UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN

TRABAJO FIN DE CARRERA

TITULO: SIMULADOR DEL PROTOCOLO DE TRANSPORTE OSI EN ENTORNO WINDOWS.

ESPECIALIDAD: TELEFONÍA Y TRANSMISIÓN DE DATOS

AUTOR: FRANCISCO JAVIER FLEITAS NEGRÍN

TUTOR: DOMINGO MARRERO MARRERO

MES/AÑO: JULIO/95

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN

TRABAJO FIN DE CARRERA

TITULO: SIMULADOR DEL PROTOCOLO DE TRANSPORTE OSI EN ENTORNO WINDOWS

Calificación : SOBRESALIENTE, 9 **Fecha :** 13-07-95

Fdo. Presidente **Fdo. Secretario** **Fdo. Vocal**

1. Introducción

El **Simulador del protocolo de transporte OSI** es un software diseñado para ejecutarse en un entorno Windows 3.1 o superior, realizado con las poderosas herramientas de la Programación Orientada a Objetos (OOP) que permite el C++ , con la filosofía de manejo de mensajes inherente a Windows y apoyado por la librería de clases ObjectWindows.

El programa representa el comportamiento del nivel más crítico dentro del modelo de referencia promovido por la Organización Internacional de Normalización (ISO) para la interconexión de redes de computadoras.

La capa de transporte no sólo representa otra capa más dentro del modelo, sino que desarrolla un papel muy importante y viene a ser realmente el corazón de la jerarquía de protocolos. Su tarea consiste en hacer que el transporte de datos se realice de forma segura, desde la máquina fuente hasta la máquina destinataria, independientemente de las redes físicas que haya entre ellas.

La aplicación está estructurada en tres bloques: un tutorial OSI, una simulación interactiva del protocolo de transporte OSI y una simulación visual del mismo.

Con la primera parte se pretende conocer el modelo OSI en general y sus funciones capa por capa. En la simulación visual y en la interactiva se permite al usuario comportarse como el nivel que se encuentra por encima del de transporte (nivel de sesión) o como subred de telecomunicación y comprobar como reacciona la capa de transporte en función de sus acciones.

Además, en la simulación interactiva, se permite que el usuario especifique valores para los parámetros que caracterizan la calidad del servicio que la capa de transporte va a utilizar, elija el tipo de protocolo de transporte que se va a emplear entre origen-destino, la información que quiere transmitir, etc. Esos datos introducidos provocarán unas actuaciones u otras por parte del nivel de transporte que podrán ser seguidas por el usuario.

Este trabajo nace por cuatro motivos básicos:

- Ofrecer un medio de apoyo para los alumnos de la E.U.I.T.T., y en concreto de la asignatura Redes de Ordenadores, con la finalidad didáctica de aclarar y ampliar conceptos que son abstractos sin un soporte hardware y software real acerca del modelo de arquitectura de red que plantea ISO, y en particular sobre su nivel de transporte.

- Desarrollar un programa siguiendo la actual tendencia de programación, la Programación Orientada a Objetos. La OOP ha tomado las mejores ideas de la programación estructurada y las ha combinado con varios conceptos nuevos y potentes que incitan a contemplar la tarea de programar desde un nuevo punta de vista.

- Continuar y ampliar la tarea de crear aplicaciones para entornos gráficos y multitarea como Windows, con su filosofía de mensajes provenientes del usuario o de otras aplicaciones, que el programador debe gestionar y responder.

- Como Trabajo Fin de Carrera, requisito fundamental para la obtención del título de Ingeniero Técnico de Telecomunicación.

2. Programación en Windows

2.1. Introducción

La programación en Windows conlleva una serie de diferencias con la programación tradicional, y por tanto, unos planteamientos totalmente opuestos a los habituales.

En primer lugar, un programa Windows opera en un entorno multitarea, donde se tiene la posibilidad de tener activos varios programas simultáneamente y poder pasar fácilmente de uno a otro. Por este motivo, tiene que compartir el acceso a los recursos del sistema con otras aplicaciones. Este acceso compartido y restringido impone requisitos con los que un programa DOS no se enfrenta y requiere elementos adecuados en el código fuente del programa Windows.

En segundo lugar, la estructura de programación Windows se orienta a sucesos, en oposición a la programación secuencial o estructurada. Los sucesos, como la pulsación de una tecla o del botón izquierdo del ratón, la selección de un elemento del menú, etc se transforman en mensajes. El trabajo del programador consiste en responder a estos mensajes.

Por ejemplo, si se pulsa un botón del ratón sobre una ventana, las cosas ocurren más o menos así:

- Windows intercepta la pulsación del botón del ratón.
- Se toma la posición y las características de la pulsación (simple, doble, botón izquierdo, botón derecho, etc).
- Según las cotas, se identifica la ventana afectada.
- Windows envía al programa un mensaje que informa de la acción del usuario en dicha ventana, y cuales son sus características.
- El programa procesa el mensaje y toma las acciones pertinentes.

Todos los mensajes se procesan siguiendo un orden temporal y unas preferencias; unos tienen más prioridad que otros. En el sistema se producirán multitud de sucesos en un corto intervalo de tiempo. El programador debe gestionar solamente parte de los mensajes enviados a cada ventana de la aplicación, aquellos que le interese. Otros son gestionados directamente por Windows, y otros que no interesan a la aplicación ni a Windows (por ejemplo, pulsaciones de teclas que no impliquen acciones programadas) se procesan en una rutina interna de Windows y se retiran de la cola de mensajes.

Las **palabras claves** son, por tanto, **objetos** (ventanas, controles ,cajas de diálogo, etc), **sucesos-mensajes y procesamiento del mensaje**; es decir, llamada a la función miembro que responda a cada mensaje que la aplicación necesita gestionar.

El presente simulador ha sido realizado con Borland C++ y la librería ObjectWindows. En los siguientes apartados se explica brevemente lo fundamental de estos dos elementos.

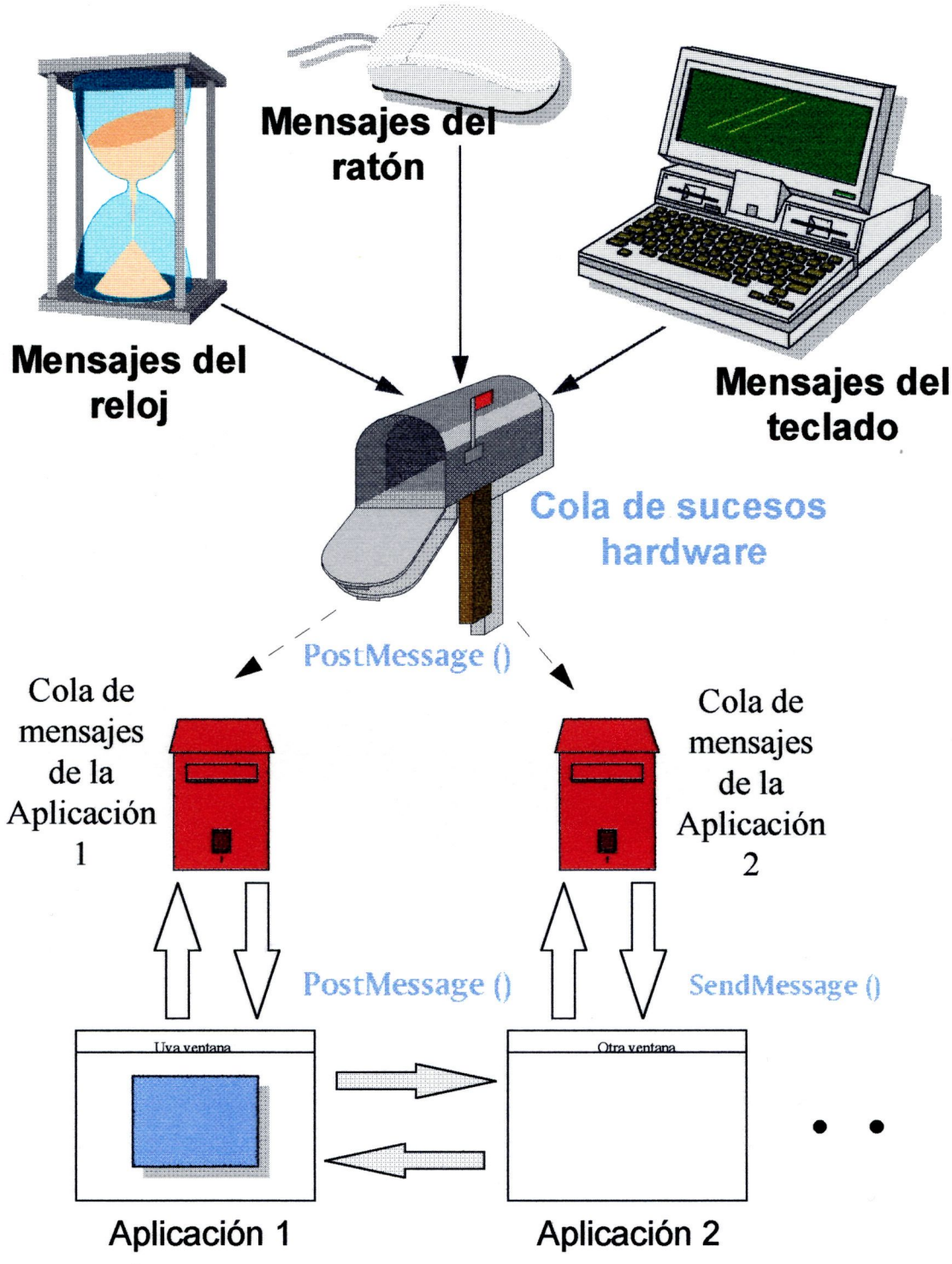
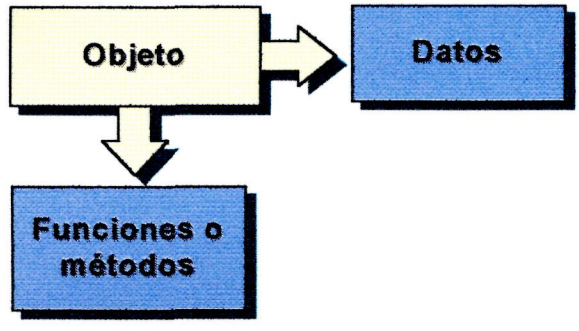


Ilustración 2–1. Esquema del flujo de mensajes

2.2. *Clases y objetos.-*

El C++ es un lenguaje de programación orientado a objetos. Los objetos son tipos de datos definidos por el usuario (se parecen a las estructuras del C o a los registros de Pascal), pero engloban también funciones diseñadas para actuar sobre estos datos. Estas funciones se llaman métodos.



Una clase es la definición de un objeto, su declaración. Por ejemplo:

```
class Empleado
{
 char nombre [61] ;
char telefono [9] ;
int sueldo;

void MarcarTelefono();
void ImprimirFactura();
}
```

No se reserva memoria para una clase, pero sí para un objeto cuando se declara. Por ejemplo:

```
Empleado recepcionista;
```

2.3. *Constructores y destructores.-*

Es muy frecuente que una cierta parte de un objeto necesite una iniciación antes de que pueda ser utilizada. C++ permite que los objetos se den a sí mismos valores iniciales cuando se crean. Esta iniciación se lleva a cabo mediante una función llamada constructor. El constructor tiene el mismo nombre que la clase.

```
class Ejemplo
{
 int x;
 char bienvenida 30 ;
public:
    Ejemplo(int y; char mensaje[30]);
        .
        .    // demás métodos
        .
}

Ejemplo::Ejemplo(int y,char mensaje[30])
{
x=y;
strcpy(bienvenida,mensaje);
}
```

Los objetos, en muchos casos, necesitan hacer una o más cosas cuando son destruidos, por ejemplo, liberar memoria que hubiese reservado anteriormente. Por ello existe el destructor que tiene el mismo nombre que la clase pero precedido por el símbolo ~.

2.4. *Herencia, encapsulamiento y polimorfismo.-*

Para que un lenguaje pueda llamarse orientado a objetos, debe proporcionar cuatro características: encapsulamiento y control de acceso, abstracción, herencia y polimorfismo.

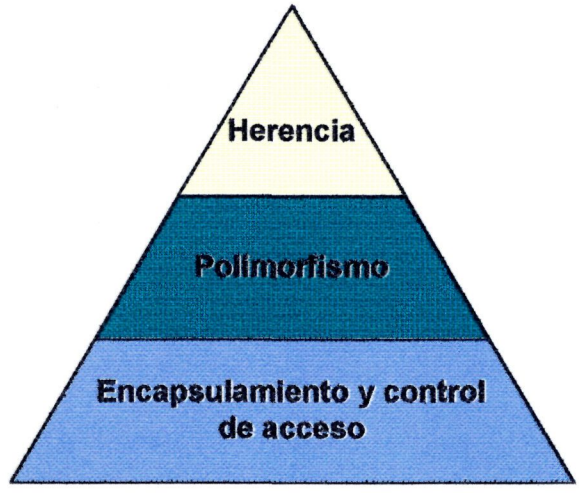


Ilustración 2–2. Bases de la Programación Orientada a Objetos

2.4.1. Encapsulamiento y control de acceso

Los datos y las funciones deben estar encapsulados en un objeto. Por defecto, todos los elementos de un objeto son privados para el resto del programa, lo cual proporciona un nivel de protección que evita que puedan ser utilizados incorrectamente. Sin embargo se puede hacer que sean públicos o protegidos, mediante las palabras clave public y protected.

Los campos public puedan ser leídos y escritos desde cualquier punto.

Los campos protected pueden ser leídos y escritos por métodos de esta clase, y por los métodos de los descendientes de dicha clase.

2.4.2. Herencia

La herencia es el proceso mediante el cual un objeto puede adquirir las propiedades de otro objeto. Por ejemplo, podríamos tener un objeto televisión con ciertos datos (volumen, canal seleccionado actualmente...) y métodos (modificar volumen y el ajuste vertical, cambiar de canal, etc). Entonces, se podría definir un nuevo objeto, un televisor en color, que herede las características del televisor normal. Además de las características de una televisión, que no tienen que redefinirse, porque han sido heredadas, un televisor en color tendría métodos para ajustar el color y el tono.

2.4.3. Polimorfismo

El polimorfismo es la capacidad que tienen los objetos hijos de redefinir los métodos de sus objetos padre si lo necesitan. Por ejemplo, un método llamado Dibujar() podría asociarse a un objeto que implementase un coche, CocheGenerico(), para que dibujase cualquier coche. Se podría entonces intentar definir un nuevo objeto, un Honda, que heredaría las propiedades del objeto Cochegenerico() pero tendría que redefinir el método Dibujar() para que dibujase un Honda, no un coche cualquiera. Esta técnica de que un mismo método muestre diferentes comportamientos dependiendo del tipo de objeto se denomina polimorfismo.

El polimorfismo es esencial para la OOP porque ayuda al programador a gestionar programas cada vez más complejos y permite crear bibliotecas de clases, que pueden ser aprovechadas y adoptadas a nuestras necesidades concretas.

2.5. Conclusión

Una filosofía de programación que encaja muy bien con la POO es la basada en mensajes de Windows.

2.6. Librería ObjectWindows

La librería ObjectWindows es una librería de clases que incorpora las ventajas de la POO para hacer que los programas Windows sean más fáciles de desarrollar. La librería suministra una estructura base para las aplicaciones Windows: comportamientos de las ventanas, cuadros de diálogo, controles, procesamiento de mensajes, etc.

El mecanismo de herencia permite derivar nuestras propias clases, con los datos y métodos que nos interesan, a partir de la clase base.

ObjectWindows ofrece tres ventajas fundamentales:

- **Encapsulamiento de la información de una ventana**

Los objetos ventana en ObjectWindows almacenan un parámetro que necesitan muchas funciones del API de Windows, un handle (un valor entero) de ventana y que pueden usar automáticamente sin necesidad de especificarlo en todas las ocasiones. Igual ocurre con los identificadores de las ventanas descendientes, los atributos de las ventanas y sus títulos.

- **Abstracción de las funciones del API de Windows**

Microsoft ha definido casi 600 funciones del API que configuran el interfaz de Windows. Aunque se puede llamar a cualquier función del API directamente desde el Borland C++, la librería permite simplificar muchas de estas llamadas.

- **Respuesta automática a los mensajes**

Con la programación tradicional se utilizan gran número de sentencias switch...case para responder a los mensajes, lo que hace difícil leer el código resultante, ya que ventanas, botones, texto, listas desplegables, y en general todo, se controla enviando y recibiendo mensajes hacia y desde ellos.

Con la librería ObjectWindows, sencillamente se define un método para cada mensaje que se quiere gestionar. El objeto llamará automáticamente al método que corresponda cuando reciba el mensaje. Para los mensajes que no tengan asignado un método definido por el usuario, ObjectWindows realiza un procesamiento por defecto. Este proceso se lleva a cabo gracias a las tablas virtuales de administración dinámica (TVAD). Cuando se declara un método que se quiere que gestione un mensaje de Windows, se le asigna un índice de administración. Por ejemplo:

```
virtual void PulsarBotonIzqda (RTMessage Msg)=
=[WM_FIRST+WM_LBUTTONDOWN];
                    ⇑ índice de administración
```

Cuando un objeto ObjectWindows recibe un mensaje, intenta emparejar el número de mensajes con la lista de índices de administración definidos. Si se puede emparejar, el objeto llama al método adecuado.

2.7. Ficheros necesarios

Los ficheros mínimos que componen una aplicación Windows son: el fichero fuente con extensión .CPP, un archivo de recursos (.RC) y el módulo de definición con extensión .DEF. Ëste último le da al enlazador información acerca de los segmentos de datos, el código, etc. ObjectWindows proporciona un módulo de definición por defecto, si no se incluye ninguno.

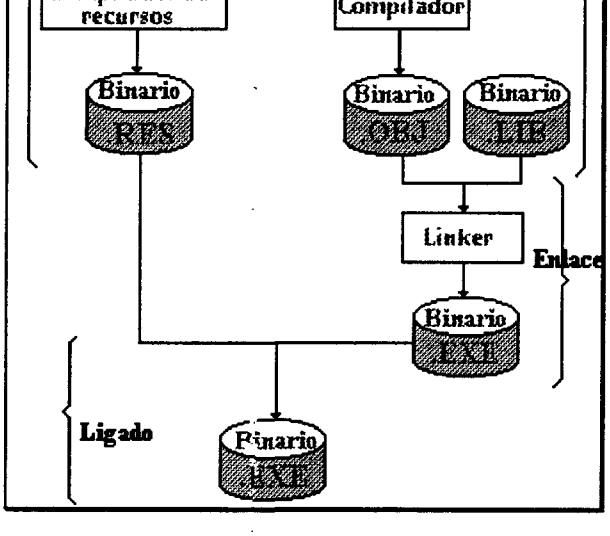


Ilustración 2–3. Desarrollo de un proyecto Windows

A diferencia de las aplicaciones para el DOS, las aplicaciones para Windows hacen uso de algo llamado recursos. Un recurso es un elemento estático del programa almacenado en el .EXE, pero que no se carga en tiempo de ejecución, sino que la aplicación lo carga y utiliza cuando lo necesita y lo descarta cuando ya no le hace falta. Ejemplos de recursos son los menús, los cuadros de diálogo, los iconos y los cursores.

El Resource Workshop de Borland permite crear y modificar los recursos de Windows.

La ventaja de los ficheros .RC es la de poder extraer los recursos de una aplicación y modificarlos a nuestro gusto, sin tener que acceder al código fuente. Podemos, por ejemplo, modificar el idioma del programa sin modificar la funcionalidad del programa.

2.8. Tipos de datos, estructuras e identificadores en Windows

Windows utiliza frecuentemente **tipos de datos** que reciben nombres especiales:

```
typedef unsigned int     WORD;
typedef unsigned long    DWORD;
typedef char far         LPSTR;
```

Un tipo de dato especial y fundamental para la programación en Windows es el handle. Se define de la siguiente forma:

```
typedef  WORD     HANDLE;
```

A su vez, existen handles específicos que se establecen para hacer el código más legible.

```
typedef  HANDLE HWND;        //Identifica a una ventana
typedef  HANDLE HICON;       //Identifica a un icono
typedef  HANDLE HDC;         //Identifica a un contexto de dispositivo
typedef  HANDLE HBITMAP;     // Identifica a un bitmap
```

```
typedef  HANDLE HBRUSH;          //Identifica a una brocha
```

Por tanto, un handle es un valor entero que se utiliza para identificar a un objeto en una aplicación, como una ventana, una pluma, un menú, etc.

La **estructura** TMessage también es crucial, puesto que contiene la información acerca de los mensajes que Windows envía a la aplicación, y está presente en la declaración y llamada de todas las funciones miembro relacionadas con los mensajes.

```
struct TMessage{
        HWND Receiver;
        WORD Message;
        union{
                WORD WParam;
                struct tagWP{
                        BYTE Lo;
                        BITE Hi;
                }WP;
        };
        union{
                DWORD LParam;
                struct tagLP{
                        WORD Lo;
                        WORD Hi;
                }LP;
        };
        long Result;
};
```

El campo Message contiene el identificador del mensaje (WM_MOVE, mensaje de mover la ventana; WM_PAINT, pintado de la ventana, etc). Los miembros LParam y WParam contienen información específica cuyo significado exacto depende del tipo de mensaje. El campo HWND es el handle de la ventana que recibe el mensaje

Los **identificadores** son constantes definidas, formadas por un prefijo, el símbolo _ y una descripción; que cumplen diversas funciones:

- Establecer una relación entre la acción del usuario y la respuesta de dichas acciones.

- Elegir distintos estilos de ventanas. Por ejemplo, si se indica para una ventana el estilo CS_NOCLOSE, no aparecerá en el menú de control la opción de Cerrar.
- Acceder a recursos predefinidos. Si el identificador es IDC_WAIT, el cursor tomará forma de reloj de arena.
- Distinguir los controles dentro de una caja de diálogo. ID_BOTONAYUDA nos permitirá acceder al botón de ayuda dentro de un cuadro.

La tabla presenta una muestra de los prefijos identificadores de Windows:

Prefijo	*Categoría*
ID	Identificador en general
CM	Identificador de un elemento de un menú
CS	Estilo de la clase
IDI	Identificador de un icono
IDC	Identificador de un cursor
WS	Estilo de la ventana
WM	Gestión de mensajes de Windows

2.9. Una aplicación Windows básica

El código necesario para crear un programa Windows básico con ObjectWindows consta de :

⇒ Una clase para representar la aplicación en ejecución derivada de la clase TApplication de la librería.

```
class TMiAplicacion : public TApplication
{
 public:
        TMiAplicacion(LPSTR Aname, HINSTANCE hInstance,
                      HINSTANCE  hPrevInstance, LPSTR lpCmdLine,
                      int nCmdShow)
        :TApplication(Aname,hInstance,hPrevInstance,lpCmdLine,nCmdShow) {}
;

virtual void InitMainWindow();
}
```

⇒ Una clase para representar la ventana principal de la aplicación derivada de la clase TWindow de la librería. La clase TWindow implementa ventanas genéricas que tienen un título, el menú de control, un icono, que pueden moverse y cambiar de tamaño, etc.

⇒ Un bucle de programa WinMain(). La función WinMain() equivale a la función main() de los programas en C, pero a diferencia de la función main() que incluye buena parte del código del programa, realiza otras tareas.

```
int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow)
{
 TMiAplicacion MiAplicacion( "Programa ejemplo", hInstance, hPrevInstance,
                             lpCmdLine,nCmdShow);
MiAplicacion.Run();
return MiAplicacion.Status;
}
```

En la primera sentencia se construye un objeto MiAplicacion, de la clase que hemos creado, derivada de TApplication, TMiAplicacion. Lo que estamos haciendo es notificar a Windows la presencia de la aplicación y sus requisitos.

En la segunda línea se llama a la función Run que ejecuta la aplicación. Entre otras cosas llama al método InitMainWindow() para inicializar la ventana principal , y a

MessageLoop que es un bucle que procesa cualquier mensaje de entrada y lo despacha a su destino, y del que no sale hasta que llega el mensaje de salir, que lleva el identificador WM_QUIT.

Finalmente se devuelve el estado de la aplicación.

A continuación se adjunta el código de una primera aplicación para ObjectWindows:

```
#include <owl.h>      //Incluye un archivo de cabecera para indicar que la aplicación
                      //es ObjectWindows
//----------------------------declaración de las clases----------------------------

class TMiAplicacion : public TApplication
{
 public:
        TMiAplicacion(LPSTR AName,HINSTANCE hInstance, HINSTANCE
                      hPrevInstance,LPSTR lpCmdLine, int nCmdShow)
              : TApplication(AName, hInstance, hPrevInstance, lpCmdLine,
                             nCmdShow) {};
        virtual void InitMainWindow();
};
_CLASSDEF(TMiVentana)
class TMiVentana : public TWindow
{
 public :
        TMiVentana(PTWindowsObject Aparent, LPSTR Atitle)
         : TWindow(AParent, ATitle) {} ;
}

//-------------------------------TMiAplicacion-------------------------------

void TMiAplicacion::InitMainWindow()
{
 MainWindow=new TMiVentana(NULL, " Mi primera ventana ");
}

//-------------------------------WinMain-------------------------------

int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow)
{
 TMiAplicacion MiAplicacion( "Programa ejemplo", hInstance, hPrevInstance,
                             lpCmdLine,nCmdShow);
```

```
MiAplicacion.Run();
return MiAplicacion.Status;
}
```

Tras compilar y ejecutar obtendremos el resultado siguiente:

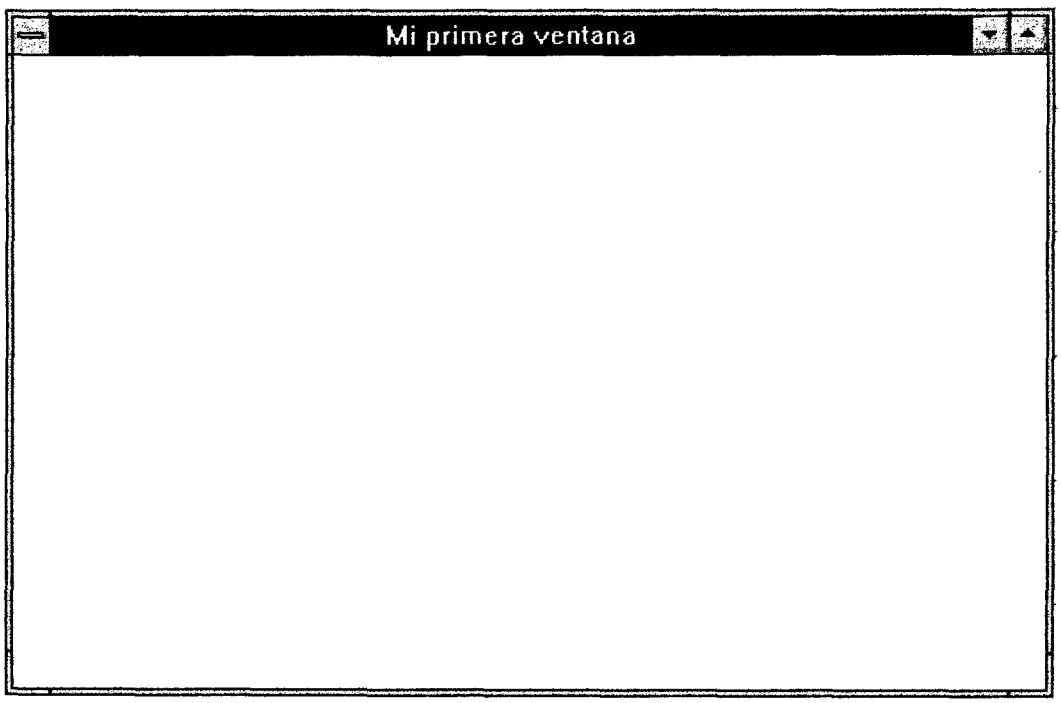


Ilustración 2–4. Una sencilla aplicación ObjectWindows

2.10. Los contextos de dispositivos y la función Paint

El aspecto visual de un programa Windows es muy importante. Siempre que queramos dibujar un gráfico, texto, pintar una zona de la pantalla, etc hemos de obtener un handle del contexto de dispositivo que vayamos a usar (pantalla, plotter, impresora, memoria...). Este handle es necesario porque cualquier función del GDI (interfaz de dispositivo gráfico) de Windows a la que llamemos, lo necesita.

El contexto de dispositivo representa una ventana en la pantalla, la pantalla entera o una impresora. En las aplicaciones DOS, los gráficos se escriben normalmente directamente en la pantalla de vídeo. Bajo Windows es diferente, se escribe en una pantalla virtual (el contexto de dispositivo) antes de copiar en la pantalla física.

Existen varias formas de obtener un contexto de dispositivo. La siguiente es una de ellas, que se utiliza para escribir un texto en pantalla:

```
HDC hdc;
hdc=GetDC(HWindow);              //Obtiene el contexto
TextOut(hdc,40,40,"Ejemplo de utilización de DC",30);  //Escribe en él
ReleaseDC(HWindow,hdc);          //Lo libera
```

La última sentencia nunca debe olvidarse porque descarga el contexto de la memoria, evitando que el sistema se quede sin ella.

El proceso explicado sólo mostrará el texto una vez. Pero si colocamos una ventana sobre él o movemos la ventana, se borrará. Se necesita un mecanismo para conseguir que el texto permanezca en pantalla. Este mecanismo es provisto por la función Paint, que se llama cuando se crea una ventana, cuando se cambia su tamaño o cuando se descubre una ventana que ha estado oculta por otras ventanas.
Nuestras ventanas, derivadas de TWindow, deben redefinir la función Paint de TWindow, según nuestros propósitos.

```
void TMiVentana::Paint (HDC DC, PAINTSTRCUT&)
{
 TextOut(DC, 40, 40, "Utilización de Paint", 22) ;
}
```

Además, la función Paint, obtiene y libera de forma automática el contexto de dispositivo, sin necesidad de la intervención del programador.

Todo lo anterior es extrapolable a la presentación de gráficos, brochas, plumas, etc. Si el dibujado de un bitmap no se encuentra dentro del método Paint, el gráfico se borrará cada vez que se produzca un mensaje WM_PAINT. Para evitarlo:

```
void Ventanahija7::Paint ( HDC DC, PAINTSTRUCT&)
{
 DibujarBitmap ( hbmpMyBitmap, HWindow, 0, 0 );
}
```

donde DibujarBitmap es una función que muestra un bitmap en una ventana. Por su importancia dentro del programa se muestra su listado y se explica el proceso necesario para emplear gráficos:

- En primer lugar, crear el bitmap y declararlo en el fichero .RC.
- Posteriormente, cargarlo en memoria. Normalmente se suele hacer en el constructor de la ventana:

  hbmpMyBitmap=LoadBitmap(GetApplication→hInstance,"CAPA5");
- Dibujarlo mediante la llamada a la función Dibujarbitmap, normalmente dentro del método Paint.
- Liberar la memoria ocupada por el bitmap cuando ya no se precise más:

  DeleteObject(hbmpMyBitmap);

La función DibujarBitmap requiere un estudio más detallado. Recibe como parámetros el handle del bitmap que hemos cargado, la ventana en donde queremos plasmar el gráfico y la posición x e y dentro de la ventana donde lo colocaremos.

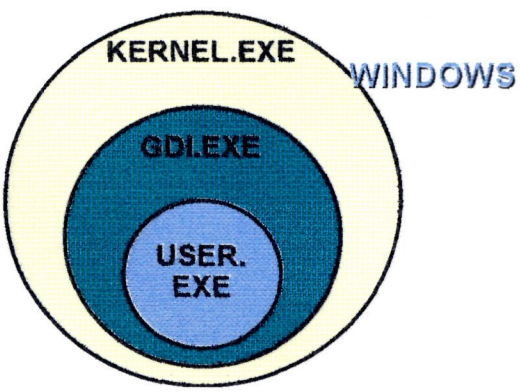
```
void DibujarBitmap(HBITMAP MyBitmap, HWND hwnd, int x, int y)
{
 HDC hdcMemory,DC;
 HGDIOBJ hbmpOld;
 BITMAP bm;

 DC=GetDC(hwnd);                          //Se obtiene el contexto de la ventana
 hdcMemory=CreateCompatible(DC);  //Crea un contexto en memoria
 hbmpOld=SelectObject(hdcMemory,MyBitmap);  //Pone el bitmap en dicho
                                                          //contexto en memoria
 GetObject(MyBitmap, sizeof(BITMAP),&bm);   //Devuelve el ancho, el alto y
                                                         //los colores del bitmap
BitBlt(DC, x, y, bm.bmWidth, bm.bmHeigth, hdcMemory, 0,0, SRCCOPY);
//Copia el bitmap desde memoria hasta la ventana
SelectObject(hdcMemory, hbmpOld);  //Deja el contexto de memoria en su
estado inicial  DeleteDC(hdcMemory);    //Borra el contexto de memoria creado
ReleaseDC(hwnd, DC);        //Libera el contexto de dispositivo en pantalla
}
```

2.11. *Funciones y mensajes de Windows*

El API de Windows tiene una amplia gama de funciones disponibles. Pueden dividirse en categorías dependiendo de la parte de Windows donde residan:

Funciones del USER.EXE, que incluyen funciones para manejar mensajes, manipular ventanas y cuadros de diálogo, controlar el Portapapeles y crear salida del sistema.

Funciones del GDI.EXE, que incluyen funciones para escribir texto, gráficos y mapas de bits en pantallas y dispositivos de salida de forma independiente del dispositivo.
Funciones del KERNEL.EXE, que incluyen funciones para gestionar la memoria, interactuar con el sistema operativo, gestionar los recursos y las comunicaciones.



Ya hemos visto como ObjectWindows utiliza las TVAD para relacionar los métodos de un objeto con los mensajes y que éstos pueden provenir del usuario, de otras aplicaciones, del propio Windows, del programa, etc.

Tanto Microsoft Windows como ObjectWindows dividen los mensajes en rangos disjuntos. Se presentan, a continuación, las constantes que más nos van a interesar:

Constante	*Rango de mensajes*	*Descripción*
CM_FIRST	0x0400-0xFFFF	Mensajes de comando

0x0000-0x7FFF	Mensajes de ventanas
0X0400-0x8FFF	Mensajes definibles por el usuario
0x8000-0x8EFF	Mensajes de los controles

De esta forma se consigue que el usuario pueda crear sus propios mensajes sin que interfieran con los mensajes definidos por Windows. Así, definiríamos un mensaje propio de la forma:

```
#define WM_CONNIND WM_USER+1
```

que se podrían enviar mediante las funciones SendMessage, PostMessage y SendDlgItemMessage.

```
DWORD SendMessage( HWND hwnd, WORD wMsg, WORD wParam,
                   LONG lParam);
```

Esta función envía un mensaje a una ventana y requiere el handle de ella y el mensaje a enviar. Además provoca que la ventana receptora procese inmediatamente el mensaje, es decir , se elude la cola de mensajes y se renuncia a lo que se estaba haciendo.

```
SendMessage ( HWindow, WM_CONNIND, 0, 0L);
```

PostMessage es similar a SendMessage, excepto que no tiene el sentido de urgencia, el mensaje es colocado en la cola de espera de mensajes de la ventana receptora. Ësta procesará el mensaje cuando le sea conveniente.

La función SendDlgItemMessage envía un mensaje a un control particular de una caja de diálogo.

2.12. Creación de ayudas

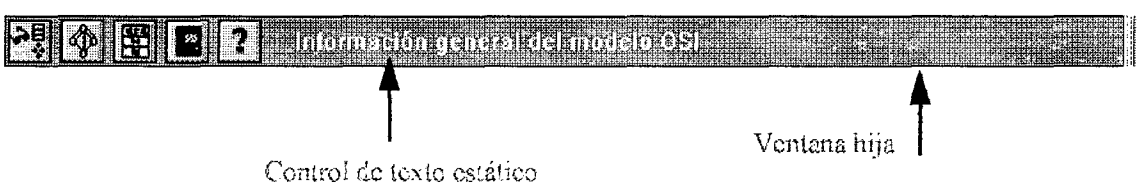
Windows proporciona un entorno consistente en el sentido de que todas las aplicaciones que se ejecutan bajo Windows muestran esencialmente el mismo aspecto. Una vez que se ha aprendido a manejar un programa, se está en condiciones de aprender fácilmente otro, puesto que ya se conoce el entorno.

Hay dos puntos donde el usuario puede obtener información, sin tener que abandonar el ordenador: la línea de estado y el menú de Ayuda.

La línea de estado

La línea de estado se sitúa generalmente en la parte inferior de la ventana principal. El «Simulador del protocolo... presenta una línea de estado para indicar el significado de las posibles selecciones que el usuario puede realizar, pero a diferencia de lo habitual, se sitúa junto a la barra rápida para economizar el espacio de la ventana.

La línea de estado consiste en una pequeña ventana hija con un control estático en su interior.



La ayuda

La estructura física de los libros obliga a que el acceso a la información escrita se realiza de forma secuencial. Mediante el ordenador podemos diseñar un sistema de modo que se acceda a la información de forma no secuencial. A este sistema se le denomina hipertexto.

En un sistema de ayuda resulta esencial el concepto de área activa; es decir, aquella palabra o porción de un gráfico que al ser pulsada nos conduce a otra página de información (jump), o bien aparece una pequeña ventana temporal (pop up) que contiene una breve información relativa al área activa.

Los siguientes elementos componen un sistema de ayuda en Windows:

- Los ficheros de texto con extensión .RTF.
- Los ficheros de imágenes .BMP cuya referencia está en el texto (opcional).
- El fichero proyecto .HPJ compuesto como mínimo de los ficheros de texto y de imágenes
- La compilación del archivo .HPJ mediante el compilador de ayuda (HC.EXE) situado en el subdirectorio BORLAND\BIN.Este proceso da como resultado un fichero binario con extensión HLP.
- La llamada a la función WinHelp en el código fuente de la aplicación con los parámetros adecuados según el propósito:

a) Si queremos que aparezca el índice de la ayuda:

WinHelp(HWindow,"AYUDA.HLP",HELP_INDEX,0L);

b) Si queremos que aparezca un tema concreto:

WinHelp(HWindow,"AYUDA.HLP",HELP_KEY,
(unsigned long) (LPSTR) "Tutorial OSI");

2.12.1.Creación de ficheros .RTF

Podemos crear un archivo de este tipo con cualquier procesador de textos que tengan la capacidad para hacerlo. La Ayuda de "Simulador del protocolo de transporte OSI" se realizó con el Microsoft Word.

Los campos clave junto con su propósito dentro de un fichero .RTF se muestran a continuación:

Código de control	*Propósito*
#	Identifica a un tema. Debe ser único
$ (Nota de pie de página)	Identifica el título de un tema (opcional). El texto al que acompaña es el del Historial de la Ayuda.

Código de control	*Propósito*
k (Nota de pie de página)	El texto al que acompaña aparece en el cuadro Buscar de la Ayuda
Texto subrayado	Creación de ventanas temporales
Texto oculto	Especifica la cadena de caracteres que identifican a los temas.

Con estas premisas, para crear un salto a un tema específico dentro de la ayuda habrá que seguir los siguientes pasos:

* Seleccionar la palabra activa y darle un estilo Tachado (si se desea un salto) o Subrayado (si lo que se desea es una ventana temporal).

* Situar el identificador a continuación de la palabra activa sin espacios en medio, con estilo Oculto.

* Ir a la página donde deseamos poner la información relativa a la palabra activa e insertar una nota al pie con la marca # seguida del identificador.

* Escribir el texto que se desee.

Se pueden incluir bitmaps en la ayuda haciendo uso de la declaración *bml* (situándolo a la izquierda), *bmc* (en el centro) o *bmr* (a la derecha) seguido del nombre del bitmap y encerrándolo entre llaves:

{bml salir.bmp}

Se pueden crear bitmaps con zonas activas de modo que si pulsamos en ellas aparezca una ventana temporal o se produzca un salto a otro tema. Estos bitmaps se crean con el Hot Spot Editor de Borland C++ y tienen extensión .shg.

