5 = tempor
      Call Form1.asciidec
     sboxes(i, j + 1) = Form1.v6
   Next j
Next i

       Call longitud    'se llama a la rutina longitud y bloques
       Call bloques
Label5.Visible = False   'se oculta el mensaje de "procesando, por favor espere"
'y la imagen
Image1.Visible = False
Cmdabrir.Enabled = True  'se activan los botones "abrir" y "cifrar/descifrar"
Cmdcd.Enabled = True
tfin = Timer 'guarda el tiempo de finalizacion del proceso
minutos = tfin - tinic 'realiza la resta para saber cuanto se llevo el proceso
minutos = minutos / 60 'el tiempo esta en segundos, se pasa a minutos
segundos = ((minutos) - Int(minutos)) * 60 'lo sobrante a segundos
Text2.Text = Int(minutos) 'se muestran en pantallas
Text5.Text = Int(segundos)
       Call guardar     'se llama a la rutina "guardar"
Cmdcd.Enabled = False 'se desactiva el boton "cifrar/descifrar"
Text1.Locked = True      'no se permite escribir en el campo de la clave
Optcifarch.Enabled = True
Optdescifarch.Enabled = True
End Sub

Private Sub cifraarchivo() 'envia la opcion seleccionada por el usuario
Form1.opcion = "B"
Call Form1.enviar
Form1.opcion = key
Call Form1.enviar
Call cajasS 'inicializa los valores de Cajas S
End Sub

Private Sub descifraarchivo() 'envia la opcion seleccionada por el usuario
Form1.opcion = "D"
Call Form1.enviar
Form1.opcion = key
Call Form1.enviar
Call cajasS 'inicializa los valores de Cajas S
End Sub
Private Sub longitud() 'esta rutina calcula en base a la longitud del archivo,
'cuantas veces se debe ejecutar el algoritmo Blowfish
Dim a As Long
Dim b As Long
Dim e As Variant
longtext = Len(file)
a = ((longtext \ 8) + 1) * 8 'se verifica si es multiplo de 8
b = a - Len(file)
If b = 8 Then  'si es multiplo de 8, solo se divide entre 8 la longitud el archivo
```

```
   contado = (Len(file) / 8)
Else
   contado = (Len(file) \ 8) + 1 'si no es multiplo de 8 debe ser la longitud del
'archivo entre 8 + 1
End If
e = Hex(contado) 'cuando ya se tiene el valor este se debe pasar a ascii para
'mandarselo al micro
 If Len(e) = 1 Then 'si su longitud es igual a 1 se le agrega un cero
'para poder pasarlo a ascii, ya que esa rutina trabaja con dos digitos
   Form1.opcion = Chr(Form1.hexascii(0 & contado))
 Else
   Form1.opcion = Chr(Form1.hexascii(e)) 'si su longitud es dos, solo se manda
'a convertir
 End If
Call Form1.enviar
End Sub

Private Sub bloques()
xau = "00"
xau = Chr(Form1.hexascii(xau))
If Len(file) <> (contado * 8) Then 'inicialmente esta rutina verifica que el numero
'de caracteres sea multiplo de 8
'si no lo es le agrega al texto caracteres ascii equivalentes a 0
 file = file & String((contado * 8) - Len(file), xau)
End If

For ini = 1 To Len(file)  'en esta parte se va segmentando el texto de 8 en 8
    If ini + 8 > Len(file) Then
       lnpos = Len(file) - ini + 1
    Else
       lnpos = 8
    End If
       lsTexto = Mid(file, ini, lnpos)
        Form1.opcion = lsTexto 'se envian las partes del texto
        Call Form1.enviar
        Call algoritmo   'se ejecuta el algoritmo Blowfish
        doctext = doctext & temp & tempor 'en doctext se guardan los resultados
        'obtenidos
       ini = ini + 7 'este proceso se repite hasta que ya se ha enviado
'todo el texto
Next ini
End Sub
Private Sub algoritmo() 'algoritmo de cifrado/descifrado solo se guardan
'los resultados finales
For contando = 1 To 8 Step 1
       Call Form1.aceptar
       Call Form1.recibir
       Call Form1.recibirS
       Call Form1.enviarS
       Call Form1.aceptar
```

```
        Call Form1.recibir
        Call Form1.recibirS
        Call Form1.enviarS
Next
        Call Form1.aceptar
        Call Form1.recibir
        temp = Form1.buferentrada 'se guardan los resultados finales
        Call Form1.aceptar
        Call Form1.recibir
        tempor = Form1.buferentrada

End Sub
Private Sub guardar() 'rutina que guarda los resultados finales en un archivo .txt
If Form1.selecta = "cifra" Then
GoTo etiq
End If
c = Len(doctext) 'esta parte de esta rutina es utilizada cuando se ha realizado
'un proceso de descifrado, en el caso de que se haya cifrado un texto menor
'a 8 caracteres o que no era multiplo de 8, se le agregan ceros,
'cuando esto se decifra se deben quitar estos ceros
'en el caso de que no hubieran ceros o fuera un proceso de cifrado
Do While c > 0 'esta parte no se ejecuta
d = Mid(doctext, c, 1)
 If d = xau Then
   doctext = Mid(doctext, 1, c - 1)
    c = c - 1
 Else
   GoTo etiq
 End If
Loop
etiq:
   Text3.Text = doctext 'en text3 se muestra el archivo cifrado o descifrado
salto:
   CommonDialog1.CancelError = False  'si el usuario decide no guardar los datos
'finaliza sin problemas
   CommonDialog1.DialogTitle = "guardar archivo como" 'titulo del cuadro de dialogo
'filtros para los archivos
   CommonDialog1.Filter = "Archivos de texto (*.txt) |*.txt|"
   CommonDialog1.InitDir = doctext
   CommonDialog1.FileName = ""
   CommonDialog1.ShowSave 'muestra el cuadro de dialogo guardar como
   ubicacion = CommonDialog1.FileName
If Trim(ubicacion) <> "" Then 'si el usuario ya puso un nombre al archivo,empieza
'el proceso de guardado
 If FileExists(ubicacion) Then 'si el nombre del archivo ya existe,da la advertencia
      gstrMsg = "El nombre de ese archivo ya existe" & vbCrLf & " ¿Desea
remplazarlo?"
      gintResponse = MsgBox(gstrMsg, vbQuestion + vbYesNo, App.Title)
      If gintResponse = vbYes Then
         Kill (ubicacion)
```

```
        Call grabar
        GoTo salir
      Else
        GoTo salto
      End If
  End If
Call grabar
End If
salir:
End Sub
Private Function FileExists(sFile As String) As Boolean
   If sFile = "" Then
      FileExists = False
   End If
   On Error Resume Next
   FileExists = ((GetAttr(sFile) And vbDirectory) = 0)
End Function
Private Function grabar()
num = FreeFile
Open ubicacion For Binary Access Write As #num
        Put #num, 1, doctext  'guarda los datos en un archivo .txt
    Close #num
      MsgBox "El archivo ha sido guardado" 'indica que el archivo ha sido guardado
End Function
Private Sub Cmdregresar_Click() 'boton regresar
Form1.MSComm1.PortOpen = False 'cierra el puerto
Unload Me 'oculta esta forma
Unload Form1 'carga la forma1
Form1.Show 'muestra la forma principal
End Sub
```

FORMA 3

```
Public confir As String
Public h As Variant
Public contad As Integer
Public texizq As String
Public texder As String
Public a As Integer
Public c As Integer
Public d As Integer
Public e As Integer
Public f As Integer
Private Sub Cmdcontinuar_Click() 'rutina del boton "continuar"
confir = "si" 'pone en confir esta palabra
End Sub
Private Sub Cmdinic_Click()
confir = ""
Text1.Text = Form1.texto 'coloca al inicio del flujograma el texto a cif/descifrar
texder = Mid(Form1.texto, 1, 4) 'divide el texto
```

```
Text2(c).Text = texder 'muestra los primeros 4 caracteres
Form1.v5 = texder  'lo pasa a decimal y luego a hexadecimal para mostrarlo
Call Form1.asciidec
Text11(c).Text = Hex(Form1.v6)
c = c + 1
texizq = Mid(Form1.texto, 5, 4) 'muestra ultimos 4 caracteres
Text2(c).Text = texizq 'igual los pasa a hexadecimal
Form1.v5 = texizq
Call Form1.asciidec
Text11(c).Text = Hex(Form1.v6)

Cmdinic.Enabled = False 'desactiva el botón "Iniciar"
Cmdinic.Visible = False 'desaparece el botón "Iniciar"
Cmdcontinuar.Enabled = True 'activa el botón "Continuar"
Cmdcontinuar.Visible = True 'aparece el botón "Continuar"
Call algoritmoC 'llama al algoritmo Blowfish
End Sub

Private Sub algoritmoC() 'algoritmo Blowfish completo
'muestra los resultados parciales
f = 17
e = 0
a = 0
c = c + 1
Call rutina '1º Iteracion del blowfish
e = e + 1
a = e
c = c + 1
Call rutina '2º Iteracion del blowfish

 Do While confir = "" 'espera que se presione el boton "continuar"
  DoEvents 'para pasar a ver las iteraciones en el otro esquema
 Loop
 confir = ""

