Esta tesis doctoral contiene un índice que enlaza a cada uno de los capítulos de la misma.
Existen asimismo botones de retorno al índice al principio y final de cada uno de los capítulos.

Ir directamente al índice

Para una correcta visualización del texto es necesaria la versión de Adobe Acrobat Reader 7.0 o posteriores

Aquesta tesi doctoral conté un índex que enllaça a cadascun dels capítols. Existeixen així mateix botons de retorn a l'index al principi i final de cadascun deis capítols .

Anar directament a l'index

Per a una correcta visualització del text és necessària la versió d' Adobe Acrobat Reader 7.0 o posteriors.

TESIS DOCTORAL

202

DISEÑO DE UNA RED DE ORDENADORES APLICADA AL CONTROL DE PROCESOS REMOTOS

Presentada en el

Departamento de Ingeniería de Sistemas y Comunicaciones
ESCUELA POLITECNICA SUPERIOR
Universidad de Alicante

Para la obtención del grado de

DOCTOR INGENIERO EN INFORMATICA

Doctorando: **Julio L. Rosa Herranz**
Director: **Dr. D. Carlos Pastor Antón**

Alicante, 1997

UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
DEPARTAMENT D'ENGINYERIA DE SISTEMES I COMUNICACIONS
DEPARTAMENTO DE INGENIERIA DE SISTEMAS Y COMUNICACIONES

Ap. Correus 99 -:- E-03080 ALACANT -:- Telf. 96 / 5903682 -:- Fax 96 / 5903464

CARLOS PASTOR ANTÓN, Catedrático de Escuela Universitaria y Doctor en Ciencias Físicas, como Director de la tesis doctoral «**Diseño de una red de ordenadores aplicada al control de procesos remotos**», realizada por D. Julio L. Rosa Herranz,

AUTORIZA, la presentación de la misma para que se proceda a su lectura.

Alicante, 10 de marzo de 1997

Fdo.: Carlos Pastor Antón

A Concha, Sergio y Raúl

AGRADECIMIENTOS

Quisiera recordar a las personas que, de una u otra forma, han prestado su abnegado apoyo para que esta tesis pudiera llegar a su conclusión.

A mi director de tesis, Carlos Pastor, por las ideas y experiencia que ha proporcionado a este proyecto. Por el impulso que ha sabido transmitir en los momentos más duros. Y, fundamentalmente, por la confianza que siempre depositó en mí y por su fe en mi trabajo.

A Pedro Jauregui, con quien he colaborado durante los últimos tres años en un proyecto común. Cada uno en su campo, desde diferente perspectiva, hemos tenido que proporcionar una solución conjunta a numerosos problemas. La experiencia ha sido sumamente enriquecedora y, a buen seguro, esta tesis no sería la misma sin su contribución. Tratando con personas como Pedro la cordialidad y el entendimiento están garantizados.

A Luis Miguel Crespo, a quien es imposible agradecer en su justa medida las ideas aportadas. Sus conocimientos y su amistad han estado incondicionalmente a mi lado.

A Francisco Candelas y Federico Botella, cuya colaboración ha sido fundamental para llevar a cabo las pruebas experimentales de la tesis.

A José Antonio Pina, en representación del Departamento de Ciencias de la Tierra y Medio Ambiente, y José Giner, como responsable de la Red Local de Sismicidad de la Privincia de Alicante, por poner a mi disposición los recursos necesarios para realizar una implementación práctica de la red de ordenadores diseñada.

A los compañeros del Departamento de Ingeniería de Sistemas y Comunicaciones, en los que siempre he encontrado apoyo y buenos consejos: Agapito, Guillermo, Augusto, Antonio Durá, Antonio Hernández, Antonio Barreres, Jenaro, Manolo Pérez, Manolo López, Fernando, Luis Miguel Jiménez, Marisol, Juan Fco., Enrique, Alberto, Fco. Javier, José Joaquín, Amparo y Javier.

A compañeros de otros Departamentos que, gustosos, se han ofrecido a resolver las dudas que les planteaba: Mikel Forcada, Marco Maruhenda, Pablo Baeza y José Delgado.

A mi familia, que sin duda es la que más a sufrido mi dedicación a esta tarea. Por la comprensión que han tenido conmigo y el ánimo que siempre me dieron.

A todos vosotros, y a aquellos que mi memoria haya podido dejar en el olvido, muchas gracias.

ÍNDICE

Resumen..... 1

Capítulo 1º.- Planteamiento de la Tesis

1.1 Introducción..... 5

1.2 Contenidos..... 9

Capítulo 2º.- Control de procesos remotos

2.1 Control de procesos..... 15

2.2 Procedimientos de control..... 23

2.3 Control distribuido..... 27

Capítulo 3º.- La red de comunicaciones digitales

3.1 Introducción..... 35

3.2 La red digital de radio..... 39

3.2.1 Pruebas iniciales..... 39

3.2.2 Elementos de la red seleccionada..... 44

3.3 El protocolo estándar AX25, V2.0..... 48

3.3.1 Estructura de la trama..... 48

3.3.2 Fases para el intercambio de información..... 56

3.4 Evaluación del rendimiento de la red de comunicaciones..... 59

Capítulo 4º.- Diseño de la red de ordenadores

4.1 Metodología de diseño..... 73

4.2 Estructura de la red..... 78

4.3 Aplicaciones en red..... 83

4.4 Arquitectura..... 86

4.4.1 Definición de niveles..... 86

4.4.2 Servicios. Primitivas de servicio..... 100

4.4.3 Funciones de los niveles..... 106

4.5 Protocolos de nivel..... 124

4.5.1 Especificación informal de los protocolos..... 125

4.5.2 Especificación formal de los protocolos..... 148

4.5.3 Otras consideraciones sobre los protocolos..... 178

Capítulo 5º.- El sistema informático remoto

5.1 Características generales del sistema..... 191

5.2 Adaptación del sistema informático remoto al control de estaciones de campo con sistemas de adquisición de datos autónomos..... 198

Capítulo 6º.- Utilización de la red de ordenadores como soporte para realizar un control distribuido sobre estaciones de registro sísmico.