2.12.2.El fichero proyecto

El fichero puede elaborarse con cualquier editor ASCII y tiene extensión .HPJ. Consta como mínimo de la sección:

[FILES] Especifican los ficheros con extensión .RTF que forman la ayuda

pero también se utiliza con asiduidad la sección:

[BITMAPS] Si se utilizan bitmaps en la ayuda

2.12.3.El fichero proyecto de la aplicación "Simulador del Protocolo de Transporte OSI"

El fichero proyecto con el que se construyó la ayuda del "Simulador del Protocolo de Transporte OSI" se incluye a continuación, explicando cada declaración, junto con una pantalla en ejecución de la ayuda.

```
[OPTIONS]                    ; Contiene opciones que controlan el HC
BMROOT=c:\tutor\ayuda        ; Especifica el directorio que contiene los bitmaps
TITLE=Ayuda de Simulador...  ; Especifica el título del archivo de ayuda

[MAP]                        ; Asocia cadenas de contexto con identificadores.
CAMPOTPDU 101                ; Necesario para crear ventanas pop-up de ayuda
CAMPOID 102                  ; al hacer clic en determinadas ventanas del
CAMPOCHECKSUM 103            ; programa
CAMPOORIGEN 104
CAMPODESTINO 105
VENTANA1 106
CAMPOSECUENCIA 107

[FILES]                      ; Contiene los ficheros de tema
```

adan.rtf

eva.rtf

osi.rtf

[BITMAPS] ; Bitmaps contenidos en la ayuda

salir.bmp

help.bmp

tpdu.bmp

esttpdu.bmp

ordenar.bmp

nuevo.bmp

automata.bmp

menu1.bmp

menu2.bmp

menu3.bmp

formgen.bmp

info.bmp

parametr.bmp

caprmosi.bmp

cregistr.bmp

visual.shg

inter.shg

rmosi.shg

mod.shg

cqos.shg

cmodo.shg

ccaida.shg

corgdest.shg

cspeed.shg

cclase.shg

cinfosup.shg

wmodo.shg

[CONFIG] ; Determina la configuración del archivo de ayuda

BrowseButtons() ; Añade botones de secuencia de hojeo

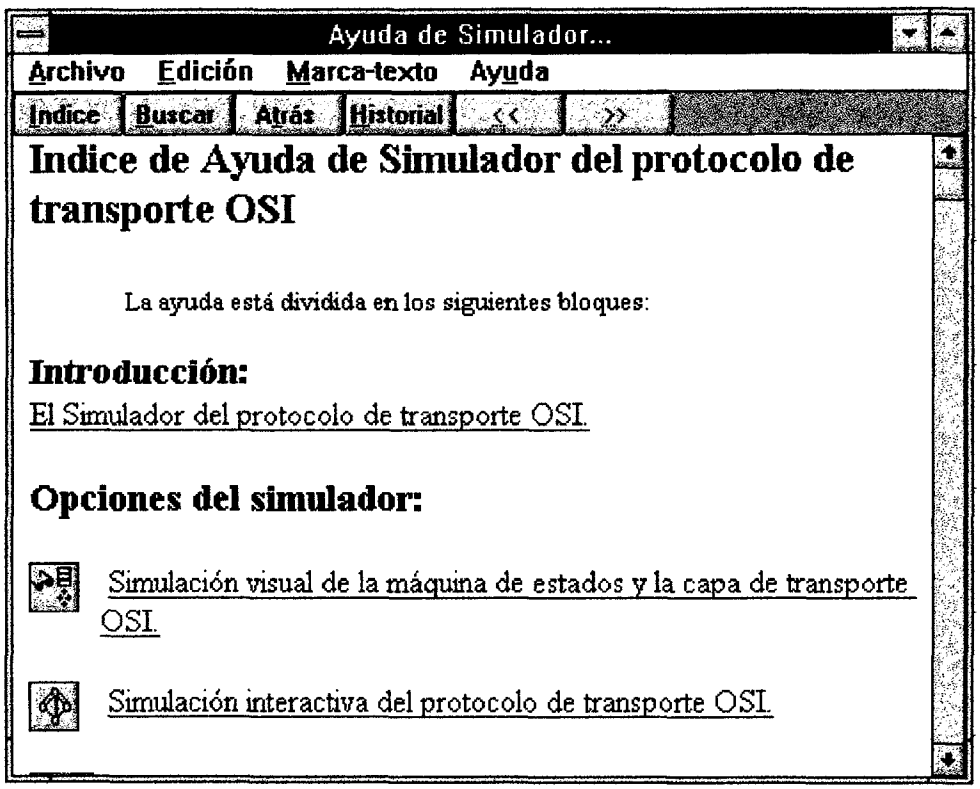


Ilustración 2–5. Ayuda de la aplicación

3. El modelo de referencia OSI

3.1. Necesidad y creación

Durante los últimos años, el funcionamiento de las redes de ordenadores ha cambiado enormemente. Hasta hace poco eran consideradas como herramientas extrañas para la investigación y sólo las utilizaban algunos especialistas. Cada fabricante de ordenadores tenía su propia arquitectura de red y en ningún caso existía la compatibilidad con la de otros fabricantes.

Sin embargo, en la actualidad, para millones de personas en el mundo entero, el correo electrónico es una realidad y las redes de ordenadores han evolucionado hasta ser consideradas como una herramienta esencial para sus usuarios en negocios, gobierno y universidades.

Como consecuencia de este desarrollo la **Organización Internacional de Normalización (ISO)** acordó una serie de Normas Internacionales para describir las arquitecturas de redes. Estas normas se conocen como el **Modelo de Referencia OSI (interconexión de sistemas abiertos).** Con ellas se pretende que los fabricantes evolucionen hacia una única arquitectura de red y que sus ordenadores puedan conectarse con los de otros fabricantes sin ningún esfuerzo y problema de compatibilidad.

La idea del modelo OSI consiste en diseñar redes como una secuencia de capas, cada una construida sobre la anterior y con una función específica. El modelo OSI tiene siete capas.

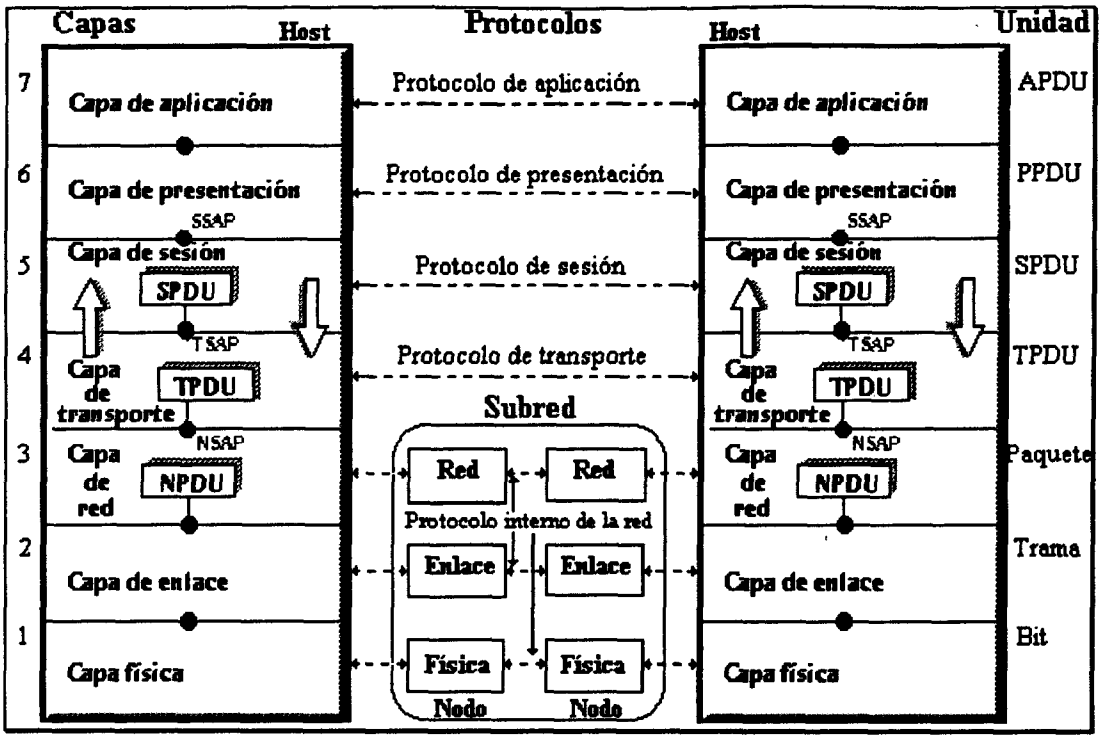


Ilustración 3–1. Modelo de referencia OSI.

3.2. Referencia básica de las capas de la arquitectura OSI

Excluyendo el nivel de transporte, se listan a continuación el resto de capas que componen el modelo de arquitectura de red OSI y su cometido principal.[1]

3.2.1. Nivel físico

Su función es la de proporcionar un camino físico para unir los sistemas que se quieren conectar. La capa física debe asegurar que cuando un extremo envía un bit de valor "1", se reciba en el otro extremo un bit de ese valor, y no un "0".

[1] Una ayuda más extensa se puede encontrar en la aplicación.

3.2.2. Nivel de enlace

La función de la capa de enlace consiste en proporcionar servicios a la capa de red, como transformar el medio de comunicación en una línea sin errores, evitar que un transmisor rápido sature a un receptor lento ...

3.2.3. Nivel de red

La función fundamental de la capa de red es la de controlar la subred de comunicaciones. Para ello debe solucionar problemas como el de una posible congestión de la red, cómo debe realizarse el encaminamiento de paquetes, cómo interconectar varias redes, etc.

3.2.4. Nivel de sesión

La capa de sesión, junto con la de presentación y aplicación constituyen las capas superiores en el RM-OSI. Las cuatro capas inferiores están diseñadas para proporcionar una comunicación fiable extremo a extremo, mientras que el objetivo de las capas superiores consiste en proporcionar servicios orientados al usuario. Una de las tareas de esta capa es la de evitar que tanto el transmisor como el receptor traten de realizar la misma operación al mismo tiempo. Para ello utiliza testigos.

3.2.5. Nivel de presentación

Se encarga de la sintaxis de los datos intercambiados. Su propósito es resolver diferencias en formatos y representación de los datos. También está relacionada con otros aspectos como la compresión de los datos y la criptografía.

3.2.6. Nivel de aplicación

La capa de aplicación contiene una variedad de protocolos de propósito general y específicos que el usuario necesita frecuentemente como los relacionados con la transferencia de ficheros, con el correo electrónico, etc .

4. Conceptos generales de la capa de transporte (ISO 8072/73 [SERVICIO/PROTOCOLO])

4.1. Introducción

La capa de transporte no sólo representa otra capa más dentro del modelo OSI, sino que desarrolla un papel muy importante y viene a ser realmente el corazón de la jerarquía de protocolos. Su tarea consiste en hacer que el transporte de datos se realice de forma segura y económica, desde la máquina fuente hasta la máquina destinataria, independientemente de la red o redes físicas que se encuentran en uso.

La capa de transporte es una capa del tipo origen-destino o extremo a extremo. Es decir, un programa en la máquina origen lleva una conversación con un programa parecido que se encuentra en la máquina destino, utilizando las cabeceras de los mensajes y campos de control. Los protocolos de las capas inferiores son entre cada máquina y su vecino inmediato, y no entre la máquina origen y destino, las cuales podrían estar separadas por muchos nodos.

La función principal de la capa de transporte consiste en aceptar los datos de la capa de sesión , dividirlos siempre que sea necesario en unidades más pequeñas, pasarlos a la capa de red y asegurarse que todas lleguen correctamente al otro extremo.

Si las redes fuesen perfectas no se necesitaría la capa de transporte o su trabajo sería mínimo.

4.2. Servicio de transporte

Se definen dos tipos de servicio de transporte según el servicio de red:

* Orientado a la conexión (Circuito virtual).
* No orientado a la conexión (Datagrama).

En el primer caso, las conexiones tienen tres fases: establecimiento, transferencia de datos y liberación.

4.2.1. Calidad de servicio

Una manera diferente de observar a la capa de transporte es la de considerar que su función es la de enriquecer la calidad de servicio (o QOS) suministrada a las capas inferiores. Si el servicio de red es impecable, la capa de transporte puede tener un trabajo relativamente sencillo. Sin embargo, si el servicio de red llega a ser deficiente, la capa de transporte tiene que llenar el hueco que existe entre aquello que los usuarios desean y lo que las capas inferiores ofrecen.

La QOS puede caracterizarse por medio de varios parámetros. El servicio de transporte OSI le permite al usuario especificar valores preferidos para estos parámetros, en el momento en que se lleva a cabo el establecimiento de la conexión.

Algunos de estos parámetros son:

- Retardo en el establecimiento de la conexión.
- Retardo de tránsito.
- Retardo en la liberación de la conexión.
- Prioridad.
- Protección

- Tasa de error residual.[2]

En algunos casos, después de observar los parámetros QOS, la capa de transporte puede llegar a comprender de inmediato que algunos de ellos son inalcanzables, en cuyo caso le dirá al usuario que llama que el intento de conexión falló.
En otros casos, la capa de transporte puede no alcanzar el objetivo deseado pero puede cambiar la QOS a una mínima aceptable. Si la máquina remota es incapaz de tratar el valor propuesto, pero puede manejar un valor por encima del mínimo, podría reducir el valor propuesto. Si no es capaz de manejar ningún valor por encima del mínimo rechazará el intento de conexión.
A este proceso se le conoce como negociación de opciones. Una vez que se produce la negociación, éstas se conservan a lo largo de toda la vida de la conexión.

4.2.2. Primitivas del servicio de transporte OSI

Las primitivas son representaciones abstractas de las interacciones entre las capas, que indican al usuario de transporte que debe efectuar una acción o notifican la acción tomada por otro usuario del servicio.

Las primitivas del servicio de transporte OSI para el establecimiento y la liberación de la conexión son:

- **Establecimiento**

  T_CONNECT.request (destino, origen, tipo_dato, qos, datos)

  T_CONNECT.indication (destino, origen, tipo_dato, qos, datos)

  T_CONNECT.response (qos, respuesta, tipo_dato, datos)

  T_CONNECT.confirm (qos, respuesta, tipo_dato, datos)

- **Liberación**

  T_DISCONNECT.request (datos)

  T_DISCONNECT.indication (causa,datos)

[2] El significado de estos parámetros se explican en la ayuda del programa

La secuencia de primitivas para el establecimiento de una conexión en un circuito virtual se muestra a través del siguiente dibujo. Los números indican el desarrollo en el tiempo.

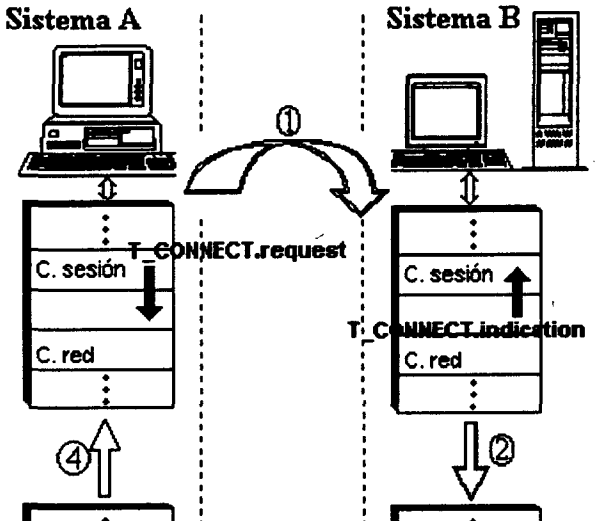


Ilustración 4–1. Establecimiento de una conexión de transporte.

Las primitivas para la transferencia de datos son:

Para un servicio de transporte tipo **circuito virtual:**

T_DATA.request (datos)

T_DATA.indication (datos)

T_EXPEDITED_DATA.request (datos)

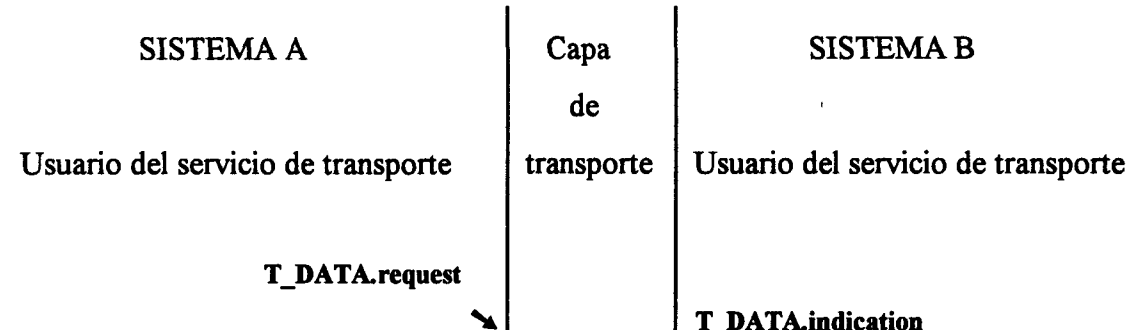
T_EXPEDITED_DATA.indication (datos)

Para un servicio datagrama:

T_UNITDATA.request (destino, origen qos, datos)

T_UNITDATA.indication (destino, origen qos, datos)

La secuencia de primitivas en una transferencia de datos para un circuito virtual es la siguiente:



4.3. Protocolos de transporte

Para controlar los diferentes tipos de transferencia de datos que pueden ser manejados por una conexión de transporte y la amplia variedad de redes existentes, la ISO ha definido cinco clases de protocolos de transporte. Existen las clases 0, 1, 2, 3, 4. A su vez divide las redes en tipo A, B, C.

Tipo A: La red es prácticamente perfecta. No hay paquetes perdidos, duplicados o dañados. No es necesario que la capa de transporte tenga servicios de recuperación de fallos, ni servicios de resecuenciamiento, etc. Las redes tipo LAN se acercan bastante a este tipo.

Tipo B: En ellas existe una entrega perfecta de paquetes, pero de cuando en cuando la capa de red emite un N-RESET debido a una congestión, a problemas de hardware. El protocolo de transporte debe resincronizarse y continuar, de tal manera que el N-RESET quede oculto para los usuarios. La mayoría de las redes públicas X-25 son tipo B.

Tipo C: La red es insegura, con pérdida y duplicación de paquetes, caídas, etc. Ejemplos de este tipo son las redes tipo WAN y las redes de radiopaquetes. Los protocolos de transporte utilizados con redes tipo C son los más complejos de todos.

Las cinco variantes OSI para el protocolo de transporte presentan las siguientes características:

- **Clase 0:**
  - ✓ Existe una única conexión de red para cada conexión de transporte solicitada.
  - ✓ La conexión de red no comete errores.
  - ✓ No hay secuenciamiento ni control del flujo.
  - ✓ Proporciona mecanismos para establecimiento y liberación de las conexiones de transporte.

- **Clase 1:**

✓ Similar a la 0, pero diseñada para recuperarse de N-RESET. Si éste se produce las entidades de transporte se resincronizan y continúan a partir del punto en que habían quedado. Para lograrlo necesitan guardar números de secuencia.

- **Clase 2:**

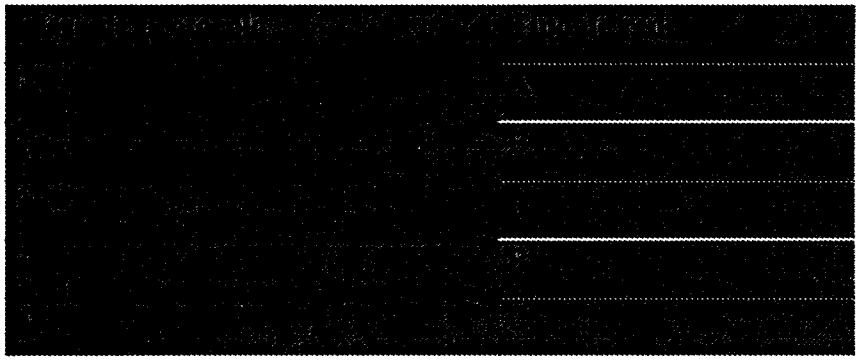
✓ Es igual que la clase 0 (para redes tipo A) pero permite la multiplexación de varios conexiones de transporte sobre una misma conexión de red.

- **Clase 3:**
  - ✓ Combina las características de las clases 1 y 2.
  - ✓ Permite la multiplexación.
  - ✓ Se recupera de los N-RESET.

- **Clase 4:**
  - ✓ Está diseñada para redes tipo C.
  - ✓ Detecta y corrige fallos de red, errores, duplicidades, caídas, etc.

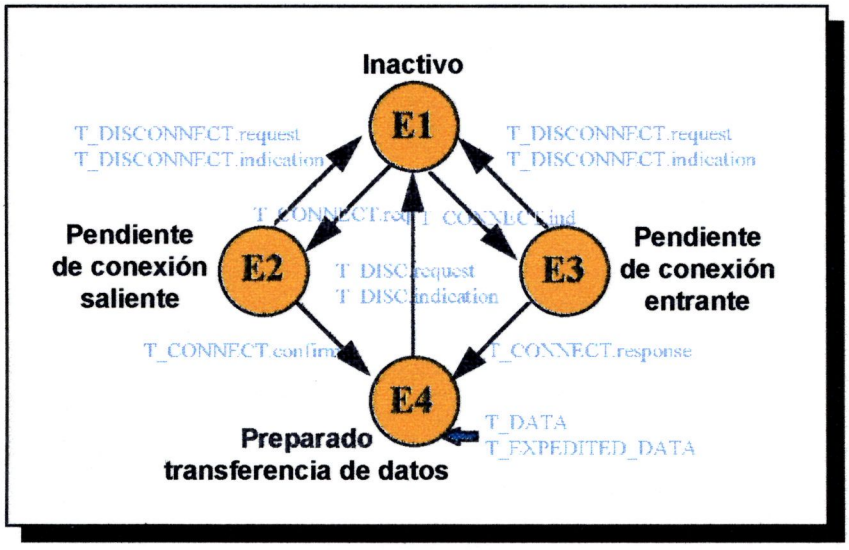
La elección de la clase de protocolo se realiza mediante negociación entre las entidades de transporte en el momento de establecer una conexión. Si el destino no puede con la clase ofertada por el origen, rechazará la conexión.

La siguiente tabla muestra el protocolo de transporte ideal según el tipo de red sobre el que se esté trabajando:



4.3.1. Máquina de estados del protocolo de transporte

La máquina de estados del protocolo de transporte para un servicio tipo circuito virtual es la siguiente:



El paso de un estado a otro se establece según la primitiva de transporte que se reciba.

4.3.2. Definición, formato y tipos de TPDU's

Los términos "tramas" y "paquetes" nos son muy familiares y se utilizan ampliamente. Con tramas y paquetes nos referimos a las unidades de datos intercambiadas entre entidades pares de la capa de enlace y red, respectivamente.

La TPDU (Unidad de datos del Protocolo de transporte) es el término equivalente para la capa de transporte.

Los datos transportados en cada primitiva T_DATA.request, los que se pasan del nivel de sesión al nivel de transporte, se denominan TSDU's (Unidades de datos del Servicio de transporte) y se entregan al nivel de sesión receptor por la primitiva T_DATA.indication. La TSDU no está limitada en longitud. Una de las tareas más importantes del protocolo es dividir la TSDU al tamaño de la TPDU que utiliza el protocolo y después rearmar las piezas, de una manera transparente, en el otro extremo.

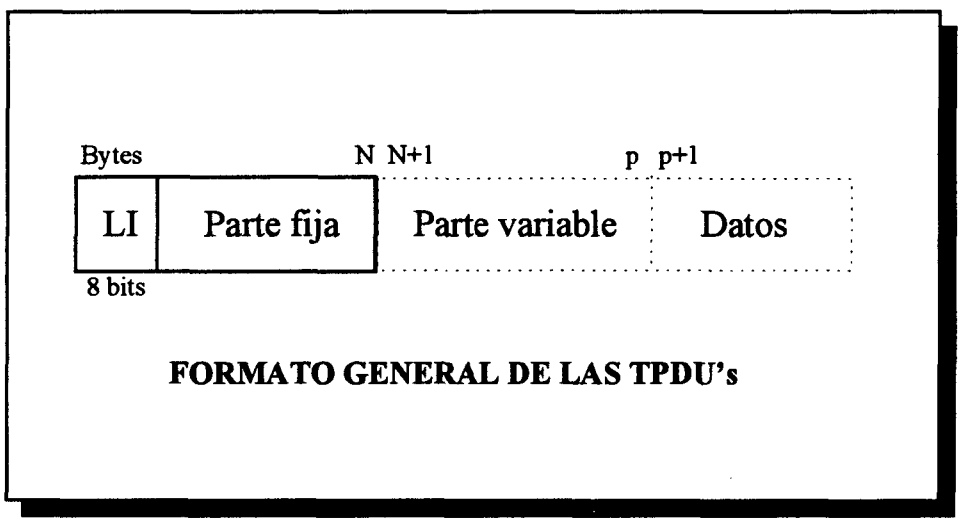
El protocolo de transporte del modelo OSI tiene diez diferentes tipos de TPDU's; cada una de ellas consta de hasta cuatro partes:

⇒ **Indicador de longitud (LI):** Da la longitud de las cabeceras fijas y de longitud variable.

⇒ **Parte fija de la cabecera:** Campos de control.

⇒ **Parte variable de la cabecera:** Campos de control (Si aparece).

⇒ **Campo de datos.** (Si aparece).



FORMATO GENERAL DE LAS TPDU's

El campo de datos contiene los cabeceros que se han añadido en las capas superiores y los datos de usuario. Los campos de control de la capa de transporte se usan para:

- Negociar el nivel de servicio entre el transmisor y el receptor durante la fase de establecimiento de la conexión.

- Manejar la transferencia de datos.

- Permitir la terminación de la conexión.

5. Aplicación: "Simulación del Protocolo de Transporte OSI"

5.1. Introducción

La aplicación **"Simulación del Protocolo de Transporte OSI"** se divide en tres partes claramente diferenciadas: *"Tutorial RM-OSI"*, *"Simulación interactiva del Protocolo de Transporte OSI"* y *"Simulación visual del nivel y máquina de protocolo de transporte"*.

A estos bloques se tiene acceso a partir de la ventana inicial: por medio de las opciones de menú o a través de la barra de botones, más rápida e intuitiva.

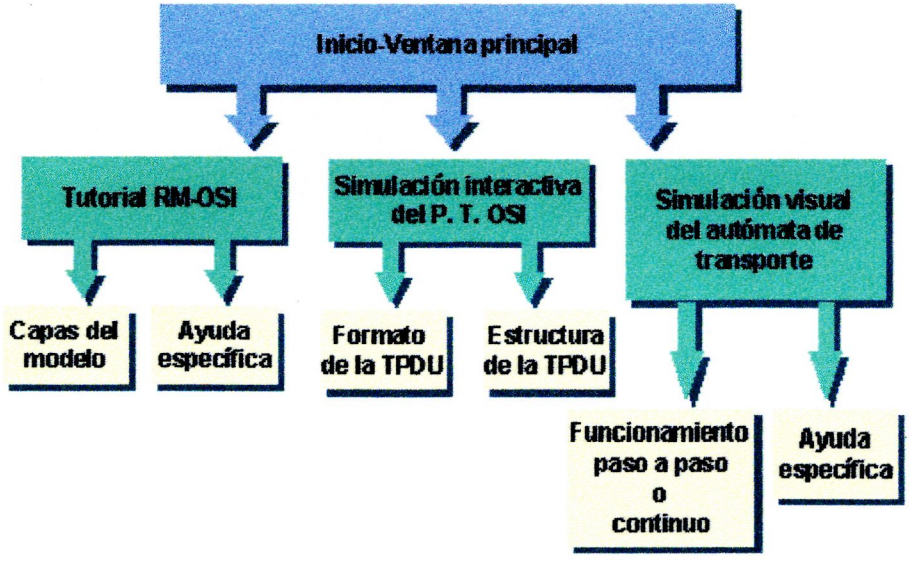


Ilustración 5–1. Flujo básico de la aplicación

Con la primera parte se pretende conocer el modelo OSI en general y sus funciones capa por capa. En la simulación visual y en la interactiva se permite al usuario comportarse como el nivel que se encuentra por encima del de transporte (nivel de sesión) o como subred de telecomunicación y comprobar como reacciona la capa de transporte en función de sus acciones.

Además, en la simulación interactiva, se permite que el usuario especifique valores para los parámetros que caracterizan la calidad del servicio que la capa de transporte va a utilizar, elija el tipo de protocolo de transporte que se va a emplear entre origen-destino, la información que quiere transmitir, etc. Esos datos introducidos provocarán unas actuaciones u otras por parte del nivel de transporte que podrán ser seguidas por el usuario.

Como toda aplicación Windows dispone de una ayuda basada en el hipertexto donde se puede obtener información sobre el manejo del programa y sobre los conceptos que éste explica.

Esta ayuda, a la que se puede acceder desde cualquier punto del programa, también está estructurada en bloques para hacer más fácil su interpretación:

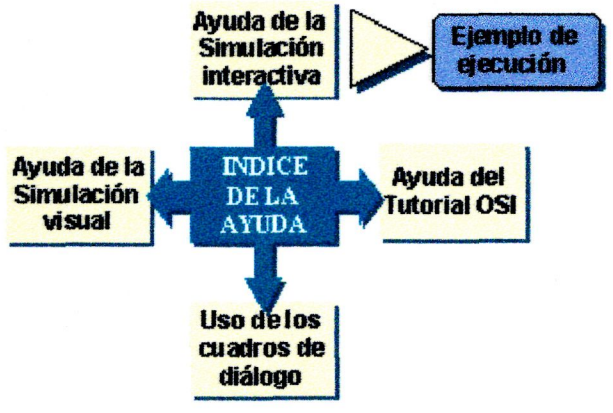


Ilustración 5–2. Disposición de la ayuda

5.2. *Configuración de colores y resolución*

El programa se realizó en una máquina 486-66Mhz con tarjeta gráfica SuperVGA que proporciona hasta 16.8 millones de colores y con Windows configurada a 640x480 pixels y 64k colores.

Existen dos versiones de la aplicación: una para usuarios que dispongan de una tarjeta gráfica similar y otra para aquellos con tarjetas VGA. Ambas versiones están preparadas para ejecutarse en cualquier sistema con independencia del dispositivo gráfico sin ningún tipo de problemas. Sin embargo, "El Simulador del Protocolo de Transporte OSI" utiliza una gran cantidad de bitmaps. Éstos pueden crearse con 16, 256, 64k colores,... Por eso los bitmaps realizado con 64k colores perderán brillo y vistosidad si se ejecutan con Windows configurado con una cantidad de colores menor. Esta cuestión de estética obligó a realizar dos versiones del programa: la inicial y otra para que los usuarios que no dispongan de una buena tarjeta gráfica obtengan, a pesar de todo, un programa agradable a la vista.

En cuanto a la resolución con la que debe configurar Windows se recomienda 640x480 pixeles pues el tamaño y la colocación de los bitmaps se realizó pensando en esta configuración. Cualquier otra permitida por su monitor también será soportada.

5.3. *Tutorial RM-OSI*

Es uno de los tres bloques que componen el programa y su función es mostrar el modelo de referencia OSI completo y estudiar las funciones más importantes de cada una de las capas que lo componen. Se accede al tutorial al elegir en el menú principal la opción Tutorial RM-OSI o al pulsar el botón Tutorial RM-OSI en la barra de botones.

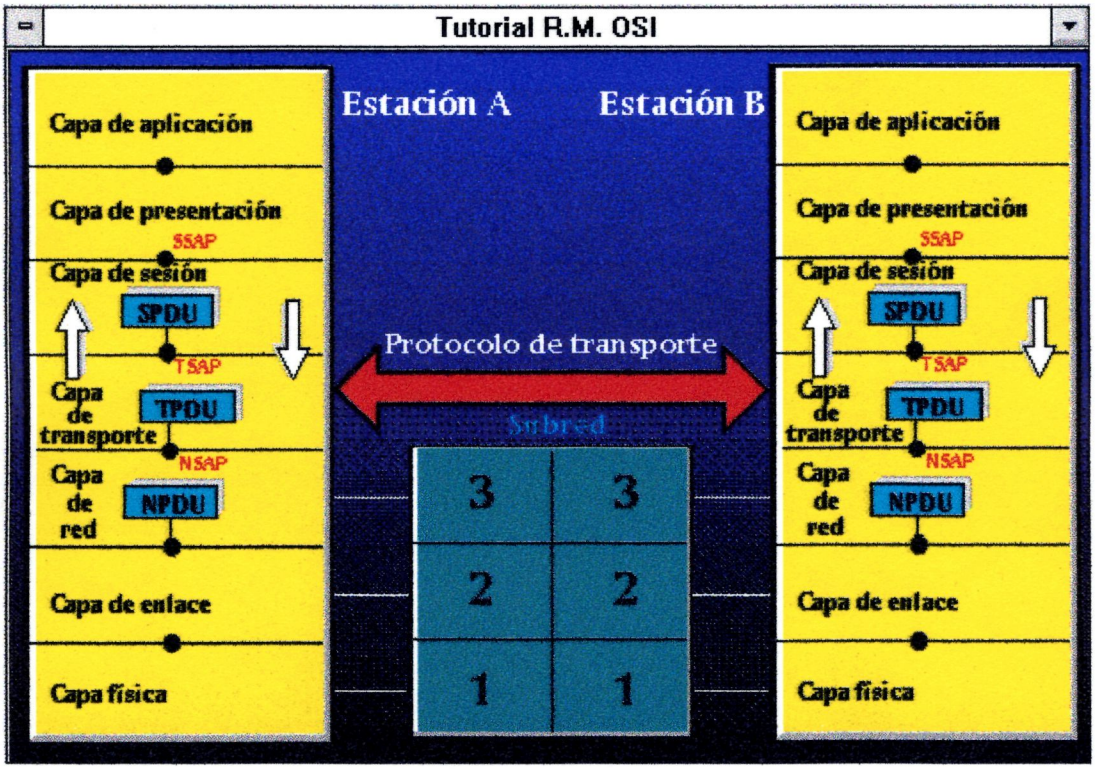


Ilustración 5–3. Ventana del Modelo de Referencia OSI

El cursor toma forma de prismáticos para incitar al usuario a "mirar" dentro de las capas del bitmap que representa el modelo OSI. "Mirar" equivaldrá a pulsar en el botón izquierdo del ratón dentro de los niveles de la arquitectura, lo que provoca la creación de otra ventana cuyo contenido especifica, de una manera gráfica, las tareas que realiza esa capa en concreto.

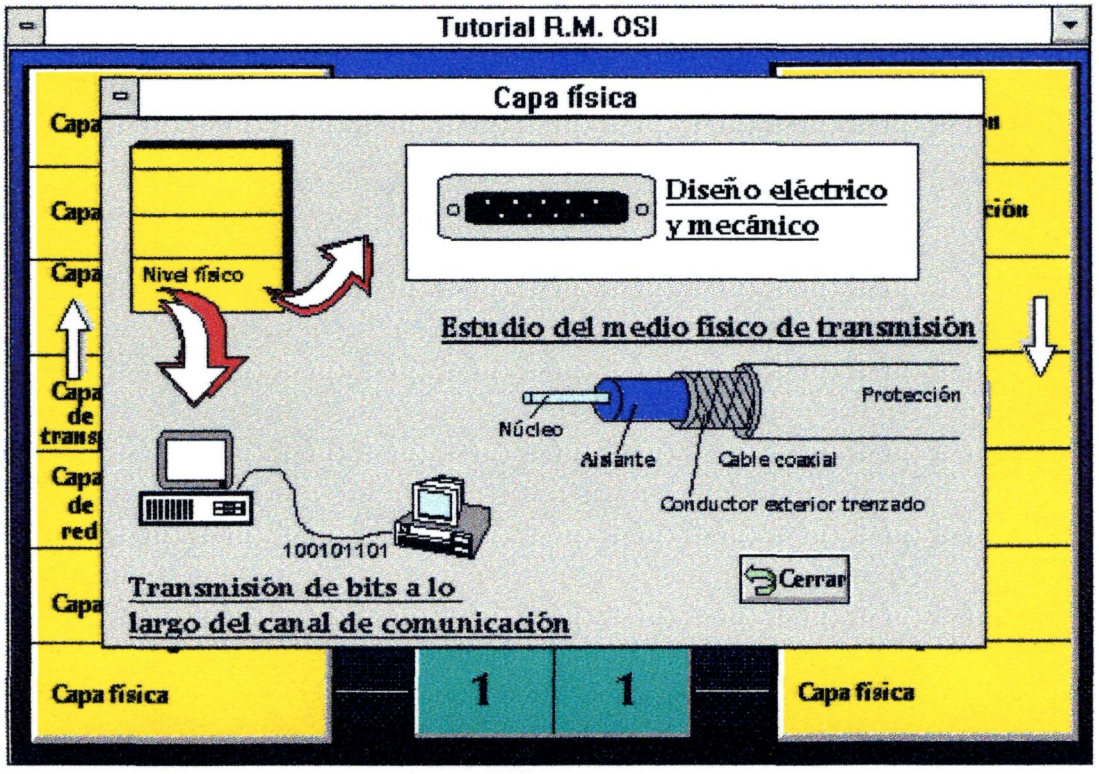


Ilustración 5–4. Ventana "Capa física" del RM-OSI

Al pulsar el ratón en este tipo de ventanas se ejecuta la ayuda referida a este nivel del modelo donde se explican las funciones de la capa de una manera más extensa.

El organigrama siguiente muestra los procesos que se realizan en esta parte de la aplicación:

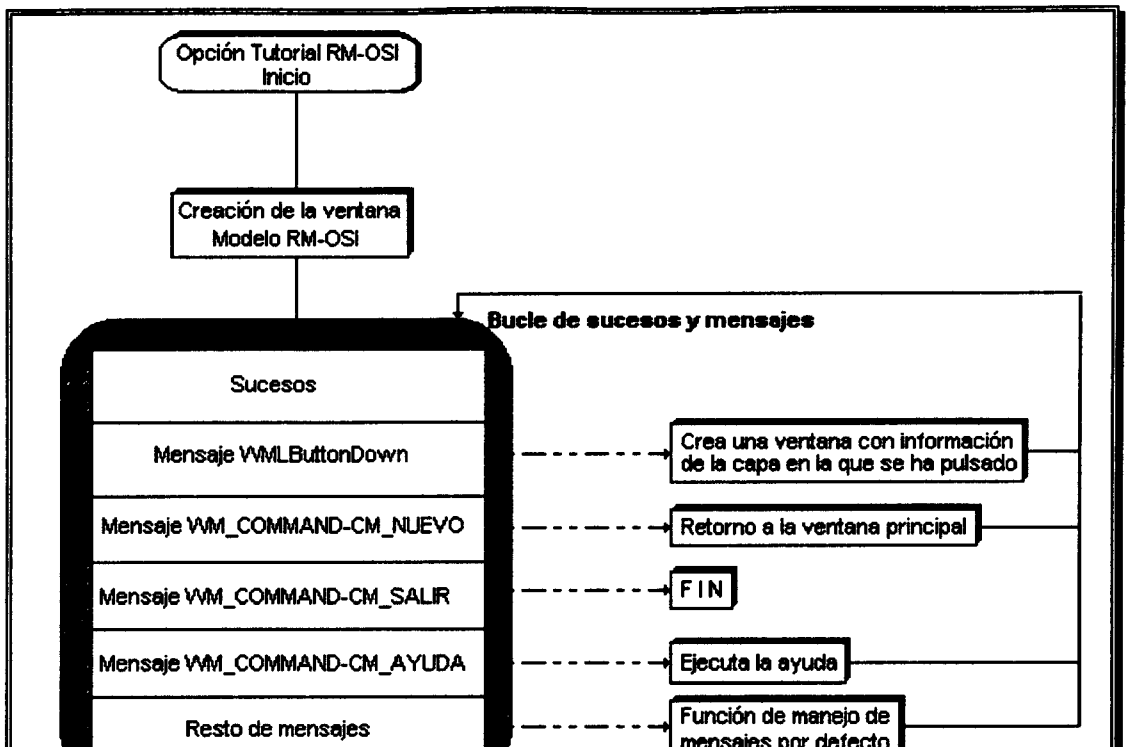


Ilustración 5–5. Flujo de mensajes en este apartado

5.4. Simulación visual del nivel y la máquina de transporte

Se entra en esta parte al seleccionar Simulación | Simulación visual en el menú raíz o bien con el botón Simulación visual de la barra rápida.