 Text2(6).Text = Text2(5).Text
 Text11(6).Text = Text11(5).Text
 Text2(7).Text = Text2(4).Text
 Text11(7).Text = Text11(4).Text
For contad = 1 To 13 Step 1 'se ven 13 iteraciones en ese esquema
 c = c + 3
 a = 2
 e = e + 1
 Text2(c).Text = ""
 Text2(c + 1).Text = ""
 Text11(c).Text = ""
 Text11(c + 1).Text = ""
 If Form1.selecta = "cifpap" Then
    Label6.Caption = e + 1
 Else
```

```
   Label6.Caption = f - 1
   f = f - 1
 End If
 Call rutina
 c = 5
 Do While confir = "" 'espera que se presione el boton "continuar"
  DoEvents          'para que la salida de la iteracion pase a ser ahora
 Loop               'la entrada
 confir = ""
 Text2(6).Text = Text2(9).Text
 Text11(6).Text = Text11(9).Text
 Text2(7).Text = Text2(8).Text
 Text11(7).Text = Text11(8).Text
Next
Text2(10).Text = Text2(9).Text
Text11(10).Text = Text11(9).Text
Text2(11).Text = Text2(8).Text
Text11(11).Text = Text11(8).Text
c = 12
a = 3
e = 15
Call rutina 'ultima iteracion (con funcion F)
c = 14
a = 5 '4
e = 17 '16
Call confirmar 'las rutinas "confirmar" son confirmaciones no automaticas
Text6(a).Text = pboxes(e) 'el usuario las da al presionar el boton "continuar"
Form1.v5 = pboxes(e)
Call Form1.asciidec
Text4(a).Text = Hex(Form1.v6)
Call pasopaso
Form1.archivos = Form1.temp
c = 15
a = 4
e = 16
Call confirmar
Text6(a).Text = pboxes(e)
Form1.v5 = pboxes(e)
Call Form1.asciidec
Text4(a).Text = Hex(Form1.v6)
Call pasopaso
Form1.archivos = Form1.archivos & Form1.temp
Text3.Text = Text2(14).Text & Text2(15).Text
Cmdcontinuar.Enabled = False
End Sub

Private Sub rutina()
      Call Form1.aceptar  'las rutinas "aceptar" son confirmaciones automaticas
      Text6(a).Text = pboxes(e)
      Form1.v5 = pboxes(e)
```

```
        Call Form1.asciidec
        Text4(a).Text = Hex(Form1.v6)
        Call pasopaso
        Form1.buferentrada = Form1.temp
        Text5.Text = Form1.temp
        Text12.Text = Text11(c).Text
        Call Form1.recibirS 'en este caso recibe que valores de caja S necesita
        Text17(0).Text = Form1.cs1 'la interfaz los busca
        Text17(1).Text = Form1.cs2
        Text17(2).Text = Form1.cs3
        Text17(3).Text = Form1.cs4
        Call mostrarS 'los muestra
        Call Form1.enviarS 'y luego se los envia
        Call funcionF
        c = c + 1
        Call pasopaso
End Sub

Private Sub funcionF()
        d = c
        c = 16
        Text2(c).Text = "" 'limpia las cajas de texto de la funcion F
        Text11(c).Text = ""
        Text2(c + 1).Text = ""
        Text11(c + 1).Text = ""
        Text2(c + 2).Text = ""
        Text11(c + 2).Text = ""
        Call pasopaso   'muestra los resultados de las funcion F
        c = 17
        Call pasopaso
        c = 18
        Call pasopaso
        c = d
End Sub

Private Sub mostrarS()
Text7.Text = Form1.s1 'muestra los valores de caja S solicitados
    If Len(Form1.s11) < 8 Then 'tambien muestra los valores en hexadecimal
        Text13.Text = String(8 - Len(Form1.s11), "0") & Form1.s11
    Else 'verifica si es menor de 8 porque si es asi es que los bits
        Text13.Text = Form1.s11 'iniciales son cero, pero el programa
    End If 'no los muestra, porque ceros a la izquierda no valen nada
Text8.Text = Form1.s2  'pero se deben mostrar porque esos ceros
    If Len(Form1.s22) < 8 Then 'a la izquierda al convertirse a ascii, son el
'el caracter NULL que borra los caracteres que le siguen
        Text14.Text = String(8 - Len(Form1.s22), "0") & Form1.s22
'por lo que en ascii no se podran 'apreciar, pero si en hexadecimal

    Else
        Text14.Text = Form1.s22
```

```
   End If
Text9.Text = Form1.s3
   If Len(Form1.s33) < 8 Then
      Text15.Text = String(8 - Len(Form1.s33), "0") & Form1.s33
   Else
Text15.Text = Form1.s33
   End If
Text10.Text = Form1.s4
   If Len(Form1.s44) < 8 Then
      Text16.Text = String(8 - Len(Form1.s44), "0") & Form1.s44
   Else
Text16.Text = Form1.s44
   End If
End Sub
Private Sub recdato()
Do While Form1.MSComm1.InBufferCount = 0
DoEvents
Loop
Form1.buferentrada = Form1.MSComm1.Input
End Sub

Private Sub pasopaso()
      Call confirmar 'las rutinas "confirmar" son confirmaciones no automaticas
      Call recdato   'el usuario las da al presionar el boton "continuar"
      Text11(c).Text = ""
      Form1.temp = Form1.buferentrada
      Text2(c).Text = Text2(c).Text & Form1.buferentrada
      Call hexadecimal
      Call confirmar
      Call recdato
      Form1.temp = Form1.temp & Form1.buferentrada
      Text2(c).Text = Text2(c).Text & Form1.buferentrada
      Call hexadecimal
      Call confirmar
      Call recdato
      Form1.temp = Form1.temp & Form1.buferentrada
      Text2(c).Text = Text2(c).Text & Form1.buferentrada
      Call hexadecimal
      Call confirmar
      Call recdato
      Form1.temp = Form1.temp & Form1.buferentrada
      Text2(c).Text = Text2(c).Text & Form1.buferentrada '& vbCrLf
      Call hexadecimal
End Sub
Private Sub hexadecimal()
h = Asc(Form1.buferentrada) 'primero se convierte a decimal el valor
h = Hex(h) 'para convertirlo finalmente a Hexadecimal
If Len(h) = 1 Then
h = "0" & h
End If
```

```
Text11(c).Text = Text11(c).Text & h
End Sub
Private Sub confirmar()
'rutina que espera a que el usuario presione al boton "continuar"
Do While confir = ""  'para enviarle al micro que ya puede enviar nuevos datos
DoEvents 'sentencia que permite salirse del lazo
Loop
confir = ""   'si se presiono el boton continuar la variable "confir"
'ya no está vacia por lo que se sale del lazo
Form1.opcion = "E"  'se limpia esa variable, y se envia una "E" al micro para
'indicarle que ya puede enviar los nuevos datos
Call Form1.enviar
End Sub
Private Sub Cmdregre_Click()
Form1.MSComm1.PortOpen = False 'cierra el puerto
Unload Me 'oculta esta forma
Unload Form1 'carga la forma1
Form1.Show 'muestra la forma principal
End Sub

Private Sub Form_Load()
c = 0
If Form1.selecta = "cifpap" Then
   Text3.Text = "Texto cifrado"
Else
   Text3.Text = "Texto descifrado" 'si es un proceso de descifrado
   Label1(c).Caption = "P18"        'las etiquetas de P se cambian
   Label1(c + 1).Caption = "P17"     'porque estos se invierten"
   Label1(c + 2).Caption = "P2"      'cuando es un proceso de descifrado"
   Label1(c + 3).Caption = "P1"
   Label1(c + 4).Caption = "P3"
   Label6.Caption = 16
End If
End Sub
```

MODULO 1

```
Option Explicit
Option Base 0 'para que los arreglos inicien de cero

' Este modulo bascajasS inicializa las cajas S para el algoritmo
' Blowfish
Public pboxes(17) As String
Public sboxes(3, 255) As Double

Public Function cajasS()

Dim arreglo As Variant
Dim i As Integer

'Esta es la caja S1 esta dividida en 4 porciones porque visual basic
'no permite arreglos de mas de 64 elementos
'------------------------------------------------------------

   'caja S1
   arreglo = Array( _
      &HD1310BA6, &H98DFB5AC, &H2FFD72DB, &HD01ADFB7, _
      &HB8E1AFED, &H6A267E96, &HBA7C9045, &HF12C7F99, _
      &H24A19947, &HB3916CF7, &H801F2E2, &H858EFC16, _
      &H636920D8, &H71574E69, &HA458FEA3, &HF4933D7E, _
      &HD95748F, &H728EB658, &H718BCD58, &H82154AEE, _
      &H7B54A41D, &HC25A59B5, &H9C30D539, &H2AF26013, _
      &HC5D1B023, &H286085F0, &HCA417918, &HB8DB38EF, _
      &H8E79DCB0, &H603A180E, &H6C9E0E8B, &HB01E8A3E, _
      &HD71577C1, &HBD314B27, &H78AF2FDA, &H55605C60, _
      &HE65525F3, &HAA55AB94, &H57489862, &H63E81440, _


      &H55CA396A, &H2AAB10B6, &HB4CC5C34, &H1141E8CE, _
      &HA15486AF, &H7C72E993, &HB3EE1411, &H636FBC2A, _
      &H2BA9C55D, &H741831F6, &HCE5C3E16, &H9B87931E, _
      &HAFD6BA33, &H6C24CF5C, &H7A325381, &H28958677, _
      &H3B8F4898, &H6B4BB9AF, &HC4BFE81B, &H66282193, _
      &H61D809CC, &HFB21A991, &H487CAC60, &H5DEC8032)