6.1 Características específicas del proceso..... 207

6.2 Aplicación de control. Modos de operación..... 212

6.3 Ventajas que aporta este procedimiento..... 225

Conclusiones..... 231

Bibliografía..... 237

Abreviaturas..... 243

Apéndices:

Apéndice I .- Listado del software diseñado para la aplicación de control y la red de ordenadores, residente en la estación central.

Apéndice II.- Listado del software diseñado para la aplicación de control y la red de ordenadores, residente en los sistemas informáticos remotos.

RESUMEN

La tesis se centra en el diseño de una red de ordenadores dedicada al control distribuido de procesos que tienen lugar en puntos geográficamente dispersos. La interconexión de los sistemas que conforman la red se realiza mediante una subred de comunicaciones digitales de radio. Para acceder al canal de comunicación se confecciona un procedimiento, que proporciona agilidad en la gestión de incidencias en la red de ordenadores y evita colisiones en el medio cuando existe información voluminosa que transmitir. Una evaluación de las prestaciones de la subred de comunicaciones, suministra datos significativos acerca de su capacidad de trabajo e información esencial para obtener su mayor productividad.

Los requisitos de las aplicaciones de control determinan los servicios que la red de ordenadores debe proporcionar. Estos servicios se estructuran dentro de un conjunto de niveles que, junto con los protocolos de nivel, conforman la arquitectura de la red. Los protocolos se especifican formalmente mediante máquinas de estados finitos (MEF). Si el protocolo es complejo, sus aspectos de control se modelan con una MEF, definiéndose un proceso generador de entradas y otro de salidas a la MEF para realizar una especificación completa. El diseño de la red finaliza con una serie de recomendaciones para la implementación de los protocolos.

La elaboración de un procedimiento que utiliza esta red de ordenadores como soporte para realizar un control distribuido sobre estaciones de registro sísmico, permite comprobar experimentalmente los planteamientos efectuados en la tesis y poner de manifiesto las ventajas que proporciona una solución de estas características. Esta implementación puede llevarse a cabo gracias a la contribución de un sistema informático dedicado a efectuar el control local sobre las estaciones remotas y capaz de atender, simultáneamente, las comunicaciones en la red. El sistema propuesto está basado en una arquitectura multicomputador, que le permite abordar la ejecución de distintas tareas de forma independiente y con una comunicación ágil entre las mismas.

CAPITULO 1

PLANTEAMIENTO DE LA TESIS

- ***Introducción***

- ***Contenidos***

1.1 INTRODUCCIÓN

La presente Tesis se enmarca dentro del programa de doctorado del Departamento de Ingeniería de Sistemas y Comunicaciones de la Universidad de Alicante. La elección del tema surge a raíz de la colaboración entre este Departamento y el de Ciencias de la Tierra y Medio Ambiente, de la misma Universidad, a principios de 1994. El trabajo, a desarrollar conjuntamente, se dirigía a buscar soluciones para mejorar las prestaciones de la Red Local de Sismicidad de la provincia de Alicante, que gestiona dicho Departamento.

La Red Local de Sismicidad de la provincia de Alicante tiene como objeto la adquisición de información, complementaria a la que suministra el Instituto Geográfico Nacional. Está orientada, fundamentalmente, hacia la detección y registro de eventos locales de baja intensidad, que por sus características escapan a los umbrales de detección de la Red Sísmica Nacional.

En el momento que se establece esta colaboración, la Red Sísmica de la Universidad de Alicante está compuesta por una serie de estaciones telemáticas de campo, conectadas mediante enlaces de radio analógicos a una unidad de registro central. Esta última se encuentra situada en el Campus de la Universidad, donde la señal recibida es demodulada y filtrada, acondicionándola para realizar un registro gráfico de la misma.

Esta conformación de la red comporta importantes problemas. La emisión desde la estación de campo debe ser simultánea y continuada, por lo que debe de disponerse de una frecuencia de radio distinta para cada enlace (estación de campo -

estación de registro central) y lo que es más importante, el sistema no es inmune a interferencias radioeléctricas ocasionadas por diversas fuentes, por lo que disminuye la fiabilidad de los registros efectuados.

Con objeto de mejorar las prestaciones de la red se estaba trabajando en dos caminos paralelos. Por una parte, la digitalización de las señales recibidas en el centro de proceso permitiría una mejora importante en cuanto al registro y tratamiento matemático de la información. Este paso era necesario para automatizar el proceso y aprovechar la infraestructura existente, pero insuficiente porque no corrige uno de los principales problemas, la fiabilidad de los registros, ya que sigue siendo dependiente de las interferencias radioeléctricas. Simultáneamente se estaba valorando la posibilidad de instalar, en nuevas estaciones de campo, sistemas capaces de digitalizar la información y almacenarla en forma de ficheros de datos. La aportación del Departamento de Ingeniería de Sistemas y Comunicaciones se centraría en el estudio de un procedimiento para enlazar las nuevas estaciones de registro, con el centro donde los ficheros obtenidos por las mismas debían procesarse.

El sistema para enlazar las estaciones de campo con la estación de proceso debía reunir los siguientes requisitos:

- Garantizar la fiabilidad de la información procesada
- Rapidez en la disposición de los ficheros a procesar
- Capacidad para enlazar un gran número de estaciones
- Coste competitivo

Considerando que el primer requisito imponía la necesidad de un enlace digital, para recuperar los posibles errores en la transmisión mediante el correspondiente protocolo de enlace, se realiza un análisis de las estaciones de adquisición de datos sísmicos existentes en el mercado.

Los productos que se ajustan a los requerimientos de los gestores de la red, disponen de una entrada/salida bajo normas RS232C y el *software* necesario para recuperar la información. La recuperación de información puede llevarse a cabo mediante una conexión directa del sistema de adquisición a un ordenador o a través de un módem de llamada automática. En este último caso debe instalarse otro módem en la estación de proceso, transmitiéndose la información por línea telefónica.