El **objetivo** de este apartado del programa es **mostrar de una forma completa la interacción de la capa de transporte con la de sesión y la de red a nivel de primitivas**, así como de explicar la implicación que tienen éstas en el cambio de estados de la máquina por la que se rige el protocolo de transporte OSI.

De una forma visual, rápida e intuitiva, **el usuario** aprenderá el manejo de primitivas OSI puesto que es él el que **actúa como nivel de sesión o incluso como red de comunicaciones** (mediante el empleo de la directiva T_DISCONNECT.indication).

A su vez el alumno dispone de una opción de **cambio de velocidad** de la simulación y también de un **modo paso a paso** para clarificar conceptos, según su grado de práctica o conocimiento.

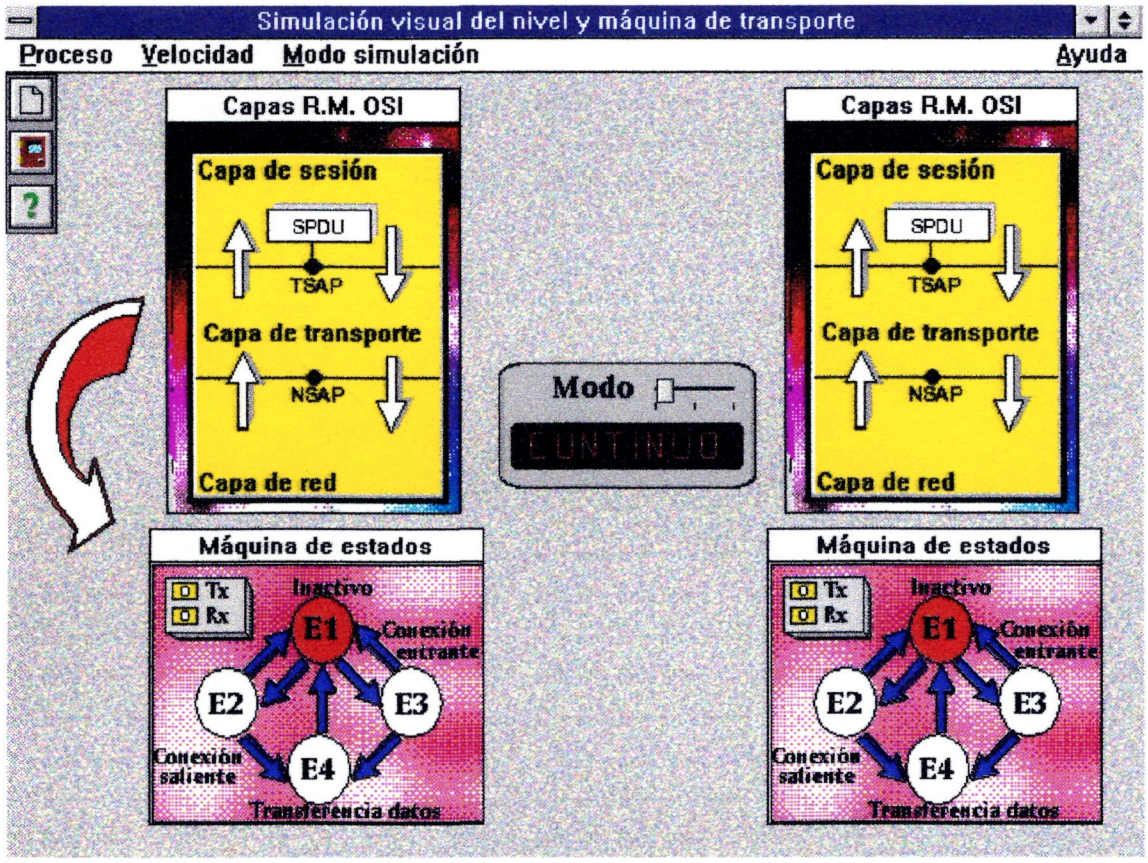


Ilustración 5–6. Pantalla en ejecución de la Simulación visual

Se crean cinco ventanas inicialmente para la "Simulación visual del nivel y la máquina de transporte". Cuatro de ellas ("Capas del RM-OSI" y "Máquina de estados") representan al sistema transmisor y al sistema receptor. Tienen como misión plasmar las relaciones entre la máquina origen y la destino. En cualquiera de las ventanas "Capas del RM-OSI" comienza la simulación cuando al pulsar el botón derecho sobre ellas aparece un menú flotante con el que fijamos un origen y un destino.

La otra ventana se utiliza como indicador del modo actual de funcionamiento: paso a paso o continuo; y para conmutar entre ellos.

5.5. *Simulación interactiva del Protocolo de Transporte OSI*

Se accede a esta sección del programa al elegir Simulación | Simulación interactiva en el menú principal o al pulsar en el botón Simulación interactiva de la barra de botones.

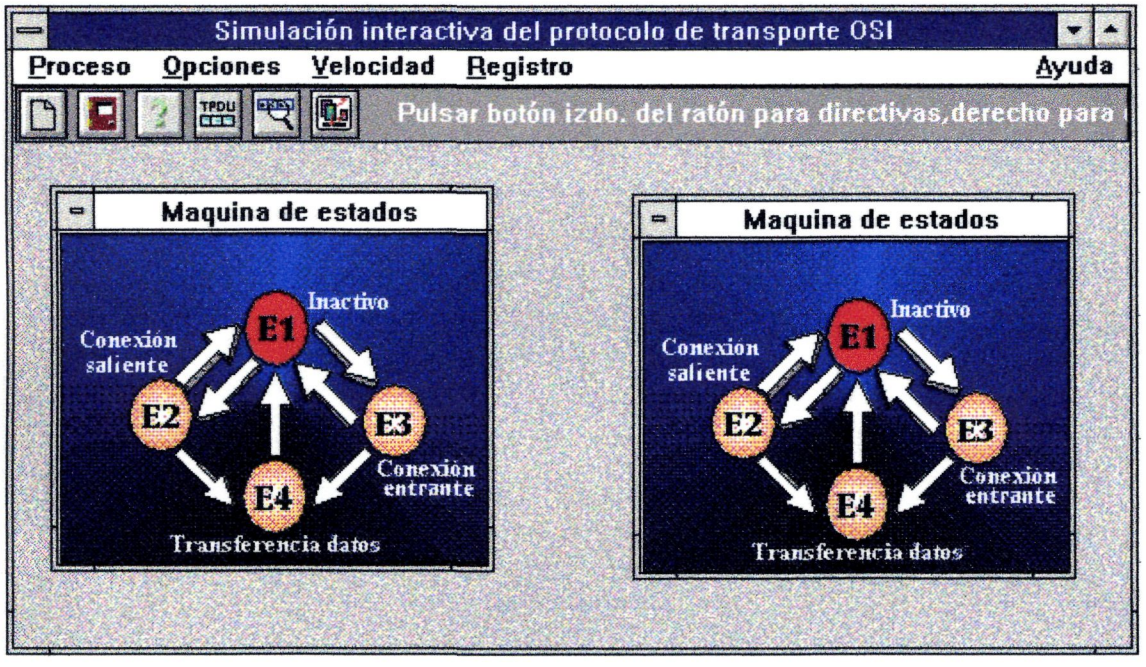


Ilustración 5–7. Pantalla en ejecución de la Simulación interactiva

Como su nombre indica, el **objetivo** de este bloque del programa es la **interactuación entre el usuario y la capa de transporte.**

El usuario dispondrá de un menú flotante a través del cual podrá fijar los orígenes y destino que considere, la información que desea transmitir, modificar los parámetros de calidad del servicio que ofrece la capa de transporte, simular una caída de la red, elegir una clase de protocolo de transporte, etc.

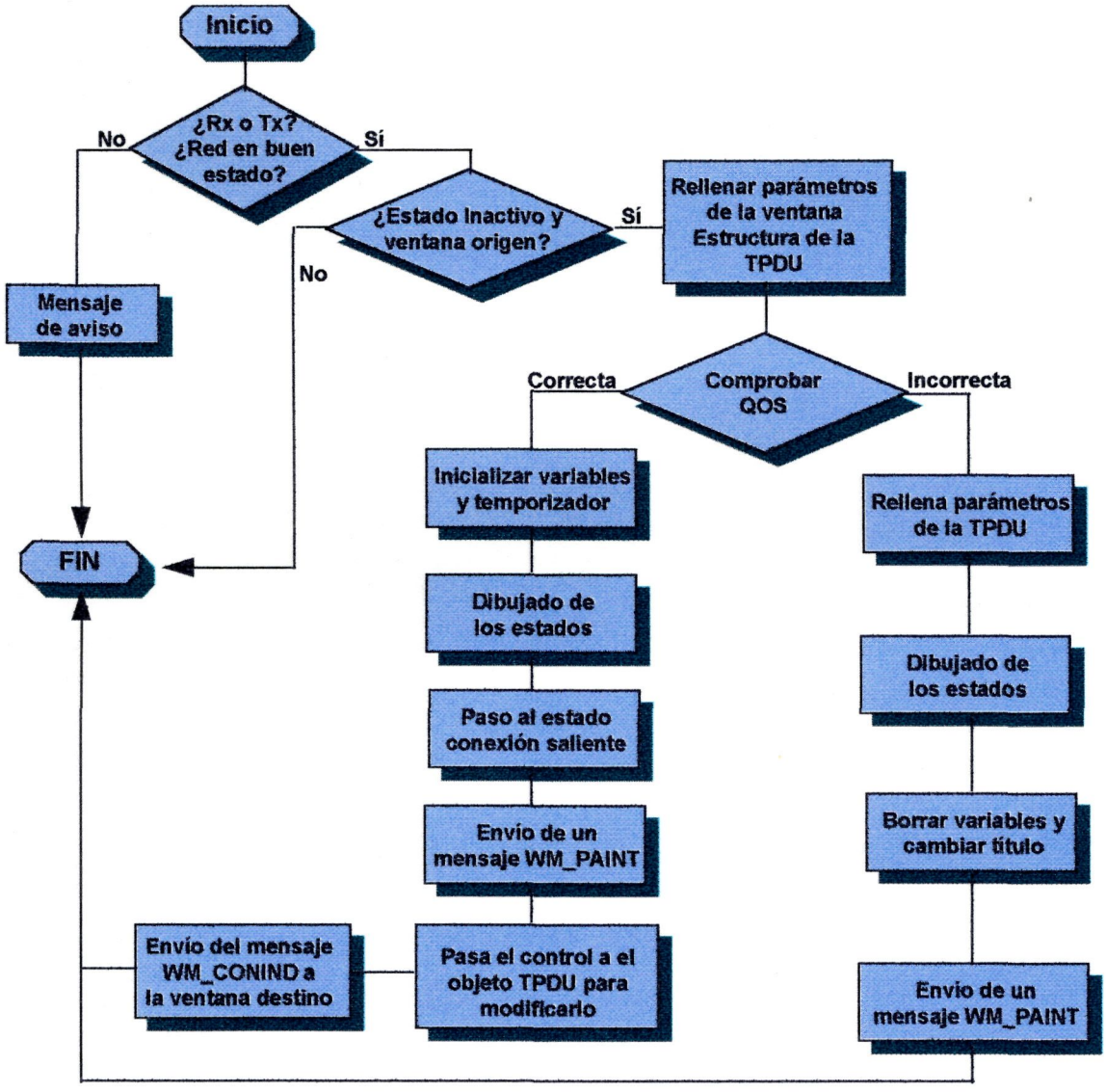
En función de sus selecciones visualizará las reacciones de la capa de transporte. Se podrá observar desde rechazos de conexiones, si no son compatibles las clases de protocolo o la calidad de servicio entre las máquinas a conectarse, a establecimientos y posterior transferencia de datos si se acierta con las combinaciones correctas de parámetros.

Además se podrá modificar la TPDU en dos lugares diferentes: cuando se encuentra en el buffer de salida del sistema transmisor o cuando se encuentra en la red. Así podrá estudiar la respuesta de la capa de transporte destino ante cambios en la TPDU.

También se puede realizar en todo momento un seguimiento de los valores que toma la TPDU y obtener ayuda sobre los parámetros de ésta.

El inicio de la simulación interactiva se produce al seleccionar en el menú de primitivas **T_CONNECT.request.** El organigrama de la respuesta a ese mensaje de comienzo, similar al esquema seguido por el resto de mensajes respuesta a las primitivas, es el siguiente:

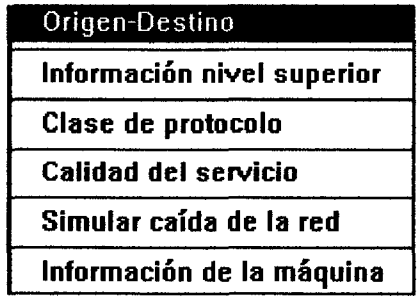
Diagrama de flujo de la función: *virtual void Directival (RTMessage Msg) = [CM_FIRST + IDM_CREQUEST]*



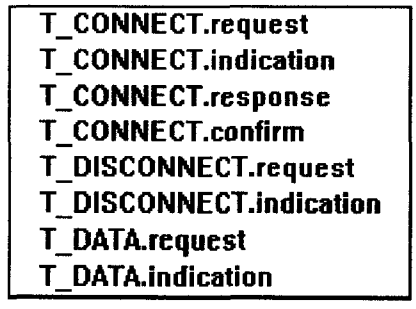
5.6. Tutorial: Ejemplo de ejecución en "Simulación Interactiva"

5.6.1. Terminología usada en el tutorial

Menú de configuración: Es el menú flotante que se obtiene pulsando en las ventanas "Máquinas de estado" el botón derecho del ratón.



Menú de primitivas: Es el menú flotante que se obtiene pulsando en las ventanas "Máquinas de estado" el botón izquierdo del ratón.

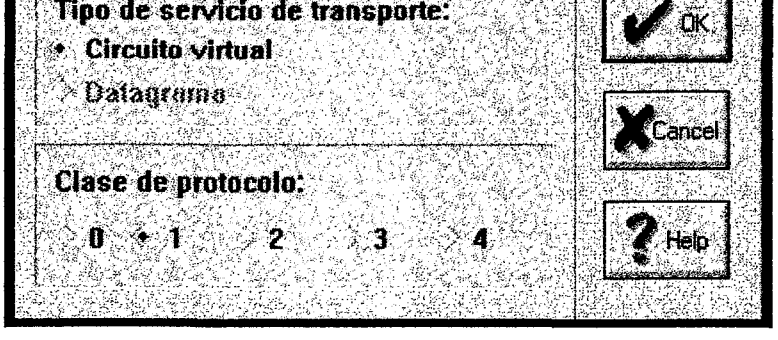


5.6.2. Establecimiento de una conexión de transporte

Para establecer una conexión de transporte el usuario deberá configurar las capas de transporte de los ordenadores a conectarse. Este paso se realiza con el menú de configuración:

- Seleccionaremos **Origen-Destino,** en primer lugar, para dar un nombre a los ordenadores a conectarse.

- La **clase de protocolo y la Calidad del servicio (QOS)** tienen unos parámetros por defecto. Si quiere especificar los suyos propios sólo debe modificar éstos. Por ejemplo, la clase de protocolo por defecto es la 1, si quiere trabajar con clase 4 seleccione Clase de protocolo. Aparece el cuadro de diálogo siguiente:



Realice el cambio. Repita el proceso para la máquina destino.

- Pulse la primitiva de petición de conexión **T_CONNECT.request** del menú de primitivas.

Observará que la máquina origen no sólo permanece en el estado **Inactivo** y no se sitúa en el estado **Pendiente de conexión saliente** como sería lo lógico, sino que ni siquiera

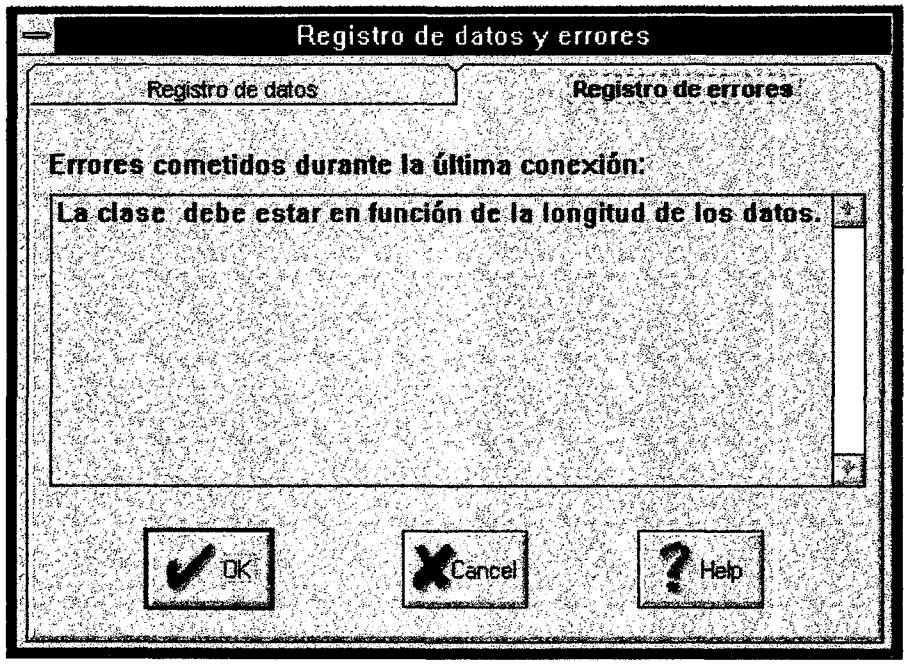
da aviso a la máquina destino del intento de conexión. Se preguntará, ¿ qué ha pasado ? ¿ Por qué no se ha establecido la conexión como era su propósito ?.

La finalidad didáctica del programa le obliga a que encuentre el error, lo subsane y vuelva a intentar el establecimiento de la conexión. Para ello cuenta con la ayuda de un **Registro de errores** donde podrá ver que ha sucedido:

- Pulsar **Registro** en el menú

- Hacer clic en la carpeta **Registro de errores**.

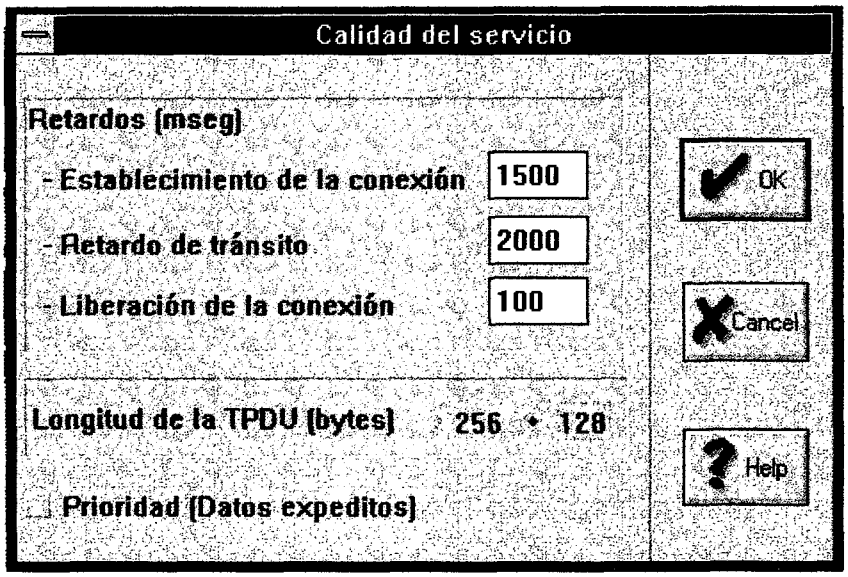
En la lista de errores podrá leer:



Por tanto, hay discordancia entre la clase elegida y uno de los parámetros QOS. Los siguientes pasos serán:

- Fijar un origen y destino.

- Seleccionar Calidad del servicio en ambas máquinas y realizar las modificaciones. En este caso será modificar el tamaño de la TPDU a la que indica la ilustración:



- Ejecutar la primitiva **T_CONNECT.request** en la máquina origen.

- Ejecutar la primitiva **T_CONNECT.response** en la máquina destino.

Como verá, ambos autómatas se han situado en el estado **Preparado Transferencia de datos**. La conexión se ha establecido.

5.6.3. Transferencia de datos normales

Una vez establecida la conexión, nos dispondremos a transferir datos. La primitiva de transporte para la transferencia de datos es **T_DATA.request**.
Al ejecutarla en la máquina origen vía menú de primitivas pueden ocurrir varias cosas:

- No ha indicado ninguna información a transmitir que simule los datos que proceden de los niveles superiores en el momento de establecer la conexión. El programa lo indicará con un mensaje de error y permitirá la introducción del mensaje.

- No se cumpla el **retardo de tránsito**, con lo cual la máquina destino se desconectará y emitirá un **T_DISCONNECT.request** al origen, con lo que se liberará la conexión. Para que esto no vuelva a suceder aumente el valor del retardo de tránsito en la QOS de las máquinas.

- Se transmite una de las TPDU's en las que se ha dividido la información y se notifica a la máquina destino mediante la primitiva **T_DATA.indication**.
Puede continuar ejecutando primitivas **T_DATA.request** hasta que se haya completado el envío de todas las TPDU's.

5.6.4. Transferencia de datos acelerados

El proceso es idéntico al explicado anteriormente pero con la utilización de la primitiva **T_EXP_DATA.request**.

5.6.5. Liberación de la conexión de transporte

La liberación de una conexión se consigue mediante la directiva **T_DISCONNECT.request**.

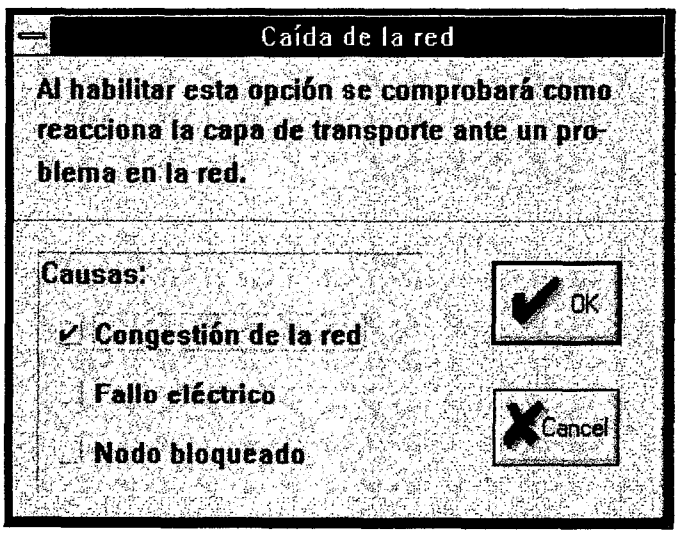
La liberación de una conexión, como ya hemos visto, puede realizarla la propia capa de transporte cuando no se cumplen los retardos, la QOS no es aceptada, etc.
Además puede liberar una conexión, si lo desea, con el menú de primitivas seleccionando **T_DISCONNECT.request**, tanto en la máquina origen como en destino. Si el autómata está en el estado Inactivo no tendrá efecto. En caso contrario volverá a Inactivo eliminando la conexión y liberando variables.

5.6.6. Simular una caída de la red-Primitiva T_DISCONNECT. indication

Para simular una caída de la red y estudiar el comportamiento de la capa de transporte:

- Seleccione **Simular caída de la red** en el menú configuración.

- Confirme cualquiera de los motivos y pulse OK.



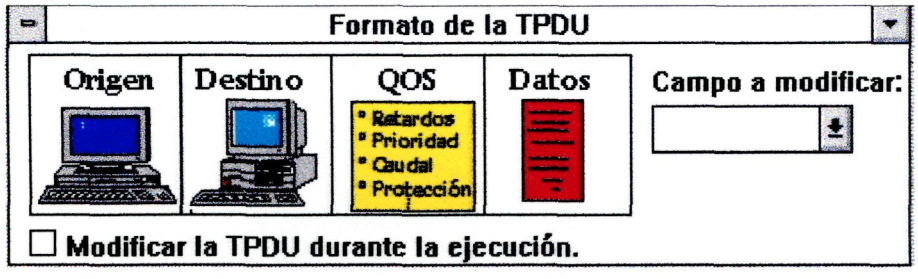
- Pulsar **T_DISCONNECT.indication** en el menú de primitivas. Esta primitiva la utiliza la propia capa de transporte (y no el usuario del servicio) cuando quiere liberar una conexión por algún problema. Este hecho determina la finalización de la conexiones y el retorno al estado Inactivo en todos los sistemas.

- Si la **clase de protocolo** actualmente en uso es la **3 ó 4**, cuando volvamos a establecer un correcto funcionamiento de la red mediante el menú de configuración-Simular caída de la red, automáticamente el programa volverá al estado en el que se encontraba antes de la caída y esperará a que el usuario transmita las TPDU's que habían quedado pendientes mediante la ejecución de los **T_DATA.request** necesarios.

- Si la **clase** es **0, 1, ó 2** la transmisión que se llevaba a cabo se dará por pérdida y correrá a cargo del usuario y no de la capa de transporte repetir el envío.

5.6.7. Modificar la TPDU de formato general

Primeramente deberá crear la ventana Formato general de la TPDU eligiendo Opciones | Mostrar | Formato de la TPDU en el menú principal o pulsando el botón TPDU de la barra de botones.



La ventana Formato de la TPDU presenta una TPDU de formato general con los campos que se encuentran prácticamente en los diez tipos de TPDU's existentes en el protocolo de transporte.

Modificar la TPDU en los niveles bajos del host

- Pulsar el botón izquierdo del ratón sobre los campos **Origen, Destino o QOS**.

- Aparecerá una sencilla **caja de entrada** donde podrá modificar el origen, el destino o la clase de protocolo actualmente seleccionados. Las consecuencias serán la desconexión por parte de la capa de transporte ante discrepancias en la TPDU.

- Hacer clic dentro del campo de datos crea una pequeña ventana con un control de edición que presenta el bloque de datos que se transmite en la actual TPDU y que podrá modificar a su antojo. Un cambio de este tipo provoca un error de checksum.

Modificar la TPDU en tiempo de ejecución

- Habilite la casilla de verificación **"Modificar la TPDU en tiempo de ejecución".**

- Elija en la caja de lista **"Campo a modificar:"** la parte de la TPDU que desea cambiar.

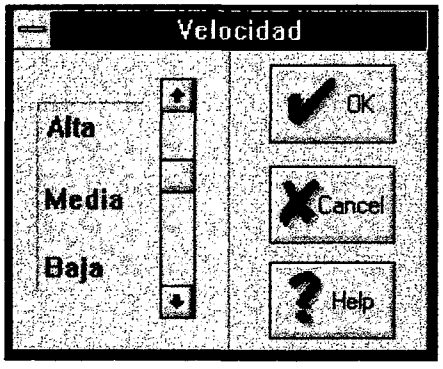
- Después de que la máquina origen haya transmitido la TPDU, en su camino por la red, se crea una caja de entrada donde podrá introducir nuevos valores. Confírmelos pulsando OK.

- Observe los cambios que se producen.

5.6.8. Modificar la velocidad de la ejecución

- Seleccione **Velocidad** en el menú principal.

- Modifique la velocidad de la simulación: baja, media, alta con la barra de desplazamiento y pulse OK.



La velocidad que marque está en relación directa con los retardos de la QOS. Con una velocidad alta los parpadeos de flechas y estados serán más rápidos y por tanto es posible que no se sobrepasen los tiempos fijados para el establecimiento y liberación de la conexión, así como para la transferencia de datos; con lo que el protocolo de transporte no rechazará conexiones al no cumplirse los timers.
Por ejemplo, si marca una velocidad baja y un retardo de establecimiento de 1500 mseg el protocolo rechazará el intento de conexión. Sin embargo, si sitúa la barra en el punto máximo, se aceptará la conexión, al menos en el aspecto referente a timers.

5.6.9. Estudiar la estructura de la TPDU

Cree la ventana Estructura de la TPDU mediante el menú. Opciones | Mostrar | Estructura de la TPDU o el botón Estructura de la TPDU.

Origen: OFRA		Destino: ULPGC	Primitiva: T_DATA.Indication
TPDU:AK	ID: 1	Nº Secuencia: 0	CheckSum: 6700

Observe los cambios que se producen en los campos de la ventana durante la ejecución. Podrá ver los tipos de TPDU's que se intercambian en función de las primitivas que elija. Si la primitiva es **T_DATA.request** tendrá la referencia del número de secuencia que lleva la TPDU y del checksum calculado en el transmisor y el receptor para tener conciencia de un posible error por este motivo.

Pulsar dentro de los distintos recuadros de la ventana para obtener ayuda.

5.7. Ficheros necesitados

Se requieren tres archivos para ejecutar la aplicación: **simula.exe, adan.hlp y bwcc.dll**. Todos deben incluirse dentro del mismo directorio. También es válido cargar la .DLL dentro de C:\WINDOWS\SYSTEM y tener los ficheros **simula.exe y adan.hlp** incluidos en un directorio, o bien copiar **adan.hlp** en C:\WINDOWS.

Simula.exe es el ejecutable de la aplicación, donde se encuentra el código y todos los recursos del programa..

El fichero adan.hlp es necesario para ejecutar la ventana de ayuda del "Simulador del protocolo de transporte OSI".

Bwcc.dll es una DLL suministrada por Borland y que se necesita para poder utilizar algunos bitmaps y botones que se emplean en las cajas de diálogo de la aplicación.

6. Herramientas utilizadas

Las herramientas utilizadas para la creación de este trabajo han sido el **Borland C++ 3.1, CorelDraw 4.0, HiJaak PRO y Microsoft Word 6.0**.

El **Borland C++ 3.1** es un compilador que soporta tanto el lenguaje C como el C++ y que provee todas las herramientas para producir programas para Windows. Además mantiene ficheros de cabecera precompilados, lo cual mejora los tiempos de compilación en un factor de diez; y facilita acceso inmediato a una ayuda interactiva que abarca el lenguaje C, el C++ y las llamadas al API de Windows.

El Borland C++ contiene una serie de utilidades que se usaron para realizar el programa:

- El *Resource Workshop*, que nos permite crear cualquier tipo de recursos (bitmaps, iconos, cursores, menús, etc) de Windows.

- El *WinSight*, con el que se puede hacer un seguimiento de los mensajes de Windows.

- El *Help Compiler*, para la compilación de la ayuda del programa.

- El *Hot Spot Editor*, para crear los bitmaps que se incluyen en la ayuda, los cuales posibilitan el salto a temas concretos dentro de ésta.

El **CorelDraw 4.0** es un completo programa de diseño vectorial. Los fondos utilizados en las ventanas de la aplicación y gran parte de los gráficos se realizaron con Corel.

La ventaja de un programa de este tipo respecto a otro de bitmap es que los diseños realizados con vectores pueden reescalarse todo lo que se desee sin perder

calidad. Sin embargo Windows no permite este tipo de gráficos, por ello hemos de convertir los dibujos creados con Corel a formato .BMP. Para ello se utilizó el HJPRO.

HiJaak PRO ofrece una variada gama de procesos, entre los que destacan la posibilidad de:

- Abrir, editar y procesar casi cualquier imagen, pues acepta innumerables formatos.
- Convertir un archivo de un formato a otro.
- Capturar pantallas o áreas en DOS y Windows.

El **Word 6.0** es un procesador de textos de Microsoft, que se eligió para evitar posibles problemas de compatibilidad (Windows es de Microsoft) y con el cual se realizó el contenido de la ayuda.

7. Bibliografía

Programación con ObjectWindows

Título: Programación de aplicaciones Windows con Borland C++ y ObjectWindows.

Autor: Ramón Franco García.

Editorial: Mac GrawHill

Título: Librería ObjectWindows.

Autor: Namir Clement-Shammas

Editorial: Anaya Multimedia.

Título: ObjectWindows for C++. Guía del usuario.

Autor: Borland International

Título: Programación en Windows con Borland C++.

Autor: Charles Petzold

Editorial: Microsoft Press

Programación en Windows

Título: Programación gráfica en Windows 3.1

Autor: Ben Ezzell

Editorial: Anaya Multimedia

Título: DLL y gestión de memoria

Autor: Mike Klein

Editorial: Anaya Multimedia

Título: Programación avanzada de gráficos en C para Windows

Autor: Lee Adams

Editorial: Mac GrawHill

C / C++

Título: Aplique Turbo C++

Autor: Herbert Schildt

Editorial: Mac GrawHill

Redes de computadoras

Título: Redes de ordenadores

Autor: Andrew S. Tanenbaum

Editorial: Prentice-Hall HispanoAmericana

Normas ISO 8072/8073

8. Anexos

8.1. Listado de la Ayuda

El presente listado corresponde a uno sólo de los ficheros .RTF por los que está constituida la ayuda. En él se puede observar, a modo de ejemplo, la utilización de las notas a pie de página, el texto oculto, tachado y subrayado, etc.

$ k Indice de Ayuda de Simulador del protocolo de transporte OSI

La ayuda está dividida en los siguientes bloques:

Introducción:

~~El Simulador del protocolo de transporte OSI.~~

Opciones del simulador:

~~{bml visual.shg}~~ ~~Simulación visual de la máquina de estados y la capa de transporte OSI.~~
~~{bml inter.shg}~~ ~~Simulación interactiva del protocolo de transporte OSI.~~
~~{bml mod.shg}~~ ~~Tutorial del modelo de referencia OSI.~~

Otras cuestiones:

~~Tutorial: Ejemplo de ejecución en "Simulación interactiva".~~
~~Uso de los cuadros de diálogo.~~
~~Barra rápida.~~ # $ k **Introducción**

El ***Simulador del protocolo de transporte OSI*** ha sido realizado por cuatro motivos básicos:

- Ofrecer un medio de apoyo para los alumnos de la E.U.I.T.T., y en concreto de la asignatura Redes de Ordenadores, con la finalidad didáctica de aclarar y ampliar conceptos que son abstractos sin un soporte hardware y software real acerca del

$ Indice
k Indice
INTRODUCCION
$ Introducción de El Simulador...
k Introducción de El Simulador...

modelo de arquitectura de red que plantea ISO, y en particular sobre su nivel de transporte.

- Desarrollar un programa siguiendo la actual tendencia de programación, la Programación Orientada a Objetos. La OOP ha tomado las mejores ideas de la programación estructurada y las ha combinado con varios conceptos nuevos y potentes que incitan a contemplar la tarea de programar desde un nuevo punta de vista.

- Continuar y ampliar la tarea de crear aplicaciones para entornos gráficos y multitarea como Windows, con su filosofía de mensajes provenientes del usuario o de otras aplicaciones, que el programador debe gestionar y responder.

- Como Trabajo Fin de Carrera, requisito fundamental para la obtención del título de Ingeniero Técnico de Telecomunicación en esta escuela.

[#$ k] Simulación visual de la máquina de estados y la capa de transporte OSI

Introducción

~~Objetivo y fundamentos de la simulación visual.~~

Manejo de la simulación visual

~~Ventanas "Capas R.M.-OSI".~~
~~Ventanas "Máquina de estados".~~
~~Ventana "Modo de simulación".~~
~~Ventana "Paso a paso".~~

Cuadros de diálogo

~~Modo de simulación.~~
~~Velocidad.~~
~~Origen-Destino.~~
~~Información del nivel superior.~~

[#] VISUAL
[$] Simulación visual
[k] Simulación visual de la máquina de estados y la capa de transporte

[#] [$] [k] Simulación interactiva del protocolo de transporte OSI

Introducción

~~Objetivo y fundamentos de la simulación interactiva.~~

Manejo de la simulación interactiva

~~Ventana Máquina de estados.~~
~~Ventana Formato de la TPDU.~~
~~Ventana Estructura de la TPDU.~~
~~Ventana Parámetros de la máquina.~~

Cuadros de diálogo

~~Velocidad.~~
~~Origen-Destino.~~
~~Información del nivel superior.~~
~~Clase de protocolo.~~
~~Calidad del servicio(QOS).~~
~~Simular caída de la red.~~

[#] INTERACTIVA
[$] Simulación interactiva
[k] Simulación interactiva del protocolo de transporte OSI

[#] [$] [k] Tutorial del modelo de referencia OSI

El tutorial del modelo de referencia OSI (RM-OSI) es una de las tres partes en las que está estructurado el programa. Su función es la de informar al usuario de una forma general acerca de este estándard promovido por ISO.