   For i = 0 To 63
      sboxes(0, i) = arreglo(i)
   Next

   arreglo = Array( _
      &HEF845D5D, &HE98575B1, &HDC262302, &HEB651B88, _
      &H23893E81, &HD396ACC5, &HF6D6FF3, &H83F44239, _
      &H2E0B4482, &HA4842004, &H69C8F04A, &H9E1F9B5E, _
      &H21C66842, &HF6E96C9A, &H670C9C61, &HABD388F0, _
```

```
    &H6A51A0D2, &HD8542F68, &H960FA728, &HAB5133A3, _
    &H6EEF0B6C, &H137A3BE4, &HBA3BF050, &H7EFB2A98, _
    &HA1F1651D, &H39AF0176, &H66CA593E, &H82430E88, _
    &H8CEE8619, &H456F9FB4, &H7D84A5C3, &H3B8B5EBE, _
    &HE06F75D8, &H85C12073, &H401A449F, &H56C16AA6, _
    &H4ED3AA62, &H363F7706, &H1BFEDF72, &H429B023D, _
    &H37D0D724, &HD00A1248, &HDB0FEAD3, &H49F1C09B, _
    &H75372C9, &H80991B7B, &H25D479D8, &HF6E8DEF7, _
    &HE3FE501A, &HB6794C3B, &H976CE0BD, &H4C006BA, _
    &HC1A94FB6, &H409F60C4, &H5E5C9EC2, &H196A2463, _
    &H68FB6FAF, &H3E6C53B5, &H1339B2EB, &H3B52EC6F, _
    &H6DFC511F, &H9B30952C, &HCC814544, &HAF5EBD09)

For i = 0 To 63     '64 To 127
    sboxes(0, i + 64) = arreglo(i)
Next

arreglo = Array( _
    &HBEE3D004, &HDE334AFD, &H660F2807, &H192E4BB3, _
    &HC0CBA857, &H45C8740F, &HD20B5F39, &HB9D3FBDB, _
    &H5579C0BD, &H1A60320A, &HD6A100C6, &H402C7279, _
    &H679F25FE, &HFB1FA3CC, &H8EA5E9F8, &HDB3222F8, _
    &H3C7516DF, &HFD616B15, &H2F501EC8, &HAD0552AB, _
    &H323DB5FA, &HFD238760, &H53317B48, &H3E00DF82, _
    &H9E5C57BB, &HCA6F8CA0, &H1A87562E, &HDF1769DB, _
    &HD542A8F6, &H287EFFC3, &HAC6732C6, &H8C4F5573, _
    &H695B27B0, &HBBCA58C8, &HE1FFA35D, &HB8F011A0, _
    &H10FA3D98, &HFD2183B8, &H4AFCB56C, &H2DD1D35B, _
    &H9A53E479, &HB6F84565, &HD28E49BC, &H4BFB9790, _
    &HE1DDF2DA, &HA4CB7E33, &H62FB1341, &HCEE4C6E8, _
    &HEF20CADA, &H36774C01, &HD07E9EFE, &H2BF11FB4, _



    &H95DBDA4D, &HAE909198, &HEAAD8E71, &H6B93D5A0, _
    &HD08ED1D0, &HAFC725E0, &H8E3C5B2F, &H8E7594B7, _
    &H8FF6E2FB, &HF2122B64, &H8888B812, &H900DF01C)


For i = 0 To 63     ' 128 To 191
    sboxes(0, i + 128) = arreglo(i)
Next

arreglo = Array( _
    &H4FAD5EA0, &H688FC31C, &HD1CFF191, &HB3A8C1AD, _
    &H2F2F2218, &HBE0E1777, &HEA752DFE, &H8B021FA1, _
    &HE5A0CC0F, &HB56F74E8, &H18ACF3D6, &HCE89E299, _
    &HB4A84FE0, &HFD13E0B7, &H7CC43B81, &HD2ADA8D9, _
    &H165FA266, &H80957705, &H93CC7314, &H211A1477, _
    &HE6AD2065, &H77B5FA86, &HC75442F5, &HFB9D35CF, _
```

```
      &HEBCDAF0C, &H7B3E89A0, &HD6411BD3, &HAE1E7E49, _
      &H250E2D, &H2071B35E, &H226800BB, &H57B8E0AF, _
      &H2464369B, &HF009B91E, &H5563911D, &H59DFA6AA, _
      &H78C14389, &HD95A537F, &H207D5BA2, &H2E5B9C5, _
      &H83260376, &H6295CFA9, &H11C81968, &H4E734A41, _
      &HB3472DCA, &H7B14A94A, &H1B510052, &H9A532915, _
      &HD60F573F, &HBC9BC6E4, &H2B60A476, &H81E67400, _
      &H8BA6FB5, &H571BE91F, &HF296EC6B, &H2A0DD915, _
      &HB6636521, &HE7B9F9B6, &HFF34052E, &HC5855664, _
      &H53B02D5D, &HA99F8FA1, &H8BA4799, &H6E85076A)

   For i = 0 To 63     ' 192 To 255
      sboxes(0, i + 192) = arreglo(i)
   Next

'------------------------------------------------------
'------------------------------------------------------
   'caja s2
   arreglo = Array( _
      &H4B7A70E9, &HB5B32944, &HDB75092E, &HC4192623, _
      &HAD6EA6B0, &H49A7DF7D, &H9CEE60B8, &H8FEDB266, _
      &HECAA8C71, &H699A17FF, &H5664526C, &HC2B19EE1, _
      &H193602A5, &H75094C29, &HA0591340, &HE4183A3E, _
      &H3F54989A, &H5B429D65, &H6B8FE4D6, &H99F73FD6, _
      &HA1D29C07, &HEFE830F5, &H4D2D38E6, &HF0255DC1, _
      &H4CDD2086, &H8470EB26, &H6382E9C6, &H21ECC5E, _
      &H9686B3F, &H3EBAEFC9, &H3C971814, &H6B6A70A1, _
      &H687F3584, &H52A0E286, &HB79C5305, &HAA500737, _
      &H3E07841C, &H7FDEAE5C, &H8E7D44EC, &H5716F2B8, _
      &HB03ADA37, &HF0500C0D, &HF01C1F04, &H200B3FF, _
      &HAE0CF51A, &H3CB574B2, &H25837A58, &HDC0921BD, _



      &HD19113F9, &H7CA92FF6, &H94324773, &H22F54701, _
      &H3AE5E581, &H37C2DADC, &HC8B57634, &H9AF3DDA7, _
      &HA9446146, &HFD0030E, &HECC8C73E, &HA4751E41, _
      &HE238CD99, &H3BEA0E2F, &H3280BBA1, &H183EB331)


   For i = 0 To 63
      sboxes(1, i) = arreglo(i)
   Next

   arreglo = Array( _
      &H4E548B38, &H4F6DB908, &H6F420D03, &HF60A04BF, _
      &H2CB81290, &H24977C79, &H5679B072, &HBCAF89AF, _
      &HDE9A771F, &HD9930810, &HB38BAE12, &HDCCF3F2E, _
      &H5512721F, &H2E6B7124, &H501ADDE6, &H9F84CD87, _
      &H7A584718, &H7408DA17, &HBC9F9ABC, &HE94B7D8C, _
```

```
    &HEC7AEC3A, &HDB851DFA, &H63094366, &HC464C3D2, _
    &HEF1C1847, &H3215D908, &HDD433B37, &H24C2BA16, _
    &H12A14D43, &H2A65C451, &H50940002, &H133AE4DD, _
    &H71DFF89E, &H10314E55, &H81AC77D6, &H5F11199B, _
    &H43556F1, &HD7A3C76B, &H3C11183B, &H5924A509, _
    &HF28FE6ED, &H97F1FBFA, &H9EBABF2C, &H1E153C6E, _
    &H86E34570, &HEAE96FB1, &H860E5E0A, &H5A3E2AB3, _
    &H771FE71C, &H4E3D06FA, &H2965DCB9, &H99E71D0F, _
    &H803E89D6, &H5266C825, &H2E4CC978, &H9C10B36A, _
    &HC6150EBA, &H94E2EA78, &HA5FC3C53, &H1E0A2DF4, _
    &HF2F74EA7, &H361D2B3D, &H1939260F, &H19C27960)

For i = 0 To 63     '64 To 127
    sboxes(1, i + 64) = arreglo(i)
Next

arreglo = Array( _
    &H5223A708, &HF71312B6, &HEBADFE6E, &HEAC31F66, _
    &HE3BC4595, &HA67BC883, &HB17F37D1, &H18CFF28, _
    &HC332DDEF, &HBE6C5AA5, &H65582185, &H68AB9802, _
    &HEECEA50F, &HDB2F953B, &H2AEF7DAD, &H5B6E2F84, _
    &H1521B628, &H29076170, &HECDD4775, &H619F1510, _
    &H13CCA830, &HEB61BD96, &H334FE1E, &HAA0363CF, _
    &HB5735C90, &H4C70A239, &HD59E9E0B, &HCBAADE14, _
    &HEECC86BC, &H60622CA7, &H9CAB5CAB, &HB2F3846E, _
    &H648B1EAF, &H19BDF0CA, &HA02369B9, &H655ABB50, _
    &H40685A32, &H3C2AB4B3, &H319EE9D5, &HC021B8F7, _
    &H9B540B19, &H875FA099, &H95F7997E, &H623D7DA8, _
    &HF837889A, &H97E32D77, &H11ED935F, &H16681281, _
    &HE358829, &HC7E61FD6, &H96DEDFA1, &H7858BA99, _
    &H57F584A5, &H1B227263, &H9B83C3FF, &H1AC24696, _
    &HCDB30AEB, &H532E3054, &H8FD948E4, &H6DBC3128, _
    &H58EBF2EF, &H34C6FFEA, &HFE28ED61, &HEE7C3C73)