La recuperación de datos mediante una conexión directa requiere el desplazamiento al lugar de ubicación del sistema de adquisición, por lo que no resulta una alternativa que oferte mejoras substanciales. El segundo procedimiento necesita de una línea telefónica que enlace cada estación de campo con la estación de proceso. Dada la ubicación geográfica de los sistemas de medida, la existencia de una infraestructura con cableado telefónico que sirva de soporte al enlace no siempre es viable. Aunque esta contingencia puede solventarse con teléfonos móviles, la conexión utilizando líneas telefónicas se hace excesivamente lenta, ya que, además del tiempo necesario para efectuar la transmisión, debe añadirse el tiempo utilizado para establecer el circuito de datos (llamada telefónica). Este último tiempo puede evitarse utilizando un enlace permanente para la conexión (línea punto a punto) entre cada estación de campo y la estación de proceso, pero esta solución resultaría desproporcionada y su mantenimiento económico insostenible.

Dado que ninguno de los procedimientos conocidos satisfacía plenamente los requisitos demandados, se establece un plan de trabajo para buscar una solución práctica al problema. No obstante, con la intención de no ceñirse a un caso excesivamente particular, se plantea como objetivo de la Tesis el diseño de un procedimiento aplicable al control de sistemas geográficamente dispersos en general.

Dentro del término *sistemas geográficamente dispersos* debe entenderse: sistemas de adquisición de datos (sísmicos, meteorológicos, medioambientales, etc.) en los cuales no se actúa sobre el fenómeno físico estudiado, centrándose el control en los sistemas de adquisición y sistemas destinados al control de procesos remotos (por ejemplo el abastecimiento de agua potable a una ciudad) en los que existe

necesidad de actuar sobre motores, válvulas y otros mecanismos que regulan el proceso y que presentan la particularidad de no estar distribuidos en una planta o en un área restringida, donde la comunicación por cable podría ser una alternativa viable.

Aunque, evidentemente, las características del sistema a controlar tiene connotaciones que pueden influir en el procedimiento de interconexión, el trabajo desarrollado en esta Tesis pretende objetivar aquellos aspectos comunes que existen en los sistemas distribuidos, con la intención de proponer un procedimiento que permita un control eficiente de los mismos. El interés no reside en el estudio de los datos que puedan proporcionar unos determinados sistemas de adquisición o en el tipo de proceso que se desea controlar, si no en el intercambio de información entre puntos geográficamente alejados de forma que, mediante esta comunicación, un conjunto de estaciones ubicadas en puntos geográficamente alejados dispongan de los datos necesarios para efectuar las funciones para las que han sido programadas.

A partir de la propuesta general se derivarán los condicionantes requeridos para aportar soluciones al caso particular del estudio de eventos sísmicos. La implementación práctica de esta solución será la base experimental que corrobore los planteamientos efectuados en la memoria.

1.2 CONTENIDOS

Planteado el objetivo de la investigación, en el capítulo 2 de la memoria se realiza un estudio sobre las características que reúnen los sistemas sobre los que se desea actuar, enmarcándolos dentro del control de procesos. Se revisan los procedimientos de control aplicados a los mismos en la actualidad y se analizan los conceptos de control distribuido, utilizados en el control de plantas industriales. Las ventajas derivadas de la utilización de un control distribuido, serán la base para aplicar este procedimiento al control de nuestros sistemas y proponer la utilización de una red de ordenadores para llevarlo a cabo.

Las características de la subred de comunicaciones, que sirve de soporte para la interconexión de los sistemas que conforman la red de ordenadores, es fundamental para diseñar la arquitectura de esta última. En el capítulo 3 de la memoria se describen las pruebas realizadas para la elección de la subred de comunicaciones, presentando los elementos de la subred digital de radio seleccionada, su operatividad y el protocolo AX25 que controla el intercambio de mensajes entre los nodos de la subred. Finalmente se evalúa el rendimiento de esta subred de comunicaciones, bajo condicionantes similares a los que debe mantener durante el trabajo real.

En el capítulo 4 se diseña la red de ordenadores enunciando, en primer lugar, la metodología empleada. Seguidamente se describen los elementos que integran la red de ordenadores propuesta, así como las aplicaciones en red necesarias para realizar el control de los sistemas de nuestro interés. En base a estas aplicaciones y

teniendo en cuenta las características específicas de la red de comunicaciones utilizada, se procede a diseñar la arquitectura de la red.

La arquitectura de una red de ordenadores se estructura en base a un conjunto de niveles jerarquizados. La definición de estos niveles y las funciones asignadas a cada uno de ellos es el siguiente trabajo que se aborda.

Para que dos niveles, que realizan las mismas funciones en sistemas diferentes (niveles pares), puedan dialogar entre sí es necesario definir un conjunto de normas (protocolo de nivel) que regule la comunicación entre ellos. El diseño de los protocolos se realiza también dentro de este capítulo, elaborando en primer lugar una especificación informal de los mismos.

La especificación formal de los protocolos más sencillos se lleva a cabo mediante máquinas de estados finitos (MEF). Cuando la complejidad del protocolo así lo requiera, sus aspectos de control se especificarán con una MEF y se completará con la definición los procesos que manejan las variables del protocolo no incluidas en la construcción de la MEF. Previamente se formula la base teórica que avala el procedimiento de especificación utilizado.

Finalmente dentro de este capítulo se aborda la cuestión de implementar los protocolos, exponiendo una serie de recomendaciones para facilitar la tarea de programación.

En el capítulo 5 se describen las características técnicas del sistema informático remoto necesario para poder confeccionar la red de ordenadores, su operatividad y las modificaciones necesarias en el mismo para adaptarlo a las características del sistema o proceso a controlar. Dado que no existía un dispositivo con estas particularidades en el mercado, que nos permitiera comprobar experimentalmente el funcionamiento de la red de ordenadores definida, se abordó, por parte del Departamento de Ingeniería de Sistemas y Comunicaciones, el diseño y la fabricación de un sistema de estas características, adaptado al control de estaciones

de medida autónomas. Esta implementación práctica se llevó a cabo a partir de las especificaciones aportadas por los estudios realizados en la Tesis, siendo objeto de una patente de invención.

Con objeto de verificar los planteamientos realizados en la memoria, en el capítulo 6 se desarrolla un procedimiento que utiliza la red de ordenadores como plataforma para realizar un control distribuido de estaciones digitales de registro de movimientos sísmicos. Para ello se analizan las características de los sistemas de adquisición de datos y los objetivos que se pretenden cubrir con el control de los mismos. En base a los resultados obtenidos, se definen los modos de operación de la aplicación de control y los comandos que permiten la comunicación de los sistemas informáticos conectados en red a través de la misma. Finalmente se analizan las ventajas que comporta la utilización de este procedimiento frente a otros empleados actualmente.