Orígenes:

~~El modelo de referencia OSI: necesidad y creación.~~

Capas de la arquitectura:

~~Capa física.~~
~~Capa de enlace.~~
~~Capa de red.~~
~~Capa de transporte.~~
~~Capa de sesión.~~
~~Capa de presentación.~~
~~Capa de aplicación.~~

[#] MODELO
[$] Tutorial R-M OSI
[k] Tutorial OSI

8.2. *Listado del código fuente: simula.cpp*

```
// ObjectWindows - (C) Copyright 1994 by Fco. Javier Fleitas Negrín

// Archivos de cabecera
#define WIN31
#define STRICT
#include <owl.h>
#include <window.h>
#include <string.h>
#include <scrollba.h>
#include <stdio.h>
#include <ctype.h>
#include <inputdia.h>
#include "simula.h"  //archivo de cabecera de la aplicación
#include <edit.h>
#include <combobox.h>
#include <listbox.h>
#include <bgrpbox.h>
#include <bchkbox.h>
#include <bbutton.h>
#include <bradio.h>
#include <static.h>
#include <button.h>
#include <bwcc.h>   //Necesario para los iconos de las cajas de dialogo

//Constantes necesarias para las ventanas de ayuda
#define CAMPOTPDU 101
#define CAMPOID 102
#define CAMPOCHECKSUM 103
#define CAMPOORIGEN 104
#define CAMPODESTINO 105
#define VENTANA1 106
#define CAMPOSECUENCIA 107

//Mensajes definidos por el programador
#define WM_CONNIND WM_USER
#define WM_DISCIND WM_USER + 1
#define WM_DISCREQ WM_USER + 2
#define WM_CCONFIRM WM_USER + 3
#define WM_DATAIND WM_USER + 4
#define WM_DATAEXPIND WM_USER + 5

//Constantes que identifican los mensajes de la barra de botones
const int IDSOLOMAQ =  201;
const int IDSALIDA=   202;
const int IDSOLOARQ =  203;
```

```
const int IDNUEVO =   204;
const int IDCERRAR =  205;
const int IDVISUAL =  206;
const int IDTPDU =    208;
const int IDSTRUCT =  209;
const int IDAYUDA =   210;
const int IDORDENAR = 211;
const int IDPASOAPASO =212;

//Variables globales
RECT coordenadas_ventana;
int ancho,alto,arriba,izda,abajo,alto_barra,alto_menu,alto_titulo,Boton=0;
BOOL vh1activas,modeloactivo,todasactivas,Infoactiva,barraprincipal,TPDUactiva,
   botonactivo;
POINT cambio_cursor;

//*********************************************************
//Definición de ls estructuras de la applicación
struct nodos{
 char parte[70],auxiliar[70];
 DWORD checksum;
 nodos *prox;
 };

struct dialogo_caida{
 BOOL conges[3];
 } red;

struct dialogo_modo{
 BOOL modo[2];
 } simulacion;

struct dialogo_qos{
 BOOL datos_expeditos;
 BOOL l256;
 BOOL l128;
 char retardo1[5];
 char retardo2[5];
 char retardo3[5];
 };

struct dialogo_clase{
 BOOL cl[5];
 BOOL tipo[2];
 };

struct dialogo_orgdest{
 char origen[15];
 char destino[15];
```

```
 };

enum estado {Inactivo,conexion_saliente,conexion_entrante,transferencia_datos};

struct calidad_servicio{
 BOOL datos_expeditos,longitud[2];
 int retardo1,retardo2,retardo3;
 };

struct unidad_datos{
 int origen,destino,clase,n_secuencia,n_secuenciaexp,ID;
 DWORD checksum;
 char info[255],primitiva[30],nombre[15];
 estado est;
 calidad_servicio c;
 BOOL caida;
 };
//**********************************************************
//**********************************************************
//Funciones globales
void delay(DWORD T)
{
 DWORD time1;

 time1=GetTickCount();
 do
 {
  ;
 }while((GetTickCount() - time1) < T);
}

void Rellenar(HWND hwnd,COLORREF color,int x,int y)
{
 HDC hdc;
 HBRUSH brush;
 HGDIOBJ brushold;

 hdc = GetDC(hwnd);
 brush=CreateSolidBrush(color);
 brushold=SelectObject(hdc,brush);
 FloodFill(hdc,x,y,RGB(0,0,0));
 SelectObject(hdc,brushold);
 DeleteObject(brush);
 ReleaseDC(hwnd, hdc);
}

void Dibujarbitmap(HBITMAP MyBitmap,HWND hwnd,int x,int y,BOOL copia)
{
 HDC hdcMemory,DC;
```

```
HGDIOBJ hbmpOld;
BITMAP bm;

DC = GetDC(hwnd);
hdcMemory = CreateCompatibleDC(DC);
hbmpOld = SelectObject(hdcMemory, MyBitmap);
GetObject(MyBitmap, sizeof(BITMAP), &bm);
if (copia)
 BitBlt(DC,x,y,bm.bmWidth, bm.bmHeight,hdcMemory,0,0,SRCCOPY);
else
 StretchBlt(DC,9,y,70,x, hdcMemory, 0, 0, bm.bmWidth, bm.bmHeight, SRCCOPY);□
SelectObject(hdcMemory, hbmpOld);
DeleteDC(hdcMemory);
ReleaseDC(hwnd, DC);
}
//************************************************************************

//************************************************************************
//Clase TMiProyecto-Definición de la clase TMiproyecto
//
//Descripción:
//Clase que representa el objeto aplicación Simulador del P.T. OSI. Sigue
//la conducta fundamental de una aplicación ObjectWindows
//
//Parámetros de creación y notas:
//Aname: Nombre de la aplicación
//hInstance: Handle de la instancia de la aplicación
//hPrevInstance: Handle de la instancia anterior
//lpCmdLine: Puntero a la línea de comandos
//nCmdShow: Tamaño inicial de la ventana
//
//Comportamiento del destructor: Ninguno
//
//Clase padre: TApplication
//
//Clases hijas : Ninguna
//
//************************************************************************
class TMiproyecto : public TApplication
{
public:
 TMiproyecto(LPSTR AName, HINSTANCE hInstance, HINSTANCE hPrevInstance,
   LPSTR lpCmdLine, int nCmdShow);

protected:

 virtual void InitMainWindow();
 virtual void InitApplication();
 char Tituloventana[70];
```

```
};
//Fin de la definicion de la clase TMiproyecto

//Uso de una macro que declara definiones de tipo (typedef) en el resto de
//clases de la aplicación
_CLASSDEF(VentanaPrincipal)
_CLASSDEF(Ventanahija1)
_CLASSDEF(Ventanahija2)
_CLASSDEF(Ventanahija4)
_CLASSDEF(Ventanahija5)
_CLASSDEF(Cdialogos)
_CLASSDEF(TPresentacion)
_CLASSDEF(Cdialogos9)
_CLASSDEF(Velocidad)
_CLASSDEF(TAcercade)
_CLASSDEF(Ventanahija3)
_CLASSDEF(TVentana_flotante)
_CLASSDEF(TVentanaBarra)
_CLASSDEF(Simvisual)
_CLASSDEF(MROSI)
_CLASSDEF(Ventanahija6)
_CLASSDEF(Ventanahija7)
_CLASSDEF(Ventanahija8)
_CLASSDEF(Ventanahija9)
_CLASSDEF(Ventanahija10)

//Punteros globales
 PVentanahija3 Info;
 PVentanahija1 maquinas[2];
 PSimvisual arq[2];
 PMROSI modelo;
 PTVentana_flotante Ventana_flotante;
 PVentanahija4 TPDU;
 PVentanahija5 infomaquina;
 PVentanahija6 campos;
 PVentanahija7 CAPA;
 PVentanahija7 niveles[16];
 PVentanahija8 modo;
 PVentanahija9 boton_pap;
 PVentanahija10 registro;
 PTVentanaBarra Barra;
 PVentanafondo fondo;
 static HWND secuencia[4];
 static PSimvisual secuenciaptrarq[2];
 static PVentanahija1 secuenciaptrmaq[2];

//************************************************************************
//Clase VentanaPrincipal-Definición de la clase VentanaPrincipal
//
```

```
//Descripción:
//Esta clase representa la ventana principal de la aplicación "Simulador del
//Protocolo de Transporte OSI.
//
//Parámetros de creación y notas:
//Carga el menú principal de la aplicación y un bitmap desde el archivo de
//recursos y comprueba la presencia del ratón y la configuración de WINDOWS.
//
//Comportamiento del destructor: Borra el bitmap de la memoria
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//*************************************************************************
class VentanaPrincipal : public TWindow
{
 public:
  VentanaPrincipal(PTWindowsObject AParent,LPSTR ATitle);

  ~VentanaPrincipal();

  void Paint(HDC DC, PAINTSTRUCT& P);

  virtual void Uso(RTMessage Msg)
   = [CM_FIRST + 1];

  virtual void Acercade(RTMessage Msg)
   = [CM_FIRST + 2];

  virtual void Activarmaquinas(RTMessage Msg)
   = [CM_FIRST + 3];

  virtual void Activararq(RTMessage Msg)
   = [CM_FIRST + 4];

  virtual void Activarregistro(RTMessage Msg)
   = [CM_FIRST + CM_REGISTRO];

  virtual void Ordenarventanas(RTMessage Msg)
   = [CM_FIRST + CM_ORDENAR];

  virtual void ActivarstructTPDU(RTMessage Msg)
   = [CM_FIRST + CM_ESTRUCTURA];

  virtual void CerrarstructTPDU(RTMessage Msg)
   = [CM_FIRST + CM_CERRAREST];

  virtual void Activarambas(RTMessage Msg)
```

```
 = [CM_FIRST + CM_AMBAS];

virtual void Cerrarambas(RTMessage Msg)
 = [CM_FIRST + CM_CERRARAMBAS];

virtual void Activarventanas(RTMessage Msg)
 = [CM_FIRST +6];

virtual void ActivarTPDU(RTMessage Msg)
 = [CM_FIRST + CM_TPDU];

virtual void CerrarTPDU(RTMessage Msg)
 = [CM_FIRST + CM_CERRARTPDU];

virtual void Salir(RTMessage Msg)
 = [CM_FIRST + 7];

virtual void Indice(RTMessage Msg)
 = [CM_FIRST + 10];

virtual void TutorialRMOSI(RTMessage Msg)
 = [CM_FIRST + CM_OSI];

virtual void Ayuda_interactiva(RTMessage Msg)
 = [CM_FIRST + CM_HELPINTER];

virtual void Ayuda_visual(RTMessage Msg)
 = [CM_FIRST + CM_HELPVISUAL];

virtual void Ejemplo_Ayuda(RTMessage Msg)
 = [CM_FIRST + CM_EJEMPLO];

virtual void Nuevo(RTMessage Msg)
 = [CM_FIRST + 9];

virtual void Velocity(RTMessage Msg)
 = [CM_FIRST + 15];

virtual void Modofuncionamiento(RTMessage Msg)
 = [CM_FIRST + CM_MODO];

virtual void WMCreate(RTMessage Msg)
 =[WM_FIRST+WM_CREATE];

virtual void WMTimer(RTMessage Msg)
 =[WM_FIRST+WM_TIMER];

virtual void WMSize(RTMessage Msg)
 = [WM_FIRST + WM_SIZE];
```

```
 virtual BOOL CanClose();

 int cont;

protected:

 LPSTR GetClassName() { return "VentanaPrincipal";};

 void GetWindowClass( WNDCLASS _FAR & );

 HMENU nuevomenu;
 int FlagVentana_Info;
 BOOL flaglineaestado;
 POINT Punto, PuntoAnterior;

private:

 void Calculardimensiones();
 void limpieza();
 HBITMAP hbmpMyBitmap;
};
//Fin de la definición de la clase VentanaPrncipal

//************************************************************************
//Clase Ventanahija1-Definición de la clase Ventanahija1
//
//Descripción:
//Clase que representa el autómata de transporte OSI para un C.V.
//
//Parámetros de creación y notas:
//Carga el bitmap del autómata desde el archivo de recursos, inicializa el
//estado de la máquina y las estructuras, así como rellena los parámetros de
//colocación, estilo y tamaño de la ventana de la estructura Attr.
//
//Comportamiento del destructor: Borra el bitmap de la memoria
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//************************************************************************
class Ventanahija1:public TWindow
{
public:

  Ventanahija1(PTWindowsObject AParent,LPSTR ATitle,BYTE WinNum,BOOL size);

  ~Ventanahija1();
```

```
void Paint(HDC DC, PAINTSTRUCT& P);

virtual void WMLButtonDown(RTMessage Msg)
 =[WM_FIRST+WM_LBUTTONDOWN];

virtual void WMRButtonDown(RTMessage Msg)
 =[WM_FIRST+WM_RBUTTONDOWN];

virtual void WMSYSCOMMAND(RTMessage Msg)
 =[WM_FIRST+WM_SYSCOMMAND];

virtual void Directiva1(RTMessage Msg)
 = [CM_FIRST + IDM_CREQUEST];

virtual void Respuestaconnind(RTMessage Msg)
 = [WM_FIRST + WM_CONNIND];

virtual void Respuestadiscreq(RTMessage Msg)
 = [WM_FIRST + WM_DISCREQ];

virtual void Directiva3(RTMessage Msg)
 = [CM_FIRST + IDM_DREQUEST];

virtual void Directiva4(RTMessage Msg)
 = [CM_FIRST + IDM_DINDICATION];

virtual void Respuestadisconnectind(RTMessage Msg)
 = [WM_FIRST + WM_DISCIND];

virtual void Directiva5(RTMessage Msg)
 = [CM_FIRST + IDM_DATAREQUEST];

virtual void Respuestadataind(RTMessage Msg)
 = [WM_FIRST + WM_DATAIND];

virtual void Directivaexpedita(RTMessage Msg)
 = [CM_FIRST + IDM_EXPDATAREQUEST];

virtual void Respuestadataexpind(RTMessage Msg)
 = [WM_FIRST + WM_DATAEXPIND];

virtual void Directiva7(RTMessage Msg)
 = [CM_FIRST + IDM_CRESPONSE];

virtual void Respuestaconnectconfirm(RTMessage Msg)
 = [WM_FIRST + WM_CCONFIRM];

virtual void Origendestino(RTMessage Msg)
```

```
 = [CM_FIRST + ORG_DEST];

 virtual void Informa(RTMessage Msg)
 = [CM_FIRST + IDM_INFORMACION];

 virtual void Clase(RTMessage Msg)
 = [CM_FIRST + IDM_CLASE];

 virtual void QOS(RTMessage Msg)
 = [CM_FIRST + IDM_CALIDAD];

 virtual void Simulacaida(RTMessage Msg)
 = [CM_FIRST + IDM_CAIDARED];

 virtual void Informacionmaquina(RTMessage Msg)
 = [CM_FIRST + IDM_INFOMAQ];

 BOOL Comprobar_qos();

 void Actuarantecaida(int n, int m);
 void Brocha(int estado,char directiva[40],HWND hwnd1);

 static PVentanahija1 ventanadestino;
 static PVentanahija1 ventanaorigen;
 static int contador;
 static int retardo;
 static int parametro_reconexion;
 static HWND wndini;
 static HWND wndnopicada;
 static BOOL reconectarse;
 dialogo_qos calidad;
 dialogo_clase c;
 dialogo_orgdest a;
 unidad_datos datos;
 calidad_servicio cal;
 DWORD time_est,time_transf;
 char msg_sesion[255];
 BOOL array_errores[14];
nodos *cab,*temp,*almacenados,*prioridad,*temp2,*almacenados2,*ptrerror,*almacen;

protected:

 LPSTR GetClassName() {return "Ventanahija1";};

 void GetWindowClass( WNDCLASS _FAR & );

 void llenarinfo(LPSTR p,LPSTR t,LPSTR s,LPSTR r,LPSTR o);
 void llenarorgdest(LPSTR p,LPSTR t);
 void menuflotante(POINT r);
```

```
  void Inicializarestructuras();
  void Inicializarventanas(int i,int j);
  void Borralista(nodos **cabeza);
  void Crealista(nodos **cabeza,char modulo[70],DWORD sum);
  BOOL movibles,terminar,continuar_tx,acelerado;
  RECT r;
  POINT point;
  int puntos[22],contadormsg,n,total_TPDUs;
  HBITMAP hbmpMyBitmap;
};
//Fin de la definición de la clase Ventanahija1

//*************************************************************************
//Clase Ventanafondo-Definición de la clase Ventanafondo
//
//Descripción:
//Clase que representa una ventana invisible que contiene a las demás
//
//Parámetros de creación y notas:
//Rellena los parámetros de colocación, estilo y tamaño de la ventana
// de la estructura Attr.□
//
//Comportamiento del destructor: Ninguno
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//*************************************************************************
class Ventanafondo: public TWindow{

 public:

  Ventanafondo(PTWindowsObject AParent,LPSTR ATitle);

  LPSTR GetClassName() { return "Ventanafondo";};

  void GetWindowClass( WNDCLASS _FAR & );
};
//Fin de la definición de la clase Ventanafondo

//*************************************************************************
//Clase MROSI-Definición de la clase MROSI
//
//Descripción:
//Clase que representa el Modelo de referencia OSI.
//
//Parámetros de creación y notas:
//Carga el bitmap del modelo desde el archivo de recursos, rellena los
```

```
//parámetros de colocación, estilo y tamaño de la ventana de la estructura
//Attr y define los rectángulos que delimitan las capas del modelo.
//
//Comportamiento del destructor: Borra el bitmap de la memoria
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//*************************************************************************
class MROSI: public TWindow{

 public:

  MROSI(PTWindowsObject AParent,LPSTR ATitle);

  ~MROSI();

  void Paint(HDC DC, PAINTSTRUCT& P);

  virtual void WMLButtonDown(RTMessage Msg)
   =[WM_FIRST+WM_LBUTTONDOWN];

  RECT TPDU1,TPDU2;

 protected:

  RECT capas[16];
  int puntos[4],n;
  BOOL red;
  HBITMAP hbmpMyBitmap,hbmpMyBitmap2;
  LPSTR GetClassName() {return "MROSI" ;};
  void GetWindowClass( WNDCLASS _FAR & );
};
//Fin de la definición de la clase MROSI

//*************************************************************************
//Clase Simvisual-Definición de la clase Simvisual
//
//Descripción:
//Clase que representa una parte del Modelo de Referencia OSI.
//
//Parámetros de creación y notas:
//Carga el bitmap adecuado desde el archivo de recursos, inicializa el
//estado del autómata y resto de variables, así como rellena los parámetros
// de colocación, estilo y tamaño de la ventana de la estructura Attr.
//
//Comportamiento del destructor: Borra el bitmap de la memoria
//
```

```
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//*************************************************************************
class Simvisual: public TWindow{

 public:

  Simvisual(PTWindowsObject AParent,LPSTR ATitle,BYTE WinNum);

  ~Simvisual();

  void Paint(HDC DC, PAINTSTRUCT& P);

  virtual void WMLButtonDown(RTMessage Msg)
   =[WM_FIRST+WM_LBUTTONDOWN];

  virtual void WMRButtonDown(RTMessage Msg)
   =[WM_FIRST+WM_RBUTTONDOWN];

  virtual void WMSYSCOMMAND(RTMessage Msg)
   =[WM_FIRST+WM_SYSCOMMAND];

  virtual void Directiva1(RTMessage Msg)
   = [CM_FIRST + IDM_CREQUEST];

  virtual void Directiva3(RTMessage Msg)
   = [CM_FIRST + IDM_DREQUEST];

  virtual void Directiva4(RTMessage Msg)
   = [CM_FIRST + IDM_DINDICATION];

  virtual void Directiva7(RTMessage Msg)
   = [CM_FIRST + IDM_CRESPONSE];

  virtual void Directiva5(RTMessage Msg)
   = [CM_FIRST + IDM_DATAREQUEST];

  virtual void Localremoto(RTMessage Msg)
   = [CM_FIRST + ORG_DEST];

  virtual void Informa(RTMessage Msg)
   = [CM_FIRST + IDM_INFORMACION];

   void Brocha(int estado,COLORREF parpadeo,char directiva[40],HWND hwnd1);

  static int contador;
  static int retardo;
```

```
  static BOOL inicializado;
  static int recorrido;
  static BOOL continuo;
  dialogo_orgdest b;
  int cont;
  estado flagestado;
  BOOL en_marcha;

 protected:

  void menuflotante(POINT r);
  char t[40];
  int puntos[4];
  HBITMAP hbmpMyBitmap;
  LPSTR GetClassName() {return "Simvisual" ;};
  void GetWindowClass( WNDCLASS _FAR & );
};
//Fin de la definición de la clase Simvisual

//**************************************************************************
//Clase Ventanahija3-Definición de la clase Ventanahija3
//
//Descripción:
//Clase que representa la estructura de la TPDU.
//
//Parámetros de creación y notas:
//Crea los controles de tipo texto estático que contiene la ventana y rellena
// los parámetros de colocación, estilo y tamaño de la ventana de la estructura
// Attr.
//
//Comportamiento del destructor: Ninguno.
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//**************************************************************************
class Ventanahija3:public TWindow
{
 public:

  Ventanahija3(PTWindowsObject AParent,LPSTR ATitle);

  virtual void WMControlColor(RTMessage Msg)
   =[WM_FIRST+WM_CTLCOLOR];

  virtual void WMLButtonDown(RTMessage Msg)
   =[WM_FIRST+WM_LBUTTONDOWN];
```

```
  void Paint(HDC DC, PAINTSTRUCT& P);

  void Fijarorigen(LPSTR texto);
  void Fijardestino(LPSTR texto);
  void Fijarprimitiva(LPSTR texto);
  void Fijartpdu(LPSTR texto);
  void Fijarid(LPSTR texto);
  void Fijarchecksum(LPSTR texto);
  void Fijarn_secuencia(LPSTR texto);
  PTStatic org,dest,tpdu,primitiva,id,checksum,n_secuencia;

protected:

  LPSTR GetClassName() {return "Ventanahija3" ;};

  void GetWindowClass( WNDCLASS _FAR & );

  void Dibujarrecuadros(HDC hDC, RECT *rect);

  void Dibujarinfo(HDC hDC);

  BOOL movible;
  HFONT Font;
  RECT rect1,rect2,rect3,rect4,rect5,rect6,rect7;
};

//Fin de la definición de la clase Ventanahija3

//*************************************************************************
//Clase Ventanahija4-Definición de la clase Ventanahija4
//
//Descripción:
//Clase que representa la ventana Formato de la TPDU.
//
//Parámetros de creación y notas:
//Carga el bitmap con un formato general desde el archivo de recursos,
//rellena los parámetros de colocación, estilo y tamaño de la ventana de la
// estructura Attr, crea un control casilla de verificación, uno tipo texto
//estático y una caja combinada. Además habilita el gestor de teclado.
//
//Comportamiento del destructor: Borra el bitmap de la memoria
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//*************************************************************************
class Ventanahija4:public TWindow
{
```

```
  void Paint(HDC DC, PAINTSTRUCT& P);

  void Fijarorigen(LPSTR texto);
  void Fijardestino(LPSTR texto);
  void Fijarprimitiva(LPSTR texto);
  void Fijartpdu(LPSTR texto);
  void Fijarid(LPSTR texto);
  void Fijarchecksum(LPSTR texto);
  void Fijarn_secuencia(LPSTR texto);
  PTStatic org,dest,tpdu,primitiva,id,checksum,n_secuencia;

protected:

  LPSTR GetClassName() {return "Ventanahija3" ;};

  void GetWindowClass( WNDCLASS _FAR & );

  void Dibujarrecuadros(HDC hDC, RECT *rect);

  void Dibujarinfo(HDC hDC);

  BOOL movible;
  HFONT Font;
  RECT rect1,rect2,rect3,rect4,rect5,rect6,rect7;
};

//Fin de la definición de la clase Ventanahija3

//*************************************************************************
//Clase Ventanahija4-Definición de la clase Ventanahija4
//
//Descripción:
//Clase que representa la ventana Formato de la TPDU.
//
//Parámetros de creación y notas:
//Carga el bitmap con un formato general desde el archivo de recursos,
//rellena los parámetros de colocación, estilo y tamaño de la ventana de la
// estructura Attr, crea un control casilla de verificación, uno tipo texto
//estático y una caja combinada. Además habilita el gestor de teclado.
//
//Comportamiento del destructor: Borra el bitmap de la memoria
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//*************************************************************************
class Ventanahija4:public TWindow
{
```

```
public:

  Ventanahija4(PTWindowsObject AParent,LPSTR ATitle);

  ~Ventanahija4();

  void Paint(HDC DC, PAINTSTRUCT& P);

  virtual void WMDestroy(RTMessage Msg)
   =[WM_FIRST+WM_DESTROY];

  virtual void WMLButtonDown(RTMessage Msg)
   =[WM_FIRST+WM_LBUTTONDOWN];

  virtual void SetupWindow();

  void Procesar();
  void llenarorgstruct(LPSTR a);
  void llenardeststruct(LPSTR b);

protected:

  LPSTR GetClassName() {return "Ventanahija4";};

  void GetWindowClass( WNDCLASS _FAR & );

  HBITMAP hbmpMyBitmap;
  PTCheckBox CheckBox;
  PTComboBox ComboBox;
};

//Fin de la definición de la clase Ventanahija4

//*************************************************************************
//Clase Ventanahija5-Definición de la clase Ventanahija5
//
//Descripción:
//Clase que representa la ventana "Parámetros de la máquina".
//
//Parámetros de creación y notas:
//Carga un bitmap desde el archivo de recursos, rellena los parámetros de
//colocación, estilo y tamaño de la ventana de la estructura Attr y crea
//controles de texto estático.
//
//Comportamiento del destructor: Borra el bitmap de la memoria
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
```

```
//
//**********************************************************************
class Ventanahija5:public TWindow
{
 public:

  Ventanahija5(PTWindowsObject AParent,LPSTR ATitle,int x,int y);

  ~Ventanahija5();

  void Paint(HDC DC, PAINTSTRUCT& P);

  virtual void WMSYSCOMMAND(RTMessage Msg)
   =[WM_FIRST+WM_SYSCOMMAND];


 protected:

  LPSTR GetClassName() {return "Ventanahija5" ;};

  void GetWindowClass( WNDCLASS _FAR & );

  void Rellenarparametros();

  HBITMAP hbmpMyBitmap;
  PTStatic nombre,tsap,id,clase,prioridad,longitud,estab,transito,liberacion;
  char n[12],t[5],i[3],c[2],p[3],l[5],r1[5],r2[5],r3[5];
  dialogo_qos param_calidad;
  unidad_datos param_datos;
};

//Fin de la definición de la clase Ventanahija5

//**********************************************************************
//Clase Ventanahija6-Definición de la clase Ventanahija6
//
//Descripción:
//Clase que representa una ventana donde se puede modificar el campo datos
//de la TPDU.
//
//Parámetros de creación y notas:
//Carga un bitmap que representa el campo de datos, crea el botón "Cerrar",
//rellena los parámetros de colocación, estilo y tamaño de la ventana de la
//estructura Attr.
//
//Comportamiento del destructor: Borra el bitmap de la memoria
//
//Clase padre: TWindow
//
```

```
//Clases hijas : Ninguna
//
//************************************************************************
class Ventanahija6:public TWindow
{
 public:

  Ventanahija6(PTWindowsObject AParent,LPSTR ATitle);□

  ~Ventanahija6();

  void Paint(HDC DC, PAINTSTRUCT& P);

  virtual void IDcerrar(RTMessage Msg)=[ID_FIRST + IDCERRAR];

 protected:

  LPSTR GetClassName() {return "Ventanahija6" ;};

  void GetWindowClass( WNDCLASS _FAR & );

  HBITMAP hbmpMyBitmap;
  PTBButton PBotoncerrar;
  PTEdit Edit1;
  char mensaje[70];
};
//Fin de la definición de la clase Ventanahija6

//************************************************************************
//Clase Ventanahija7-Definición de la clase Ventanahija7
//
//Descripción:
//Clase que representa las capas del modelo OSI.
//
//Parámetros de creación y notas:
//Carga sólo el bitmap identificativo de la capa que se indica a través del
//parámetro entero j que se pasa al constructor, crea un botón para cerrar la
//ventana, así como rellena los parámetros de colocación, estilo y tamaño de
// la ventana de la estructura Attr.
//
//Comportamiento del destructor: Borra el bitmap de la memoria y el botón.
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//************************************************************************
class Ventanahija7:public TWindow
{
```

```
public:

  Ventanahija7(PTWindowsObject AParent,LPSTR ATitle,int i);

  ~Ventanahija7();

  virtual void WMLButtonDown(RTMessage Msg)
   =[WM_FIRST+WM_LBUTTONDOWN];

   virtual void WMMove(RTMessage Msg)
   =[WM_FIRST+WM_MOVE];

  void Paint(HDC DC, PAINTSTRUCT& P);

  virtual void IDcerrar(RTMessage Msg)=[ID_FIRST + IDCERRAR]
  {
   SendMessage(HWindow, WM_CLOSE, 0, NULL);
  }

  RECT boton;
  int m;

protected:

  LPSTR GetClassName() {return "Ventanahija7" ;};

  void GetWindowClass( WNDCLASS _FAR & );

  HBITMAP hbmpMyBitmap;
  PTBButton PBotoncerrar;
  int ayuda;
};
//Fin de la definición de la clase Ventanahija7

//*************************************************************************
//Clase Ventanahija8-Definición de la clase Ventanahija8
//
//Descripción:
//Clase que representa la ventana Modo de funcionamiento y que simula un
//visualizador digital y una barra de desplazamiento para conmutar el modo.
//
//Parámetros de creación y notas:
//Carga los bitmaps necesarios para la realización y movimiento de la barra,
// inicializa variables y rellena los parámetros de colocación, estilo y
//tamaño de la ventana de la estructura Attr.
//
//Comportamiento del destructor: Borra los bitmaps de la memoria
//
//Clase padre: TWindow
```

```
//
//Clases hijas : Ninguna
//
//*************************************************************************
class Ventanahija8:public TWindow
{
 public:

  Ventanahija8(PTWindowsObject AParent,LPSTR ATitle);

  ~Ventanahija8();

  void Paint(HDC DC, PAINTSTRUCT& P);

  virtual void WMLButtonDown(RTMessage Msg)
   =[WM_FIRST+WM_LBUTTONDOWN];

  virtual void WMLButtonUp(RTMessage Msg)
   =[WM_FIRST+WM_LBUTTONUP];

  virtual void WMMouseMove(RTMessage Msg)
   =[WM_FIRST+WM_MOUSEMOVE];

  HBITMAP hbmpMyBitmap,hbmpMyBitmap2,hbmpMyBitmap3,hbmpMyBitmap4;
  PTBButton PBotonmodo;
  BOOL mover;
  int x,y;

 protected:

  LPSTR GetClassName() {return "Ventanahija8" ;};

  void GetWindowClass( WNDCLASS _FAR & );
};
//Fin de la definición de la clase Ventanahija8

//*************************************************************************
//Clase Ventanahija9-Definición de la clase Ventanahija9
//
//Descripción:
//Clase que representa la ventana Paso a paso y que , a través del botón
//Play, avanza paso a paso la simulación.
//
//Parámetros de creación y notas:
//Carga un bitmap, crea el botón Play, inicializa variables y rellena los
// parámetros de colocación, estilo y tamaño de la ventana de la estructura
// Attr.
//
//Comportamiento del destuctor: Borra el bitmap y el botón de la memoria
```

```
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//***************************************************************************
class Ventanahija9:public TWindow
{
 public:

  Ventanahija9(PTWindowsObject AParent,LPSTR ATitle);

  ~Ventanahija9();

  void Paint(HDC DC, PAINTSTRUCT& P);

  virtual void IDpasoapaso(RTMessage Msg)
   =[ID_FIRST + IDPASOAPASO];

  void Inicializar(int indice1,int indice2);

 protected:

  LPSTR GetClassName() {return "Ventanahija9" ;};

  void GetWindowClass( WNDCLASS _FAR & );
  void Dibujarrecuadros(HDC hDC, RECT *rect);
  void Inicializartitulos();

  PTBButton PBotonpasoapaso;
  HBITMAP hbmpMyBitmap;
  int cont;
};
//Fin de la definición de la clase Ventanahija9

//***************************************************************************
//Clase TVentana_flotante-Definición de la clase TVentana_flotante
//
//Descripción:
//Clase que representa la ventana de ayuda que se activa al situarse el
//cursor sobre la barra de botones.
//
//Parámetros de creación y notas:
//Rellena los parámetros de colocación, estilo y tamaño de la ventana de la
// estructura Attr y crea un tipo de letra lógica.
//
//Comportamiento del destructor: Borra el tipo de letra.
//
//Clase padre: TWindow
```

```
//
//Clases hijas : Ninguna
//
//*************************************************************************
class TVentana_flotante : public TWindow
{
 public:

  TVentana_flotante (PTWindowsObject AParent, LPSTR ATitle);
  ~TVentana_flotante();
  void Paint(HDC DC, PAINTSTRUCT& P);

 protected:

  virtual void GetWindowClass(WNDCLASS& ClaseVentana);
  virtual LPSTR GetClassName(){ return "TVentana_flotante";}
  HANDLE hHelv;
};
//Fin de la definición de la clase TVentana_flotante

//*************************************************************************
//Clase TVentanaBarra-Definición de la clase TVentanaBarra
//
//Descripción:
//Clase que crea la ventana que contiene la barra de botones y la línea de
//estado para la ayuda.
//
//Parámetros de creación y notas:
//Rellena los parámetros de colocación, estilo y tamaño de la ventana de la
// estructura Attr y crea los botones que conforman la barra según la opción
//del programa determinada por el parámetro entero k que se pasa en el
//constructor. Además crea un control de texto estático.
//
//Comportamiento del destructor: Borra los botones y el control estático.
//
//Clase padre: TWindow
//
//Clases hijas : Ninguna
//
//*************************************************************************
class TVentanaBarra: public TWindow
{
 public:

  TVentanaBarra (PTWindowsObject AParent, LPSTR ATitle,int k);
  ~TVentanaBarra();

  PTBButton PBotonAyuda;
  PTBButton PBotonmaquinas;
```

```
PTBButton PBotonarquitecturas;
PTBButton PBotonsalida;
PTBButton PBotonnuevo;
PTBButton PBotonvisual;
PTBButton PBotontpdu;
PTBButton PBotonstruct;
PTBButton PBotonordenar;

virtual void IDAyuda(RTMessage Msg)=[ID_FIRST + IDAYUDA]
{
 WinHelp(HWindow,"ADAN.HLP",HELP_INDEX,0L);
}

virtual void IDSolomaq(RTMessage Msg)=[ID_FIRST + IDSOLOMAQ]
{
 if (vh1activas==FALSE)
        SendMessage(Parent->HWindow, WM_COMMAND, CM_MAQ, NULL);
}

virtual void IDSoloarq(RTMessage Msg)=[ID_FIRST + IDSOLOARQ]
{
 if (modeloactivo==FALSE)
  SendMessage(Parent->HWindow, WM_COMMAND, CM_ARQ, NULL);
}

virtual void IDnuevo(RTMessage Msg)=[ID_FIRST + IDNUEVO]
{
 SendMessage(Parent->HWindow, WM_COMMAND, CM_NUEVO, NULL);
}

virtual void IDsalida(RTMessage Msg)=[ID_FIRST + IDSALIDA]
{
 SendMessage(Parent->HWindow, WM_COMMAND, CM_SALIR, NULL);
}

virtual void IDsimvisual(RTMessage Msg)=[ID_FIRST + IDVISUAL]
{
 SendMessage(Parent->HWindow, WM_COMMAND, CM_VEN, NULL);
}

virtual void IDtpdu(RTMessage Msg)=[ID_FIRST + IDTPDU]
{
 SendMessage(Parent->HWindow, WM_COMMAND, CM_TPDU, NULL);
 EnableWindow(Barra->PBotontpdu->HWindow,FALSE);
}

virtual void IDstruct(RTMessage Msg)=[ID_FIRST + IDSTRUCT]
{
 SendMessage(Parent->HWindow, WM_COMMAND, CM_ESTRUCTURA, NULL);
```

```
  EnableWindow(Barra->PBotonstruct->HWindow,FALSE);
  }

  virtual void IDordenar(RTMessage Msg)=[ID_FIRST + IDORDENAR]
  {
   SendMessage(Parent->HWindow, WM_COMMAND, CM_ORDENAR, NULL);
  }

  void Mostrarmensaje(char* mensaje);

  virtual void WMControlColor(RTMessage Msg)
   =[WM_FIRST+WM_CTLCOLOR];

protected:

  virtual void GetWindowClass(WNDCLASS& ClaseVentana);
  virtual LPSTR GetClassName(){ return "TVentanaBarra";}
  PTStatic lineaestado;□
  HBRUSH brush;
  COLORREF color;
  int x;
};
//Fin de la definición de la clase TVentanaBarra

//**************************************************************************
//Clase Cdialogos-Definición de la clase Cdialogos
//
//Descripción:
//Clase que representa el cuadro de diálogo Origen-Destino con el que se da
//un nombre al transmisor y al receptor.
//
//Parámetros de creación y notas:
//Crea los controles de edición origen y destino y transfiere a una estructura
//el contenido de éstos según el mecanismo de transferencia a través de un
//buffer.
//
//Comportamiento del destructor: Ninguno
//
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//**************************************************************************
class Cdialogos: public TDialog
{
 public:

  Cdialogos(PTWindowsObject AParent,LPSTR AName);
  virtual void Botonayuda(RTMessage Msg)=[ID_FIRST+IDHELP];
```

```
};
//Fin de la declaracion de la clase Cdialogos

//*************************************************************************
//Clase Velocidad-Definición de la clase Velocidad
//
//Descripción:
//Clase que representa el cuadro de diálogo Velocidad y con el que se modifica
//la rapidez de la simulación.
//
//Parámetros de creación y notas:
//Ninguna.
//
//Comportamiento del destructor: Ninguno
//
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//*************************************************************************
class Velocidad: public TDialog
{
 public:

  Velocidad(PTWindowsObject AParent,LPSTR AName);

  virtual void WMVScroll(RTMessage Msg)
  =[ WM_FIRST + WM_VSCROLL];

  virtual void SetupWindow();
  virtual void Ok(RTMessage Msg);
  virtual void Botonayuda(RTMessage Msg)=[ID_FIRST+IDHELP];
  int Pos;
 };
//Fin de la declaracion de la clase Velocidad

//*************************************************************************
//Clase Cdialogos2-Definición de la clase Cdialogos2
//
//Descripción:
//Clase que representa el cuadro de diálogo Clase de protocolo y con el cual
//se selecciona la clase que deseamos.
//
//Parámetros de creación y notas:
//Crea los controles tipo caja de grupo y radiobotones que conforman el cuadro
//y transfiere su contenido inicial de un buffer.
//
//Comportamiento del destructor: Ninguno
//
```

```
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//************************************************************************
class Cdialogos2: public TDialog
{
 public:

  PTBGroupBox Clasegrupo,Primitivasgrupo,Protocologrupo;

  Cdialogos2(PTWindowsObject AParent,LPSTR AName);
  virtual void Botonayuda(RTMessage Msg)=[ID_FIRST+IDHELP];
 };
//Fin de la declaracion de la clase Cdialogos2

//************************************************************************
//Clase Cdialogos3-Definición de la clase Cdialogos3
//
//Descripción:
//Clase que representa el cuadro de diálogo Información del nivel superior a
//través del cual se introduce los datos que simulan los que ha introducido
//el usuario más lo que han añadido las capas superiores.
//
//Parámetros de creación y notas:
//Crea un control de edición y transfiere su contenido inicial de un buffer.
//
//Comportamiento del destructor: Ninguno
//
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//************************************************************************
class Cdialogos3: public TDialog
{
 public:

  Cdialogos3(PTWindowsObject AParent,LPSTR AName);