For i = 0 To 63     ' 128 To 191
    sboxes(1, i + 128) = arreglo(i)
Next

arreglo = Array( _
    &H5D4A14D9, &HE864B7E3, &H42105D14, &H203E13E0, _
    &H45EEE2B6, &HA3AAABEA, &HDB6C4F15, &HFACB4FD0, _
    &HC742F442, &HEF6ABBB5, &H654F3B1D, &H41CD2105, _
    &HD81E799E, &H86854DC7, &HE44B476A, &H3D816250, _
    &HCF62A1F2, &H5B8D2646, &HFC8883A0, &HC1C7B6A3, _
    &H7F1524C3, &H69CB7492, &H47848A0B, &H5692B285, _
    &H95BBF00, &HAD19489D, &H1462B174, &H23820E00, _
    &H58428D2A, &HC55F5EA, &H1DADF43E, &H233F7061, _
    &H3372F092, &H8D937E41, &HD65FECF1, &H6C223BDB, _
    &H7CDE3759, &HCBEE7460, &H4085F2A7, &HCE77326E, _
    &HA6078084, &H19F8509E, &HE8EFD855, &H61D99735, _
```

```
      &HA969A7AA, &HC50C06C2, &H5A04ABFC, &H800BCADC, _
      &H9E447A2E, &HC3453484, &HFDD56705, &HE1E9EC9, _
      &HDB73DBD3, &H105588CD, &H675FDA79, &HE3674340, _
      &HC5C43465, &H713E38D8, &H3D28F89E, &HF16DFF20, _
      &H153E21E7, &H8FB03D4A, &HE6E39F2B, &HDB83ADF7)

   For i = 0 To 63     ' 192 To 255
      sboxes(1, i + 192) = arreglo(i)
   Next
'------------------------------------------------------
'------------------------------------------------------
   ' Caja S3
   arreglo = Array( _
      &HE93D5A68, &H948140F7, &HF64C261C, &H94692934, _
      &H411520F7, &H7602D4F7, &HBCF46B2E, &HD4A20068, _
      &HD4082471, &H3320F46A, &H43B7D4B7, &H500061AF, _
      &H1E39F62E, &H97244546, &H14214F74, &HBF8B8840, _
      &H4D95FC1D, &H96B591AF, &H70F4DDD3, &H66A02F45, _
      &HBFBC09EC, &H3BD9785, &H7FAC6DD0, &H31CB8504, _
      &H96EB27B3, &H55FD3941, &HDA2547E6, &HABCA0A9A, _
      &H28507825, &H530429F4, &HA2C86DA, &HE9B66DFB, _
      &H68DC1462, &HD7486900, &H680EC0A4, &H27A18DEE, _
      &H4F3FFEA2, &HE887AD8C, &HB58CE006, &H7AF4D6B6, _
      &HAACE1E7C, &HD3375FEC, &HCE78A399, &H406B2A42, _
      &H20FE9E35, &HD9F385B9, &HEE39D7AB, &H3B124E8B, _
      &H1DC9FAF7, &H4B6D1856, &H26A36631, &HEAE397B2, _
      &H3A6EFA74, &HDD5B4332, &H6841E7F7, &HCA7820FB, _
      &HFB0AF54E, &HD8FEB397, &H454056AC, &HBA489527, _
      &H55533A3A, &H20838D87, &HFE6BA9B7, &HD096954B)

   For i = 0 To 63
      sboxes(2, i) = arreglo(i)
   Next

   arreglo = Array( _
      &H55A867BC, &HA1159A58, &HCCA92963, &H99E1DB33, _
      &HA62A4A56, &H3F3125F9, &H5EF47E1C, &H9029317C, _
      &HFDF8E802, &H4272F70, &H80BB155C, &H5282CE3, _
      &H95C11548, &HE4C66D22, &H48C1133F, &HC70F86DC, _
      &H7F9C9EE, &H41041F0F, &H404779A4, &H5D886E17, _
      &H325F51EB, &HD59BC0D1, &HF2BCC18F, &H41113564, _
      &H257B7834, &H602A9C60, &HDFF8E8A3, &H1F636C1B, _
      &HE12B4C2, &H2E1329E, &HAF664FD1, &HCAD18115, _
      &H6B2395E0, &H333E92E1, &H3B240B62, &HEEBEB922, _
      &H85B2A20E, &HE6BA0D99, &HDE720C8C, &H2DA2F728, _
      &HD0127845, &H95B794FD, &H647D0862, &HE7CCF5F0, _
      &H5449A36F, &H877D48FA, &HC39DFD27, &HF33E8D1E, _
      &HA476341, &H992EFF74, &H3A6F6EAB, &HF4F8FD37, _
      &HA812DC60, &HA1EBDDF8, &H991BE14C, &HDB6E6B0D, _
      &HC67B5510, &H6D672C37, &H2765D43B, &HDCD0E804, _
```

```
    &HF1290DC7, &HCC00FFA3, &HB5390F92, &H690FED0B)

  For i = 0 To 63     '64 To 127
    sboxes(2, i + 64) = arreglo(i)
  Next

  arreglo = Array( _
    &H667B9FFB, &HCEDB7D9C, &HA091CF0B, &HD9155EA3, _
    &HBB132F88, &H515BAD24, &H7B9479BF, &H763BD6EB, _
    &H37392EB3, &HCC115979, &H8026E297, &HF42E312D, _
    &H6842ADA7, &HC66A2B3B, &H12754CCC, &H782EF11C, _
    &H6A124237, &HB79251E7, &H6A1BBE6, &H4BFB6350, _
    &H1A6B1018, &H11CAEDFA, &H3D25BDD8, &HE2E1C3C9, _
    &H44421659, &HA121386, &HD90CEC6E, &HD5ABEA2A, _
    &H64AF674E, &HDA86A85F, &HBEBFE988, &H64E4C3FE, _
    &H9DBC8057, &HF0F7C086, &H60787BF8, &H6003604D, _
    &HD1FD8346, &HF6381FB0, &H7745AE04, &HD736FCCC, _
    &H83426B33, &HF01EAB71, &HB0804187, &H3C005E5F, _
    &H77A057BE, &HBDE8AE24, &H55464299, &HBF582E61, _
    &H4E58F48F, &HF2DDFDA2, &HF474EF38, &H8789BDC2, _
    &H5366F9C3, &HC8B38E74, &HB475F255, &H46FCD9B9, _
    &H7AEB2661, &H8B1DDF84, &H846A0E79, &H915F95E2, _
    &H466E598E, &H20B45770, &H8CD55591, &HC902DE4C)

  For i = 0 To 63     ' 128 To 191
    sboxes(2, i + 128) = arreglo(i)
  Next

  arreglo = Array( _
    &HB90BACE1, &HBB8205D0, &H11A86248, &H7574A99E, _
    &HB77F19B6, &HE0A9DC09, &H662D09A1, &HC4324633, _
    &HE85A1F02, &H9F0BE8C, &H4A99A025, &H1D6EFE10, _
    &H1AB93D1D, &HBA5A4DF, &HA186F20F, &H2868F169, _
    &HDCB7DA83, &H573906FE, &HA1E2CE9B, &H4FCD7F52, _
    &H50115E01, &HA70683FA, &HA002B5C4, &HDE6D027, _
    &H9AF88C27, &H773F8641, &HC3604C06, &H61A806B5, _
    &HF0177A28, &HC0F586E0, &H6058AA, &H30DC7D62, _
    &H11E69ED7, &H2338EA63, &H53C2DD94, &HC2C21634, _
    &HBBCBEE56, &H90BCB6DE, &HEBFC7DA1, &HCE591D76, _
    &H6F05E409, &H4B7C0188, &H39720A3D, &H7C927C24, _
    &H86E3725F, &H724D9DB9, &H1AC15BB4, &HD39EB8FC, _
    &HED545578, &H8FCA5B5, &HD83D7CD3, &H4DAD0FC4, _
    &H1E50EF5E, &HB161E6F8, &HA28514D9, &H6C51133C, _
    &H6FD5C7E7, &H56E14EC4, &H362ABFCE, &HDDC6C837, _
    &HD79A3234, &H92638212, &H670EFA8E, &H406000E0)

  For i = 0 To 63     ' 192 To 255
    sboxes(2, i + 192) = arreglo(i)
  Next
'-------------------------------------------------------
```