Un apartado dedicado a conclusiones, recoge los aspectos más relevantes de los estudios desarrollados en la Tesis.

La memoria concluye con una relación de reseñas bibliográficas utilizadas en la confección de la misma, una lista de abreviaturas y dos apéndices. En estos se presentan los listados del *software* diseñado para realizar un control distribuido de estaciones digitales de registro sísmico, sirviéndose de las prestaciones proporcionadas por la red de ordenadores.

CAPITULO 2

CONTROL DE PROCESOS REMOTOS

- ***Control de Procesos***
- ***Procedimientos de Control***
- ***Control Distribuido***

2.1 CONTROL DE PROCESOS

Como paso previo al diseño de un procedimiento para controlar sistemas geográficamente dispersos, es conveniente realizar un estudio sobre las particularidades que reúnen los procesos a los que nos interesa aplicar el control, así como analizar los procedimientos de control utilizados en la actualidad para este fin. Este estudio permitirá establecer bases sólidas sobre las que fundamentar la solución elegida para este propósito.

Los sistemas de control automático han adquirido, en los últimos años, gran importancia en la industria al aportar soluciones a varios de los problemas no resueltos en los sistemas clásicos, como:

- La falta de integración de un proceso como un todo
- La dependencia de resultados de un operador humano
- El tratamiento costoso y lento de gran cantidad de información
- Un funcionamiento no óptimo del proceso de producción
- La existencia de situaciones peligrosas para las personas

El control automático se aplica a gran variedad de sistemas, con la finalidad de aumentar la productividad y homogeneidad del producto, causando un beneficio económico notable gracias a un menor coste y al mejor uso de las materias primas [No&Angulo_87].

Básicamente el control aparece cuando se desea que una propiedad o variable de salida, que caracteriza un *proceso*, tome un determinado valor o conjunto de

valores, estando el proceso normalmente sometido a la acción de otra variable de entrada o señal de control sobre la que se puede actuar [No&Angulo_87].

No todos los procesos pueden gobernar por si solos la variable de salida y, normalmente, resulta necesario utilizar un *sistema de control* alrededor al proceso para obtener la respuesta deseada. Así, un sistema de control es un conjunto de componentes interconectados, entre los que se incluye el proceso a controlar, cuyo objetivo es proporcionar una respuesta deseada a la salida del mismo sistema [Kuo_82].

Puesto que la teoría de los sistemas lineales que sirve de base para el análisis de un sistema supone una relación de causa-efecto para cada componente del sistema, estos componentes pueden representarse como bloques con señales de entrada y de salida. La relación causa-efecto de un componente viene dada por la relación entre sus señales de entrada y las señales de salida que genera [Dorf_86].

Dentro del control de procesos pueden distinguirse dos tipos básicos de sistemas de control:

- Sistemas de control de lazo cerrado
- Sistemas de control en lazo abierto

Los *sistemas de control de lazo cerrado* o sistemas realimentados, permiten aumentar la precisión y la inmunidad a las perturbaciones (señales internas o externas al sistema que tienden a afectar adversamente sobre el valor de la salida de un sistema), mediante una comparación de la señal de mando del sistema (también llamada consigna o referencia) y la salida de este. El esquema básico de este tipo de control se muestra en la figura 2.1.

La comparación se efectúa como diferencia entre la señal de entrada y la señal de salida realimentada. Como resultado de esta comparación se obtiene una señal de error que se aplica a un controlador. Ese controlador se encarga de que la entrada del

proceso tome los valores adecuados para reducir el error y llevar la salida al valor deseado [Ogata_93].

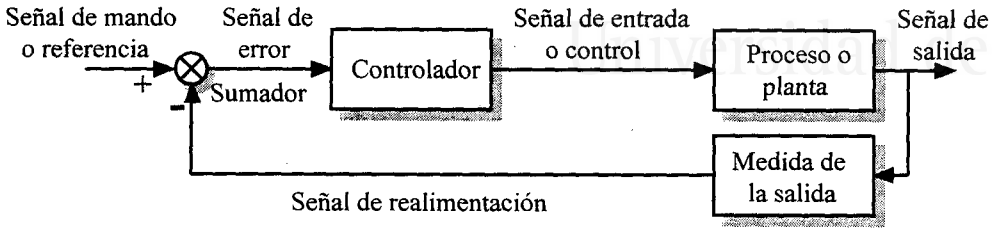

Figura 2.1 Esquema básico de control en lazo cerrado

Se puede decir que se establece una realimentación negativa al restarse la salida de la señal de mando. Con todo ello, un sistema realimentado tiende a mantener una relación entre la salida y la señal de mando.

Los *sistemas en lazo abierto* se caracterizan porque la salida no tiene efecto sobre la acción de control (no se realimenta). Para cada entrada o señal de mando corresponde una operación fija y la precisión del sistema depende de la calibración.

Aunque los sistemas en lazo abierto son los más sencillos y económicos, un sistema de este tipo no cumple su función ante perturbaciones y no permite conocer si el proceso alcanzará los valores de salida deseados. Se pueden utilizar cuando no se presentan perturbaciones y se conoce la relación entre la entrada y la salida [No&Angulo_87], [Dorf_86] y [Ogata_93].

Dentro de un sistema de control la *referencia* establece el valor que la salida del proceso debe tener en cada instante, fijándose su valor a un nivel superior. Se dice que el sistema actúa como *regulador*, si la referencia se mantiene fija. En este caso el control solo trata de corregir las variaciones de la salida producidas por perturbaciones. Si la referencia va cambiando su valor el sistema actúa como *servomecanismo* y el control actúa para que la salida se adapte a los valores de referencia, independientemente de las perturbaciones.

Entre el controlador y el proceso se debe colocar el *actuador* adecuado (por ejemplo un motor o una válvula) para que la señal física de control del proceso sea la requerida [No&Angulo_87].

Debe considerarse que, aunque se ha expuesto un sistema de control como un sistema con una señal de entrada y otra señal de salida, lo normal es que un sistema de control tenga varias entradas y salidas, pudiendo requerir varios controladores.