  virtual void Botonayuda(RTMessage Msg)=[ID_FIRST+IDHELP];
 };
//Fin de la declaracion de la clase Cdialogos3

//************************************************************************
//Clase Cdialogos4-Definición de la clase Cdialogos4
//
//Descripción:
//Clase que representa el cuadro de diálogo Calidad del servicio donde se
//pueden modificar los parámetros QOS que configuran por defecto la capa de
```

```
//transporte.
//
//Parámetros de creación y notas:
//Crea los controles de la caja y transfiere su contenido inicial de un buffer.
//
//Comportamiento del destructor: Ninguno
//
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//*************************************************************************
class Cdialogos4: public TDialog
{
 PTBGroupBox grupolong;□
 public:

  Cdialogos4(PTWindowsObject AParent,LPSTR AName);
  virtual void Botonayuda(RTMessage Msg)=[ID_FIRST+IDHELP];
 };
//Fin de la declaracion de la clase Cdialogos4

//*************************************************************************
//Clase Cdialogos5-Definición de la clase Cdialogos5
//
//Descripción:
//Clase que representa el cuadro de diálogo Simular caída de la red.
//
//Parámetros de creación y notas:
//Crea los controles y transfiere su contenido inicial de un buffer global.
//
//Comportamiento del destructor: Ninguno
//
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//*************************************************************************
class Cdialogos5: public TDialog
{
 public:

  Cdialogos5(PTWindowsObject AParent,LPSTR AName);
 };
//Fin de la declaracion de la clase Cdialogos5

//*************************************************************************
//Clase Cdialogos6-Definición de la clase Cdialogos6
//
```

```
//Descripción:
//Clase que representa el cuadro de diálogo Registro de datos y errores y con
//el que se tiene un referencia de los errores que se ha cometido y los datos
//que se ha tx y rx.
//
//Parámetros de creación y notas:
//Carga los bitmaps que simulan separadores de una carpeta y crea los
// controles tipo cajas de lista que componen el cuadro.
//
//Comportamiento del destructor: Borra los bitmaps y las listas de memoria.
//
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//***************************************************************************
class Cdialogos6: public TDialog
{
 public:

  Cdialogos6(PTWindowsObject AParent,LPSTR AName);

  ~Cdialogos6();

  virtual void SetupWindow();

  virtual void WMLButtonDown(RTMessage Msg)
    =[WM_FIRST+WM_LBUTTONDOWN];

  virtual void Botonayuda(RTMessage Msg)=[ID_FIRST+IDHELP];

  HBITMAP hbmpMyBitmap,hbmpMyBitmap2;
  PTListBox Lista_datostx,Lista_datosrx,Lista_errores;
  PTStatic texto_datostx,texto_datosrx,texto_errores;
 };
//Fin de la declaracion de la clase Cdialogos6

//***************************************************************************
//Clase Cdialogos7-Definición de la clase Cdialogos7
//
//Descripción:
//Clase que representa el cuadro de diálogo Modo de funcionamiento y que
//permite conmutar el modo de trabajo de la simulación entre continuo y paso
//a paso en la opción Simulación visual del programa.
//
//Parámetros de creación y notas:
//Crea los controles del cuadro edición y transfiere su contenido inicial de
// un buffer.
//
```

```
//Comportamiento del destructor: Ninguno
//
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//***************************************************************************
class Cdialogos7: public TDialog
{
 PTBGroupBox grupomodo;
 public:

 Cdialogos7(PTWindowsObject AParent,LPSTR AName);
 virtual void Botonayuda(RTMessage Msg)=[ID_FIRST+IDHELP];
 };
//Fin de la declaracion de la clase Cdialogos7

//***************************************************************************
//Clase Cdialogos8-Definición de la clase Cdialogos8
//
//Descripción:
//Clase que representa cuadros de diálogo con mensajes de advertencia o
//preguntas.
//
//Parámetros de creación y notas:
//Ninguno.
//
//Comportamiento del destructor: Ninguno
//
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//***************************************************************************
class Cdialogos8: public TDialog
{
 public:

 Cdialogos8(PTWindowsObject AParent,LPSTR AName);
 virtual void BotonNo(RTMessage Msg)=[ID_FIRST+IDNO];
 };
//Fin de la declaracion de la clase Cdialogos8

//***************************************************************************
//Clase Cdialogos9-Definición de la clase Cdialogos9
//
//Descripción:
//Clase que representa el cuadro de diálogo Acerca de en el que se obtiene
//información del programa y del autor.
```

```
//
//Parámetros de creación y notas:
//Inicializa variables y crea un timer.
//
//Comportamiento del destructor: Elimina el timer.
//
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//*************************************************************************
class Cdialogos9: public TDialog
{
 public:

  Cdialogos9(PTWindowsObject AParent,LPSTR AName);
  virtual void SetupWindow();
  virtual void WMTimer(RTMessage Msg)
    =[WM_FIRST+WM_TIMER];

  ~Cdialogos9();

  int contador;
  BOOL descendente;
 };
//Fin de la declaracion de la clase Cdialogos9

//*************************************************************************
//Clase TPresentacion-Definición de la clase TPresentacion
//
//Descripción:
//Clase que representa el cuadro de diálogo no modal que aparece al ejecutar
//la aplicación.
//
//Parámetros de creación y notas:
//Ninguna.
//
//Comportamiento del destructor: Ninguno
//
//Clase padre: TDialog
//
//Clases hijas : Ninguna
//
//*************************************************************************
class TPresentacion: public TDialog
{
 public:

  TPresentacion(PTWindowsObject AParent):TDialog(AParent,"presentacion"){};
```

```
};
//Fin de la declaracion de la clase Presentacion


//*********************************************************************
//COMIENZA LA DECLARACIÓN DE LAS FUNCIONES MIEMBRO DE TODAS //LAS CLASES
//*********************************************************************

//***********************************************************
//METODOS DE LA CLASE CDIALOGOS
//***********************************************************
Cdialogos::Cdialogos(PTWindowsObject AParent,LPSTR AName):
                        TDialog(AParent,AName)
{
 new TEdit(this,IDC_TEXTO,15);
 new TEdit(this,IDC_DESTINO,15);
 if (todasactivas)
  TransferBuffer= &(((Simvisual*)Parent)->b);
 else
  TransferBuffer= &(((Ventanahija1*)Parent)->a);
}

void Cdialogos::Botonayuda(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Cuadro de diálogo
Origen-Destino");
}
//***********************************************************
//FIN DE LOS METODOS DE LA CLASE CDIALOGOS
//***********************************************************

//***********************************************************
//METODOS DE LA CLASE VELOCIDAD
//***********************************************************

 Velocidad::Velocidad(PTWindowsObject AParent,LPSTR AName):
                        TDialog(AParent,AName)
 {}

 void Velocidad::WMVScroll(RTMessage Msg)
 {
  HWND Scroll;
  const int PageStep = 30;

  Scroll = (HWND)Msg.LP.Hi;
  Pos = GetScrollPos( Scroll, SB_CTL );
  switch ( Msg.WParam )
  {
            case SB_LINEUP:
```

```
                Pos--;
                break;
        case SB_LINEDOWN:
                Pos++;
                break;
        case SB_PAGEUP:
                Pos -= PageStep;
                break;
        case SB_PAGEDOWN:
                Pos += PageStep;
                break;
        case SB_THUMBPOSITION:
        case SB_THUMBTRACK:
                Pos = Msg.LP.Lo;
                break;
 }
 SetScrollPos( Scroll, SB_CTL, Pos, TRUE );
}

void Velocidad::SetupWindow()
{
 TWindowsObject::SetupWindow();
 SetScrollRange(GetDlgItem(HWindow,IDC_BARRA),SB_CTL,100,400,FALSE);
 SetScrollPos(GetDlgItem(HWindow,IDC_BARRA),SB_CTL,maquinas[0]->retardo,TRUE);
};

void Velocidad::Ok(RTMessage Msg)
{
 if (vh1activas==TRUE)
  maquinas[0]->retardo=Pos;
 else
 {
 if (todasactivas)
 {
  arq[0]->retardo=Pos;
  maquinas[0]->retardo=Pos;
 }
}
TDialog::Ok(Msg);
}

void Velocidad::Botonayuda(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Cuadro de diálogo
Velocidad");
}
//*********************************************************
//FIN DE LOS METODOS DE LA CLASE VELOCIDAD
//*********************************************************
```

```
//*********************************************************
//METODOS DE LA CLASE CDIALOGOS2
//*********************************************************

Cdialogos2::Cdialogos2(PTWindowsObject AParent,LPSTR AName):
                        TDialog(AParent,AName)
{
 Clasegrupo=new TBGroupBox(this,IDC_GRUPOCLASE);
 new TRadioButton(this,IDC0,Clasegrupo);
 new TRadioButton(this,IDC1,Clasegrupo);
 new TRadioButton(this,IDC_2,Clasegrupo);
 new TRadioButton(this,IDC_3,Clasegrupo);
 new TRadioButton(this,IDC_4,Clasegrupo);
 Protocologrupo=new TBGroupBox(this,IDC_GRUPOPROTOCOLO);
 new TRadioButton(this,IDC_CV,Protocologrupo);
 new TRadioButton(this,IDC_DATAGRAMA,Protocologrupo);
 TransferBuffer= &(((Ventanahija1*)Parent)->c);
}

void Cdialogos2::Botonayuda(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Cuadro de diálogo
Clase de protocolo");
}
//*********************************************************
//FIN DE LOS METODOS DE LA CLASE CDIALOGOS2
//*********************************************************

//*********************************************************
//METODOS DE LA CLASE CDIALOGOS3
//*********************************************************

Cdialogos3::Cdialogos3(PTWindowsObject AParent,LPSTR AName):
                        TDialog(AParent,AName)
{
 new TEdit(this,IDC_INFO,255);
 if (vh1activas) //si no se pone puede dar problemas al estar en Simvisual
   TransferBuffer= &(((Ventanahija1*)Parent)->msg_sesion);
}

void Cdialogos3::Botonayuda(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Cuadro de diálogo
Información del nivel superior");
}
//*********************************************************
//FIN DE LOS METODOS DE LA CLASE CDIALOGOS3
//*********************************************************
```

```
//*********************************************************
//METODOS DE LA CLASE CDIALOGOS4
//*********************************************************

Cdialogos4::Cdialogos4(PTWindowsObject AParent,LPSTR AName):
                        TDialog(AParent,AName)
{
 new TBCheckBox(this,IDC_PRIORIDAD,NULL);
 grupolong=new TBGroupBox(this,IDC_GRUPOLONGITUD);
 new TRadioButton(this,IDC_256,grupolong);
 new TRadioButton(this,IDC_128,grupolong);

 new TEdit(this,IDC_RET1,5);
 new TEdit(this,IDC_RET2,5);
 new TEdit(this,IDC_RET3,5);
 TransferBuffer= &((Ventanahija1*)Parent)->calidad;
}

void Cdialogos4::Botonayuda(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Cuadro de diálogo
Calidad del servicio");
}
//*********************************************************
//FIN DE LOS METODOS DE LA CLASE CDIALOGOS4
//*********************************************************
//*********************************************************
//METODOS DE LA CLASE CDIALOGOS5
//*********************************************************

Cdialogos5::Cdialogos5(PTWindowsObject AParent,LPSTR AName):
                        TDialog(AParent,AName)
{
 new TBCheckBox(this,IDC_CONGESTION,NULL);
 new TBCheckBox(this,IDC_FALLOELEC,NULL);
 new TBCheckBox(this,IDC_BLOQUEADO,NULL);
 TransferBuffer= &red;
}
//*********************************************************
//FIN DE LOS METODOS DE LA CLASE CDIALOGOS5
//*********************************************************

//*********************************************************
//METODOS DE LA CLASE CDIALOGOS6
//*********************************************************

Cdialogos6::Cdialogos6(PTWindowsObject AParent,LPSTR AName):
                        TDialog(AParent,AName)
```

```
{
 Lista_datostx=new TListBox(this,ID_LISTADATOSTX,NULL);
 Lista_datosrx=new TListBox(this,ID_LISTADATOSRX,NULL);
 Lista_errores=new TListBox(this,ID_LISTAERRORES,NULL);
 texto_datostx=new TStatic(this,ID_TEXTTX,30);
 texto_datosrx=new TStatic(this,ID_TEXTRX,30);
 texto_errores=new TStatic(this,ID_TEXTERROR,50);
 hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "SEPARADOR");
 hbmpMyBitmap2 = LoadBitmap(GetApplication()->hInstance, "SEPARADOR2");
}

Cdialogos6::~Cdialogos6()
{
 DeleteObject(hbmpMyBitmap);
 DeleteObject(hbmpMyBitmap2);
 delete Lista_datostx;
 delete Lista_datosrx;
 delete Lista_errores;
}

void Cdialogos6::SetupWindow()
{
 char cadena_error[150];
 nodos *aux;

 TDialog::SetupWindow();

 if (maquinas[0]->ventanaorigen!=NULL)
 {
  switch (maquinas[0]->ventanaorigen->datos.est)
  {
   case Inactivo: Lista_datostx->AddString("No se ha transmitido nada.");
                  break;
   case conexion_saliente: Lista_datostx->AddString("H.T.#T_C.r");
                           break;
   case transferencia_datos: Lista_datostx->AddString("H.T.#T_C.r");
                             Lista_datostx->AddString("H.T.#T_C.c");
                             aux=maquinas[0]->ventanaorigen->almacen;
                             while ((aux!=NULL) && (aux!=maquinas[0]->ventanaorigen->temp))
                             {
                              strcpy(cadena_error,"H.T.#T_D.r»»");
                     strcat(cadena_error,aux->parte);
                              Lista_datostx->AddString(cadena_error);
                              aux=aux->prox;
                             }
                             break;
  }

 switch (maquinas[0]->ventanadestino->datos.est)
```

```
 {
  case Inactivo: Lista_datosrx->AddString("No se ha recibido nada.");
                 break;
  case conexion_entrante: Lista_datosrx->AddString("H.T.#T_C.i");
                          break;
  case transferencia_datos: Lista_datosrx->AddString("H.T.#T_C.i");
                            Lista_datosrx->AddString("H.T.#T_C.c");
                            aux=maquinas[0]->ventanadestino->almacen;
                            while (aux!=NULL)
                            {
                             strcpy(cadena_error,"H.T.#T_D.i««");
                             strcat(cadena_error,aux->parte);
                             Lista_datosrx->AddString(cadena_error);
                    aux=aux->prox;
                    }
                            break;
 }
 }
 for (int k=0;k<14;k++)
 {
  if ((maquinas[0]->array_errores[k]) || (maquinas[1]->array_errores[k]))
  {□
   LoadString(GetApplication()->hInstance,k,cadena_error,250);
   Lista_errores->AddString(cadena_error);
   maquinas[0]->array_errores[k]=FALSE;
   maquinas[1]->array_errores[k]=FALSE;
  }
 }
}

void Cdialogos6::Botonayuda(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Cuadro Registro de errores y datos");
}

void Cdialogos6::WMLButtonDown(RTMessage Msg)
{
 POINT Punto;
 RECT rect_errores,rect_datos;

 SetRect(&rect_errores,214,4,430,24);
 SetRect(&rect_datos,4,4,217,26);
 Punto.x=Msg.LP.Lo;
 Punto.y=Msg.LP.Hi;
 if (PtInRect(&rect_errores,Punto))
 {
  ShowWindow(Lista_datostx->HWindow,SW_HIDE);
  ShowWindow(Lista_datosrx->HWindow,SW_HIDE);
  ShowWindow(texto_datostx->HWindow,SW_HIDE);
```

```
 ShowWindow(texto_datosrx->HWindow,SW_HIDE);
 ShowWindow(Lista_errores->HWindow,SW_SHOW);
 ShowWindow(texto_errores->HWindow,SW_SHOW);
 Dibujarbitmap(hbmpMyBitmap,HWindow,3,2,TRUE);
 }
 else
 {
 if (PtInRect(&rect_datos,Punto))
 {
  ShowWindow(Lista_datostx->HWindow,SW_SHOW);
  ShowWindow(Lista_datosrx->HWindow,SW_SHOW);
  ShowWindow(texto_datostx->HWindow,SW_SHOW);
  ShowWindow(texto_datosrx->HWindow,SW_SHOW);
  ShowWindow(texto_errores->HWindow,SW_HIDE);
  ShowWindow(Lista_errores->HWindow,SW_HIDE);
  Dibujarbitmap(hbmpMyBitmap2,HWindow,4,2,TRUE);
 }
 }
}
//*********************************************************
//FIN DE LOS METODOS DE LA CLASE CDIALOGOS6
//*********************************************************

//*********************************************************
//METODOS DE LA CLASE CDIALOGOS7
//*********************************************************

Cdialogos7::Cdialogos7(PTWindowsObject AParent,LPSTR AName):
                        TDialog(AParent,AName)
{
 grupomodo=new TBGroupBox(this,IDC_GRUPOMODO);
 new TRadioButton(this,IDC_CONTINUO,grupomodo);
 new TRadioButton(this,IDC_PASOAPASO,grupomodo);
 TransferBuffer= &simulacion;
 }

void Cdialogos7::Botonayuda(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Cuadro de diálogo Modo de simulación");
}

//*********************************************************
//FIN DE LOS METODOS DE LA CLASE CDIALOGOS7
//*********************************************************
//*********************************************************
//METODOS DE LA CLASE CDIALOGOS8
//*********************************************************

Cdialogos8::Cdialogos8(PTWindowsObject AParent,LPSTR AName):
```

```
                    TDialog(AParent,AName)
{}

void Cdialogos8::BotonNo(RTMessage)
{
 CloseWindow();
}

//*********************************************************
//FIN DE LOS METODOS DE LA CLASE CDIALOGOS8
//*********************************************************
//*********************************************************
//METODOS DE LA CLASE CDIALOGOS9
//*********************************************************

Cdialogos9::Cdialogos9(PTWindowsObject AParent,LPSTR AName):
                    TDialog(AParent,AName)
{
 contador=0;
 descendente=TRUE;
}

void Cdialogos9::SetupWindow()
{
 TDialog::SetupWindow();
 SetTimer(HWindow,2, 1000,NULL);
 };

void Cdialogos9::WMTimer(RTMessage)
{
 int linea;

 linea= (int) SendDlgItemMsg(ID_MITEXTO,EM_GETLINECOUNT,0,0);
 if (descendente)
 {
  SendDlgItemMsg(ID_MITEXTO,EM_LINESCROLL,0,MAKELONG(1,0));
  contador++;
 }
 else
 {
  SendDlgItemMsg(ID_MITEXTO,EM_LINESCROLL,0,MAKELONG(-1,0));
  contador--;
 }
 if (contador==linea)
   descendente=FALSE;
 if (contador==0)
   descendente=TRUE;
}
```

```
Cdialogos9::~Cdialogos9()
{
 KillTimer(HWindow,2);
}
//*********************************************************
//FIN DE LOS METODOS DE LA CLASE CDIALOGOS9
//*********************************************************

//*********************************************************************
//METODOS DE LA CLASE TVENTANA_FLOTANTE
//*********************************************************************
void TVentana_flotante::GetWindowClass(WNDCLASS& ClaseVentana)
{
  TWindow::GetWindowClass(ClaseVentana);
  ClaseVentana.hbrBackground = CreateSolidBrush(RGB(255,255,0));
  ClaseVentana.lpszClassName = "TVentana_flotante";
}


TVentana_flotante::TVentana_flotante (PTWindowsObject AParent, LPSTR ATitle)
            :TWindow(AParent, ATitle)
{
 LOGFONT logfont;

 Attr.Style |=  WS_CHILD | WS_BORDER | WS_CLIPSIBLINGS;

 Attr.X=NULL;
 Attr.Y= NULL;
 Attr.W=NULL;
 Attr.H=18;
 logfont.lfHeight=15;
 logfont.lfWidth=9;
 logfont.lfUnderline=0;
 logfont.lfStrikeOut=0;
 logfont.lfWeight=FW_BOLD;
 lstrcpy(logfont.lfFaceName,"Helv");
 hHelv=CreateFontIndirect(&logfont);
}

TVentana_flotante::~TVentana_flotante()
{
 DeleteObject(hHelv);
}

void TVentana_flotante::Paint(HDC DC,PAINTSTRUCT& )
{
 HFONT hSystem=(HFONT)SelectObject(DC,hHelv);
 SetBkColor(DC, RGB(255, 255, 0));
 switch(Boton){
```

```
            case 1: TextOut(DC, 3, 0, "Salir", 5);
                                    break;
            case 2: TextOut(DC, 3, 0, "Ayuda", 5);
                                    break;
            case 3: TextOut(DC, 3, 0, "Simulación interactiva", 22);
                                    break;
            case 4: TextOut(DC, 3, 0, "Tutorial RM-OSI", 15);
                                    break;
            case 5: TextOut(DC, 3, 0, "Nuevo", 5);
                                    break;
            case 6: TextOut(DC, 3, 0, "Simulación visual", 17);
                                    break;
            case 7: TextOut(DC, 3, 0, "TPDU", 4);
                                    break;
            case 8: TextOut(DC, 3, 0, "Estructura de la TPDU", 21);
                                    break;
            case 9: TextOut(DC, 3, 0, "Ordenar ventanas", 16);
                                    break;
            default : ;
                                     break;
  }
 SelectObject(DC,hSystem);
}
//*************************************************************
// FIN DE LOS METODOS DE LA CLASE TVENTANA_FLOTANTE
//*************************************************************

//*************************************************************************
// METODOS DE LA CLASE TVENTANABARRA
//*************************************************************************
void TVentanaBarra::GetWindowClass(WNDCLASS& ClaseVentana)
{
  TWindow::GetWindowClass(ClaseVentana);
  ClaseVentana.hbrBackground = CreateSolidBrush(RGB(128,128,128));
  ClaseVentana.lpszClassName = "TVentanaBarra";
 }

TVentanaBarra::TVentanaBarra (PTWindowsObject AParent, LPSTR ATitle,int k)
                    :TWindow(AParent, ATitle)
{

  Attr.Style |= ((todasactivas) ? WS_CHILD : WS_POPUP) | WS_BORDER |
WS_CLIPSIBLINGS;
  Attr.X=((todasactivas) ? 0 : izda+5);
  Attr.Y=((todasactivas) ? 0 : arriba+alto_barra); //42
  Attr.H=((todasactivas) ? 90 : 29);
  Attr.W=((todasactivas) ? 29 : ancho-9);
  switch (k)
  {
```

```
 case 0:    PBotonarquitecturas = new TBButton(this, IDSOLOARQ, NULL, 60,0, NULL,NULL, FALSE);
            PBotonmaquinas = new TBButton(this, IDSOLOMAQ, NULL,30,0, NULL, NULL,FALSE);
            PBotonvisual = new TBButton(this, IDVISUAL, NULL, 0, 0, NULL, NULL, FALSE);
            PBotonAyuda = new TBButton(this, IDAYUDA, NULL,120,0 , NULL, NULL,FALSE);
            PBotonsalida = new TBButton(this, IDSALIDA, NULL, 90, 0, NULL, NULL, FALSE);
            lineaestado=new TStatic(this,-1,"",165,5,450,20,200);
            break;
 case 1: PBotonnuevo = new TBButton(this, IDNUEVO, NULL, 0, 0, NULL, NULL, FALSE);
        PBotonAyuda = new TBButton(this, IDAYUDA, NULL,((todasactivas) ? 0 :
60),((todasactivas) ? 60 : 0) , NULL, NULL, FALSE);
            PBotonsalida = new TBButton(this, IDSALIDA, NULL, ((todasactivas) ? 0 :
30),((todasactivas) ? 30 : 0), NULL, NULL, FALSE);
            lineaestado=new TStatic(this,-1,"",100,5,ancho,20,200);
            break;
 case 2: PBotonnuevo = new TBButton(this, IDNUEVO, NULL, 0, 0, NULL, NULL, FALSE);
        PBotonAyuda = new TBButton(this, IDAYUDA, NULL,60,0 , NULL, NULL, FALSE);
           PBotonsalida = new TBButton(this, IDSALIDA, NULL, 30,0, NULL, NULL, FALSE);
            PBotontpdu = new TBButton(this, IDTPDU, NULL, 90,0, NULL, NULL, FALSE);
            PBotonstruct = new TBButton(this, IDSTRUCT, NULL, 120,0, NULL, NULL,FALSE);
            PBotonordenar = new TBButton(this, IDORDENAR, NULL, 150,0, NULL, NULL,FALSE);
            lineaestado=new TStatic(this,-1,"",195,5,ancho,20,200);
            break;
 }
 x=k;
 color=RGB(128,128,128);
 brush=CreateSolidBrush(color);
 }

TVentanaBarra::~TVentanaBarra()
{
 switch(x)
 {
 case 0: delete PBotonarquitecturas;
            delete PBotonmaquinas;
            delete PBotonvisual;
            delete PBotonAyuda;
            delete PBotonsalida;
      break;
 case 1: delete PBotonnuevo;
        delete PBotonAyuda;
            delete PBotonsalida;
            break;
 case 2: delete PBotonnuevo;
        delete PBotonAyuda;
            delete PBotonsalida;
            delete PBotontpdu;
        delete PBotonstruct;
            delete PBotonordenar;
            break;
```

```
}
delete lineaestado;
DeleteObject(brush);
}

void TVentanaBarra::Mostrarmensaje(char* mensaje)
{
lineaestado->SetText(mensaje);
}

void TVentanaBarra::WMControlColor(RTMessage Msg)
{
SetTextColor(HDC(Msg.WParam),RGB(255,255,255));
SetBkColor(HDC(Msg.WParam),RGB(128,128,128));
Msg.Result=(long)brush;
}
//*************************************************************
// FIN DE LOS METODOS DE LA CLASE TVENTANABARRA
//*************************************************************
//*************************************************************
// METODOS DE LA CLASE VENTANAFONDO
//*************************************************************

Ventanafondo::Ventanafondo(PTWindowsObject AParent,
                           LPSTR ATitle): TWindow(AParent,ATitle)
{
Attr.Style=WS_VISIBLE | WS_CHILD | WS_CLIPCHILDREN | WS_CLIPSIBLINGS;
Attr.X=0;
Attr.Y=30;
Attr.W=ancho;
Attr.H=alto-alto_barra;
}

void Ventanafondo::GetWindowClass( WNDCLASS _FAR & h)
{
TWindow::GetWindowClass(h);
h.hbrBackground=(CreateSolidBrush(RGB(191,191,191)));
}
/*************************************************************
// FIN DE LOS METODOS DE LA CLASE VENTANAFONDO
//*************************************************************
//*************************************************************
// METODOS DE LA CLASE VENTANAHIJA1
//*************************************************************

Ventanahija1::Ventanahija1(PTWindowsObject AParent,
                           LPSTR ATitle,BYTE WinNum,BOOL size):
        TWindow(AParent,ATitle)□
{
```

```
if (size==FALSE)
{
 Attr.Style |=WS_CHILD | WS_CAPTION | WS_CLIPSIBLINGS | WS_BORDER;
 movibles=FALSE;
 switch (WinNum) {
        case 0:
                Attr.X=80;
                Attr.Y=2*abs((alto-462)/3)+240;
                Attr.H=147+alto_titulo;
                Attr.W=189;
                break;
        case 1:
                Attr.X=ancho-220;
                Attr.Y=2*abs((alto-462)/3)+240;
                Attr.H=147+alto_titulo;
                Attr.W=189;
                break;
          }
puntos[0]=96; puntos[1]=25; puntos[2]=37; puntos[3]=66;
puntos[4]=96; puntos[5]=105; puntos[6]=141; puntos[7]=67;
puntos[8]=68; puntos[9]=72; puntos[10]=71; puntos[11]=43;
puntos[12]=123; puntos[13]=72; puntos[14]=122; puntos[15]=43;
puntos[16]=69; puntos[17]=116; puntos[18]=120; puntos[19]=116;
puntos[20]=96; puntos[21]=67;□
hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "EST64K");
datos.est=Inactivo;
}
else{
 Attr.Style |=WS_CHILD | WS_CAPTION | WS_THICKFRAME
        | WS_CLIPSIBLINGS | WS_SYSMENU;
 movibles=TRUE;
   switch (WinNum) {
        case 0:
                Attr.X=40;
                Attr.Y=18;
                Attr.H=178+alto_titulo;
                Attr.W=228;
                break;
        case 1:
                Attr.X=ancho-268;
                Attr.Y=18;
                Attr.H=178+alto_titulo;
                Attr.W=228;
                break;
          }
puntos[0]=103; puntos[1]=51; puntos[2]=46; puntos[3]=80;
puntos[4]=103; puntos[5]=123; puntos[6]=164; puntos[7]=80;
puntos[8]=73; puntos[9]=78; puntos[10]=75; puntos[11]=51;
puntos[12]=147; puntos[13]=64; puntos[14]=123; puntos[15]=68;
```

```
 puntos[16]=76; puntos[17]=129; puntos[18]=130; puntos[19]=129;
 puntos[20]=106; puntos[21]=76;
 hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "est7");
 calidad.datos_expeditos=FALSE;
 calidad.l256=TRUE;
 calidad.l128=FALSE;
 strcpy(calidad.retardo1,"1500");
 strcpy(calidad.retardo2,"2000");
 strcpy(calidad.retardo3,"100");
 for (int i=0;i<5;i++)
  c.cl[i]=FALSE;
 c.cl[1]=TRUE;
 c.tipo[0]=TRUE;
 c.tipo[1]=FALSE;
 strcpy(a.origen,"OFRA");
 strcpy(a.destino,"ULPGC");
 for (i=0;i<3;i++)
  red.conges[i]=FALSE;
 cal.datos_expeditos=FALSE;
 cal.longitud[0]=calidad.l256;
 cal.longitud[1]=calidad.l128;
 cal.retardo1=1500; cal.retardo2=2000; cal.retardo3=100;
 datos.clase=1;
 datos.caida=FALSE;
 cab=NULL;
 prioridad=NULL;
 almacenados=NULL;
 almacenados2=NULL;
 ptrerror=NULL;
 almacen=NULL;
 ventanaorigen=NULL;
 ventanadestino=NULL;
 terminar=FALSE;
 continuar_tx=FALSE;
 reconectarse=FALSE;
 acelerado=FALSE;
 contadormsg=0;
 n=1;
 total_TPDUs=-1;
 for (i=0;i<14;i++)
   array_errores[i]=FALSE;
 Inicializarestructuras();
 }
SetRect(&r,1,1,100,100);
retardo=200;
strcpy(msg_sesion,"Información de los niveles superiores.");
}

int Ventanahija1::contador;
```

```
int Ventanahija1::retardo;
int Ventanahija1::parametro_reconexion;
BOOL Ventanahija1::reconectarse;
HWND Ventanahija1::wndini;
HWND Ventanahija1::wndnopicada;
PVentanahija1 Ventanahija1::ventanaorigen;
PVentanahija1 Ventanahija1::ventanadestino;

Ventanahija1::~Ventanahija1()
{
 DeleteObject(hbmpMyBitmap);
 if (vh1activas)
 {
  Borralista(&almacen);
  Borralista(&cab);
  Borralista(&prioridad);
 }
}

void Ventanahija1::Paint(HDC ,PAINTSTRUCT& )
{
 COLORREF c;

 Dibujarbitmap(hbmpMyBitmap,HWindow,0,0,TRUE);
 c=RGB(255,0,0);
 switch (datos.est){
    case Inactivo:
            Rellenar(HWindow,c,puntos[0],puntos[1]);
            break;
    case conexion_saliente:
            Rellenar(HWindow,c,puntos[2],puntos[3]);
            break;
    case conexion_entrante:
            Rellenar(HWindow,c,puntos[6],puntos[7]);
            break;
    case transferencia_datos:
            Rellenar(HWindow,c,puntos[4],puntos[5]);
            break;
  }
 }


void Ventanahija1::GetWindowClass( WNDCLASS _FAR & h)
{
 HCURSOR CursorHand;
 TWindow::GetWindowClass(h);
 h.style |=CS_NOCLOSE | CS_VREDRAW | CS_HREDRAW;
 CursorHand = LoadCursor( GetApplication()->hInstance, "mano");
 h.hCursor = CursorHand;
```

```
}

void Ventanahija1::Inicializarestructuras()
{
 contador=1;
 datos.origen=-1;  datos.destino=-1;
 datos.est=Inactivo;
 strcpy(datos.primitiva," ");□
 strcpy(datos.nombre," ");
 strcpy(datos.info,msg_sesion);
 datos.n_secuencia=0;
 datos.n_secuenciaexp=0;
 datos.ID=-1;
 datos.c=cal;
 datos.checksum=0;
 SetWindowText(HWindow,"Máquina de estados");
 if (temp!=NULL)
  temp=NULL;
 if (temp2!=NULL)
  temp2=NULL;
 time_transf=0;
 if (almacen!=NULL)
  Borralista(&almacen);
}

void Ventanahija1::Inicializarventanas(int i, int j)
{
 wndini=maquinas[i]->HWindow;
 ventanaorigen=maquinas[i];
 wndnopicada=maquinas[j]->HWindow;
 ventanadestino=maquinas[j];
 contador++;
}

void Ventanahija1::llenarinfo(LPSTR p,LPSTR t,LPSTR s,LPSTR r,LPSTR o)
{
 if (Infoactiva==TRUE)
 {
  Info->Fijarprimitiva(p);
  Info->Fijartpdu(t);
  Info->Fijarid(s);
  Info->Fijarchecksum(r);
  Info->Fijarn_secuencia(o);
  }
 }

void Ventanahija1::llenarorgdest(LPSTR p,LPSTR t)
{
 if (Infoactiva)
```

```
{
Info->Fijarorigen(p);
Info->Fijardestino(t);
}
}

void Ventanahija1::Brocha(int estado,char directiva[40],HWND hwnd1)
{
RECT rect;
COLORREF colores;
HDC hdc;

hdc=GetDC(hwnd1);
if (vh1activas)
{
SetRect(&rect,1,1,180,14);
SetTextColor(hdc,RGB(255,0,0));
SetBkMode(hdc,TRANSPARENT);
ExtTextOut(hdc,1,1,ETO_CLIPPED,&rect,directiva,strlen(directiva),NULL);
}
for (int i=1;i<7;i++)
{
if ((i%2)==0)
  colores=RGB(255,0,0);
else
  colores=RGB(255,255,255);

switch (estado){
  case 1:
          Rellenar(hwnd1,colores,puntos[2],puntos[3]);
          Rellenar(hwnd1,colores,puntos[8],puntos[9]);
   break;
  case 2:
          Rellenar(hwnd1,colores,puntos[6],puntos[7]);
          Rellenar(hwnd1,colores,puntos[12],puntos[13]);
          break;
  case 3:
          Rellenar(hwnd1,colores,puntos[4],puntos[5]);
          Rellenar(hwnd1,colores,puntos[16],puntos[17]);
          break;
  case 4:
          Rellenar(hwnd1,colores,puntos[4],puntos[5]);
          Rellenar(hwnd1,colores,puntos[18],puntos[19]);
          break;
  case 5:
          Rellenar(hwnd1,colores,puntos[0],puntos[1]);
          Rellenar(hwnd1,colores,puntos[14],puntos[15]);
          break;
  case 6:
```

```
            Rellenar(hwnd1,colores,puntos[0],puntos[1]);
            Rellenar(hwnd1,colores,puntos[10],puntos[11]);
            break;
    case 7:
            Rellenar(hwnd1,colores,puntos[4],puntos[5]);
            break;
    case 8:
            Rellenar(hwnd1,colores,puntos[0],puntos[1]);
            Rellenar(hwnd1,colores,puntos[20],puntos[21]);
            break;
    case 9:
            Rellenar(hwnd1,colores,puntos[0],puntos[1]);
            break;

  }
  delay(retardo);
 }
 ReleaseDC(hwnd1,hdc);

}

void Ventanahija1::menuflotante(POINT r)
{
 HMENU hmenu;

 hmenu=CreatePopupMenu();
 AppendMenu(hmenu,MF_ENABLED,IDM_CREQUEST,"T_CONNECT.request");
 AppendMenu(hmenu,MF_ENABLED,IDM_CINDICATION,"T_CONNECT.indication");
AppendMenu(hmenu,MF_ENABLED,IDM_CRESPONSE,"T_CONNECT.response");
AppendMenu(hmenu,MF_ENABLED,IDM_CCONFIRM,"T_CONNECT.confirm");
AppendMenu(hmenu,MF_ENABLED,IDM_DREQUEST,"T_DISCONNECT.request");
AppendMenu(hmenu,MF_ENABLED,IDM_DINDICATION,"T_DISCONNECT.indication");
AppendMenu(hmenu,MF_ENABLED,IDM_DATAREQUEST,"T_DATA.request");
AppendMenu(hmenu,MF_ENABLED,IDM_DATAINDICATION,"T_DATA.indication");
 if (calidad.datos_expeditos)
 {
AppendMenu(hmenu,MF_ENABLED,IDM_EXPDATAREQUEST,"T_EXPEDITED_DATA.request");
AppendMenu(hmenu,MF_ENABLED,IDM_EXPDATAINDICATION,"T_EXPEDITED_DATA.indication");
 }
 TrackPopupMenu(hmenu,TPM_LEFTBUTTON |
TPM_CENTERALIGN,r.x,r.y,0,HWindow,NULL);
 DestroyMenu(hmenu);
}


void Ventanahija1::WMLButtonDown(RTMessage Msg)
{
 POINT p;
 if (vh1activas)
```

```
 {
  p.x=Msg.LP.Lo;
  p.y=Msg.LP.Hi;
  ClientToScreen(HWindow,&p);
  menuflotante(p);
 }
}

void Ventanahija1::WMRButtonDown(RTMessage Msg)
{
 HMENU hmenu;
 POINT m;

 if (((maquinas[0]->HWindow)==HWindow) && (contador==1))
   Inicializarventanas(0,1);
 else
 {
  if (contador==1)
   Inicializarventanas(1,0);
 }

 hmenu=CreatePopupMenu();
 AppendMenu(hmenu,MF_ENABLED,ORG_DEST,"Origen-Destino");
 AppendMenu(hmenu,MF_SEPARATOR,-1,NULL);
AppendMenu(hmenu,MF_ENABLED,IDM_INFORMACION,"Información nivel superior");
 AppendMenu(hmenu,MF_SEPARATOR,-1,NULL);
 AppendMenu(hmenu,MF_ENABLED,IDM_CLASE,"Clase de protocolo");
 AppendMenu(hmenu,MF_SEPARATOR,-1,NULL);
 AppendMenu(hmenu,MF_ENABLED,IDM_CALIDAD,"Calidad del servicio");
 AppendMenu(hmenu,MF_SEPARATOR,-1,NULL);
 AppendMenu(hmenu,MF_ENABLED,IDM_CAIDARED,"Simular caída de la red");
 AppendMenu(hmenu,MF_SEPARATOR,-1,NULL);
 AppendMenu(hmenu,MF_ENABLED,IDM_INFOMAQ,"Información de la máquina");

 m.x=Msg.LP.Lo;
 m.y=Msg.LP.Hi;
 ClientToScreen(HWindow,&m);

 TrackPopupMenu(hmenu,TPM_LEFTBUTTON |
TPM_CENTERALIGN,m.x,m.y,0,HWindow,NULL);
 DestroyMenu(hmenu);
}

BOOL Ventanahija1::Comprobar_qos()
{
 BOOL salida=TRUE;

 if ((datos.clase<3) && (cal.datos_expeditos))
 {
```

```
 array_errores[12]=TRUE;
 salida=FALSE;
 }

 if (((datos.clase>2) && (cal.longitud[0])) || ((datos.clase<3) && (cal.longitud[1])))
 {
 array_errores[2]=TRUE;
 salida=FALSE;
 }

 if ((cal.retardo1<=0) || (cal.retardo2<=0) || (cal.retardo3<=0))
 {
 array_errores[3]=TRUE;
 salida=FALSE;
 }

 if ( ((cal.retardo1<1000) || (cal.retardo2<1000) || (cal.retardo3<100)) && (datos.clase<3))
 {
 array_errores[11]=TRUE;
 salida=FALSE;
 }

 if (datos.clase!=ventanaorigen->datos.clase)
 {
 array_errores[4]=TRUE;
 salida=FALSE;
 }