```
'-------------------------------------------------------
   'caja S4
   arreglo = Array( _
      &H3A39CE37, &HD3FAF5CF, &HABC27737, &H5AC52D1B, _
      &H5CB0679E, &H4FA33742, &HD3822740, &H99BC9BBE, _
      &HD5118E9D, &HBF0F7315, &HD62D1C7E, &HC700C47B, _
      &HB78C1B6B, &H21A19045, &HB26EB1BE, &H6A366EB4, _
      &H5748AB2F, &HBC946E79, &HC6A376D2, &H6549C2C8, _
      &H530FF8EE, &H468DDE7D, &HD5730A1D, &H4CD04DC6, _
      &H2939BBDB, &HA9BA4650, &HAC9526E8, &HBE5EE304, _
      &HA1FAD5F0, &H6A2D519A, &H63EF8CE2, &H9A86EE22, _
      &HC089C2B8, &H43242EF6, &HA51E03AA, &H9CF2D0A4, _
      &H83C061BA, &H9BE96A4D, &H8FE51550, &HBA645BD6, _
      &H2826A2F9, &HA73A3AE1, &H4BA99586, &HEF5562E9, _
      &HC72FEFD3, &HF752F7DA, &H3F046F69, &H77FA0A59, _
      &H80E4A915, &H87B08601, &H9B09E6AD, &H3B3EE593, _
      &HE990FD5A, &H9E34D797, &H2CF0B7D9, &H22B8B51, _
      &H96D5AC3A, &H17DA67D, &HD1CF3ED6, &H7C7D2D28, _
      &H1F9F25CF, &HADF2B89B, &H5AD6B472, &H5A88F54C)

   For i = 0 To 63
      sboxes(3, i) = arreglo(i)
   Next

   arreglo = Array( _
      &HE029AC71, &HE019A5E6, &H47B0ACFD, &HED93FA9B, _
      &HE8D3C48D, &H283B57CC, &HF8D56629, &H79132E28, _
      &H785F0191, &HED756055, &HF7960E44, &HE3D35E8C, _
      &H15056DD4, &H88F46DBA, &H3A16125, &H564F0BD, _
      &HC3EB9E15, &H3C9057A2, &H97271AEC, &HA93A072A, _
      &H1B3F6D9B, &H1E6321F5, &HF59C66FB, &H26DCF319, _



      &H7533D928, &HB155FDF5, &H3563482, &H8ABA3CBB, _
      &H28517711, &HC20AD9F8, &HABCC5167, &HCCAD925F, _
      &H4DE81751, &H3830DC8E, &H379D5862, &H9320F991, _
      &HEA7A90C2, &HFB3E7BCE, &H5121CE64, &H774FBE32, _
      &HA8B6E37E, &HC3293D46, &H48DE5369, &H6413E680, _
      &HA2AE0810, &HDD6DB224, &H69852DFD, &H9072166, _
      &HB39A460A, &H6445C0DD, &H586CDECF, &H1C20C8AE, _
      &H5BBEF7DD, &H1B588D40, &HCCD2017F, &H6BB4E3BB, _
      &HDDA26A7E, &H3A59FF45, &H3E350A44, &HBCB4CDD5, _
      &H72EACEA8, &HFA6484BB, &H8D6612AE, &HBF3C6F47)

   For i = 0 To 63     '64 To 127
      sboxes(3, i + 64) = arreglo(i)
   Next
```

```
    arreglo = Array( _
       &HD29BE463, &H542F5D9E, &HAEC2771B, &HF64E6370, _
       &H740E0D8D, &HE75B1357, &HF8721671, &HAF537D5D, _
       &H4040CB08, &H4EB4E2CC, &H34D2466A, &H115AF84, _
       &HE1B00428, &H95983A1D, &H6B89FB4, &HCE6EA048, _
       &H6F3F3B82, &H3520AB82, &H11A1D4B, &H277227F8, _
       &H611560B1, &HE7933FDC, &HBB3A792B, &H344525BD, _
       &HA08839E1, &H51CE794B, &H2F32C9B7, &HA01FBAC9, _
       &HE01CC87E, &HBCC7D1F6, &HCF0111C3, &HA1E8AAC7, _
       &H1A908749, &HD44FBD9A, &HD0DADECB, &HD50ADA38, _
       &H339C32A, &HC6913667, &H8DF9317C, &HE0B12B4F, _
       &HF79E59B7, &H43F5BB3A, &HF2D519FF, &H27D9459C, _
       &HBF97222C, &H15E6FC2A, &HF91FC71, &H9B941525, _
       &HFAE59361, &HCEB69CEB, &HC2A86459, &H12BAA8D1, _
       &HB6C1075E, &HE3056A0C, &H10D25065, &HCB03A442, _
       &HE0EC6E0E, &H1698DB3B, &H4C98A0BE, &H3278E964, _
       &H9F1F9532, &HE0D392DF, &HD3A0342B, &H8971F21E)

    For i = 0 To 63     ' 128 To 191
       sboxes(3, i + 128) = arreglo(i)
    Next

    arreglo = Array( _
       &H1B0A7441, &H4BA3348C, &HC5BE7120, &HC37632D8, _
       &HDF359F8D, &H9B992F2E, &HE60B6F47, &HFE3F11D, _
       &HE54CDA54, &H1EDAD891, &HCE6279CF, &HCD3E7E6F, _
       &H1618B166, &HFD2C1D05, &H848FD2C5, &HF6FB2299, _
       &HF523F357, &HA6327623, &H93A83531, &H56CCCD02, _
       &HACF08162, &H5A75EBB5, &H6E163697, &H88D273CC, _
       &HDE966292, &H81B949D0, &H4C50901B, &H71C65614, _
       &HE6C6C7BD, &H327A140A, &H45E1D006, &HC3F27B9A, _
       &HC9AA53FD, &H62A80F00, &HBB25BFE2, &H35BDD2F6, _



       &H71126905, &HB2040222, &HB6CBCF7C, &HCD769C2B, _
       &H53113EC0, &H1640E3D3, &H38ABBD60, &H2547ADF0, _
       &HBA38209C, &HF746CE76, &H77AFA1C5, &H20756060, _
       &H85CBFE4E, &H8AE88DD8, &H7AAAF9B0, &H4CF9AA7E, _
       &H1948C25C, &H2FB8A8C, &H1C36AE4, &HD6EBE1F9, _
       &H90D4F869, &HA65CDEA0, &H3F09252D, &HC208E69F, _
       &HB74E6132, &HCE77E25B, &H578FDFE3, &H3AC372E6)