Cada vez con mayor frecuencia, las tareas del controlador o controladores de un proceso vienen siendo desempeñadas por un sistema computador. En este caso debe tenerse en cuenta que el computador no trabaja con señales continuas si no con señales discretas. Se requiere entonces elementos de conversión, que adapten las señales analógicas presentes en el proceso a un formato digital inteligible por el ordenador y viceversa. Estos elementos son los convertidores analógico-digitales (ADC, A*nalog to Digital Converter*) y digitales-analógicos (DAC, D*igital to Analog Converter*). Con la incorporación de estos elementos, un sistema de control discreto general puede verse representado en la figura 2.2 [Aracil_87].

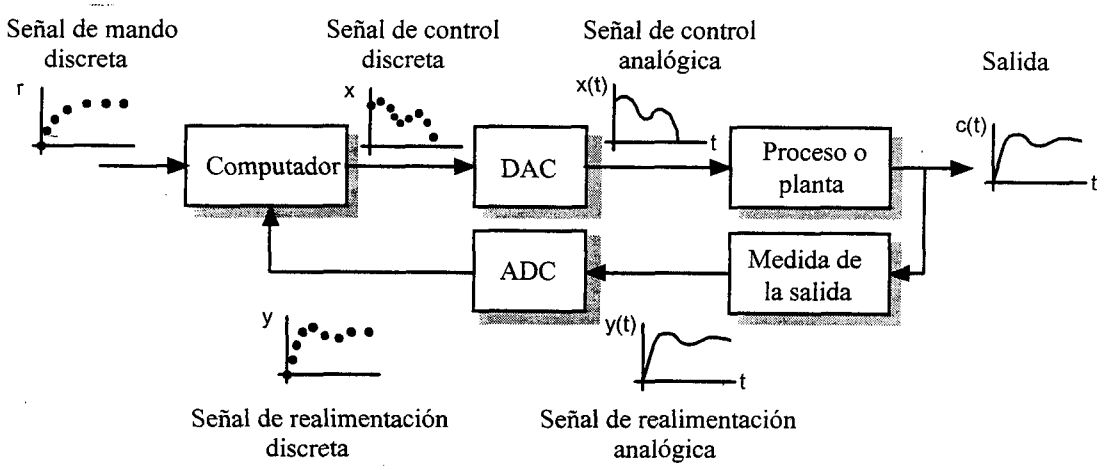

Figura 2.2 Esquema de un sistema de control discreto

En este esquema el bloque de ADC también hace la función de *muestreador*, tomando los valores de la señal analógica realimentada en ciertos instantes de tiempo. Por otro lado, el bloque DAC no solo debe convertir los valores digitales

dados por el computador en ciertos instantes del tiempo a valores reales, si no que debe formar una señal continua. Lo normal es que mantenga el valor de la última salida proporcionada por el computador, hasta que llega el siguiente dato.

Así, el funcionamiento básico del sistema viene marcado por instantes de tiempo, de forma que en cada uno de ellos el computador toma una muestra del valor de la salida y del valor de la entrada deseada correspondiente. El programa que actúa como regulador realiza los cálculos necesarios y obtiene un valor de salida que se manda al DAC. Para un correcto control del proceso lo habitual es que el programa regulador, además de considerar los valores obtenidos en cada instante, opere también con un conjunto de valores anteriores.

Una cuestión importante en el diseño de sistemas de control discretos, es la elección de la duración del intervalo de tiempo entre cada dos muestras de la señal de salida. Lo ideal sería tomar intervalos lo más pequeños posibles, con objeto de que los valores discretos puedan representarse como evolución de las señales analógicas. Pero las características técnicas del sistema y los tiempos de cálculo necesarios en el computador, limitan los valores mínimos del tiempo entre muestras y la elección de ese valor puede ser una cuestión crítica.

Las ventajas de la utilización de un computador frente a un sistema analógico son varias:

- Se mejora la calidad de un sistema de control al poder realizar funciones de control más complejas.
- Resulta fácil cambiar la estrategia de control modificando los programas del computador (bien por orden de un operador humano o bien por ordenes generadas por el propio sistema).
- Se obtiene mayor precisión en los cálculos
- Se evitan los errores típicos de los circuitos analógicos (influencia de temperatura, saturación...).

- Es posible realizar operaciones adicionales al control (almacenamiento de datos, análisis estadísticos...).

Como inconveniente podemos citar que un sistema con computador es más caro que un regulador convencional, aunque esta situación se puede salvar si el computador controla diferentes bucles (con tecnología convencional el coste aumenta linealmente conforme al número de bucles controlado, mientras que con el uso de un computador este coste es más estable) [Aracil_87].

Debe considerarse que no siempre todos los elementos de un sistema de control tienen por que estar localizados en un mismo lugar. Por ejemplo el controlador que regula la presión de una tubería de agua, normalmente se suele encontrar ubicado a varios kilómetros del lugar donde se encuentra el proceso. Tanto la señal de salida que realimenta al controlador (medidas de presión), como las señales de control sobre válvulas de apertura y cierre para conseguir la presión deseada, deben salvar estas distancias. En este tipo de casos está centrado nuestro interés, y en ellos el bucle de control debe incluir los sistemas de comunicación necesarios.

Los sistemas de comunicación utilizados influyen en el comportamiento del sistema introduciendo un retardo en las señales. Conceptualmente el tratamiento de este tipo de sistemas es el mismo que si no intervienen sistemas de comunicación, pero el retardo introducido debe tenerse en cuenta al describir las funciones de transferencia o las ecuaciones de estado del sistema.

Un proceso muy importante dentro del control es la medida de la señal de salida. En todo sistema de control es preciso disponer de sensores para medir los valores de las variables a controlar. Habitualmente los sensores se comunican directamente y mediante señales analógicas con el proceso de control, pero a veces el proceso de medida lo llevan a cabo sistemas complejos *(sistemas de adquisición de datos o sistemas inteligentes de medida)* que digitalizan la señal proporcionada por los sensores y son capaces de elaborar información más útil, que suministran al

proceso de control según este lo solicita. Estos sistemas suelen llevar su propio procesador y pueden comunicarse con el proceso de control, a través de líneas de comunicación digitales, con protocolos que hacen muy fiable la transmisión de información.