 if (cal.datos_expeditos!=ventanaorigen->cal.datos_expeditos)
 {
 array_errores[13]=TRUE;
 salida=FALSE;
 }

 if ((cal.retardo1>ventanaorigen->cal.retardo1) || (cal.retardo2>ventanaorigen-
>cal.retardo2))
 {
 array_errores[5]=TRUE;
 salida=FALSE;
 }

 if ((HWindow==wndnopicada) && ( (strcmp(a.origen,ventanaorigen->a.destino)) ||
(strcmp(a.destino,ventanaorigen->a.origen)) ) )
  {
 array_errores[6]=TRUE;
 salida=FALSE;
 }

 if (time_est-ventanaorigen->time_est>datos.c.retardo1)
```

```
 {
  array_errores[7]=TRUE;
  salida=FALSE;
 }

 if (time_transf-ventanaorigen->time_transf>datos.c.retardo2)□
 {
  array_errores[8]=TRUE;
  salida=FALSE;
 }

 return salida;
}

void Ventanahija1::Directiva1(RTMessage )
{
 strcpy(datos.primitiva,"T_CONNECT.request");
 if ((datos.origen<=0) || (datos.destino<=0) || (datos.caida))
 {
  if (datos.caida)
  {
   array_errores[1]=TRUE;
   MessageBox(HWindow,"La red ha caído.Debe fijar correctamente esta opción.","¡¡
ERROR !!",MB_OK | MB_ICONHAND);
  }
  else
  {
   array_errores[0]=TRUE;
  MessageBox(HWindow,"No hay transmisor ni receptor fijado.","¡¡ ERROR !!",MB_OK | MB_ICONHAND);
  }
 }
 else
 {
  if ((datos.est==Inactivo) && (HWindow==wndini))
  {
   llenarinfo(datos.primitiva,"CR","1"," "," ");
   if (Comprobar_qos())
   {
    time_est=GetTickCount();□
    datos.ID++;
    Brocha(1,datos.primitiva,wndini);
    datos.est=conexion_saliente;
    SendMessage(ventanaorigen->HWindow,WM_PAINT,0,0L);
    if (TPDUactiva)
     TPDU->Procesar();
    SendMessage(ventanadestino->HWindow,WM_CONNIND,0,0L);
    }
   else
   {
```

```
    llenarinfo("T_DISCONNECT.request","DR","-1"," "," ");
    Brocha(9,datos.primitiva,wndini);
    Borralista(&cab);
    Inicializarestructuras();
    SendMessage(ventanaorigen->HWindow,WM_PAINT,0,0L);
    SetWindowText(wndnopicada,"Máquina de estados");
    llenarorgdest(" "," ");
   }
  }
 else
  GetModule()->ExecDialog(new Cdialogos8(this,"DIALOG_4"));
 }
}

void Ventanahija1::Respuestaconnind(RTMessage)
{
 strcpy(datos.primitiva,"T_CONNECT.indication");

 if ((datos.est==Inactivo) && (HWindow==wndnopicada))
 {
  llenarinfo(datos.primitiva,"CC","1"," "," ");
  time_est=GetTickCount();
  if (Comprobar_qos())
  {
   datos.ID=ventanaorigen->datos.ID;
   Brocha(2,datos.primitiva,wndnopicada);
   ventanadestino->datos.est=conexion_entrante;
   SendMessage(HWindow,WM_PAINT,0,0L);
  }
  else
   {
    Brocha(9,datos.primitiva,wndnopicada);
    SendMessage(ventanaorigen->HWindow,WM_DISCREQ,0,0L);
    Inicializarestructuras();
    llenarorgdest("  ","   ");
    llenarinfo(" "," ","-1"," "," ");
    SendMessage(HWindow,WM_PAINT,0,0L);
   }
 }
}

//Respuesta cuando una maquina envia un T_DISCONNECT.request a la otra
void Ventanahija1::Respuestadiscreq(RTMessage )
{
 strcpy(datos.primitiva,"T_DISCONNECT.request");
 llenarinfo(datos.primitiva,"DR","-1"," "," ");

 switch (datos.est)
 {
```

```
 case conexion_saliente: Brocha(6,datos.primitiva,HWindow);
                                    break;
 case conexion_entrante: Brocha(5,datos.primitiva,HWindow);
                                    break;
 case transferencia_datos: Brocha(8,datos.primitiva,HWindow);
                                    break;
 }
 Borralista(&cab);
 Inicializarestructuras();
 llenarorgdest("  ","   ");
 llenarinfo(" "," ","-1"," "," ");
 SendMessage(HWindow,WM_PAINT,0,0L);
}

//Respuesta cuando el usuario pulsa un T_DISCONNECT.request
void Ventanahija1::Directiva3(RTMessage )
{
 strcpy(datos.primitiva,"T_DISCONNECT.request");
 llenarinfo(datos.primitiva,"DR","1"," "," ");

 switch (datos.est){
   case conexion_saliente:
              Brocha(6,datos.primitiva,HWindow);
                                    break;
 case conexion_entrante:
              Brocha(5,datos.primitiva,HWindow);
                                    break;
 case transferencia_datos:
                                    Brocha(8,datos.primitiva,HWindow);
                                    break;
 }
 Borralista(&cab);
 Borralista(&prioridad);
 Inicializarestructuras();
 llenarorgdest("  ","   ");
 llenarinfo(" "," ","-1"," "," ");
 SendMessage(HWindow,WM_PAINT,0,0L);

 if ((maquinas[0]->HWindow)==HWindow)
   SendMessage(maquinas[1]->HWindow,WM_DISCIND,0,0L);
 else
   SendMessage(maquinas[0]->HWindow,WM_DISCIND,0,0L);

}

void Ventanahija1::Directiva4(RTMessage)
{
strcpy(datos.primitiva,"T_DISCONNECT.indication");
```

```
 if (datos.caida)
 {
  llenarinfo(datos.primitiva,"DC","1"," "," ");
  SendMessage(wndini,WM_DISCIND,0,0L);
  SendMessage(wndnopicada,WM_DISCIND,0,0L);
  }
 else
MessageBox(HWindow,"La red se encuentra en perfecto estado.","Información",MB_OK | MB_ICONINFORMATION);
 }

 void Ventanahija1::Respuestadisconnectind(RTMessage)
 {
  strcpy(datos.primitiva,"T_DISCONNECT.indication");
  llenarinfo(datos.primitiva,"DC","1"," "," ");

  switch (datos.est){
   case conexion_saliente:
                Brocha(6,datos.primitiva,HWindow);
                                break;
   case conexion_entrante:
                Brocha(5,datos.primitiva,HWindow);
                                break;
   case transferencia_datos:
                                Brocha(8,datos.primitiva,HWindow);
                                break;
  }
  Borralista(&cab);
  Borralista(&prioridad);
  Inicializarestructuras();
  llenarorgdest("  ","   ");
  llenarinfo(" "," "," ","-1"," "," ");
  SendMessage(HWindow,WM_PAINT,0,0L);
  }

 void Ventanahija1::Directiva5(RTMessage )
 {
  char conversion[6],conversion2[3];

  strcpy(datos.primitiva,"T_DATA.request");
  if ((datos.est==transferencia_datos) && (HWindow==wndini))
  {
   if ((ventanadestino->ptrerror!=NULL) && (temp==NULL))
   {
    if (GetModule()->ExecDialog(new Cdialogos8(this,"DIALOG_3"))==IDOK)
    {
     Borralista(&cab);
     cab=temp=ventanadestino->ptrerror;
     ventanadestino->ptrerror=NULL;
     SendMessage(maquinas[0]->HWindow,WM_PAINT,0,0L);
```

```
    SendMessage(maquinas[1]->HWindow,WM_PAINT,0,0L);
   }
   else
   {
    Borralista(&ventanadestino->ptrerror);
    return;
   }
  }
  if (temp==NULL)
  {
   MessageBox(HWindow,"No existen datos del nivel superior fijados.","¡¡ ERROR
!!",MB_OK | MB_ICONHAND);
    array_errores[10]=TRUE;
    if (GetModule()->ExecDialog(new Cdialogos8(this,"MASDATOS"))==IDOK)
    {
     Borralista(&cab);
     Borralista(&ventanadestino->cab);//esto no sabes si es asi y como va a funcionar
     continuar_tx=TRUE;
     datos.n_secuencia=0;
     ventanadestino->datos.n_secuencia=0;
     SendMessage(HWindow,WM_COMMAND,IDM_INFORMACION,0);
    }
   }
  else
  {
   if (datos.caida)
    Actuarantecaida(datos.n_secuencia,datos.n_secuenciaexp);
   else
   {
    time_transf=GetTickCount();
    itoa(temp->checksum,conversion,10);
    itoa(datos.n_secuencia,conversion2,10);
    llenarinfo(datos.primitiva,"DT","1",conversion,conversion2);
    Brocha(7,datos.primitiva,wndini);
    SendMessage(HWindow,WM_PAINT,0,0L);
    if (TPDUactiva)
     TPDU->Procesar();
    Crealista(&almacen,temp->parte,temp->checksum);
    SendMessage(wndnopicada,WM_DATAIND,0,0L);
    if (temp!=NULL)
    {
     temp=temp->prox;
     datos.n_secuencia++;
    }
   }
  }
 }
else
 GetModule()->ExecDialog(new Cdialogos8(this,"DIALOG_4"));
```

```
}

void Ventanahija1::Respuestadataind(RTMessage)
{
 char conversion[6],conversion2[2];

 strcpy(datos.primitiva,"T_DATA.indication");

 if ((datos.est==transferencia_datos) && (HWindow==wndnopicada))
 {
  time_transf=GetTickCount();
  itoa(ventanaorigen->temp->checksum,conversion,10);
  itoa(datos.n_secuencia,conversion2,10);
  if (Comprobar_qos())
  {
   Brocha(7,datos.primitiva,wndnopicada);
   Crealista(&cab,ventanaorigen->temp->parte,datos.checksum);
   Crealista(&almacen,ventanaorigen->temp->parte,datos.checksum);
   if (!strcmp(ventanaorigen->temp->auxiliar," "))
     llenarinfo(datos.primitiva,"AK","1",conversion,conversion2);
   else
   {
    llenarinfo(datos.primitiva,"AK","1","ERROR",conversion2);
    array_errores[9]=TRUE;
    Crealista(&ptrerror,ventanaorigen->temp->parte,ventanaorigen->temp->checksum);
    temp=almacen;
    while (temp->prox!=NULL) temp=temp->prox;
    strcpy(temp->parte,ventanaorigen->temp->auxiliar);
   }
   SendMessage(HWindow,WM_PAINT,0,0L);
   datos.n_secuencia++;
  }
  else
  {
   llenarinfo("T_DISCONNECT.request","DR","-1"," "," ");
   Brocha(8,"T_DISCONNECT.request",wndnopicada);
   Borralista(&cab);
   Inicializarestructuras();
   SendMessage(HWindow,WM_PAINT,0,0L);
   array_errores[9]=TRUE;
   SendMessage(wndini,WM_DISCIND,0,0L);
  }
  time_transf=0;
  ventanaorigen->time_transf=0;
 }
}

void Ventanahija1::Directivaexpedita(RTMessage)
{
```

```
 char conversion[6],conversion2[3];

 strcpy(datos.primitiva,"T_EXP_DATA.request");
 if ((datos.est==transferencia_datos) && (HWindow==wndini))
 {
  if (temp2==NULL)
  {
   array_errores[10]=TRUE;
   if (GetModule()->ExecDialog(new Cdialogos8(this,"DIALOG_2"))==IDOK)
   {
    Borralista(&prioridad);
   Borralista(&ventanadestino->prioridad);
    continuar_tx=TRUE;
    acelerado=TRUE;
    datos.n_secuenciaexp=0;
    ventanadestino->datos.n_secuenciaexp=0;
    SendMessage(HWindow,WM_COMMAND,IDM_INFORMACION,NULL);
   }
  }
  else
  {
   if (datos.caida)
    Actuarantecaida(datos.n_secuencia,datos.n_secuenciaexp);
   else
   {
    time_transf=GetTickCount();
    itoa(temp2->checksum,conversion,10);
    itoa(datos.n_secuenciaexp,conversion2,10);
    llenarinfo(datos.primitiva,"ED","1",conversion,conversion2);
    Brocha(7,datos.primitiva,wndini);
    SendMessage(HWindow,WM_PAINT,0,0L);
    if (TPDUactiva)
     TPDU->Procesar();
    SendMessage(wndnopicada,WM_DATAEXPIND,0,0L);
    Crealista(&almacen,temp2->parte,temp2->checksum);
    if (temp2!=NULL)
    {
     temp2=temp2->prox;
     datos.n_secuenciaexp++;
    }
   }
  }
 }
 else
  GetModule()->ExecDialog(new Cdialogos8(this,"DIALOG_4"));
}

void Ventanahija1::Respuestadataexpind(RTMessage)
{
```

```
char conversion[6],conversion2[2];

strcpy(datos.primitiva,"T_EXP_DATA.indication");
if ((datos.est==transferencia_datos) && (HWindow==wndnopicada))
{
 time_transf=GetTickCount();
 itoa(ventanaorigen->temp2->checksum,conversion,10);
 itoa(datos.n_secuenciaexp,conversion2,10);
 if (Comprobar_qos())
 {
  Brocha(7,datos.primitiva,wndnopicada);
  Crealista(&prioridad,ventanaorigen->temp2->parte,datos.checksum);
  Crealista(&almacen,ventanaorigen->temp2->parte,datos.checksum);
  if (!strcmp(ventanaorigen->temp2->auxiliar," "))
    llenarinfo(datos.primitiva,"EA","1",conversion,conversion2);
  else
  {
   llenarinfo(datos.primitiva,"EA","1","ERROR",conversion2);
   array_errores[9]=TRUE;
  }
  SendMessage(HWindow,WM_PAINT,0,0L);
  datos.n_secuenciaexp++;
 }
 else
 {
  llenarinfo("T_DISCONNECT.request","DR","-1"," "," ");
  Brocha(8,"T_DISCONNECT.request",wndnopicada);
  Borralista(&prioridad);□
  Inicializarestructuras();
  SendMessage(HWindow,WM_PAINT,0,0L);
  array_errores[9]=TRUE;
  SendMessage(wndini,WM_DISCIND,0,0L);
 }
 time_transf=0;
 ventanaorigen->time_transf=0;
}
}

void Ventanahija1::Directiva7(RTMessage )
{
strcpy(datos.primitiva,"T_CONNECT.response");

if ((datos.est==conexion_entrante) && (HWindow==wndnopicada) && (!datos.caida))
{
 if (TPDUactiva)
    TPDU->Procesar();
 if (Comprobar_qos())
 {
  llenarinfo(datos.primitiva,"CR","1"," "," ");
```

```
   Brocha(4,datos.primitiva,wndnopicada);
   datos.est=transferencia_datos;
   SendMessage(HWindow,WM_PAINT,0,0L);
   SendMessage(wndini,WM_CCONFIRM,0,0L);
  }
  else
  {
   SendMessage(HWindow,WM_DISCIND,0,0L);
   SendMessage(wndini,WM_DISCIND,0,0L);
  }
 }
 else
 {
  if ((datos.caida) && (HWindow==wndnopicada))
  {
   SendMessage(HWindow,WM_DISCIND,0,0L);
   SendMessage(wndini,WM_DISCIND,0,0L);
   }
  else
    GetModule()->ExecDialog(new Cdialogos8(this,"DIALOG_4"));
  }
}

void Ventanahija1::Respuestaconnectconfirm(RTMessage)
{
 strcpy(datos.primitiva,"T_CONNECT.confirm");

 if (datos.est==conexion_saliente)
 {
   llenarinfo(datos.primitiva,"CC","1"," "," ");
   Brocha(3,datos.primitiva,wndini);
   datos.est=transferencia_datos;
   SendMessage(wndini,WM_PAINT,0,0L);
  }
}

void Ventanahija1::Origendestino(RTMessage)
{
 if ((datos.est==Inactivo) && (HWindow==wndini))
 {
  if ((GetModule()->ExecDialog(new Cdialogos(this,"ORG_DES"))==IDOK) &&
(!reconectarse))
  {
   SetWindowText(maquinas[0]->wndini,a.origen);//cambia el titulo de la ventana cuyo
handle se pasa
   SetWindowText(maquinas[0]->wndnopicada,a.destino);
   strcpy(ventanadestino->a.destino,a.origen);
   strcpy(ventanadestino->a.origen,a.destino);
   strcpy(ventanaorigen->datos.nombre,a.origen);
```

```
  strcpy(ventanadestino->datos.nombre,a.destino);
  llenarorgdest(a.origen,a.destino);
  datos.origen=1000;
  datos.destino=2000;
  ventanadestino->datos.origen=2000;
  ventanadestino->datos.destino=1000;
 }
 }
}

void Ventanahija1::QOS(RTMessage)
{
 if (datos.est==Inactivo)
 {
  if (GetModule()->ExecDialog(new Cdialogos4(this,"QOS2"))==IDOK)
  {
   int m;

   cal.datos_expeditos=calidad.datos_expeditos;
   cal.longitud[0]=calidad.l256;
   cal.longitud[1]=calidad.l128;
   m=atoi(calidad.retardo1);
   cal.retardo1=m;
   m=atoi(calidad.retardo2);
   cal.retardo2=m;
   m=atoi(calidad.retardo3);
   cal.retardo3=m;
   datos.c=cal;
  }
 }
}

void Ventanahija1::Clase(RTMessage)
{
 if (datos.est==Inactivo)
 {
  if (GetModule()->ExecDialog(new Cdialogos2(this,"CLASE"))==IDOK)
  {
   int i=0;

   while (c.cl[i]==FALSE) i++;
   datos.clase=i;
  }
 }
}

void Ventanahija1::Informa(RTMessage)
{
 int tamanobloque,j,n_ascii;
```

```
 char bloque[70];
 DWORD total;

 if (((datos.est==Inactivo) && (HWindow==wndini)) || (continuar_tx))
 {
 if ((GetModule()->ExecDialog(new Cdialogos3(this,"INFORMACION"))==IDOK) &&
(!reconectarse))
 {
 strcpy(datos.info,msg_sesion);
 if (datos.clase<3)
  tamanobloque=68;
 else
  tamanobloque=39;

 if (acelerado)
  tamanobloque=16;

 while (terminar==FALSE)
 {
 total=0;
 j=0;
 strcpy(bloque," ");
 while ((contadormsg!=n*tamanobloque) && (msg_sesion[contadormsg] != '\0'))
 {
 bloque[j]=msg_sesion[contadormsg];
 n_ascii=tolower(bloque[j]);
 total=total+n_ascii;
 j++;
 contadormsg++;
 }

 if (msg_sesion[contadormsg]=='\0')
 {
 bloque[j]='\0';
 terminar=TRUE;
 }
 else
 bloque[tamanobloque]='\0';

 n++;
 contadormsg--;

 if (acelerado)
 Crealista(&prioridad,bloque,total);
 else
 Crealista(&cab,bloque,total);
 datos.checksum=total;
 }
 total_TPDUs=n;
```

```
  terminar=FALSE;
  contadormsg=0;
  n=1;
  continuar_tx=FALSE;
  if (acelerado)
   temp2=prioridad;
  else
   temp=cab;
  acelerado=FALSE;
 }
 }
}

void Ventanahija1::Simulacaida(RTMessage)
{
 if ( GetModule()->ExecDialog(new Cdialogos5(this,"CAIDA"))==IDOK)
 {
  maquinas[0]->datos.caida=FALSE;
  maquinas[1]->datos.caida=FALSE;□
  for (int j=0;j<3;j++)
  {
   if (red.conges[j])
   {
    maquinas[0]->datos.caida=TRUE;
    maquinas[1]->datos.caida=TRUE;
   }
  }
  if ((reconectarse) && (!datos.caida))
  {
   if (Comprobar_qos())
   {
    if (parametro_reconexion==0)
     Inicializarventanas(0,1);
    else
     Inicializarventanas(1,0);
    SendMessage(ventanaorigen->HWindow,WM_PAINT,0,0L);
    SendMessage(ventanadestino->HWindow,WM_PAINT,0,0L);
    if (TPDUactiva)
            SendMessage(TPDU->HWindow,WM_PAINT,0,0L);
    SetWindowText(ventanaorigen->HWindow,ventanaorigen->a.origen);
    SetWindowText(ventanadestino->HWindow,ventanaorigen->a.destino);
    ventanaorigen->time_est=GetTickCount();
    ventanaorigen->datos.ID++;
    Brocha(1,datos.primitiva,wndini);
    ventanaorigen->datos.est=conexion_saliente;
    SendMessage(ventanaorigen->HWindow,WM_PAINT,0,0L);
    SendMessage(ventanadestino->HWindow,WM_CONNIND,0,0L);
    llenarinfo(datos.primitiva,"CR","1"," "," ");
    Brocha(4,datos.primitiva,wndnopicada);
```

```
    ventanadestino->datos.est=transferencia_datos;
    ventanaorigen->datos.origen=1000;
    ventanaorigen->datos.destino=2000;
    SendMessage(wndnopicada,WM_PAINT,0,0L);
    SendMessage(wndini,WM_CCONFIRM,0,0L);
   }
   else
    Borralista(&cab);
   reconectarse=FALSE;
  }
 }
}

void Ventanahija1::Informacionmaquina(RTMessage )
{
 infomaquina=new Ventanahija5(this,"Parámetros de la máquina",Attr.X-30,Attr.Y+190);
 GetApplication()->MakeWindow(infomaquina);
}

void Ventanahija1::Actuarantecaida(int n, int m)
{
 nodos *p;

 if ((datos.clase)>2)
 {
  reconectarse=TRUE;
  while (ventanaorigen->temp!=NULL)
  {
   Crealista(&almacenados,ventanaorigen->temp->parte,ventanaorigen->temp->checksum);
   ventanaorigen->temp=ventanaorigen->temp->prox;
  }
  while (ventanaorigen->temp2!=NULL)
  {
   Crealista(&almacenados2,ventanaorigen->temp2->parte,ventanaorigen->temp2->checksum);
   ventanaorigen->temp2=ventanaorigen->temp2->prox;
  }
  if (wndini==maquinas[0]->HWindow)
    parametro_reconexion=0;
  else
    parametro_reconexion=1;
 }
 SendMessage(ventanaorigen->HWindow,WM_DISCIND,0,0L);
 SendMessage(ventanadestino->HWindow,WM_DISCIND,0,0L);
 ventanaorigen->temp=ventanaorigen->cab=ventanaorigen->almacenados;
 almacenados=NULL;
 ventanaorigen->temp2=ventanaorigen->prioridad=ventanaorigen->almacenados2;
 almacenados2=NULL;
 if (datos.clase>2)
 {
```

```
 ventanaorigen->datos.n_secuencia=n;
 ventanadestino->datos.n_secuencia=n;
 ventanaorigen->datos.n_secuenciaexp=m;
 ventanadestino->datos.n_secuenciaexp=m;
 }
}


void Ventanahija1::Crealista(nodos **cabeza,char modulo[70],DWORD sum)
{
 nodos *p,*q;

 p=new nodos;
 p->prox=NULL;
 strcpy(p->parte," ");
 strcpy(p->auxiliar," ");
 strcpy(p->parte,modulo);
 p->checksum=sum;

 if (*cabeza==NULL)
  *cabeza=p;
 else
 {
 q=*cabeza;
 while (q->prox!=NULL) q=q->prox;
 q->prox=p;
 }
}

void Ventanahija1::Borralista(nodos **cabeza)
{
 nodos *aux;

 if (*cabeza==NULL)
  return;
 else
 {
 while (*cabeza!=NULL)
 {
  aux=*cabeza;
  *cabeza=(*cabeza)->prox;
  delete aux;
 }
 }
}

void Ventanahija1::WMSYSCOMMAND(RTMessage Msg)
{
 if (movibles==FALSE)
```

```
{   /* si es una órden del sistema... */
 if ((Msg.WParam & 0xfff0)== SC_MOVE)
   {        /* bloquea cualquier intento de mover la ventana... */
   ;   /* y en su lugar muestra un mensaje */
   }
}
else
{
   if ((Msg.WParam & 0xfff0)== SC_SIZE)
   {
    ;
    }
   else
    DefWindowProc(HWindow,Msg.Message,Msg.WParam,Msg.LParam);

 }
}

//************************************************************
// FIN DE LOS METODOS DE LA CLASE VENTANAHIJA1
//************************************************************

//************************************************************
// METODOS DE LA CLASE MROSI
//************************************************************

MROSI::MROSI(PTWindowsObject AParent,
                                LPSTR ATitle):
            TWindow(AParent,ATitle)
{
 Attr.Style |=WS_CHILD | WS_CAPTION | WS_MINIMIZEBOX
        | WS_CLIPSIBLINGS | WS_SYSMENU;
 Attr.X=abs((ancho-541)/2);
 Attr.Y=abs((alto-alto_barra-alto_titulo-378)/2);
 Attr.H=358+alto_titulo;
 Attr.W=541;
 SetRect(&TPDU1,65,170,115,190);
 SetRect(&TPDU2,435,170,490,190);
 puntos[0]=148; puntos[1]=157; puntos[2]=36; puntos[3]=138;
 hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "MODELOOSI");
 hbmpMyBitmap2 = LoadBitmap(GetApplication()->hInstance, "RED");
 red=FALSE;
 capas[0].left=15;
 capas[0].top=300;
 capas[0].right=166;
 capas[0].bottom=350;
 for (int cont=1;cont<7;cont++)
 {
  capas[cont].left=capas[0].left;
```

```
 capas[cont].right=capas[0].right;
 capas[cont].top=capas[cont-1].top-48;
 capas[cont].bottom=capas[cont-1].bottom-48;
 }
capas[7].left=71;
capas[7].top=174;
capas[7].right=118;
capas[7].bottom=190;

capas[8].left=380;
capas[8].top=300;
capas[8].right=533;
capas[8].bottom=350;
for (cont=9;cont<15;cont++)
{
 capas[cont].left=capas[8].left;
 capas[cont].right=capas[8].right;
 capas[cont].top=capas[cont-1].top-48;
 capas[cont].bottom=capas[cont-1].bottom-48;
 }
capas[15].left=440;
capas[15].top=173;
capas[15].right=485;
capas[15].bottom=187;
for (n=0;n<16;n++)
  niveles[NULL]=NULL;
n=0;
}

MROSI::~MROSI()
{
 DeleteObject(hbmpMyBitmap);
 DeleteObject(hbmpMyBitmap2);
}

void MROSI::Paint(HDC ,PAINTSTRUCT& )
{
 Dibujarbitmap(hbmpMyBitmap,HWindow,0,0,TRUE);
}

void MROSI::WMLButtonDown(RTMessage Msg)
{
 POINT punto;
 BOOL inside=FALSE;
 int i=0;
 char cp[25];
 RECT rect;

punto.x=Msg.LP.Lo;
```

```
punto.y=Msg.LP.Hi;
SetRect(&rect,202,200,338,342);

while ( (inside==FALSE) && (i<15) ) {
  if (PtInRect(&capas[i],punto))
   inside=TRUE;
  i++;
}

sprintf(cp," Capa ");
switch (i){
        case 1: strcat(cp,"física");
                break;
        case 2: strcat(cp,"de enlace");
                break;
        case 3: strcat(cp,"de red");
                break;
        case 4: strcat(cp,"de transporte");
                if (PtInRect(&capas[7],punto))
                 i=8;
                break;
        case 5: strcat(cp,"de sesión");
                break;
        case 6: strcat(cp,"de presentación");
                break;
        case 7: strcat(cp,"de aplicación");
                break;
        case 9: strcat(cp,"física");
                break;
        case 10: strcat(cp,"de enlace");
                break;
        case 11: strcat(cp,"de red");
                break;
        case 12: strcat(cp,"de transporte");
                if (PtInRect(&capas[15],punto))
                 i=16;
                break;
        case 13: strcat(cp,"de sesión");
                break;
        case 14: strcat(cp,"de presentación");
                break;
        case 15: strcat(cp,"de aplicación");
                break;

}

if (inside)
{
 GetApplication()->MakeWindow(CAPA=new Ventanahija7(Parent,cp,i));
 SetWindowPos(CAPA->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE |
```

```
SWP_NOSIZE);
   if (n<16)
   {
    niveles[n]=CAPA;
    CAPA->m=n;
    n++;
   }
  }
  if (PtInRect(&rect,punto))
  {
   if (!red)
   {
    HPEN hpen;
    HDC hdc;
    int l,k;
    l=272;□
    k=273;
    hdc=GetDC(HWindow);
    hpen=CreatePen(PS_SOLID,1,RGB(255,255,255));
    SelectObject(hdc,hpen);
    for (int j=0;j<70;j++)
    {
     MoveTo(hdc,202,l);
     LineTo(hdc,338,l);
     MoveTo(hdc,202,k);
     LineTo(hdc,338,k);
     delay(1);
     l=l--;
     k=k++;
    }
    DeleteObject(hpen);
    ReleaseDC(HWindow,hdc);
    Dibujarbitmap(hbmpMyBitmap2,HWindow,202,203,TRUE);
    red=TRUE;
    }
    else
    {
     SendMessage(HWindow,WM_PAINT,0,0L);
     red=FALSE;
    }
   }
  }

  void MROSI::GetWindowClass( WNDCLASS _FAR & h)
  {
   HINSTANCE Instanc;
   TWindow::GetWindowClass(h);
   Instanc=MROSI::GetModule()-> hInstance;
   h.style |=CS_NOCLOSE | CS_VREDRAW | CS_HREDRAW;
```

```
 h.hIcon=LoadIcon(Instanc,"TUTORIAL");
 h.hCursor=LoadCursor( GetApplication()->hInstance, "prisma");
 }
//*************************************************************
// FIN DE LOS METODOS DE LA CLASE MROSI
//*************************************************************
//*************************************************************
// METODOS DE LA CLASE SIMVISUAL
//*************************************************************
 Simvisual::Simvisual(PTWindowsObject AParent,
                                LPSTR ATitle,BYTE WinNum):
            TWindow(AParent,ATitle)
{
 Attr.Style |=WS_CHILD | WS_CAPTION | WS_CLIPSIBLINGS | WS_BORDER;
 switch (WinNum) {
        case 0:
                Attr.X=90;
                Attr.Y=abs((alto-462)/3);
                Attr.H=220+alto_titulo;
                Attr.W=166;
                break;
        case 1:
                Attr.X=ancho-210;
                Attr.Y=abs((alto-462)/3);
                Attr.H=220+alto_titulo;
                Attr.W=166;
                break;
        }
  puntos[0]=125; puntos[1]=89; puntos[2]=40; puntos[3]=65;
  hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "arq64K");
  retardo=200;
  contador=1;
  flagestado=Inactivo;
  continuo=TRUE;
  inicializado=FALSE;
  strcpy(b.origen,"OFRA");
  strcpy(b.destino,"ULPGC");
  en_marcha=FALSE;
}

Simvisual::~Simvisual()
{
 DeleteObject(hbmpMyBitmap);
}

int Simvisual::contador;
int Simvisual::retardo;
int Simvisual::recorrido;
BOOL Simvisual::inicializado;
```

```
BOOL Simvisual::continuo;

void Simvisual::Paint(HDC ,PAINTSTRUCT& )
{
 Dibujarbitmap(hbmpMyBitmap,HWindow,0,0,TRUE);
}

void Simvisual::menuflotante(POINT r)
{
 HMENU hmenu;

 hmenu=CreatePopupMenu();
 AppendMenu(hmenu,MF_ENABLED,IDM_CREQUEST,"T_CONNECT.request");
AppendMenu(hmenu,MF_ENABLED,IDM_CINDICATION,"T_CONNECT.indication");
AppendMenu(hmenu,MF_ENABLED,IDM_CRESPONSE,"T_CONNECT.response");
AppendMenu(hmenu,MF_ENABLED,IDM_CCONFIRM,"T_CONNECT.confirm");
AppendMenu(hmenu,MF_ENABLED,IDM_DREQUEST,"T_DISCONNECT.request");
AppendMenu(hmenu,MF_ENABLED,IDM_DINDICATION,"T_DISCONNECT.indication");
AppendMenu(hmenu,MF_ENABLED,IDM_DATAREQUEST,"T_DATA.request");
AppendMenu(hmenu,MF_ENABLED,IDM_DATAINDICATION,"T_DATA.indication");
 TrackPopupMenu(hmenu,TPM_LEFTBUTTON |
TPM_CENTERALIGN,r.x,r.y,0,HWindow,NULL);
 DestroyMenu(hmenu);
 }

 void Simvisual::Brocha(int est,COLORREF parpadeo,char directiva[23],HWND hwnd1)
 {
  RECT rect;
  COLORREF colores;
  HDC hdc;

  hdc=GetDC(hwnd1);
  SetTextColor(hdc,RGB(255,0,0));
  SetBkMode(hdc,TRANSPARENT);
  if (parpadeo==RGB(255,0,0))
  {
   SetRect(&rect,17,34,153,54);
   ExtTextOut(hdc,17,32,ETO_CLIPPED,&rect,directiva,strlen(directiva),NULL);
   }
  else
  {
   SetRect(&rect,17,175,153,195);
   ExtTextOut(hdc,17,173,ETO_CLIPPED,&rect,directiva,strlen(directiva),NULL);
  }
  for (int i=1;i<8;i++)
  {
  if ((i%2)==0)
    colores=parpadeo;
  else
```

```
   colores=RGB(255,255,255);

  switch (est){
   case 1:
            Rellenar(hwnd1,colores,puntos[0],puntos[1]);
            break;
   case 2:
            Rellenar(hwnd1,colores,124,159);
            break;
   case 3:
            Rellenar(hwnd1,colores,42,139);
            break;
   case 4:
            Rellenar(hwnd1,colores,puntos[2],puntos[3]);
            break;

   }
  delay(retardo);
 }

 ReleaseDC(hwnd1,hdc);
}

void Simvisual::WMRButtonDown(RTMessage Msg)
{
 HMENU hmenu;
 POINT m;

  if (((arq[0]->HWindow)==HWindow) && (contador==1))
  {
   secuencia[0]=arq[0]->HWindow;
   secuencia[1]=maquinas[0]->HWindow;
   secuencia[2]=maquinas[1]->HWindow;
   secuencia[3]=arq[1]->HWindow;
   secuenciaptrarq[0]=arq[0];
   secuenciaptrmaq[0]=maquinas[0];
   secuenciaptrmaq[1]=maquinas[1];
   secuenciaptrarq[1]=arq[1];
   }
  else
  {
   if (contador==1)
   {
    secuencia[0]=arq[1]->HWindow;
    secuencia[1]=maquinas[1]->HWindow;
    secuencia[2]=maquinas[0]->HWindow;
    secuencia[3]=arq[0]->HWindow;
    secuenciaptrarq[0]=arq[1];
    secuenciaptrmaq[0]=maquinas[1];
```

```
    secuenciaptrmaq[1]=maquinas[0];
    secuenciaptrarq[1]=arq[0];
   }
  }
 contador++;

 hmenu=CreatePopupMenu();
 AppendMenu(hmenu,MF_ENABLED,ORG_DEST,"Origen-Destino");
 AppendMenu(hmenu,MF_SEPARATOR,-1,NULL);
 AppendMenu(hmenu,MF_ENABLED,IDM_INFORMACION,"Información nivel superior");

 m.x=Msg.LP.Lo;
 m.y=Msg.LP.Hi;
 ClientToScreen(HWindow,&m);

 TrackPopupMenu(hmenu,TPM_LEFTBUTTON |
 TPM_CENTERALIGN,m.x,m.y,0,HWindow,NULL);
 DestroyMenu(hmenu);
}

void Simvisual::WMLButtonDown(RTMessage Msg)
{
 POINT p;
 p.x=Msg.LP.Lo;
 p.y=Msg.LP.Hi;
 ClientToScreen(HWindow,&p);□
 menuflotante(p);
}

void Simvisual::Directiva1(RTMessage )
{
 if (!en_marcha)
 {
  if (inicializado)
  {
   strcpy(t,"T_CONNECT.request");
   if ((flagestado==Inactivo) && (HWindow==secuencia[0]) && (continuo))
   {
    Brocha(1,RGB(255,0,0),t,secuencia[0]);
    flagestado=conexion_saliente;
    Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),21,17);
    secuenciaptrmaq[0]->Brocha(1,t,secuencia[1]);
    secuenciaptrmaq[0]->datos.est=conexion_saliente;
    SendMessage(secuencia[1],WM_PAINT,0,0L);
    Brocha(2,RGB(255,128,64),"N_CONNECT.request",secuencia[0]);
    Brocha(3,RGB(255,128,64),"N_CONNECT.ind",secuencia[3]);
    Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),19,30);
    secuenciaptrmaq[1]->Brocha(2,t,secuencia[2]);
    secuenciaptrmaq[1]->datos.est=conexion_entrante;
```

```
  SendMessage(secuencia[2],WM_PAINT,0,0L);
  Brocha(4,RGB(255,0,0),"T_CONNECT.ind",secuencia[3]);
  secuenciaptrarq[1]->flagestado=conexion_entrante;
  SendMessage(secuencia[0],WM_PAINT,0,0L);
  SendMessage(secuencia[3],WM_PAINT,0,0L);
 }
 else
 {
 if ((flagestado==Inactivo) && (HWindow==secuencia[0]) && (!continuo))
 {
  Brocha(1,RGB(255,0,0),t,secuencia[0]);
  recorrido=1;
  secuenciaptrarq[0]->en_marcha=TRUE;
  secuenciaptrarq[1]->en_marcha=TRUE;
  }
 }
}
 else
  MessageBox(HWindow,"El origen y el destino no se han indicado.\r\n Debe primero fijarcorrectamente esta opción.","¡¡ ERROR
!!",MB_OK | MB_ICONHAND);
 }
}

void Simvisual::Directiva3(RTMessage )
{
 if (!en_marcha)
 {
  strcpy(t,"T_DISCONNECT.req");
  if (continuo)
  {
   switch (flagestado){
    case conexion_saliente:
                                Brocha(1,RGB(255,0,0),t,secuencia[0]);
                                flagestado=Inactivo;
                                Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),21,17);
                                secuenciaptrmaq[0]->Brocha(6,t,secuencia[1]);□
                                secuenciaptrmaq[0]->datos.est=Inactivo;
                                SendMessage(secuencia[1],WM_PAINT,0,0L);
                                Brocha(2,RGB(255,128,64),"N_DISCONNECT.req",secuencia[0]);
                                Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                                Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),19,30);
                                secuenciaptrmaq[1]->Brocha(5,t,secuencia[2]);
                                secuenciaptrmaq[1]->datos.est=Inactivo;
                                SendMessage(secuencia[2],WM_PAINT,0,0L);
                                Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[3]);
                                secuenciaptrarq[1]->flagestado=Inactivo;
                                SendMessage(secuencia[0],WM_PAINT,0,0L);
                                SendMessage(secuencia[3],WM_PAINT,0,0L);
                                break;
```

```
case conexion_entrante:
                                Brocha(1,RGB(255,0,0),t,secuencia[3]);
                                flagestado=Inactivo;
                                Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),21,17);
                                secuenciaptrmaq[1]->Brocha(5,t,secuencia[2]);
                                secuenciaptrmaq[1]->datos.est=Inactivo;
                                SendMessage(secuencia[2],WM_PAINT,0,0L);
                                Brocha(2,RGB(255,128,64),"N_DISCONNECT.req",secuencia[3]);
                                Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                                Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),19,30);
                                secuenciaptrmaq[0]->Brocha(6,t,secuencia[1]);
                                secuenciaptrmaq[0]->datos.est=Inactivo;
                                SendMessage(secuencia[1],WM_PAINT,0,0L);
                                Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[0]);
                                secuenciaptrarq[0]->flagestado=Inactivo;
                                SendMessage(secuencia[3],WM_PAINT,0,0L);
                               SendMessage(secuencia[0],WM_PAINT,0,0L);
                                break;
case transferencia_datos:
                        if (secuencia[0]==HWindow)
        {
                         Brocha(1,RGB(255,0,0),t,secuencia[0]);□
                         flagestado=Inactivo;
                         Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),21,17);
                         secuenciaptrmaq[0]->Brocha(8,t,secuencia[1]);
                         secuenciaptrmaq[0]->datos.est=Inactivo;
                         SendMessage(secuencia[1],WM_PAINT,0,0L);
                         Brocha(2,RGB(255,128,64),"N_DISCONNECT.req",secuencia[0]);
                         Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                         Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),19,30);
                         secuenciaptrmaq[1]->Brocha(8,t,secuencia[2]);
                         secuenciaptrmaq[1]->datos.est=Inactivo;
                         SendMessage(secuencia[2],WM_PAINT,0,0L);
                         Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[3]);
                         secuenciaptrarq[1]->flagestado=Inactivo;
                         SendMessage(secuencia[0],WM_PAINT,0,0L);
                         SendMessage(secuencia[3],WM_PAINT,0,0L);
                         }
                        else
                        {
                         Brocha(1,RGB(255,0,0),t,secuencia[3]);
                         flagestado=Inactivo;
                         Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),21,17);
                         secuenciaptrmaq[1]->Brocha(8,t,secuencia[2]);
                         secuenciaptrmaq[1]->datos.est=Inactivo;
                         SendMessage(secuencia[2],WM_PAINT,0,0L);
                         Brocha(2,RGB(255,128,64),"N_DISCONNECT.req",secuencia[3]);
                         Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                         Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),19,30);
```

```
                    secuenciaptrmaq[0]->Brocha(8,t,secuencia[1]);
                    secuenciaptrmaq[0]->datos.est=Inactivo;
                    SendMessage(secuencia[1],WM_PAINT,0,0L);
                    Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[0]);
                    secuenciaptrarq[0]->flagestado=Inactivo;
                    SendMessage(secuencia[3],WM_PAINT,0,0L);
                    SendMessage(secuencia[0],WM_PAINT,0,0L);
                    }
                   break;
 }
 contador=1;
 inicializado=FALSE;
 SetWindowText(secuencia[0],"Capas RM-OSI");
 SetWindowText(secuencia[1],"Máquina de estados");
 SetWindowText(secuencia[2],"Máquina de estados");
 SetWindowText(secuencia[3],"Capas RM-OSI");
}
else
{
 secuenciaptrarq[0]->en_marcha=TRUE;
 secuenciaptrarq[1]->en_marcha=TRUE;
 switch (flagestado){
   case conexion_saliente:
                              Brocha(1,RGB(255,0,0),t,secuencia[0]);
                              flagestado=Inactivo;
                              recorrido=4;
                              break;
   case conexion_entrante:
                              Brocha(1,RGB(255,0,0),t,secuencia[3]);
                              flagestado=Inactivo;
                              recorrido=5;
                              break;
   case transferencia_datos:
                   if (secuencia[0]==HWindow)
          {
                    Brocha(1,RGB(255,0,0),t,secuencia[0]);
                    flagestado=Inactivo;
          recorrido=6;
                    }
                   else
                   {
                    Brocha(1,RGB(255,0,0),t,secuencia[3]);
                    flagestado=Inactivo;
           recorrido=7;
                    }
                   break;
 }
}
}
```

```
}

void Simvisual::Directiva7(RTMessage )
{
 if (!en_marcha)
 {
  strcpy(t,"T_CONNECT.resp");
  if ((flagestado==conexion_entrante) && (continuo))
  {
   Brocha(1,RGB(255,0,0),t,secuencia[3]);
   flagestado=transferencia_datos;
   Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),21,17);
   secuenciaptrmaq[1]->Brocha(4,t,secuencia[2]);
   secuenciaptrmaq[1]->datos.est=transferencia_datos;
   SendMessage(secuencia[2],WM_PAINT,0,0L);
   Brocha(2,RGB(255,128,64),"N_CONNECT.resp",secuencia[3]);
   Brocha(3,RGB(255,128,64),"N_CONNECT.confirm",secuencia[0]);
   Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),19,30);
   secuenciaptrmaq[0]->Brocha(3,t,secuencia[1]);
   secuenciaptrmaq[0]->datos.est=transferencia_datos;
   SendMessage(secuencia[1],WM_PAINT,0,0L);
   Brocha(4,RGB(255,0,0),"T_CONNECT.confirm",secuencia[0]);
   secuenciaptrarq[0]->flagestado=transferencia_datos;
   SendMessage(secuencia[3],WM_PAINT,0,0L);
   SendMessage(secuencia[0],WM_PAINT,0,0L);
  }
  else
  {
   if ((flagestado==conexion_entrante) && (!continuo))
   {
    Brocha(1,RGB(255,0,0),t,secuencia[3]);
    recorrido=2;
    flagestado=transferencia_datos;
    secuenciaptrarq[0]->en_marcha=TRUE;
    secuenciaptrarq[1]->en_marcha=TRUE;
   }
  }
 }
}

void Simvisual::Directiva4(RTMessage )
{
 RECT rect;

 SetRect(&rect,17,175,153,195);
 if (!en_marcha)
 {
  strcpy(t,"T_DISCONNECT.ind");
  if (continuo)
```

```
{
 switch (flagestado){
   case conexion_saliente:
                                Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                                Brocha(4,RGB(255,0,0),t,secuencia[0]);
                                Brocha(1,RGB(255,0,0),"",secuencia[0]);
                                flagestado=Inactivo;
                                Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),19,30);
                                secuenciaptrmaq[0]->Brocha(6,t,secuencia[1]);
                                secuenciaptrmaq[0]->datos.est=Inactivo;
                                SendMessage(secuencia[1],WM_PAINT,0,0L);
                               Brocha(2,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                               Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                               Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),19,30);
                               secuenciaptrmaq[1]->Brocha(5,t,secuencia[2]);
                               secuenciaptrmaq[1]->datos.est=Inactivo;
                               SendMessage(secuencia[2],WM_PAINT,0,0L);
                               Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[3]);
                               secuenciaptrarq[1]->flagestado=Inactivo;
                               SendMessage(secuencia[0],WM_PAINT,0,0L);
                               SendMessage(secuencia[3],WM_PAINT,0,0L);
                               break;
   case conexion_entrante:
                               Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                               Brocha(4,RGB(255,0,0),t,secuencia[3]);
                               Brocha(1,RGB(255,0,0),"",secuencia[3]);
                               flagestado=Inactivo;
                               Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),19,30);
                               secuenciaptrmaq[1]->Brocha(5,t,secuencia[2]);
                               secuenciaptrmaq[1]->datos.est=Inactivo;
                               SendMessage(secuencia[2],WM_PAINT,0,0L);
                               Brocha(2,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                               Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                               Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),19,30);
                               secuenciaptrmaq[0]->Brocha(6,t,secuencia[1]);
                               secuenciaptrmaq[0]->datos.est=Inactivo;
                               SendMessage(secuencia[1],WM_PAINT,0,0L);
                               Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[0]);
                               secuenciaptrarq[0]->flagestado=Inactivo;
                               SendMessage(secuencia[3],WM_PAINT,0,0L);
                               SendMessage(secuencia[0],WM_PAINT,0,0L);
                               break;
   case transferencia_datos:
                              if (secuencia[0]==HWindow)
                              {
                               Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                               Brocha(4,RGB(255,0,0),t,secuencia[0]);
                               Brocha(1,RGB(255,0,0),"",secuencia[0]);
                               flagestado=Inactivo;
```

```
                    Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),19,30);
                    secuenciaptrmaq[0]->Brocha(8,t,secuencia[1]);
                    secuenciaptrmaq[0]->datos.est=Inactivo;
                    SendMessage(secuencia[1],WM_PAINT,0,0L);
                    Brocha(2,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                    Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                    Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),19,30);
                    secuenciaptrmaq[1]->Brocha(8,t,secuencia[2]);
                    secuenciaptrmaq[1]->datos.est=Inactivo;
                    SendMessage(secuencia[2],WM_PAINT,0,0L);
                    Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[3]);
                    secuenciaptrarq[1]->flagestado=Inactivo;
                    SendMessage(secuencia[0],WM_PAINT,0,0L);
                    SendMessage(secuencia[3],WM_PAINT,0,0L);
                    }
                   else
                   {
                    Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                    Brocha(4,RGB(255,0,0),t,secuencia[3]);
                    Brocha(1,RGB(255,0,0),"",secuencia[3]);
                     flagestado=Inactivo;
                     Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),19,30);
                     secuenciaptrmaq[1]->Brocha(8,t,secuencia[2]);
                     secuenciaptrmaq[1]->datos.est=Inactivo;
                     SendMessage(secuencia[2],WM_PAINT,0,0L);
                     Brocha(2,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                     Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                     Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),19,30);
                     secuenciaptrmaq[0]->Brocha(8,t,secuencia[1]);
                     secuenciaptrmaq[0]->datos.est=Inactivo;
                     SendMessage(secuencia[1],WM_PAINT,0,0L);
                     Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[0]);
                     secuenciaptrarq[0]->flagestado=Inactivo;
                     SendMessage(secuencia[3],WM_PAINT,0,0L);
                     SendMessage(secuencia[0],WM_PAINT,0,0L);
                     }
                    break;
  }
 contador=1;
 inicializado=FALSE;
 SetWindowText(secuencia[0],"Capas RM-OSI");
 SetWindowText(secuencia[1],"Máquina de estados");
 SetWindowText(secuencia[2],"Máquina de estados");
 SetWindowText(secuencia[3],"Capas RM-OSI");
}
else
{
 secuenciaptrarq[0]->en_marcha=TRUE;□
 secuenciaptrarq[1]->en_marcha=TRUE;
```

```
  switch (flagestado){
    case conexion_saliente:
                                    Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                                    Brocha(4,RGB(255,0,0),t,secuencia[0]);
                                    Brocha(1,RGB(255,0,0),"",secuencia[0]);
                                    InvalidateRect(secuencia[0],&rect,TRUE);
                                    flagestado=Inactivo;
                                    recorrido=4;
                                    break;
    case conexion_entrante:
                                    Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                                    Brocha(4,RGB(255,0,0),t,secuencia[3]);
                                    Brocha(1,RGB(255,0,0),"",secuencia[3]);
                                    InvalidateRect(secuencia[3],&rect,TRUE);
                                    flagestado=Inactivo;
                                    recorrido=5;
                                    break;
    case transferencia_datos:
                            if (secuencia[0]==HWindow)
                            {
                             Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                             Brocha(4,RGB(255,0,0),t,secuencia[0]);
                             Brocha(1,RGB(255,0,0),"",secuencia[0]);
                             InvalidateRect(secuencia[0],&rect,TRUE);
                             flagestado=Inactivo;
                             recorrido=6;
                             }
                            else
                            {
                             Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                             Brocha(4,RGB(255,0,0),t,secuencia[3]);
                             Brocha(1,RGB(255,0,0),"",secuencia[3]);
                             InvalidateRect(secuencia[3],&rect,TRUE);
                              flagestado=Inactivo;
                             recorrido=7;
                             }
                            break;
   }
  }
 }
}

void Simvisual::Directiva5(RTMessage )
{
 if (!en_marcha)
 {
  strcpy(t,"T_DATA.request");
  if ((flagestado==transferencia_datos) && (continuo) && (secuencia[0]==HWindow))
  {
```

```
  Brocha(1,RGB(255,0,0),t,secuencia[0]);
  Rellenar(secuenciaptrmaq[0]->HWindow,RGB(255,0,0),21,17);
  secuenciaptrmaq[0]->Brocha(7,t,secuencia[1]);
  Brocha(2,RGB(255,128,64),"N_DATA.request",secuencia[0]);
  Brocha(3,RGB(255,128,64),"N_DATA.ind",secuencia[3]);
  Rellenar(secuenciaptrmaq[1]->HWindow,RGB(255,0,0),19,30);
  secuenciaptrmaq[1]->Brocha(7,t,secuencia[2]);
  Brocha(4,RGB(255,0,0),"T_DATA.ind",secuencia[3]);
  SendMessage(secuencia[0],WM_PAINT,0,0L);
  SendMessage(secuencia[3],WM_PAINT,0,0L);
 }
 else
 {
  if ((flagestado==transferencia_datos) && (!continuo) && (secuencia[0]==HWindow))
  {
   Brocha(1,RGB(255,0,0),t,secuencia[0]);
   recorrido=3;
   secuenciaptrarq[0]->en_marcha=TRUE;
   secuenciaptrarq[1]->en_marcha=TRUE;
  }
 }
}
}

void Simvisual::Localremoto(RTMessage)
{
 if ((flagestado==Inactivo) && (HWindow==secuencia[0]))
 {
  if (GetModule()->ExecDialog(new Cdialogos(this,"ORG_DES"))==IDOK)
  {
   SetWindowText(secuencia[0],b.origen);
   SetWindowText(secuencia[1],b.origen);
   SetWindowText(secuencia[2],b.destino);
   SetWindowText(secuencia[3],b.destino);
   inicializado=TRUE;
  }
 }
}

void Simvisual::Informa(RTMessage)
{
 if ((flagestado==Inactivo) && (HWindow==secuencia[0]))
 {
  GetModule()->ExecDialog(new Cdialogos3(this,"INFORMACION"));
 }
}

void Simvisual::WMSYSCOMMAND(RTMessage Msg)
{
```

```
if ((Msg.WParam & 0xfff0)== SC_MOVE)
{       // bloquea cualquier intento de mover la ventana...
 ;   // y en su lugar muestra un mensaje
 }
else
  DefWindowProc(HWindow,Msg.Message,Msg.WParam,Msg.LParam);
}

void Simvisual::GetWindowClass( WNDCLASS _FAR & h)
{
TWindow::GetWindowClass(h);
h.style |=CS_NOCLOSE | CS_VREDRAW | CS_HREDRAW;
 }
//*************************************************************
// FIN DE LOS METODOS DE LA CLASE Simvisual
//*************************************************************
//*************************************************************
// METODOS DE LA CLASE VENTANAHIJA3
//*************************************************************
 Ventanahija3::Ventanahija3(PTWindowsObject AParent,LPSTR ATitle)
             :TWindow(AParent,ATitle)
{
 Attr.Style |=WS_CHILD | WS_BORDER | WS_CLIPSIBLINGS;
 movible=TRUE;
 Attr.X=0;
 Attr.Y=abajo-130;
 Attr.H=56;
 Attr.W=ancho;
 new TStatic(this,-1,"Origen:",8,4,55,18,7);
 new TStatic(this,-1,"Destino:",200,4,60,18,8);
 new TStatic(this,-1,"Primitiva:",390,5,65,15,10);
 new TStatic(this,-1,"TPDU:",8,30,40,15,5);
 new TStatic(this,-1,"ID:",105,30,30,15,3);
 new TStatic(this,-1,"N° Secuencia:",185,30,180,15,13);
 new TStatic(this,-1,"CheckSum:",385,30,80,15,9);

 if (maquinas[0]->ventanaorigen!=NULL)
 {
  org=new TStatic(this,-1,maquinas[0]->ventanaorigen->datos.nombre,63,4,125,18,40);
  dest=newTStatic(this,-1,maquinas[0]->ventanadestino->datos.nombre,260,4,120,18,40);
 }
 else
 {
  org=new TStatic(this,-1,"",63,4,125,18,40);
  dest=new TStatic(this,-1,"",260,4,120,18,40);□
 }
 primitiva=new TStatic(this,-1,"",455,5,170,15,40);
 tpdu=new TStatic(this,-1,"",50,30,30,15,40);
 id=new TStatic(this,-1,"",140,30,30,15,40);
```

```
 n_secuencia=new TStatic(this,-1,"",280,30,30,15,40);
 checksum=new TStatic(this,-1,"",470,30,70,15,40);
 SetRect(&rect1,3,1,195,26);
 SetRect(&rect2,195,1,385,26);
 SetRect(&rect3,385,1,ancho-13,26);
 SetRect(&rect4,3,26,100,51);
 SetRect(&rect5,100,26,180,51);
 SetRect(&rect6,180,26,380,51);
 SetRect(&rect7,380,26,ancho-13,51);
}

void Ventanahija3::Fijarorigen(LPSTR texto)
{
 org->SetText(texto);
}

void Ventanahija3::Fijardestino(LPSTR texto)
{
 dest->SetText(texto);
}

void Ventanahija3::Fijartpdu(LPSTR texto)
{
 tpdu->SetText(texto);
}

void Ventanahija3::Fijarprimitiva(LPSTR texto)
{
 primitiva->SetText(texto);
}

void Ventanahija3::Fijarid(LPSTR texto)
{
id->SetText(texto);
}

void Ventanahija3::Fijarchecksum(LPSTR texto)
{
 checksum->SetText(texto);
}

void Ventanahija3::Fijarn_secuencia(LPSTR texto)
{
 n_secuencia->SetText(texto);
}

void Ventanahija3::WMControlColor(RTMessage Msg)
{
 SetBkColor( HDC(Msg.WParam),RGB(192,192,192));
```

```
Msg.Result=(long)GetStockObject(LTGRAY_BRUSH);
}


void Ventanahija3::Dibujarrecuadros(HDC hDC, RECT *rect)
{
 int x1, x2, y1, y2;
 HPEN hPen, hOldPen;
 HBRUSH hOldBrush;
 POINT pArray[3];

 hDC=GetDC(HWindow);
 x1 = rect->left;
 x2 = rect->right;
 y1 = rect->top;
 y2 = rect->bottom;

 hOldBrush = (HBRUSH)SelectObject(hDC, GetStockObject(NULL_BRUSH));
 hOldPen = (HPEN)SelectObject(hDC, GetStockObject(WHITE_PEN));

 Rectangle(hDC, x1, y1, x2, y2);

 pArray[0].x = x1 + 2;
 pArray[1].y = pArray[0].y = y2 - 3;
 pArray[2].x = pArray[1].x = x2 - 3;
 pArray[2].y = y1 + 2;

 Polyline(hDC, pArray, 3);
 hPen = CreatePen(PS_SOLID, 1, RGB(128, 128, 128));
 SelectObject(hDC, hPen);
 pArray[0].x = x1;
 pArray[1].y = pArray[0].y = y2-1;
 pArray[2].x = pArray[1].x = x2-1;
 pArray[2].y = y1;

 Polyline(hDC, pArray, 3);
 pArray[1].x = pArray[0].x = x1 + 2;
 pArray[0].y = y2 - 3;
 pArray[2].y = pArray[1].y = y1 + 2;
 pArray[2].x = x2 - 3;
 Polyline(hDC, pArray, 3);
 SelectObject(hDC, hOldBrush);
 DeleteObject(SelectObject(hDC, hOldPen));
 ReleaseDC(HWindow,hDC);
}

void Ventanahija3::Dibujarinfo(HDC hDC)
{
 Dibujarrecuadros(hDC, &rect1);
```

```
 Dibujarrecuadros(hDC, &rect2);
 Dibujarrecuadros(hDC, &rect3);
 Dibujarrecuadros(hDC, &rect4);
 Dibujarrecuadros(hDC, &rect5);
 Dibujarrecuadros(hDC, &rect6);
 Dibujarrecuadros(hDC, &rect7);
}

void Ventanahija3::Paint(HDC DC,PAINTSTRUCT& )
{
 Dibujarinfo(DC);
}

void Ventanahija3::WMLButtonDown(RTMessage Msg)
{
 POINT punto;

 punto.x=Msg.LP.Lo;
 punto.y=Msg.LP.Hi;

 if (PtInRect(&rect1,punto))
  WinHelp(HWindow,"ADAN.HLP",HELP_CONTEXTPOPUP,CAMPOORIGEN);
 if (PtInRect(&rect2,punto))
  WinHelp(HWindow,"ADAN.HLP",HELP_CONTEXTPOPUP,CAMPODESTINO);
 if (PtInRect(&rect3,punto))
  WinHelp(HWindow,"ADAN.HLP",HELP_CONTEXTPOPUP,VENTANA1);
 if (PtInRect(&rect4,punto))
  WinHelp(HWindow,"ADAN.HLP",HELP_CONTEXTPOPUP,CAMPOTPDU);
 if (PtInRect(&rect5,punto))
  WinHelp(HWindow,"ADAN.HLP",HELP_CONTEXTPOPUP,CAMPOID);
 if (PtInRect(&rect6,punto))
  WinHelp(HWindow,"ADAN.HLP",HELP_CONTEXTPOPUP,CAMPOSECUENCIA);
 if (PtInRect(&rect7,punto))
WinHelp(HWindow,"ADAN.HLP",HELP_CONTEXTPOPUP,CAMPOCHECKSUM);
}

void Ventanahija3::GetWindowClass( WNDCLASS _FAR & h)
{
 TWindow::GetWindowClass(h);
 h.style |=CS_NOCLOSE | CS_VREDRAW | CS_HREDRAW;
 h.hCursor=LoadCursor(NULL,IDC_CROSS);
 h.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
}
//*************************************************************+****
// FIN DE LOS METODOS DE LA CLASE VENTANAHIJA3
//*************************************************************

//*************************************************************
// METODOS DE LA CLASE VENTANAHIJA4
```

```
//*************************************************************

Ventanahija4::Ventanahija4(PTWindowsObject AParent,
                           LPSTR ATitle):
            TWindow(AParent,ATitle)
{
 Attr.Style |=WS_CHILD | WS_CAPTION | WS_CLIPSIBLINGS | WS_SYSMENU ;
 Attr.X=abs((ancho-450)/2);
 Attr.Y=(abs((alto-344)/2))+150;
 Attr.H=110+alto_titulo;
 Attr.W=450;
 hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "FORMATOTPDU");
 CheckBox=new TCheckBox(this,ID_EJECUCION,"Modificar la TPDU durante laejecución.",10,86,300,25,NULL);
 new TStatic(this,-1,"Campo a modificar:",320,10,130,20,18,NULL);
 ComboBox=new
 TComboBox(this,ID_CAMPO,320,30,100,80,CBS_DROPDOWNLIST,10,NULL);
 EnableKBHandler();
}

void Ventanahija4::SetupWindow()
{
 TWindow::SetupWindow();
 CheckBox->Uncheck();
 ComboBox->AddString("Origen");
 ComboBox->AddString("Destino");
 ComboBox->AddString("QOS");
}

Ventanahija4::~Ventanahija4()
{
 DeleteObject(hbmpMyBitmap);
}

void Ventanahija4::Paint(HDC ,PAINTSTRUCT& )
{
 Dibujarbitmap(hbmpMyBitmap,HWindow,10,5,TRUE);
 }

void Ventanahija4::GetWindowClass( WNDCLASS _FAR & h)
{
 TWindow::GetWindowClass(h);
 h.style |=CS_VREDRAW | CS_HREDRAW;
 h.hCursor=LoadCursor( GetApplication()->hInstance, "CURSOR_1");
}


void Ventanahija4::WMDestroy(RTMessage Msg)
{
 HMENU menu;
```

```
menu=GetMenu(Parent->HWindow);
EnableMenuItem(menu,CM_TPDU,MF_BYCOMMAND | MF_ENABLED);
EnableMenuItem(menu,CM_CERRARTPDU,MF_BYCOMMAND | MF_GRAYED);
EnableMenuItem(menu,CM_AMBAS,MF_BYCOMMAND | MF_GRAYED);
EnableMenuItem(menu,CM_CERRARAMBAS,MF_BYCOMMAND | MF_GRAYED);
EnableWindow(Barra->PBotontpdu->HWindow,TRUE);
TPDUactiva=FALSE;
DefWndProc(Msg);
}

void Ventanahija4::Procesar()
{
 char texto[100];
 int seleccion,Nuevaclase;
 strcpy(texto," ");

 if (CheckBox->GetCheck()==BF_CHECKED)
 {
  seleccion=ComboBox->GetSelIndex();
  switch (seleccion)
  {
   case 0: if (GetApplication()->ExecDialog(new TInputDialog(this,"Modificar TPDU","Nuevodestino:",texto,sizeof texto))==IDOK)
                {
                 strcpy(maquinas[0]->ventanaorigen->a.destino,texto);□
                 llenardeststruct(texto);
                }
               break;
   case 1: if (GetApplication()->ExecDialog(new TInputDialog(this,"Modificar TPDU","Nuevoorigen:", texto,sizeof texto))==IDOK)
           {
                 strcpy(maquinas[0]->ventanaorigen->a.origen,texto);
                 llenarorgstruct(texto);
           }
                 break;
   case 2: if (GetApplication()->ExecDialog(new TInputDialog(this,"Modificar TPDU","Nuevaclase:", texto,sizeof texto))==IDOK)
                {
                 Nuevaclase=atoi(texto);
                 if ((Nuevaclase<0) || (Nuevaclase>4))
                  delay(20);
                 else
        {
                  maquinas[0]->ventanaorigen->datos.clase=Nuevaclase;
                  for (int j=0;j<5;j++)
                          maquinas[0]->ventanaorigen->c.cl[j]=FALSE;
                  maquinas[0]->ventanaorigen->c.cl[Nuevaclase]=TRUE;
        }
                }
               break;
 }
```

```
  SendMessage(maquinas[0]->HWindow,WM_PAINT,0,0L);
  SendMessage(maquinas[1]->HWindow,WM_PAINT,0,0L);
 }
}

void Ventanahija4::llenarorgstruct(LPSTR a)
{
 if (Infoactiva)
  Info->Fijarorigen(a);
}

void Ventanahija4::llenardeststruct(LPSTR b)
{
 if (Infoactiva)
  Info->Fijardestino(b);
}

void Ventanahija4::WMLButtonDown(RTMessage Msg)
{
 RECT rect_datos,rect_org,rect_dest,rect_qos;
 POINT punto;
 char texto[100];
 int Nuevaclase;

 punto.x=Msg.LP.Lo;
 punto.y=Msg.LP.Hi;
 SetRect(&rect_datos,230,7,306,81);
 SetRect(&rect_org,4,7,81,81);
 SetRect(&rect_dest,79,7,155,81);
 SetRect(&rect_qos,154,7,230,81);
 strcpy(texto," ");

 if (maquinas[0]->ventanaorigen==NULL)
    MessageBox(HWindow,"Configure en primer lugar la capa de transporte con losparámetros que aparecen al pulsar con el botón derecho la ventana Máquina de estados.","¡¡ERROR !!",MB_OK | MB_ICONHAND);
 else
 {
  if (PtInRect(&rect_datos,punto))
  {
   GetApplication()->MakeWindow(campos=new Ventanahija6(Parent,"Campo de datos"));
   BringWindowToTop(campos->HWindow);
  }
  if (PtInRect(&rect_org,punto))
  {
   if (GetApplication()->ExecDialog(new TInputDialog(this,"Modificar TPDU","Nuevo origen:",texto,sizeof texto))==IDOK)
   {
    strcpy(maquinas[0]->ventanaorigen->a.origen,texto);
    llenarorgstruct(texto);
   }
```

```
 }
 if (PtInRect(&rect_dest,punto))
 {
  if (GetApplication()->ExecDialog(new TInputDialog(this,"Modificar TPDU","Nuevodestino:",texto,sizeof texto))==IDOK)
  {
   strcpy(maquinas[0]->ventanaorigen->a.destino,texto);
   llenardeststruct(texto);
  }
 }
 if (PtInRect(&rect_qos,punto))
 {
  if (GetApplication()->ExecDialog(new TInputDialog(this,"Modificar TPDU","Nueva clase:",texto,sizeof texto))==IDOK)
  {
   Nuevaclase=atoi(texto);
   if ((Nuevaclase<0) || (Nuevaclase>4))
     delay(20);
   else
   {
    maquinas[0]->ventanaorigen->datos.clase=Nuevaclase;
    for (int j=0;j<5;j++)
            maquinas[0]->ventanaorigen->c.cl[j]=FALSE;
    maquinas[0]->ventanaorigen->c.cl[Nuevaclase]=TRUE;
   }
  }

 }
}
}

//*************************************************************
// FIN DE LOS METODOS DE LA CLASE VENTANAHIJA4
//*************************************************************

//*************************************************************
// METODOS DE LA CLASE VENTANAHIJA5
//*************************************************************

Ventanahija5::Ventanahija5(PTWindowsObject AParent,
                           LPSTR ATitle,int x,int y):
        TWindow(AParent,ATitle)
{
 Attr.Style |=WS_POPUP | WS_CAPTION | WS_CLIPSIBLINGS | WS_SYSMENU;
 Attr.X=x;
 Attr.Y=y;
 Attr.H=180+alto_titulo;
 Attr.W=280;
 hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "PARAMETROS");
 param_datos=PVentanahija1(Parent)->datos;
 param_calidad=PVentanahija1(Parent)->calidad;□
```

```
 Rellenarparametros();
 nombre=new TStatic(this,-1,n,189,15,80,15,strlen(n));
 tsap=new TStatic(this,-1,t,145,47,50,15,strlen(t));
 id=new TStatic(this,-1,i,231,47,15,15,strlen(i));
 clase=new TStatic(this,-1,c,54,72,15,15,strlen(c));
 prioridad=new TStatic(this,-1,p,250,72,18,15,strlen(p));
 longitud=new TStatic(this,-1,l,143,72,30,15,strlen(l));
 estab=new TStatic(this,-1,r1,230,101,35,15,strlen(r1));
 transito=new TStatic(this,-1,r2,230,126,35,15,strlen(r2));
 liberacion=new TStatic(this,-1,r3,230,152,35,15,strlen(r3));

}

Ventanahija5::~Ventanahija5()
{
 DeleteObject(hbmpMyBitmap);
}

void Ventanahija5::Rellenarparametros()
{
 if (param_calidad.datos_expeditos)
  strcpy(p,"Sí");
 else
  strcpy(p,"No");
 strcpy(n,param_datos.nombre);
 sprintf(c,"%d",param_datos.clase);
 sprintf(i,"%d",param_datos.ID);
 sprintf(t,"%d",param_datos.origen);
 if (param_datos.c.longitud[0])
  strcpy(l,"256");
 else
  strcpy(l,"1024");
 sprintf(r1,"%d",param_datos.c.retardo1);
 sprintf(r2,"%d",param_datos.c.retardo2);
 sprintf(r3,"%d",param_datos.c.retardo3);
}

void Ventanahija5::Paint(HDC ,PAINTSTRUCT& )
{
 Dibujarbitmap(hbmpMyBitmap,HWindow,0,0,TRUE);
  switch(param_datos.est){
          case 0: Rellenar(HWindow,RGB(0,255,0),11,104);
                  break;
          case 1: Rellenar(HWindow,RGB(0,255,0),11,125);
                  break;
          case 2: Rellenar(HWindow,RGB(0,255,0),11,147);
                  break;
          case 3: Rellenar(HWindow,RGB(0,255,0),11,171);
                  break;
```

```
 }
}

void Ventanahija5::GetWindowClass( WNDCLASS _FAR & h)
{
 TWindow::GetWindowClass(h);
 h.style |=CS_VREDRAW | CS_HREDRAW;
 h.hCursor = LoadCursor( 0, IDC_ARROW);
}

void Ventanahija5::WMSYSCOMMAND(RTMessage Msg)
{
 if ((Msg.WParam & 0xfff0)== SC_MOVE)
 {       /* bloquea cualquier intento de mover la ventana... */
  ;   /* y en su lugar muestra un mensaje */
 }
 else
    DefWindowProc(HWindow,Msg.Message,Msg.WParam,Msg.LParam);
}

//*************************************************************
// FIN DE LOS METODOS DE LA CLASE VENTANAHIJA5
//*************************************************************



//*************************************************************
// METODOS DE LA CLASE VENTANAHIJA6
//*************************************************************
 Ventanahija6::Ventanahija6(PTWindowsObject AParent,
                                  LPSTR ATitle):
                TWindow(AParent,ATitle)
{
  Attr.Style |=WS_CHILD | WS_BORDER | WS_CAPTION | WS_CLIPSIBLINGS | WS_SYSMENU;
  Attr.X=180;
  Attr.Y=210;
  Attr.H=140+alto_titulo;
  Attr.W=275;
  hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "DATOS");
  strcpy(mensaje,"No hay datos. Debe fijarlos.");
  if (maquinas[0]->ventanaorigen->temp!=NULL)
    strcpy(mensaje,maquinas[0]->ventanaorigen->temp->parte);
Edit1=new TEdit(this,IDC_CAMPODATOS,mensaje,60,31,210,60,strlen(mensaje),TRUE);
  PBotoncerrar = new TBButton(this, IDCERRAR, NULL, 105, 100, NULL, NULL, FALSE);
}

Ventanahija6::~Ventanahija6()
{
 DeleteObject(hbmpMyBitmap);
}
```

```
void Ventanahija6::IDcerrar(RTMessage)
{
 char texto[70];

 Edit1->GetText(texto,sizeof texto);
 if (maquinas[0]->ventanaorigen!=NULL)
 {
  if ((strcmp(texto,mensaje)) && (maquinas[0]->ventanaorigen->temp!=NULL))
   strcpy(maquinas[0]->ventanaorigen->temp->auxiliar,texto);
 }
 SendMessage(HWindow, WM_CLOSE, 0, NULL);
}

void Ventanahija6::Paint(HDC ,PAINTSTRUCT& )
{
 Dibujarbitmap(hbmpMyBitmap,HWindow,0,8,TRUE);
}

void Ventanahija6::GetWindowClass( WNDCLASS _FAR & h)
{
 TWindow::GetWindowClass(h);
 h.style |=CS_VREDRAW | CS_HREDRAW;
 h.hCursor = LoadCursor( 0, IDC_ARROW);
 h.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
 }

//*************************************************************
// FIN DE LOS METODOS DE LA CLASE VENTANAHIJA6
//*************************************************************


//*************************************************************
// METODOS DE LA CLASE VENTANAHIJA7
//*************************************************************

 Ventanahija7::Ventanahija7(PTWindowsObject AParent,
                            LPSTR ATitle,int j):
          TWindow(AParent,ATitle)
{
 int x,y;

 Attr.Style |=WS_CHILD | WS_CAPTION | WS_CLIPSIBLINGS | WS_SYSMENU;
 Attr.X=100;
 Attr.Y=75;
 Attr.H=265+alto_titulo; //285;
 Attr.W=441;
 ayuda=j;
 switch (j) {
```

```
case 1: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA1");
            x=316;
            y=217;
            break;
case 2: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA2");
            x=195;
            y=229;
            break;
case 3: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA3");
            x=170;
            y=231;
            break;
case 4: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA4");
            x=190;
            y=225;
            break;
case 5: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA5");
            x=347;
            y=224;
            break;
case 6: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA6");
            x=90;
            y=102;
            break;
case 7: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA7");
            x=365;
            y=215;
            break;
case 8: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance,"BLOQUETPDU");
            x=118;
            y=230;
            break;
case 9: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA1");
            x=316;
            y=217;
            break;
case 10: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA2");□
            x=195;
            y=229;
            break;
case 11: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA3");
            x=170;
            y=231;
            break;
case 12: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA4");
            x=190;
            y=224;
            break;
case 13: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA5");
```

```
                x=347;
                y=224;
                break;
        case 14: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA6");
                x=90;
                y=102;
                break;
        case 15: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CAPA7");
                x=365;
                y=215;
                break;
        case 16: hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "BLOQUETPDU");
                x=118;
                y=230;
                break;
 }
 PBotoncerrar = new TBButton(this, IDCERRAR, NULL, x, y, NULL, NULL, FALSE);
SetRect(&boton,x-5,y-5,x+60,y+30);
if ((m>=0) && (m<16))
{
 niveles[m]=NULL;
 for (int k=0;k<16;k++)
 {
  if (niveles[k]!=NULL)
    InvalidateRect(niveles[k]->HWindow,&(niveles[k])->boton,TRUE);
 }
}
}

Ventanahija7::~Ventanahija7()
{
 DeleteObject(hbmpMyBitmap);
 delete(PBotoncerrar);
}

void Ventanahija7::GetWindowClass( WNDCLASS _FAR & h)
{
 TWindow::GetWindowClass(h);
 h.style |=CS_VREDRAW | CS_HREDRAW;
 h.hCursor = LoadCursor( 0, IDC_ARROW);
 }

void Ventanahija7::Paint(HDC ,PAINTSTRUCT& )
{
 Dibujarbitmap(hbmpMyBitmap,HWindow,0,0,TRUE);
}

void Ventanahija7::WMLButtonDown(RTMessage)
{
```

```
 switch (ayuda) {
          case 1: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa física");
                  break;
          case 2: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de enlace");
                  break;
          case 3: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de red");
                  break;
          case 4: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de transporte");
                  break;
          case 5: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de sesión");
                  break;
          case 6: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de presentación");
                  break;
          case 7: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de aplicación");
                  break;
          case 9: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa física");
                  break;
          case 10: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de enlace");
                   break;
          case 11: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de red");
                   break;
          case 12: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de transporte");
                   break;
          case 13: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de sesión");
                   break;
          case 14: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de presentación");
                   break;
          case 15: WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Capa de aplicación");
                   break;
  }
 InvalidateRect(HWindow,&boton,TRUE);
}

void Ventanahija7::WMMove(RTMessage Msg)
{
 InvalidateRect(HWindow,&boton,TRUE);
 for (int i=0;i<8;i++)
 {
  if (niveles[i]!=NULL)
   InvalidateRect(niveles[i]->HWindow,&(niveles[i])->boton,TRUE);
 }
}
//*************************************************************
// FIN DE LOS METODOS DE LA CLASE VENTANAHIJA7
//*************************************************************
//*************************************************************
// METODOS DE LA CLASE VENTANAHIJA8
//*************************************************************
```

```
Ventanahija8::Ventanahija8(PTWindowsObject AParent,
                           LPSTR ATitle):
          TWindow(AParent,ATitle)
{
 Attr.Style |=WS_CHILD;
 Attr.X=(ancho/2)-50; //270;
 Attr.Y=160;
 Attr.H=80;
 Attr.W=150;
 hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CONTINUO");
 hbmpMyBitmap2 = LoadBitmap(GetApplication()->hInstance, "PALANCA");
 hbmpMyBitmap3 = LoadBitmap(GetApplication()->hInstance, "MODOPULSADO");
 hbmpMyBitmap4 = LoadBitmap(GetApplication()->hInstance, "FONDO");
 botonactivo=FALSE;
 mover=FALSE;
 x=92;
 y=12;
}

Ventanahija8::~Ventanahija8()
{
 DeleteObject(hbmpMyBitmap);
 DeleteObject(hbmpMyBitmap2);
 DeleteObject(hbmpMyBitmap3);
 DeleteObject(hbmpMyBitmap4);
}

void Ventanahija8::WMLButtonDown(RTMessage Msg)
{
 POINT Punto;
 RECT boton;

 Punto.x=Msg.LP.Lo;
 Punto.y=Msg.LP.Hi;
 SetRect(&boton,x-4,y-2,x+11,y+16);

 if(PtInRect(&boton,Punto))
  mover=TRUE;
}

void Ventanahija8::WMMouseMove(RTMessage Msg)
{
 if ((mover) && (Msg.LP.Lo>90) && (Msg.LP.Lo<132))
 {
  Dibujarbitmap(hbmpMyBitmap4,HWindow,x,12,TRUE);
  Dibujarbitmap(hbmpMyBitmap3,HWindow,Msg.LP.Lo,12,TRUE);
  x=Msg.LP.Lo;
 }
}
```

```
void Ventanahija8::WMLButtonUp(RTMessage)
{
 if (mover)
 {
  mover=FALSE;
  if (x<113)
  {
   hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CONTINUO");
   arq[0]->continuo=TRUE;
   simulacion.modo[0]=TRUE;
   simulacion.modo[1]=FALSE;
   if (botonactivo)□
    delete(boton_pap);
   botonactivo=FALSE;
  }
  else
  {
   hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "PASOAPASO");
   arq[0]->continuo=FALSE;
   simulacion.modo[0]=FALSE;
   simulacion.modo[1]=TRUE;
   if (!botonactivo)
   {
    boton_pap=new Ventanahija9(Parent,NULL);
    GetApplication()->MakeWindow(boton_pap);
    botonactivo=TRUE;
   }
  }
  SendMessage(HWindow,WM_PAINT,0,0L);
 }
}

void Ventanahija8::GetWindowClass( WNDCLASS _FAR & h)
{
 TWindow::GetWindowClass(h);
 h.style |=CS_VREDRAW | CS_HREDRAW;
 h.hCursor = LoadCursor( 0, IDC_ARROW);
 }

void Ventanahija8::Paint(HDC ,PAINTSTRUCT& )
{
 Dibujarbitmap(hbmpMyBitmap,HWindow,0,0,TRUE);
 Dibujarbitmap(hbmpMyBitmap2,HWindow,x,y,TRUE);
}

//*************************************************************
// FIN DE LOS METODOS DE LA CLASE VENTANAHIJA8
//*************************************************************
```

```
//*************************************************************
// METODOS DE LA CLASE VENTANAHIJA 9
//*************************************************************

 Ventanahija9::Ventanahija9(PTWindowsObject AParent,
                                    LPSTR ATitle):