    For i = 0 To 63     ' 192 To 255
       sboxes(3, i + 192) = arreglo(i)
    Next
'-------------------------------------------------------
End Function
```

Anexo 3: SOFTWARE PARA VALIDACION DE RESULTADOS

VISUAL BASIC BLOWFISH Versión 6 (VB6).

Se utiliza la versión en Visual Basic del algoritmo Blowfish de Bruce Schneier que se especifica en el libro Criptografía Aplicada[27]. Esta versión fue publicada el 20 de Noviembre del 2003. El software descargado difiere un poco con el que se muestra en el libro de B.S.

DERECHOS DE AUTOR.
El código fue escrito en Visual Basic por David Ireland y fue patentado en el 2003 por D.I. Management Services Pty Limited. Fue publicado para uso libre para fines personales respetando tales derechos. La información pertinente se puede encontrar en la siguiente página web con su título en inglés: "Contains cryptography software by David Ireland cuyo enlace es www.di-mgt.com.au. Comentarios o abusos de tal aplicación se pueden hacer a code@di-mgt.com.au y las últimas versiones se pueden descargar en www.di-mgt.com.au/crypto.html.

PARTES DEL VB6.

Consta de 11 módulos y de 2 proyectos en Visual Basic.
Los módulos principales son:

1. basBlowfishByteFns: el principal grupo de funciones que sirven para llamar los otros programas.
2. basBlowfishFileFns: grupo de funciones para cifrado de archivos.
3. basBlowfish: el algoritmo blowfish básico en código Visual Basic.
4. basBlfArrays: los arreglos P y las cajas S del algoritmo Blowfish.
5. basUnsignedWord: utilidades para operaciones sin asignación.
6. basConvert: conversión de utilidades para cadenas de bytes, en código Hexadecimal y parabras.
7. basFileAPI: grupo de funciones para leer y escribir archivos usando funciones de Windows
8. basRadix64: funciones para codificar cadenas en binario a formato base 64 y viceversa (aka radix64, Transfer Encoding, Printable Encoding).

Modules que contienen funciones de medición:

9. basTestBlowfish: ejemplos de uso de Blowfish y juego de mediciones
10. basTestRadix64: ejemplos de codificación y decodificación base64/radix64.
11. basAPITimer: funciones de tiempo de Litwin, Getz, Gilbert.

[27] APPLIED CRYPTOGRAPHY, Bruce Schneier Pg 647.

Los dos Proyectos VB:

- Blowfish.vbp: Demo de cifrado Blowfish
- BlowfishEx.vbp: Demo extendido que muestra modos ECB y CBC.

PARA UTILIZA: Se deben agregar todos los archivos en un solo proyecto de Visual Basic o importarlos a un módulo.

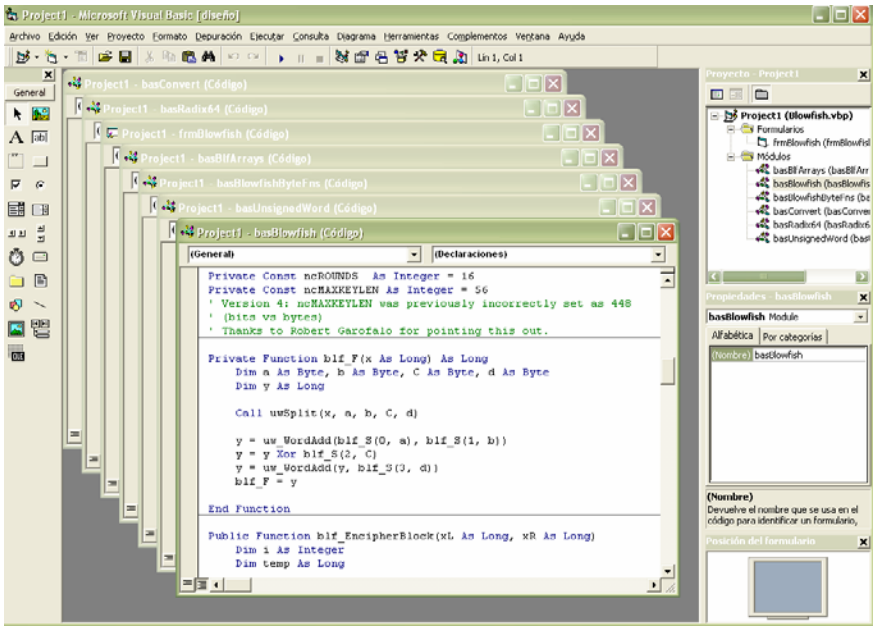


Fig. A-1: Apariencia del BV6.

Las funciones principales ECB son para longitud variable de datos que se utilizan para cifrar y descifrar cadenas de bytes y para la generación de las subclaves:

- blf_BytesEnc(abData)
- blf_BytesDec(abData)
- blf_BytesRaw(abData, bEncrypt)
- blf_FileEnc(sFileIn, sFileOut)
- blf_FileDec(sFileIn, sFileOut)

Funciones CBC

- blf_BytesEncCBC
- blf_BytesDecCBC
- blf_BytesEncRawCBC
- blf_BytesDecRawCBC
- blf_FileEncCBC
- blf_FileDecCBC

FUNCIONES BASICAS DEL BLOWFISH

- blf_KeyInit(abKey): Sirve para ingresar la clave en el espacio creado para la clave dentro del bloque.
- blf_Key: (redundante).
- blf_Initialise(aKey, nKeyBytes): Ingresa la clave actual en la generación de las subclaves.
- blf_EncryptBytes(aBytes): Cifra bloques de 8 bytes con la clave actual.
- blf_DecryptBytes(aBytes): Descifra bloques de 8 bytes con la clave actual.

Para iniciar el proceso Blowfish (recordar que todos los archivos involucrados deben estar en un mismo proyecto), hacer clic en el icono INICIAR , o seleccionándolo desde el menú ejecutar en la barra de menús. Aparecerá el bloque siguiente:

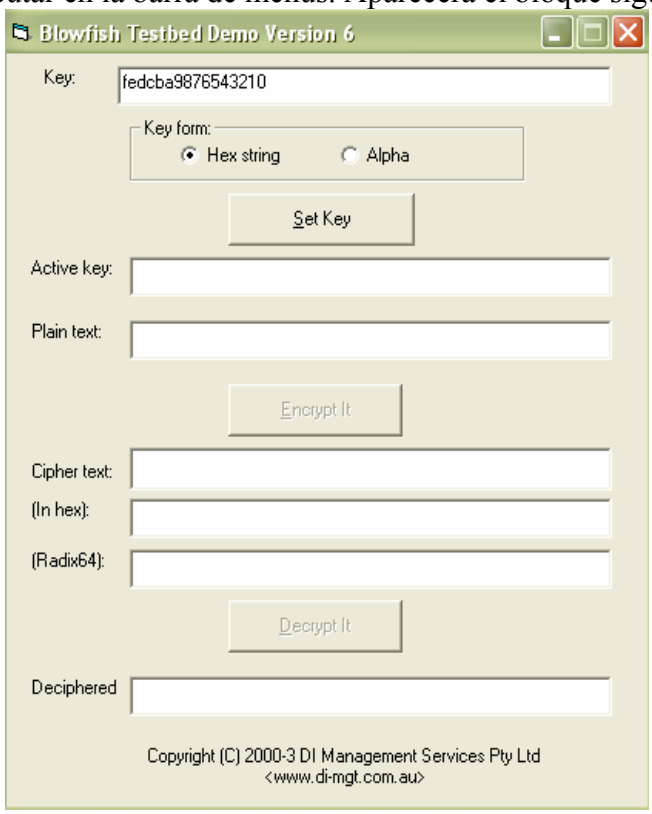


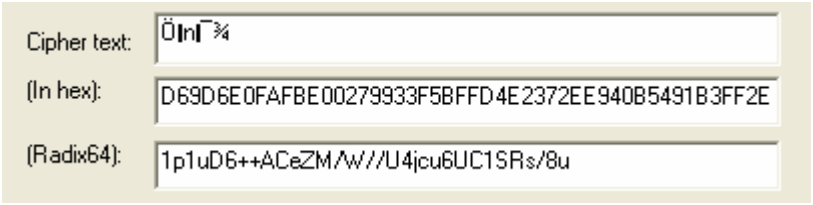
Fig. A-2: Forma ejecutable del BV6.

El primer espacio es para la clave pública que puede ser ingresada con formato Hexadecimal o como alfanumérica, según se seleccione la opción. Una vez seleccionada la clave, dar clic en el botón SET KEY para generar la clave activa (correspondiente a la clave extendida generada por las subclaves generadas en el Blowfish, la cual no es muy importante en nuestro objetivo, ya que lo que importa es comparar el cifrado de esta versión con la del presente trabajo). Esta clave activa es el resultado final del proceso de generación de subclaves que se hace previo a cifrar el mensaje.

En el espacio que dice “Plain Text” se debe escribir el mensaje que se desea cifrar.

Después dar clic en el botón “Encrypt It” para cifrar el texto original con la clave activa generada. Después aparecerá el mensaje cifrado en tres tipos de formato:

- Alfanumérico
- Hexadecimal
- Radix64 (redundante).
-

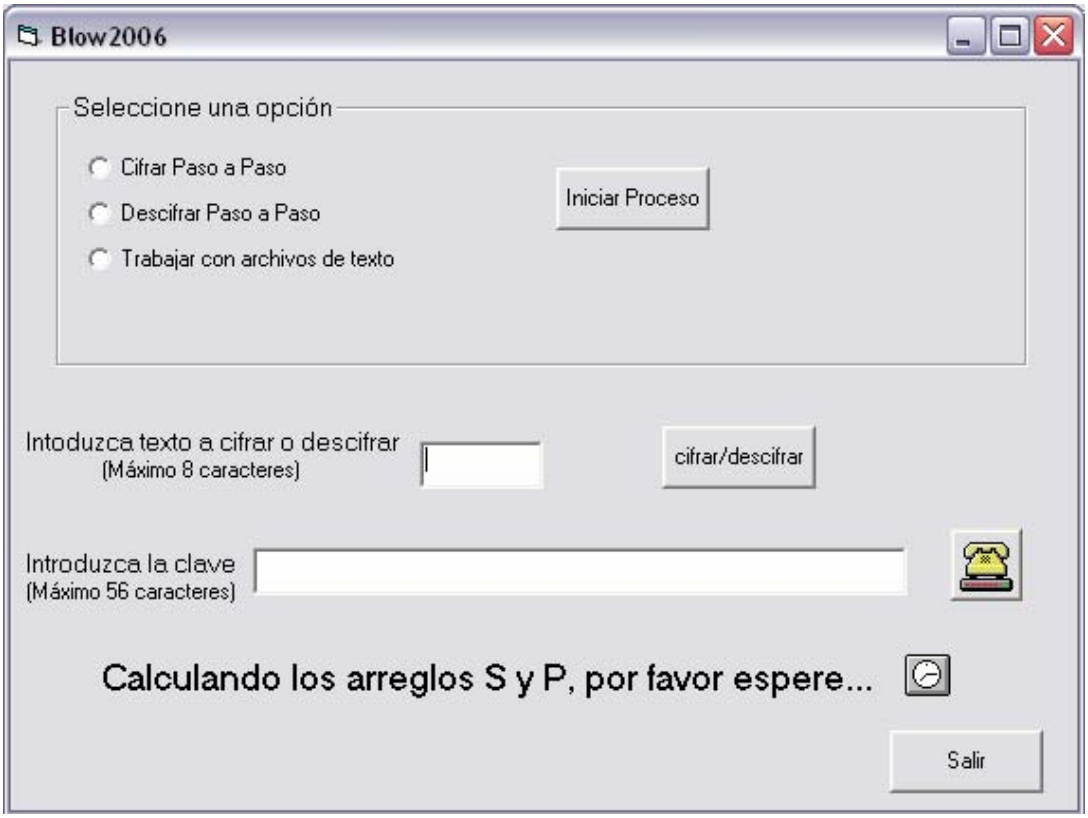


Por último dar clic en el botón “Decrypt It” para recuperar el mensaje original.

Este programa agrega siempre ocho caracteres extra (carácter ascii equivalente a “08”) al texto a cifrar para dar mayor seguridad al proceso de cifrado y si el texto es menor a ocho caracteres (64 bits) o no tiene una longitud que sea múltiplo de ocho, agrega el carácter ascii equivalente al número de caracteres faltantes, el número de veces que sea necesario, es decir, si se quiere cifrar el texto: “prueba”, en ascii-hexadecimal “707275656261” son seis caracteres, faltan dos caracteres para tener los ocho caracteres (64 bits) que el Blowfish requiere como mínimo, el programa le agregará dos veces el carácter ascii equivalente a “02” mas los ocho caracteres ascii equivalente a “08” que siempre le agrega, por lo que el programa cifrara:
“707275656261**0202080808080808080808**”
si el texto a cifrar fuera de cinco caracteres le faltarían tres caracteres, por lo que le agregaría tres veces el carácter ascii equivalente a “03” , así en los demás casos.
Otra particularidad de este programa es que solo puede descifrar el texto que acaba de cifrar, no puede descifrar texto introducido por el usuario.

Anexo 4: Tabla de Propiedades de Los Objetos

Forma 1

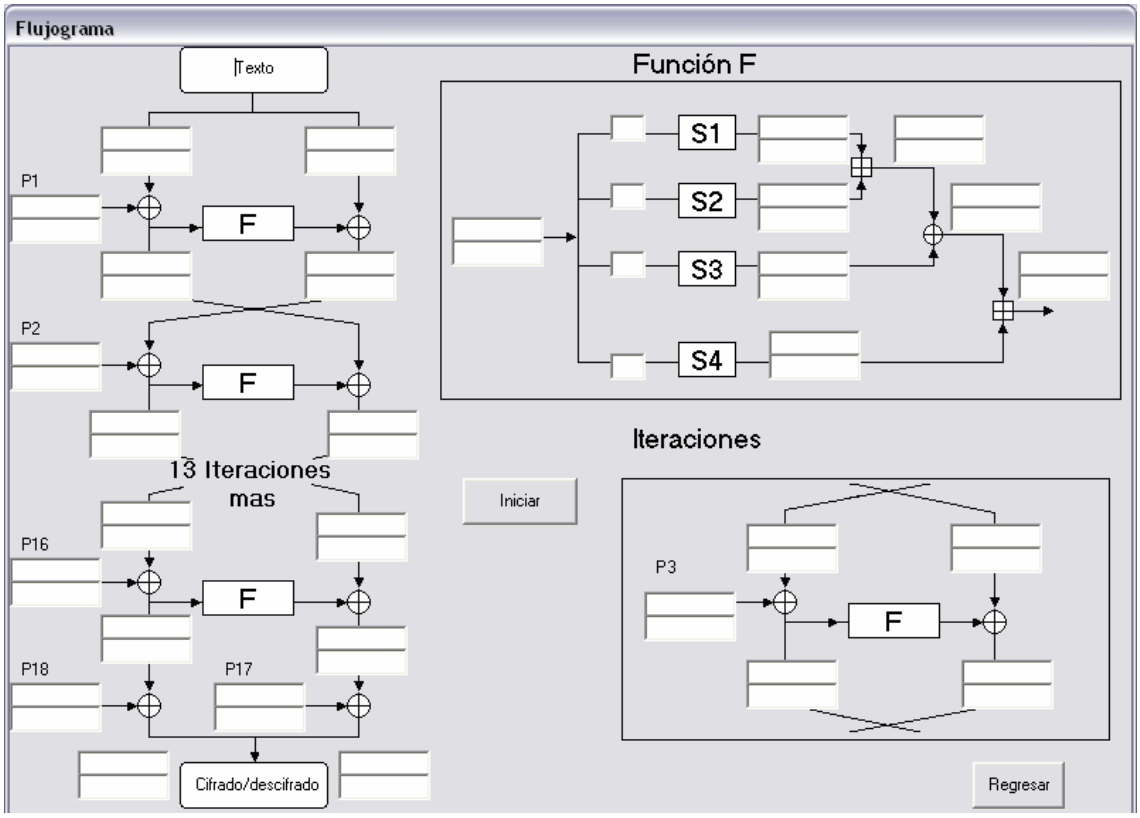


Control	Propiedad	Valor	Control	Propiedad	Valor
Form1	Name	Form1	Text1	Name	Text1
	Caption	Blow2006		Locked	True
Optcifpap	Name	Optcifpap		MaxLength	8
	Caption	Cifrar Paso a Paso		Text	""
Optdescifpap	Name	Optdescifpap	Text2	Name	Text2
	Caption	Descifrar Paso a Paso		Locked	True
Optarchivos	Name	Optarchivos		MaxLength	56
	Caption	Trabajar con archivos de texto		Text	""
Cmdiniciarproc	Name	Cmdiniciarproc	MSComm1	Name	MSComm1
	Caption	Iniciar Proceso		Commport	1
	Enabled	False		Settings	9600,n,8,1
Cmdcifrardescifrar	Name	Cmdcifrardescifrar	Label6	Name	Label6
	Caption	cifrar/descifrar		Caption	Procesando, por favor espere,,,
	Enabled	False		Visible	False
Image1	Name	Image1	cmdsalir	Name	cmdsalir
	Visible	False			
	Picture	TIME.bmp			

Forma 2

Control	Propiedad	Valor	Control	Propiedad	Valor
Form2	Name	Form2	Text1	Name	Text1
	ControlBox	False		MaxLength	56
	Caption	cifrar y descifrar archivos		Locked	True
Optcifarch	Name	Optcifarch		Text	""
	Caption	Cifrar Archivos	Text3	Name	Text3
Optdescifarch	Name	Optdescifarch		Locked	True
	Caption	Descifrar Archivos		Multiline	True