En aquellas situaciones donde el proceso se encuentra alejado del controlador y además se utiliza un sistema inteligente de medida para determinar la señal de salida, este último puede situarse en las proximidades del controlador o de la planta. En el primer caso se puede enviar la señal de salida, convenientemente amplificada y modulada, mediante enlaces analógicos hasta el sistema digitalizador. Mientras que con la segunda alternativa, al estar situado el sistema de adquisición en las proximidades del proceso, el envío de información al controlador se realiza mediante un enlace digital.

Este último procedimiento es el que mejor se ajusta a uno de los requisitos buscados, ya que garantiza una mayor fiabilidad de los registros entregados al sistema controlador. En ambos casos el sistema de comunicación interviene en el bucle de control, con la diferencia que las transmisiones analógicas están sometidas a interferencias que pueden desvirtuar la señal de salida recibida en el controlador, mientras que los errores producidos por interferencias en las transmisiones digitales pueden ser recuperadas mediante el correspondiente protocolo de enlace.

Tratando con sistemas de comunicación digitales entre el controlador y el proceso, es necesario que las señales de control que van a actuar como entrada a la planta, sean convenientemente adaptadas antes de ser entregadas a la misma. En este sentido podemos encontrar, como dispositivo intercalado entre el controlador y el proceso, desde un simple conversor digital analógico hasta un complejo sistema computador, capaz de generar señales de control en base a informaciones suministradas por el controlador remoto y que a su vez puede actuar como controlador local en determinadas circunstancias. Esta última situación es la que se abordará en la memoria.

Además del interés en el propio control se busca la posibilidad de poder acceder de forma remota, y a través del mismo soporte de comunicaciones, a elementos adicionales a la propia planta que estén situados en las proximidades de la misma: sistemas de adquisición, sistemas de control local, sistemas de alimentación eléctrica y cualquier otro elemento auxiliar necesario para el correcto funcionamiento de la planta remota, aunque éste no forme parte del propio bucle de control.

En consecuencia, la plataforma de comunicaciones propuesta en esta memoria debe cubrir dos objetivos:

- Formar parte del propio bucle de control en sistemas de control remotos.
- Servir de soporte a mensajes que permitan una supervisión a distancia de los distintos elementos que conforman el sistema remoto en su conjunto.

2.2 PROCEDIMIENTOS DE CONTROL

Considerando la aplicación del computador al control de procesos, esta puede llevarse a cabo a distintos niveles, con diferentes tipos de control y asignando diversas funciones al computador.

Atendiendo a la implicación del computador en el propio proceso, podemos distinguir tres tipos de actuación [No&Angulo_87]:

Labores de vigilancia. Es la forma más elemental de participación . El computador vigila el proceso sin intervenir en el control y se dedica a recopilar datos, almacenarlos y manipularlos, para elaborar información sobre la evolución del proceso y generar alarmas si se sobrepasan ciertos umbrales fijados.

Labores de supervisión. Además de las funciones realizadas en el caso anterior, el computador interviene en el bucle de control generando señales de mando para los actuadores a partir de la información obtenida del proceso.

Control digital directo. El computador se hace cargo de la adquisición de datos, de la elaboración de las ordenes de control y de su envío a los actuadores, siendo el corazón del bucle de control y ejecutando los algoritmos y estrategias preestablecidas.

Cuando interesa controlar simultáneamente varios procesos, existen diversas alternativas sobre como distribuir los elementos del sistema dentro del lugar de trabajo para aprovechar al máximo los equipos y la tecnología disponibles.

Básicamente se puede hablar de dos alternativas: *control centralizado* y *control distribuido* [No&Angulo_87].

En un *control centralizado* el computador, con la instrumentación de control y todos los componentes auxiliares (núcleo de control), se comunica individualmente con cada sensor actuador situado junto al proceso, independientemente de que núcleo de control y proceso estén en una zona cercana o alejados entre sí.

La utilización de una configuración de este tipo puede deberse a la necesidad de realizar un control conjunto sobre sistemas de control donde varios procesos dependen unos de otros; o también al hecho de emplear tecnología algo anticuada. Aunque se resuelve satisfactoriamente el control, el sistema de comunicaciones entre el núcleo de control y los procesos es muy costoso y complejo, siendo mayor cuanto más alejados estén estos entre sí [No&Angulo_87].

El control centralizado es la configuración típica más utilizada y, aunque su uso resulta más lógico en el control de procesos locales, también se emplea en bastantes ocasiones para realizar el control de procesos distantes.

En un *control distribuido* los recursos de cálculo se reparten por todo el sistema, aproximándolos todo lo posible a los lugares donde se necesitan. Así se instalan controladores junto a los procesos, los cuales se comunican con un computador central que actúa como supervisor, gestor y presentador de información. Entre los controladores y el computador central se transmiten señales de mando y medidas de cada salida.

Un ejemplo sobre control de procesos remotos, donde pueden comprobarse claramente las ventajas de utilizar un control distribuido, lo constituye la gestión del abastecimiento de agua potable en una ciudad. Un procedimiento de control de esta envergadura debe contar con un sistema de gestión centralizado, donde esté disponible toda la información necesaria para racionalizar el aprovechamiento del agua. El sistema central, para cubrir sus fines, necesita conocer datos puntuales del

estado de sus redes de abastecimiento, distribución y saneamiento; así como un importante caudal de información procedente de otras fuentes distintas a la propia red de aguas, como pueden ser: sistema de información metereológica (régimen de lluvias), sistemas de gestión de clientes (consumos, actividad, usos...), etc.

Sin embargo, el sistema de gestión centralizado, necesario para el conocimiento global del estado de la red, no tiene porqué controlar directamente cualquier proceso de la red como normalmente ocurre. Esto conlleva que los sistemas de comunicación se saturen de información procedente de los sensores situados en las distintas plantas. Información que, en la mayoría de ocasiones, no se utiliza por parte del controlador central para realizar una actuación sobre la planta. La gran cantidad de sensores y actuadores distribuidos en una red de estas características aconseja minimizar la información transmitida entre controlador central y plantas con el fin de reducir los sistemas de comunicación que, a menudo, son bastante complejos y caros.