                TWindow(AParent,ATitle)
 {
  Attr.Style |=WS_CHILD;
  ShowWindow(Parent->HWindow,SW_SHOWMAXIMIZED);
  Attr.X=(ancho/2)-20;
  Attr.Y=300;
  Attr.H=64;
  Attr.W=95;
  hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "PIES1");
  PBotonpasoapaso = new TBButton(this, IDPASOAPASO, NULL, 10, 23, NULL, NULL,
FALSE);
  cont=1;
 }

 Ventanahija9::~Ventanahija9()
 {
  delete(PBotonpasoapaso);
  DeleteObject(hbmpMyBitmap);
 }

 void Ventanahija9::Paint(HDC DC,PAINTSTRUCT& )
 {
  RECT marco;
  marco.left = 1;
  marco.right =94;
  marco.top = 1;
  marco.bottom = 64;
  Dibujarrecuadros(DC,&marco);
  Dibujarbitmap(hbmpMyBitmap,HWindow,55,6,TRUE);
 }

 void Ventanahija9::GetWindowClass( WNDCLASS _FAR & h)
 {
  TWindow::GetWindowClass(h);
  h.style |=CS_VREDRAW | CS_HREDRAW;□
  h.hCursor = LoadCursor( 0, IDC_ARROW);
  h.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
 }

 void Ventanahija9::Dibujarrecuadros(HDC hDC, RECT *rect)
 {
  int x1, x2, y1, y2;
  HPEN hPen, hOldPen;
```

```
 HBRUSH hOldBrush;
 POINT pArray[3];

 hDC=GetDC(HWindow);
 x1 = rect->left;
 x2 = rect->right;
 y1 = rect->top;
 y2 = rect->bottom;

 hOldBrush = (HBRUSH)SelectObject(hDC, GetStockObject(NULL_BRUSH));
 hOldPen = (HPEN)SelectObject(hDC, GetStockObject(WHITE_PEN));□

 Rectangle(hDC, x1, y1, x2, y2);

 pArray[0].x = x1 + 2;
 pArray[1].y = pArray[0].y = y2 - 3;
 pArray[2].x = pArray[1].x = x2 - 3;
 pArray[2].y = y1 + 2;

 Polyline(hDC, pArray, 3);
 hPen = CreatePen(PS_SOLID, 1, RGB(128, 128, 128));
 SelectObject(hDC, hPen);
 pArray[0].x = x1;
 pArray[1].y = pArray[0].y = y2-1;
 pArray[2].x = pArray[1].x = x2-1;
 pArray[2].y = y1;

 Polyline(hDC, pArray, 3);
 pArray[1].x = pArray[0].x = x1 + 2;
 pArray[0].y = y2 - 3;
 pArray[2].y = pArray[1].y = y1 + 2;
 pArray[2].x = x2 - 3;
 Polyline(hDC, pArray, 3);
 SelectObject(hDC, hOldBrush);
 DeleteObject(SelectObject(hDC, hOldPen));
 ReleaseDC(HWindow,hDC);
}

void Ventanahija9::Inicializar(int indice1,int indice2)
{
 cont=1;
 secuenciaptrarq[0]->recorrido=0;
 SendMessage(secuencia[indice1],WM_PAINT,0,0L);
 SendMessage(secuencia[indice2],WM_PAINT,0,0L);
 secuenciaptrarq[0]->en_marcha=FALSE;
 secuenciaptrarq[1]->en_marcha=FALSE;
}

void Ventanahija9::Inicializartitulos()
```

```
{
 SetWindowText(secuencia[0],"Capas RM-OSI");
 SetWindowText(secuencia[1],"Máquina de estados");
 SetWindowText(secuencia[2],"Máquina de estados");
 SetWindowText(secuencia[3],"Capas RM-OSI");
}

void Ventanahija9::IDpasoapaso(RTMessage )
{
 char t[40];
 static BOOL cambio=FALSE;

 cambio=(cambio==TRUE) ? FALSE : TRUE;
 if (secuenciaptrarq[0]->inicializado)
 {
  if (cambio)
   hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "PIES1");
  else
   hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "PIES2");
  SendMessage(HWindow,WM_PAINT,0,0L);

  switch(secuenciaptrarq[0]->recorrido){
     case 1: switch(cont){
                          case 1: Rellenar(secuenciaptrmaq[0]->HWindow,RGB(0,255,0),21,17);
                                  secuenciaptrmaq[0]->Brocha(1,t,secuencia[1]);
                                  secuenciaptrmaq[0]->datos.est=conexion_saliente;
                                  SendMessage(secuencia[1],WM_PAINT,0,0L);
                                  break;
                          case 2: secuenciaptrarq[0]->Brocha(2,RGB(255,128,64),"N_CONNECT.req",secuencia[0]);
                                  break;
             case 3: secuenciaptrarq[1]->Brocha(3,RGB(255,128,64),"N_CONNECT.ind",secuencia[3]);
                                  break;
                 case 4: Rellenar(secuenciaptrmaq[1]->HWindow,RGB(0,255,0),19,30);
                            secuenciaptrmaq[1]->Brocha(2,t,secuencia[2]);
                       secuenciaptrmaq[1]->datos.est=conexion_entrante;
                        SendMessage(secuencia[2],WM_PAINT,0,0L);
                            break;
                          case 5: secuenciaptrarq[1]->Brocha(4,RGB(255,0,0),"T_CONNECT.ind",secuencia[3]);
                        secuenciaptrarq[1]->flagestado=conexion_entrante;
                                  break;
                         }
                 cont++;
                 if (cont==6)
                 {
                  secuenciaptrarq[0]->flagestado=conexion_saliente;
            Inicializar(0,3);
                 }
                 break;
```

```
case 2: switch(cont){
                    case 1: Rellenar(secuenciaptrmaq[1]->HWindow,RGB(0,255,0),21,17);
                            secuenciaptrmaq[1]->Brocha(4,t,secuencia[2]);
                    secuenciaptrmaq[1]->datos.est=transferencia_datos;
                            SendMessage(secuencia[2],WM_PAINT,0,0L);
                            break;
                    case 2: secuenciaptrarq[1]->Brocha(2,RGB(255,128,64),"N_CONNECT.resp",secuencia[3]);
                            break;
      case 3: secuenciaptrarq[0]->Brocha(3,RGB(255,128,64),"N_CONNECT.confirm",secuencia[0]);
                            break;
                    case 4: Rellenar(secuenciaptrmaq[0]->HWindow,RGB(0,255,0),19,30);
                            secuenciaptrmaq[0]->Brocha(3,t,secuencia[1]);
                            secuenciaptrmaq[0]->datos.est=transferencia_datos;
                            SendMessage(secuencia[1],WM_PAINT,0,0L);
                            break;
                    case 5: secuenciaptrarq[0]->Brocha(4,RGB(255,0,0),"T_CONNECT.confirm",secuencia[0]);
                            secuenciaptrarq[0]->flagestado=transferencia_datos;
                            break;
          }
    cont++;
          if (cont==6)
          {
           secuenciaptrarq[0]->flagestado=transferencia_datos;
           Inicializar(3,0);
          }
          break;

case 3: switch(cont){
                    case 1: Rellenar(secuenciaptrmaq[0]->HWindow,RGB(0,255,0),21,17);
                            secuenciaptrmaq[0]->Brocha(7,t,secuencia[1]);
                            break;
                    case 2: secuenciaptrarq[0]->Brocha(2,RGB(255,128,64),"N_DATA.request",secuencia[0]);
                            break;
      case 3: secuenciaptrarq[1]->Brocha(3,RGB(255,128,64),"N_DATA.ind",secuencia[3]);
                            break;
                    case 4: Rellenar(secuenciaptrmaq[1]->HWindow,RGB(0,255,0),19,30);
                          secuenciaptrmaq[1]->Brocha(7,t,secuencia[2]);
        break;
                    case 5: secuenciaptrarq[1]->Brocha(4,RGB(255,0,0),"T_DATA.ind",secuencia[3]);
          break;
          }
          cont++;
          if (cont==6)
           Inicializar(0,3);
          break;

case 4: switch(cont){
                    case 1: Rellenar(secuenciaptrmaq[0]->HWindow,RGB(0,255,0),21,17);
                          secuenciaptrmaq[0]->Brocha(6,t,secuencia[1]);
```

```
                    secuenciaptrmaq[0]->datos.est=Inactivo;
                    SendMessage(secuencia[1],WM_PAINT,0,0L);
                        break;
                case 2: secuenciaptrarq[0]->Brocha(2,RGB(255,128,64),"N_DISCONNECT.req",secuencia[0]);
                        break;
        case 3: secuenciaptrarq[1]->Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);
                        break;
                case 4: Rellenar(secuenciaptrmaq[1]->HWindow,RGB(0,255,0),19,30);
                    secuenciaptrmaq[1]->Brocha(5,t,secuencia[2]);
                        secuenciaptrmaq[1]->datos.est=Inactivo;
                    SendMessage(secuencia[2],WM_PAINT,0,0L);
            break;
                case 5: secuenciaptrarq[1]->Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[3]);
                    secuenciaptrarq[1]->flagestado=Inactivo;
            break;
            }
            cont++;
            if (cont==6)
            {
             Inicializar(0,3);
          secuenciaptrarq[0]->contador=1;
            secuenciaptrarq[0]->inicializado=FALSE;
         Inicializartitulos();
            }
            break;

case 5: switch(cont){
                case 1: Rellenar(secuenciaptrmaq[1]->HWindow,RGB(0,255,0),21,17);
                    secuenciaptrmaq[1]->Brocha(5,t,secuencia[2]);
                    secuenciaptrmaq[1]->datos.est=Inactivo;
                        SendMessage(secuencia[2],WM_PAINT,0,0L);
                        break;
                case 2: secuenciaptrarq[1]->Brocha(2,RGB(255,128,64),"N_DISCONNECT.req",secuencia[3]);
                    break;
        case 3: secuenciaptrarq[0]->Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                    break;
                case 4: Rellenar(secuenciaptrmaq[0]->HWindow,RGB(0,255,0),19,30);
                    secuenciaptrmaq[0]->Brocha(6,t,secuencia[1]);
                        secuenciaptrmaq[0]->datos.est=Inactivo;
                        SendMessage(secuencia[1],WM_PAINT,0,0L);
                        break;
                case 5: secuenciaptrarq[0]->Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[0]);
                        secuenciaptrarq[0]->flagestado=Inactivo;
                        break;
            }
            cont++;
            if (cont==6)
            {
             Inicializar(3,0);
```

```
            secuenciaptrarq[0]->contador=1;
            secuenciaptrarq[0]->inicializado=FALSE;
        Inicializartitulos();
            }
            break;

case 6: switch(cont){
                case 1: Rellenar(secuenciaptrmaq[0]->HWindow,RGB(0,255,0),21,17);
                        secuenciaptrmaq[0]->Brocha(8,t,secuencia[1]);
                        secuenciaptrmaq[0]->datos.est=Inactivo;
                            SendMessage(secuencia[1],WM_PAINT,0,0L);
                            break;
                case 2: secuenciaptrarq[0]->Brocha(2,RGB(255,128,64),"N_DISCONNECT.req",secuencia[0]);
                            break;
      case 3: secuenciaptrarq[1]->Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[3]);□
                            break;
                case 4: Rellenar(secuenciaptrmaq[1]->HWindow,RGB(0,255,0),19,30);
                        secuenciaptrmaq[1]->Brocha(8,t,secuencia[2]);
                        secuenciaptrmaq[1]->datos.est=Inactivo;
                            SendMessage(secuencia[2],WM_PAINT,0,0L);
                            break;
                case 5: secuenciaptrarq[1]->Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[3]);
                            secuenciaptrarq[1]->flagestado=Inactivo;
                            break;
            }
    cont++;
            if (cont==6)
            {
             Inicializar(0,3);
            secuenciaptrarq[0]->contador=1;
            secuenciaptrarq[0]->inicializado=FALSE;
        Inicializartitulos();
            }
            break;

case 7: switch(cont){
                case 1: Rellenar(secuenciaptrmaq[1]->HWindow,RGB(0,255,0),21,17);
                        secuenciaptrmaq[1]->Brocha(8,t,secuencia[2]);
                    secuenciaptrmaq[1]->datos.est=Inactivo;
                        SendMessage(secuencia[2],WM_PAINT,0,0L);
                            break;
                case 2: secuenciaptrarq[1]->Brocha(2,RGB(255,128,64),"N_DISCONNECT.req",secuencia[3]);
                            break;
      case 3: secuenciaptrarq[0]->Brocha(3,RGB(255,128,64),"N_DISCONNECT.ind",secuencia[0]);
                            break;
                case 4: Rellenar(secuenciaptrmaq[0]->HWindow,RGB(0,255,0),19,30);
                    secuenciaptrmaq[0]->Brocha(8,t,secuencia[1]);
                            secuenciaptrmaq[0]->datos.est=Inactivo;
                    SendMessage(secuencia[1],WM_PAINT,0,0L);
```

```
                                break;
                        case 5: secuenciaptrarq[0]->Brocha(4,RGB(255,0,0),"T_DISCONNECT.ind",secuencia[0]);
                            secuenciaptrarq[0]->flagestado=Inactivo;
                            break;
                    }
                cont++;
                    if (cont==6)
                    {
                     Inicializar(3,0);
                     secuenciaptrarq[0]->contador=1;
                     secuenciaptrarq[0]->inicializado=FALSE;
                 Inicializartitulos();
                    }
                    break;
  }
 }
}

//*************************************************************
// FIN DE LOS METODOS DE LA CLASE VENTANAHIJA9
//*************************************************************
//*************************************************************
// METODOS DE LA CLASE VENTANAPRINCIPAL
//*************************************************************
 void VentanaPrincipal::Paint(HDC ,PAINTSTRUCT& )
 {
 if (todasactivas)
  Dibujarbitmap(hbmpMyBitmap,HWindow,abs((alto-462)/3)+115+(alto-407)/3,abs((alto-462)/3)+115,FALSE);
 }

 void VentanaPrincipal::Modofuncionamiento(RTMessage)
 {
 if (GetModule()->ExecDialog(new Cdialogos7(this,"MODO_SIMULACION"))==IDOK)
 {
  arq[0]->continuo=simulacion.modo[0];
  if (simulacion.modo[0])
  {
   modo->hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "CONTINUO");
   modo->x=92;
   if (botonactivo)
     delete(boton_pap);
   botonactivo=FALSE;
  }
  else
  {
   modo->hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "PASOAPASO");
   modo->x=124;
   if (!botonactivo)
   {
```

```
   boton_pap=new Ventanahija9(this,"botón");
   GetApplication()->MakeWindow(boton_pap);
   botonactivo=TRUE;
  }
 }
}
}

void VentanaPrincipal::Velocity(RTMessage)
{
 GetModule()->ExecDialog(new Velocidad(this,"VELOCIDAD"));
}

void VentanaPrincipal::Activarregistro(RTMessage)
{
 GetModule()->ExecDialog(new Cdialogos6(this,"REGISTRO"));
}

void VentanaPrincipal::Activarmaquinas(RTMessage)
{
 HMENU menu;

 delete(Barra);
 barraprincipal=FALSE;
 Barra=new TVentanaBarra(this,NULL,2);
 GetApplication()->MakeWindow(Barra);
 fondo=new Ventanafondo(this,NULL);
 GetApplication()->MakeWindow(fondo);
 vh1activas=TRUE;
 for (int i=0;i<2;i++)
 {
 maquinas[i]=new Ventanahija1(fondo,"Maquina de estados",i,TRUE);
 GetApplication()->MakeWindow(maquinas[i]);
 }
 AssignMenu(16);
 SetWindowText(HWindow,"Simulación interactiva del protocolo de transporte OSI");
};

void VentanaPrincipal::Ordenarventanas(RTMessage)
{□
MoveWindow(maquinas[0]->HWindow,40,48,228,198,TRUE);
MoveWindow(maquinas[1]->HWindow,ancho-268,48,228,198,TRUE);
if (TPDUactiva)
 MoveWindow(TPDU->HWindow,abs((ancho-450)/2),(abs((alto-344)/2))+180,450,130,TRUE);
 }

void VentanaPrincipal::Activararq(RTMessage)
{
HMENU menu;
```

```
 delete(Barra);
 barraprincipal=FALSE;
 Barra=new TVentanaBarra(this,NULL,1);
 GetApplication()->MakeWindow(Barra);
 fondo=new Ventanafondo(this,NULL);
 GetApplication()->MakeWindow(fondo);
 modeloactivo=TRUE;
 modelo=new MROSI(fondo,"Tutorial R.M. OSI");
 GetApplication()->MakeWindow(modelo);
 AssignMenu(14);
 SetWindowText(HWindow,"Tutorial del modelo de referencia OSI");
};



void VentanaPrincipal::ActivarstructTPDU(RTMessage)
{
 HMENU menu;

 Infoactiva=TRUE;

 Info=new Ventanahija3(fondo,"Información");
 GetApplication()->MakeWindow(Info);
 menu=GetMenu(HWindow);
 EnableMenuItem(menu,5,MF_BYCOMMAND | MF_GRAYED);
 if (TPDUactiva)
  EnableMenuItem(menu,CM_CERRARAMBAS,MF_BYCOMMAND | MF_ENABLED);
 else
  EnableMenuItem(menu,CM_CERRARAMBAS,MF_BYCOMMAND | MF_GRAYED);
 EnableMenuItem(menu,CM_AMBAS,MF_BYCOMMAND | MF_GRAYED);
 EnableMenuItem(menu,CM_CERRAREST,MF_BYCOMMAND | MF_ENABLED);
 EnableWindow(Barra->PBotonstruct->HWindow,FALSE);
}

void VentanaPrincipal::CerrarstructTPDU(RTMessage)
{
 HMENU menu;

 Infoactiva=FALSE;
 delete(Info);
 menu=GetMenu(HWindow);
 EnableMenuItem(menu,CM_CERRAREST,MF_BYCOMMAND | MF_GRAYED);
 EnableMenuItem(menu,5,MF_BYCOMMAND | MF_ENABLED);
EnableMenuItem(menu,CM_CERRARAMBAS,MF_BYCOMMAND | MF_GRAYED);
 if (Infoactiva)
   EnableMenuItem(menu,CM_AMBAS,MF_BYCOMMAND | MF_GRAYED);
 else
   EnableMenuItem(menu,CM_AMBAS,MF_BYCOMMAND | MF_ENABLED);
```

```
 EnableWindow(Barra->PBotonstruct->HWindow,TRUE);
}

void VentanaPrincipal::Activarambas(RTMessage)
{
 HMENU menu;

 Infoactiva=TRUE;
 TPDUactiva=TRUE;
 TPDU=new Ventanahija4(fondo,"Formato de la TPDU");
 GetApplication()->MakeWindow(TPDU);
 Info=new Ventanahija3(fondo,"Información");
 GetApplication()->MakeWindow(Info);
 menu=GetMenu(HWindow);
 EnableMenuItem(menu,5,MF_BYCOMMAND | MF_GRAYED);
 EnableMenuItem(menu,CM_TPDU,MF_BYCOMMAND | MF_GRAYED);
 EnableMenuItem(menu,CM_AMBAS,MF_BYCOMMAND | MF_GRAYED);
 EnableMenuItem(menu,CM_CERRAREST,MF_BYCOMMAND | MF_ENABLED);
 EnableMenuItem(menu,CM_CERRARTPDU,MF_BYCOMMAND | MF_ENABLED);
 EnableMenuItem(menu,CM_CERRARAMBAS,MF_BYCOMMAND | MF_ENABLED);
 EnableWindow(Barra->PBotontpdu->HWindow,FALSE);
 EnableWindow(Barra->PBotonstruct->HWindow,FALSE);
}

void VentanaPrincipal::Cerrarambas(RTMessage)
{
 HMENU menu;

 Infoactiva=FALSE;
 TPDUactiva=FALSE;
 delete(Info);
 delete(TPDU);
 menu=GetMenu(HWindow);
 EnableMenuItem(menu,5,MF_BYCOMMAND | MF_ENABLED);
 EnableMenuItem(menu,CM_TPDU,MF_BYCOMMAND | MF_ENABLED);
 EnableMenuItem(menu,CM_AMBAS,MF_BYCOMMAND | MF_ENABLED);
 EnableMenuItem(menu,CM_CERRAREST,MF_BYCOMMAND | MF_GRAYED);
 EnableMenuItem(menu,CM_CERRARTPDU,MF_BYCOMMAND | MF_GRAYED);
 EnableMenuItem(menu,CM_CERRARAMBAS,MF_BYCOMMAND | MF_GRAYED);
 EnableWindow(Barra->PBotontpdu->HWindow,TRUE);
 EnableWindow(Barra->PBotonstruct->HWindow,TRUE);
}

void VentanaPrincipal::ActivarTPDU(RTMessage)
{
 HMENU menu;

 TPDUactiva=TRUE;
 TPDU=new Ventanahija4(fondo,"Formato de la TPDU");
```

```
GetApplication()->MakeWindow(TPDU);
menu=GetMenu(HWindow);
EnableMenuItem(menu,CM_TPDU,MF_BYCOMMAND | MF_GRAYED);
EnableMenuItem(menu,CM_AMBAS,MF_BYCOMMAND | MF_GRAYED);
EnableMenuItem(menu,CM_CERRARTPDU,MF_BYCOMMAND | MF_ENABLED);
if (Infoactiva)
 EnableMenuItem(menu,CM_CERRARAMBAS,MF_BYCOMMAND | MF_ENABLED);
else
 EnableMenuItem(menu,CM_CERRARAMBAS,MF_BYCOMMAND | MF_GRAYED);
EnableWindow(Barra->PBotontpdu->HWindow,FALSE);
}

void VentanaPrincipal::CerrarTPDU(RTMessage)
{
HMENU menu;

TPDUactiva=FALSE;
delete(TPDU);
menu=GetMenu(HWindow);
EnableMenuItem(menu,CM_TPDU,MF_BYCOMMAND | MF_ENABLED);
EnableMenuItem(menu,CM_CERRARTPDU,MF_BYCOMMAND | MF_GRAYED);
EnableMenuItem(menu,CM_CERRARAMBAS,MF_BYCOMMAND | MF_GRAYED);
if (Infoactiva)
 EnableMenuItem(menu,CM_AMBAS,MF_BYCOMMAND | MF_GRAYED);
else
 EnableMenuItem(menu,CM_AMBAS,MF_BYCOMMAND | MF_ENABLED);
EnableWindow(Barra->PBotontpdu->HWindow,TRUE);
}

void VentanaPrincipal::Activarventanas(RTMessage)
{
HMENU menu;

todasactivas=TRUE;
delete(Barra);
barraprincipal=FALSE;
ShowWindow(HWindow,SW_SHOWMAXIMIZED);
Barra=new TVentanaBarra(this,NULL,1);
GetApplication()->MakeWindow(Barra);
maquinas[0]=new Ventanahija1(this,"Máquina de estados",0,FALSE);
arq[0]=new Simvisual(this,"Capas R.M. OSI",0);
arq[1]=new Simvisual(this,"Capas R.M. OSI",1);
maquinas[1]=new Ventanahija1(this,"Máquina de estados",1,FALSE);
modo=new Ventanahija8(this,"Modo");
GetApplication()->MakeWindow(modo);

for (int i=0;i<2;i++)
 GetApplication()->MakeWindow(maquinas[i]);
```

```
 for (i=0;i<2;i++)
   GetApplication()->MakeWindow(arq[i]);

 AssignMenu(31);
 SetWindowText(HWindow,"Simulación visual del nivel y máquina de transporte");
 hbmpMyBitmap = LoadBitmap(GetApplication()->hInstance, "FLECHA");
};

void VentanaPrincipal::Acercade(RTMessage)
{
 GetModule()->ExecDialog(new Cdialogos9(this,"DIALOG_1"));
};

void VentanaPrincipal::Uso(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_HELPONHELP,0);
};

void VentanaPrincipal::Indice(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_INDEX,0L);
}

void VentanaPrincipal::TutorialRMOSI(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Tutorial OSI");
}

void VentanaPrincipal::Ayuda_visual(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Simulación visual de la máquina de estados y la capa de
transporte");
}

void VentanaPrincipal::Ejemplo_Ayuda(RTMessage)
{
   WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Tutorial: Ejemplo de ejecución en Simulación
interactiva");
}

void VentanaPrincipal::Ayuda_interactiva(RTMessage)
{
 WinHelp(HWindow,"ADAN.HLP",HELP_KEY,(unsigned long) (LPSTR) "Simulacióninteractiva del protocolo de transporte OSI");
}

void VentanaPrincipal::WMTimer(RTMessage)
{
 PuntoAnterior=Punto;
 GetCursorPos(&Punto);
```

```
if ( !(WindowFromPoint(Punto)==WindowFromPoint(PuntoAnterior)) && (flaglineaestado) )
{
 flaglineaestado=FALSE;
 Barra->Mostrarmensaje(" ");
}


if(!(WindowFromPoint(Punto)==WindowFromPoint(PuntoAnterior)) && (FlagVentana_Info)){
  FlagVentana_Info=0;
  DestroyWindow(Ventana_flotante->HWindow);
  Barra->Mostrarmensaje(" ");
}

if (barraprincipal)
{

 if((Barra->PBotonsalida->HWindow==WindowFromPoint(Punto))
                                  && (!FlagVentana_Info)){
          Ventana_flotante = new TVentana_flotante (this, NULL);
          Ventana_flotante->Attr.W=40;
          Ventana_flotante->Attr.X=90;
          Ventana_flotante->Attr.Y=30;
          GetApplication()->MakeWindow(Ventana_flotante);
          FlagVentana_Info=1;
          Boton=1;
          SetWindowPos(Ventana_flotante->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
          Barra->Mostrarmensaje("Sale de la aplicación");
 }

 if((Barra->PBotonAyuda->HWindow==WindowFromPoint(Punto))
                                  && (!FlagVentana_Info)){
          Ventana_flotante = new TVentana_flotante (this, NULL);
          Ventana_flotante->Attr.W=52;
          Ventana_flotante->Attr.X=120;
          Ventana_flotante->Attr.Y=30;
          GetApplication()->MakeWindow(Ventana_flotante);
          FlagVentana_Info=1;
          Boton=2;
          SetWindowPos(Ventana_flotante->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
          Barra->Mostrarmensaje("Ejecuta el sistema de Ayuda de la aplicación");
 }

 if((Barra->PBotonmaquinas->HWindow==WindowFromPoint(Punto))
                                  && (!FlagVentana_Info)){
          Ventana_flotante = new TVentana_flotante (this, NULL);
          Ventana_flotante->Attr.W=158;
         Ventana_flotante->Attr.X=30;
          Ventana_flotante->Attr.Y=30;
```

```
            GetApplication()->MakeWindow(Ventana_flotante);
            FlagVentana_Info=1;
            Boton=3;
            SetWindowPos(Ventana_flotante->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
            Barra->Mostrarmensaje("Funcionamiento de la capa de transporte a nivel
interactivo");
   }

  if((Barra->PBotonarquitecturas->HWindow==WindowFromPoint(Punto))
                                    && (!FlagVentana_Info)){
            Ventana_flotante = new TVentana_flotante (this, NULL);
            Ventana_flotante->Attr.W=118;
            Ventana_flotante->Attr.X=60;
            Ventana_flotante->Attr.Y=30;
            GetApplication()->MakeWindow(Ventana_flotante);
            FlagVentana_Info=1;
            Boton=4;
            SetWindowPos(Ventana_flotante->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
            Barra->Mostrarmensaje("Información general del modelo OSI");
   }

  if((Barra->PBotonvisual->HWindow==WindowFromPoint(Punto))
                                    && (!FlagVentana_Info)){
            Ventana_flotante = new TVentana_flotante (this, NULL);
            Ventana_flotante->Attr.W=128;
            Ventana_flotante->Attr.X=0;
            Ventana_flotante->Attr.Y=30;
            GetApplication()->MakeWindow(Ventana_flotante);
            FlagVentana_Info=1;
            Boton=6;
            SetWindowPos(Ventana_flotante->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
            Barra->Mostrarmensaje("Funcionamiento de la capa de transporte sólo a nivel
visual");
   }
 }

 if ((vh1activas) || (modeloactivo) || (todasactivas))
 {
  if((Barra->PBotonsalida->HWindow==WindowFromPoint(Punto))
                                    && (!FlagVentana_Info)){
            Ventana_flotante = new TVentana_flotante (this, NULL);
            Ventana_flotante->Attr.W=40;
            Ventana_flotante->Attr.X=((todasactivas) ? 30 : 30);
            Ventana_flotante->Attr.Y=((todasactivas) ? 35 : 30);
            GetApplication()->MakeWindow(Ventana_flotante);
            FlagVentana_Info=1;
            Boton=1;
            SetWindowPos(Ventana_flotante->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
            Barra->Mostrarmensaje("Sale de la aplicación");
```

```
  }

 if((Barra->PBotonAyuda->HWindow==WindowFromPoint(Punto)) && (!FlagVentana_Info)){
            Ventana_flotante = new TVentana_flotante (this, NULL);
            Ventana_flotante->Attr.W=52;
            Ventana_flotante->Attr.X=((todasactivas) ? 30 : 60);
            Ventana_flotante->Attr.Y=((todasactivas) ? 65 : 30);
            GetApplication()->MakeWindow(Ventana_flotante);
            FlagVentana_Info=1;
            Boton=2;
            SetWindowPos(Ventana_flotante->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
            Barra->Mostrarmensaje("Ejecuta el sistema de Ayuda de la aplicación");
  }

 if((Barra->PBotonnuevo->HWindow==WindowFromPoint(Punto)) && (!FlagVentana_Info)){
            Ventana_flotante = new TVentana_flotante (this, NULL);
            Ventana_flotante->Attr.W=52;
            Ventana_flotante->Attr.X=((todasactivas) ? 30 : 2);
            Ventana_flotante->Attr.Y=((todasactivas) ? 5 : 30);
            GetApplication()->MakeWindow(Ventana_flotante);
            FlagVentana_Info=1;
            Boton=5;
            SetWindowPos(Ventana_flotante->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
            Barra->Mostrarmensaje("Retorna a la ventana principal para nueva opción");
            }
}

if (modeloactivo==TRUE)
{
 if(modelo->HWindow==WindowFromPoint(Punto))
 {
  if (!flaglineaestado)
  {
   Barra->Mostrarmensaje("Hacer click en las capas para más información");
   flaglineaestado=TRUE;
  }
  cambio_cursor=Punto;
  ScreenToClient(modelo->HWindow,&cambio_cursor);
  if ((PtInRect(&(modelo->TPDU1),cambio_cursor)) || (PtInRect(&(modelo->TPDU2),cambio_cursor)))
  {
   SetCursor(LoadCursor( GetApplication()->hInstance, "apuntar"));
   Barra->Mostrarmensaje("Muestra la formación de la TPDU");
   flaglineaestado=TRUE;
  }
 }
}

if (todasactivas)
{
```

```
  if (modo->HWindow==WindowFromPoint(Punto))
  {
   ScreenToClient(modo->HWindow,&Punto);
   if (((Punto.y)>8) && ((Punto.y)<24)  && ((Punto.x)>84) && ((Punto.x)<137))
    SetCursor(LoadCursor( GetApplication()->hInstance, "apuntar"));
   }
  }

 if(vh1activas)
 {
  if ( ((maquinas[0]->HWindow==WindowFromPoint(Punto)) || (maquinas[1]-
>HWindow==WindowFromPoint(Punto))) && (!flaglineaestado))
  {
   Barra->Mostrarmensaje("Pulsar botón izdo. del ratón para directivas,derecho para
configurar");
   flaglineaestado=TRUE;
  }
  if ((TPDUactiva) && (TPDU->HWindow==WindowFromPoint(Punto)) && (!flaglineaestado))
  {
   Barra->Mostrarmensaje("Hacer click en los campos para modificarlos.");
   flaglineaestado=TRUE;
  }

  if((Barra->PBotontpdu->HWindow==WindowFromPoint(Punto))
                              && (!FlagVentana_Info)){
                        Ventana_flotante = new TVentana_flotante (this, NULL);
                        Ventana_flotante->Attr.W=50;
                        Ventana_flotante->Attr.X=90;
                        Ventana_flotante->Attr.Y=30;
                        GetApplication()->MakeWindow(Ventana_flotante);
                        FlagVentana_Info=1;
                        Boton=7;
                        SetWindowPos(Ventana_flotante-
>HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
                        Barra->Mostrarmensaje("Muestra una TPDU general modificable por el
usuario");
  }

  if((Barra->PBotonstruct->HWindow==WindowFromPoint(Punto))&& (!FlagVentana_Info)){
                        Ventana_flotante = new TVentana_flotante (this, NULL);
                        Ventana_flotante->Attr.W=160;
                        Ventana_flotante->Attr.X=120;
                        Ventana_flotante->Attr.Y=30;
                        GetApplication()->MakeWindow(Ventana_flotante);
                        FlagVentana_Info=1;
                        Boton=8;
                        SetWindowPos(Ventana_flotante->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
                        Barra->Mostrarmensaje("Muestra los campos de la TPDU durante la
ejecución");
```

```
  }

  if((Barra->PBotonordenar->HWindow==WindowFromPoint(Punto)) && (!FlagVentana_Info)){
                        Ventana_flotante = new TVentana_flotante (this, NULL);
                        Ventana_flotante->Attr.W=130;
                        Ventana_flotante->Attr.X=150;
                        Ventana_flotante->Attr.Y=30;
                        GetApplication()->MakeWindow(Ventana_flotante);
                        FlagVentana_Info=1;
                        Boton=9;
                        SetWindowPos(Ventana_flotante->HWindow,HWND_TOP,1,1,5,5,SWP_NOMOVE | SWP_NOSIZE);
                        Barra->Mostrarmensaje("Coloca las ventanas en su posición original");
  }
 }
}

void VentanaPrincipal::WMCreate(RTMessage )
{
 int i,j,k;
 PTPresentacion A;
 DWORD time=GetTickCount();

 A=new TPresentacion(this);
 GetModule()->MakeWindow(A);    //Se crea una caja de diálogo Modeless.

 do {
  ;
 }while ((GetTickCount()-time)<2000);

  MessageBeep(0);

 delete(A);  // Se Borra.

 Barra=new TVentanaBarra(this,NULL,0);
 GetApplication()->MakeWindow(Barra);
 Barra->Mostrarmensaje("Alumno: Fco. Javier Fleitas Negrín  Tutor: Domingo Marrero");
 SetTimer(HWindow,1, 50,NULL);
}

void VentanaPrincipal::Calculardimensiones()
{
 GetWindowRect(HWindow,&coordenadas_ventana);
 ancho=coordenadas_ventana.right-coordenadas_ventana.left;
 izda=coordenadas_ventana.left;
 arriba=coordenadas_ventana.top;
 abajo=coordenadas_ventana.bottom;
 alto=abajo-arriba;
}
```

```
void VentanaPrincipal::WMSize(RTMessage)
{
 Calculardimensiones();
 MoveWindow(Barra->HWindow,0,0,((todasactivas) ? 29 : ancho),((todasactivas) ? 90 :
29),TRUE);

 if (modeloactivo)
 {
  MoveWindow(fondo->HWindow,0,30,ancho,alto-alto_barra,TRUE);
  MoveWindow(modelo->HWindow,abs((ancho-541)/2),abs((alto-alto_barra-alto_titulo-388)/2),541,358+alto_titulo,TRUE);
 }

 if (vh1activas)
   MoveWindow(fondo->HWindow,0,30,ancho,alto-alto_barra,TRUE);

 if (ancho<460)
  MessageBox(HWindow,"Perderá la visión global del proceso si disminuye tanto el ancho de la ventana.","Recomendación",MB_OK |
MB_ICONHAND);
}

 void VentanaPrincipal::Salir(RTMessage)
 {
  CloseWindow();
 };

 void VentanaPrincipal::limpieza()
 {
  int i;

  delete(Barra);
  if (todasactivas==TRUE)
  {
   DeleteObject(hbmpMyBitmap);
   for (i=0;i<2;i++)
    delete(maquinas[i]);
   for (i=0;i<2;i++)
    delete(arq[i]);
   delete modo;
   if (botonactivo)
    delete(boton_pap);
   todasactivas=FALSE;
   InvalidateRect(HWindow,NULL,TRUE);
  }

  if (modeloactivo==TRUE)
  {
   delete(fondo);
   modeloactivo=FALSE;
  }
```

```
if (Infoactiva==TRUE)
{
 delete(Info);
 Infoactiva=FALSE;
}

if (TPDUactiva==TRUE)
{
 delete(TPDU);
 TPDUactiva=FALSE;
}

if (vh1activas==TRUE)
{
 for (i=0;i<2;i++)
  delete(maquinas[i]);
 delete(fondo);
 vh1activas=FALSE;
}

 barraprincipal=TRUE;
 Barra=new TVentanaBarra(this,NULL,0);
 GetApplication()->MakeWindow(Barra);
}

void VentanaPrincipal::Nuevo(RTMessage)
{
 HMENU res;
 limpieza();
 SendMessage(HWindow,WM_PAINT,0,0L);
 AssignMenu(8);
 SetWindowText(HWindow,"Simulador del protocolo de transporte OSI");
};

BOOL VentanaPrincipal::CanClose()
{
 int eleccion ;
 HMENU hmenu;

 eleccion=MessageBox(HWindow,"¿Desea abandonar esta
 aplicación?","Pregunta",MB_YESNO | MB_ICONQUESTION)==IDYES;
 hmenu=GetMenu(HWindow);
 if (GetMenuState(hmenu,9,MF_BYCOMMAND)==MF_ENABLED && eleccion)
 {
  limpieza();
  delete(Barra);
 }
WinHelp(HWindow,"ADAN.HLP",HELP_QUIT,0L);□
```

```
return eleccion;
}


void VentanaPrincipal::GetWindowClass( WNDCLASS _FAR & w)
{
HINSTANCE Instancia;
TWindow::GetWindowClass(w);
Instancia=VentanaPrincipal::GetModule()-> hInstance;
w.hIcon=LoadIcon(Instancia,"ICON_5");
w.hbrBackground=(CreateSolidBrush(RGB(191,191,191)));
};

VentanaPrincipal::VentanaPrincipal(PTWindowsObject AParent,LPSTR ATitle):
TWindow(AParent,ATitle)
 {
  int MousePresent;
  HDC hdcInfo;
  int resultado;
  TEXTMETRIC tm;
  char num[7];

  AssignMenu(8);
  Attr.Style |= WS_MAXIMIZE ;
  vh1activas=FALSE;
  modeloactivo=FALSE;
  todasactivas=FALSE;
  Infoactiva=FALSE;
  TPDUactiva=FALSE;
  barraprincipal=TRUE;
  simulacion.modo[0]=TRUE;
  simulacion.modo[1]=FALSE;
  cont=1;
  MousePresent=GetSystemMetrics(SM_MOUSEPRESENT);
  if (!MousePresent)
       MessageBox(HWindow,"El ratón no está instalado. No se aprovechará al 100% elprograma.","¡¡ ERROR !!",MB_OK |
MB_ICONHAND);
  alto_menu=GetSystemMetrics(SM_CYMENU);
  alto_titulo=GetSystemMetrics(SM_CYCAPTION);
  alto_barra=alto_menu+alto_titulo+4;
  FlagVentana_Info=0;
  flaglineaestado=FALSE;
  hdcInfo=CreateIC("DISPLAY",NULL,NULL,NULL);
  resultado=GetDeviceCaps(hdcInfo,NUMCOLORS);
  if (resultado<100)
   MessageBox(HWindow,"La configuración actual de colores no es la ideal, los bitmaps perderán vistosidad. Si quiere mejorar calidad
y su tarjeta se lo permite configure Windows a 64k colores.","Información", MB_OK | MB_ICONINFORMATION);
  GetTextMetrics(hdcInfo,&tm);
  if (tm.tmHeight>19)
```

```
    MessageBox(HWindow,"Windows está configurado con FONT tipo Large. La resolución ideal es 640x480 pixels y FONT normal,
pero podrá trabajar perfectamente con la seleccionada actualmente.", "Información" , MB_OK | MB_ICONINFORMATION);
   DeleteDC(hdcInfo);
   };

VentanaPrincipal::~VentanaPrincipal()
{
 KillTimer(HWindow,1);
}

//************************************************************
// FIN DE LOS METODOS DE LA CLASE VENTANAPRINCIPAL
//************************************************************

//************************************************************
// METODOS DE LA CLASE TMIPROYECTO
//************************************************************

 TMiproyecto::TMiproyecto(LPSTR AName, HINSTANCE hInstance, HINSTANCE
hPrevInstance,  LPSTR lpCmdLine, int nCmdShow) :
 TApplication(AName, hInstance, hPrevInstance, lpCmdLine, nCmdShow)
  {
   strcpy(Tituloventana,"Simulador del protocolo de transporte OSI ( Ventana secundaria )");
  };

void TMiproyecto::InitApplication()
{
 strcpy(Tituloventana,"Simulador del protocolo de transporte OSI");
}

void TMiproyecto::InitMainWindow()
{
 MainWindow = new VentanaPrincipal(NULL, Tituloventana);
}

#pragma argsused
int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,LPSTR lpCmdLine, int nCmdShow)
{
 TMiproyecto Miproyecto("Simulador del nivel de transporte", hInstance,   hPrevInstance, lpCmdLine, SW_MAXIMIZE);
  Miproyecto.Run();
 return Miproyecto.Status;
}
```