Cmdabrir	Name	Cmdabrir		ScrollBars	2-Vertical
	Caption	Abrir archivo		Text	""
	Enabled	False	Text4	Name	Text4
Cmdregresar	Name	Cmdregresar		Locked	True
	Caption	Regresar		Multiline	True
Cmdcd	Name	Cmdcd		ScrollBars	2-Vertical
	Caption	cifrar/descifrar		Text	""
	Enabled	False	Image1	Name	Image1
Label5	Name	Label5		Visible	False
	Caption	Procesando, por favor espere,,,		Picture	TIME.bmp
	Visible	False	CommonDialog1	Name	CommonDialog1
Text2	Name	Text2	Text5	Name	Text5
	Appearance	0-Flat		Appearance	0-Flat
	Text	00		Text	00
	Locked	True		Locked	Trae

Forma 3



Control	Propiedad	Valor	Control	Propiedad	Valor
Form3	Name	Form3	image1	Name	image1
	ControlBox	False		Picture	flujograma.gif
	Caption	Flujograma	image2	Name	image2
Text1	Name	Text1		Picture	iteracion.gif
	Locked	True	image3	Name	image3
	text	Texto		Picture	función F.gif
Text3	Name	Text3	Cmdinic	Name	Cmdinic
	Locked	True		Caption	Iniciar
	text	cifrado/descifrado	Cmdregresar	Name	Cmdregresar
Cmdcontinuar (Esta oculto bajo el boton Cmdinic)	Name	Cmdcontinuar		Caption	Regresar
	Caption	Continuar			
	enabled	False			
	visible	False			

Las cajas de texto utilizadas son:

Matriz de controles: Text2(0-18), Text4(0-5), Text6(0-5), Text11(0-18), Text17(0-3).

Cajas de texto individuales: Text1, Text3, Text5, Text7, Text8, Text9,Text10,Text12, Text13, Text14, Text15, Text16.
Todas tienen las siguientes propiedades: **Locked** = True y **Text** = “”

Anexo 5: Software utilizado para la construcción de las tarjetas.

EAGLE VERSIÓN 4.08r2

Software utilizado para el diseño de circuitos impresos.

Este programa posee un panel de control a partir del cual se puede generar o editar: circuitos teóricos (archivos .sch) y circuitos impreso (archivos .pcb)
De esta manera el programa permite dibujar circuitos y generar cada una de las caras de un circuito impreso, también plantilla de perforaciones y máscaras de soldadura.
El programa provee una amplia gama de librerías de componentes, conectores, sensores, etc. y permite la generación de nuevas librerías y la edición de las librerías existentes.

La versión de prueba del programa que está disponible en forma gratuita en Internet[28] tiene las siguientes limitaciones:

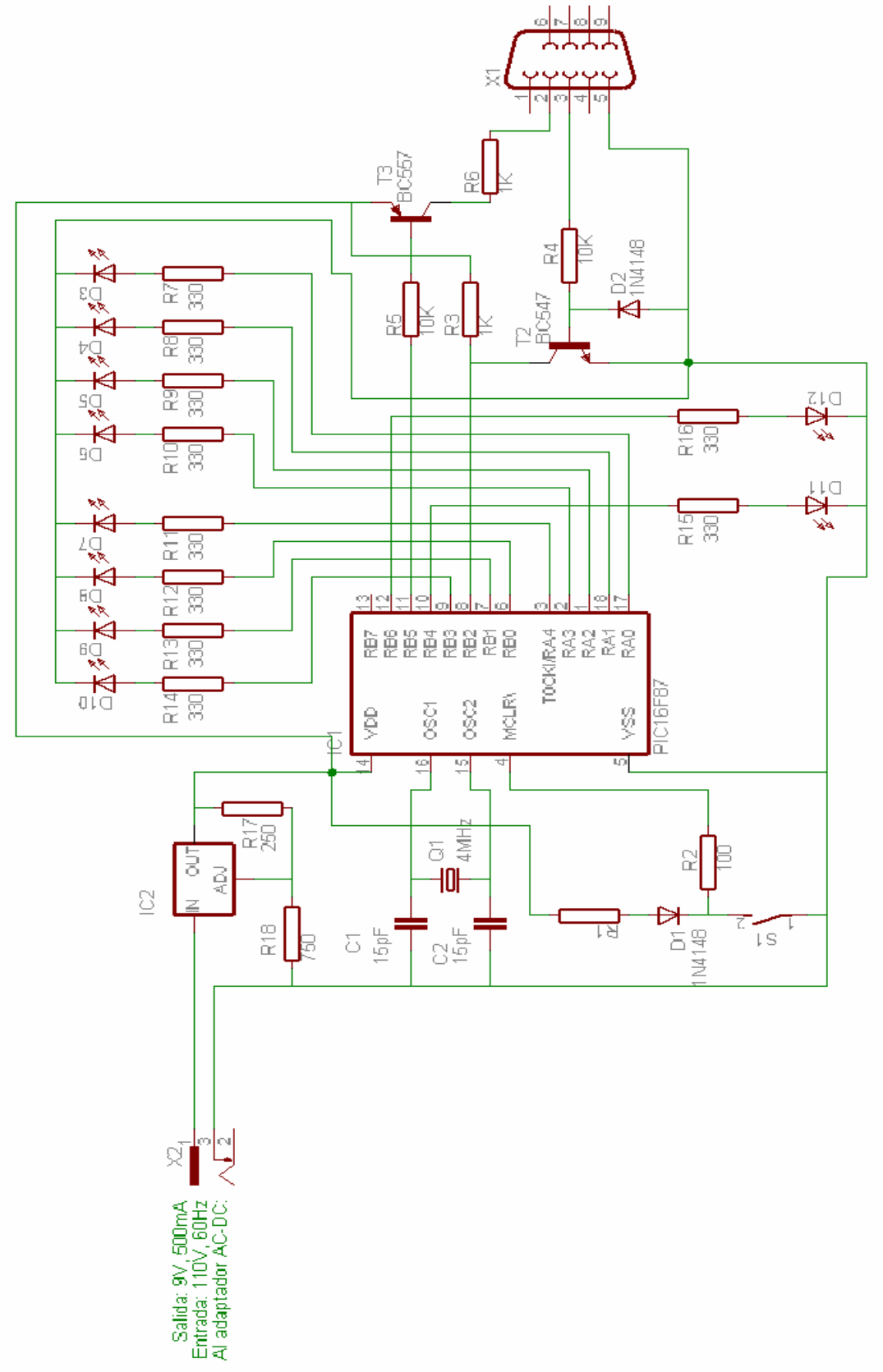
• Tamaño máximo de la tarjeta : 100 x 80mm.
• Cantidad de caras de circuito impreso disponibles : 2 (bottom y top)

Diseño de circuitos teóricos (Schematics)
En este entorno se permite la edición de circuitos con la simbología utilizada normalmente en electrónica, con el comando USE o su respectivo botón se accede a las librerías disponibles, donde se pueden ver el nombre y esquema real de los diferentes elementos.

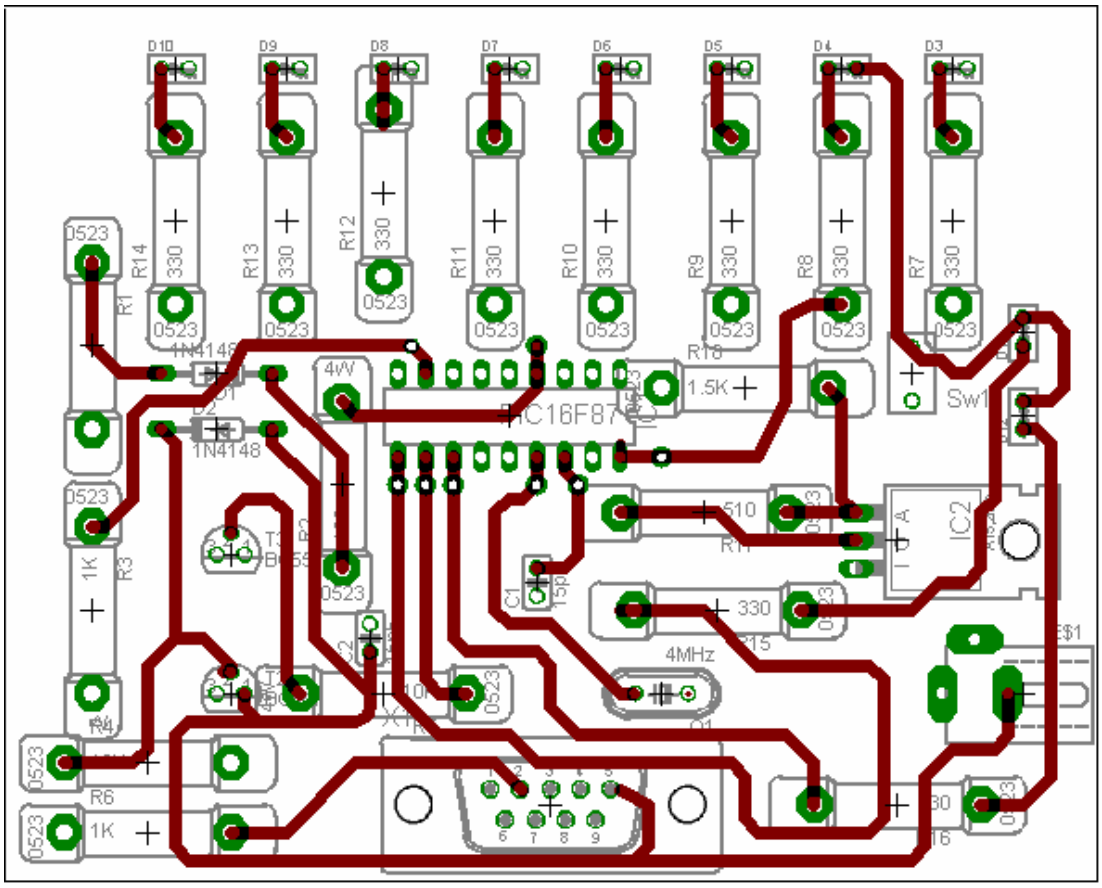
Diseño de circuitos impresos (PCB)
Existen dos formas diferentes de generar un circuito impreso, a partir de un circuito teórico (schematics) o haciendo el circuito de una vez en este entorno, aquí los elementos se ven en forma y dimensiones reales, los usuarios pueden ir dibujando las pistas manualmente o con la herramienta AutoRoute para que el programa genere las pistas de manera automática, siempre y cuando haya sido creado a partir de un circuito teórico ya que si fue creado de una vez en ese entorno las pistas deben dibujarse manualmente siempre

[28] http://www.cadsoft.de/freeware.htm

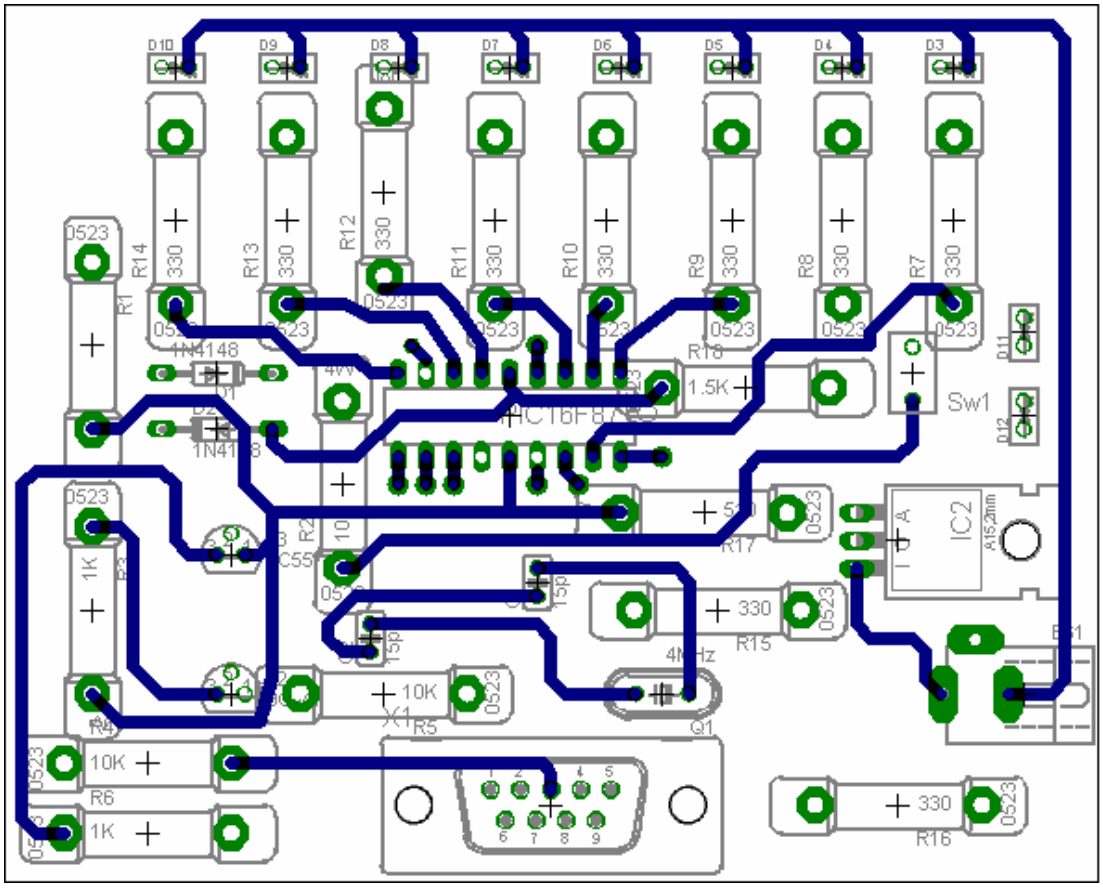
Anexo 6: Diagrama Esquemático de las Tarjetas.



Anexo 7: Diagrama de circuito impreso de las tarjetas.



Cara superior de la tarjeta.



Cara inferior de la tarjeta.