Para realizar un control como el descrito es aconsejable utilizar una solución distribuida, donde en los distintos puntos de interés de la red de aguas (depósitos, puntos de distribución, sistemas de riego en parques, estaciones depuradoras, etc.) se realice un control local (con sensores y actuadores locales) sobre estos subsistemas, utilizando sistemas de control sencillos.

Los subsistemas estarían comunicados con el centro de gestión, que recibiría información más elaborada y generaría ordenes de control de más alto nivel para los subsistemas. De esta forma se transmite información más concreta por la red, reduciéndose considerablemente el tráfico y consiguientemente se puede restringir la infraestructura de comunicaciones.

Utilizando este procedimiento, el sistema de gestión central se encargaría de las operaciones de control de alto nivel, que afectan al comportamiento global de la red, coordinando de alguna manera los subsistemas. El sistema central dispondría de la misma información que con actuaciones directas sobre las plantas, pues cualquier

información relevante que pudiera producirse en un subsistema sería comunicado al centro de control, sin necesidad de que éste gaste recursos para ir interrogando a cada planta sobre su estado.

Las principales ventajas de esta solución podrían resumirse en: un menor coste en comunicaciones, un control dedicado para los puntos importantes de la red que resulta más eficaz y una menor carga de trabajo en el sistema de gestión central, pudiendo éste dedicar la parte liberada de sus recursos a otras tareas más generales dentro de la red.

2.3 CONTROL DISTRIBUIDO

Conforme a lo enunciado en el apartado precedente, en una configuración descentralizada o distribuida los diversos controladores que intervienen en la misma se sitúan junto a los procesos a controlar, formando éstos parte de un proceso global.

Con objeto de que los sistemas distribuidos ofrezcan fiabilidad, estén preparados para seguir trabajando después de fallos y sus recursos (por ejemplo las comunicaciones) admitan la máxima carga de trabajo, se ha evolucionado hacia un conjunto de productos normalizados llamados *módulos*, que pueden interconectarse de diversas formas para ajustarse a las necesidades de los procesos a controlar. Estos módulos pueden realizar tareas de comunicación, toma de datos, salidas de actuación, control de un proceso, etc.

Para realizar las funciones de cada sistema de control local se puede utilizar uno o varios módulos, que se agrupan en una unidad llamada *nodo*. Se consigue así un sistema de control flexible y adaptable a distintas aplicaciones. Las conexiones entre módulos pertenecientes a un mismo nodo pueden realizarse mediante protocolos específicos o comerciales. Si son conectados conjuntos de módulos ubicados en nodos diferentes, se emplean protocolos comerciales normalizados.

Los diferentes nodos se conectan entre sí y con el computador supervisor, mediante una *subred de comunicaciones*. En consecuencia los nodos deben disponer de módulos de comunicación que se encarguen del envío y recepción de mensajes. El medio físico que sirve de soporte a la subred de comunicaciones, varía dependiendo de la ubicación espacial de los nodos; así, para áreas restringidas suele utilizarse el

cable coaxial o la fibra óptica, mientras que para distancias geográficas amplias puede usarse la infraestructura telefónica o enlaces digitales de radio. Los nodos del sistema deben ser capaces de reiniciarse o reconfigurarse en casos de fallos graves.

El computador central que actúa como supervisor normalmente dispone de un interfase de usuario para permitir la actuación de un operador sobre el sistema. Este interfase puede considerarse como un módulo especial que forma parte de un nodo más del sistema distribuido.

En la figura 2.3 se muestra un esquema básico de control distribuido. Cada nodo de control estará compuesto por un módulo de comunicaciones, módulos de adquisición de datos proporcionados por sensores, módulos de control, etc.

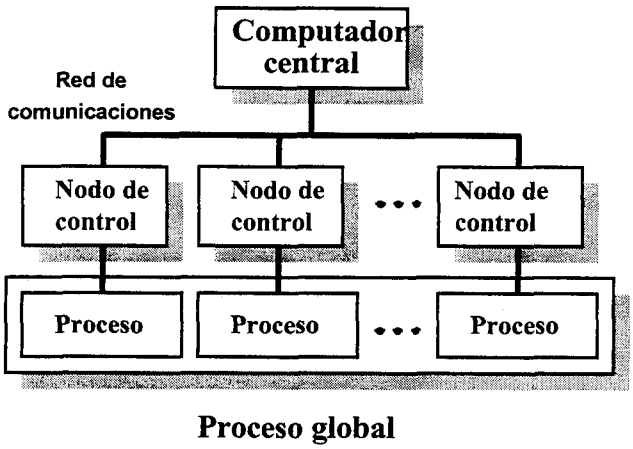

Figura 2.3 Esquema básico de un control distribuido

La subred de comunicaciones juega un papel fundamental en el correcto funcionamiento de un sistema de control distribuido, de hecho, las prestaciones de la subred de comunicaciones utilizada determina, en gran medida, la capacidad de control del sistema. Así, la subred de comunicaciones debe ser flexible para soportar diferentes configuraciones, debe permitir acceder a una base de datos (centralizada o distribuida entre los nodos, dependiendo de las características del proceso a controlar) y tiene que garantizar las prestaciones adecuadas, incluso en casos de máxima carga.

En lo que hace referencia a la topología de la subred, existen dos grandes bloques. Por un lado, puede utilizarse un *canal compartido* por todos los nodos de la red, siendo posible topologías tipo *bus* o tipo *anillo*. O bien, es posible usar *canales específicos* para la conexión de unos nodos con otros, utilizando en este caso topologías tipo *estrella*, *árbol* o *completas* (donde cada nodo esta conectado con el resto). En el caso de utilizar un medio compartido el protocolo debe incluir métodos de acceso al canal, que garanticen un emisor único [No&Angulo_87]. El estudio para determinar qué topología se adapta mejor a los requerimientos de nuestra subred de comunicaciones, será desarrollado en el siguiente capítulo.

Se puede generalizar el esquema básico de un control distribuido, presentando una descentralización en varios niveles con una estructura piramidal como la mostrada en la figura 2.4.

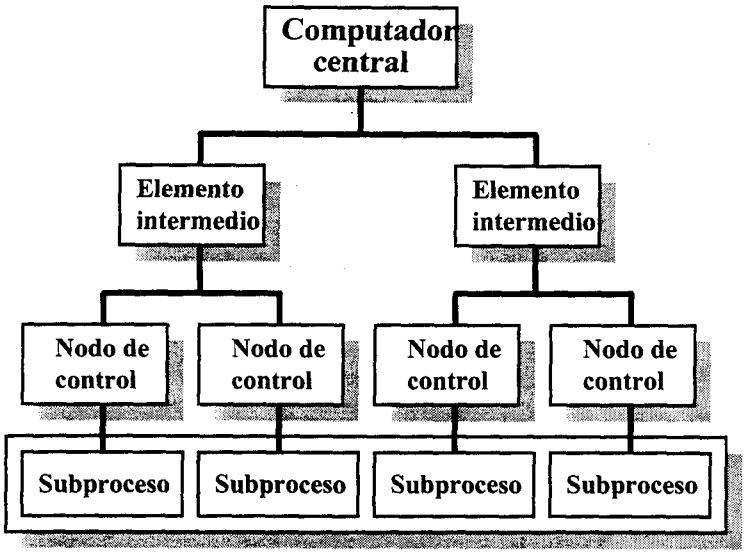

Figura 2.4 Esquema de un control distribuido jerarquizado

Esta figura puede representar, por ejemplo, el control de un proceso de una fábrica con dos secciones de fabricación, cada una con su propio equipo de control y una coordinación entre ambas ejercida por un computador supervisor.

La asignación de funciones a cada uno de los distintos niveles en los que se ha descompuesto el proceso global, presenta interesantes problemas si se quiere llegar a un aprovechamiento máximo del equipo de control. El sistema resultante se denomina jerarquizado, por la estructura de funciones de control [Aracil_87].

En un control *distribuido jerarquizado* se puede considerar que el control directo o supervisor de un proceso concreto es solo una parte del control total que puede implementarse en una situación compleja, con diversos procesos interconectados entre sí. De esta forma se puede disponer de un sistema global para abordar el control, basado en una estructura piramidal, con diversos tipos de computadores organizados en diferentes niveles y conectados por distintos tipos de comunicaciones.

El nivel más bajo, el de control local, tiene como misión el control realimentado de un proceso. El computador de este nivel realiza la medida a través de los sensores, compara con la señal de mando y determina la señal para los actuadores según un algoritmo de control seleccionado. La señal de mando y el algoritmo a emplear los determina el nivel superior al que serán remitidas las posibles alarmas generadas en el control.

Un nivel de supervisión, por encima del anterior, puede coordinar diversos controles directos y comprobar los valores y tendencias de las variables para determinar si es necesario efectuar alguna corrección o generar alguna alarma. Un tercer nivel puede gestionar el control de toda una área y tratar de optimizar la energía y los materiales empleados, estableciendo las condiciones de funcionamiento de niveles inferiores de supervisión. Por último, niveles superiores pueden integrar todas las áreas y planificar el control en su conjunto, organizando también la comunicación entre áreas.

El control distribuido aplicado a sistemas donde se controlan diferentes procesos, o a procesos complejos que pueden dividirse en subprocesos con cierta

independencia, conlleva una serie de ventajas con respecto al control centralizado convencional [Sánchez_93].

Las ventajas más importantes pueden ser estas:

- Es posible resolver problemas muy complejos, ya que disminuye la dificultad del control y se puede lograr un aumento de la potencia de control.
- Debido a su estructura modular y jerarquizada resulta muy sencillo la integración de sistemas de proceso ya diseñados. Así, por ejemplo, se puede actualizar un sistema de control convencional a un control distribuido, considerando en sistema antiguo como nivel inferior y diseñando nuevos niveles superiores que mejoren la integración.
- Se consiguen sistemas más flexibles. Es más fácil realizar un cambio a nivel de los procesos controlados, puesto que los cambios básicamente (o solamente) afectan al nodo de control local.
- Se logra una proximidad del control a los procesos controlados. Esto permite ahorrar costes de conexión, de sensores y actuadores.
- Hay una mayor modularidad, menor complejidad de diseño e instalación.
- Los sistemas resultan más fiables. Hay posibilidades de protección y recuperación de información en las comunicaciones, reconfiguración en caso de fallos, etc.
- El fallo de uno de los subsistemas no impide que el resto continúe desempeñando sus funciones.

Las cualidades que presentan los procedimientos de control distribuido frente a los procedimientos centralizados, determinan su idoneidad para ser aplicados al control de procesos remotos.

Los distintos tipos de computadores que intervienen en un control distribuido, independientemente del nivel de control en el que actúen, constituyen un conjunto de

sistemas informáticos autónomos. Consiguientemente puede diseñarse una *red de ordenadores* para interconectar estos sistemas y utilizar las prestaciones de la misma, como soporte de las aplicaciones de usuario (aplicaciones de control) que deben distribuirse entre los sistemas que conforman la red.

El objetivo fundamental de la tesis será el diseño de una red de ordenadores, que proporcione los servicios necesarios para que el control distribuido de procesos remotos en general se efectúe de una forma eficiente.

La eficacia del control dependerá, en buena medida, de la capacidad de la red de ordenadores para tramitar la información que se genera en el proceso. Un adecuado diseño de la red obliga a realizar un análisis sobre el tipo y cantidad de información que debe intercambiarse, para dotar al sistema de un soporte adecuado de comunicaciones. Este análisis se efectuará en el capítulo 3 de la memoria.

Por otra parte, es necesario llevar a cabo un estudio de las aplicaciones que deben ejecutarse en red, para facilitar la realización de un control de estas características. La red también debe posibilitar la acción a distancia sobre elementos próximos a los nodos de control local, ubicados en puntos remotos. Estos aspectos serán desarrollados en el capítulo 4 como paso previo al diseño de la red de ordenadores.

CAPITULO 3

LA RED DE COMUNICACIONES DIGITALES

- ***Introducción***
- ***La red digital de radio***
- ***El protocolo AX25***
- ***Evaluación del rendimiento de la red de comunicaciones***

3.1 INTRODUCCIÓN

Los conceptos de *red de ordenadores* y *red de comunicaciones* a veces no aparecen debidamente diferenciados, induciendo a confusión. Este hecho se produce porque, en muchas ocasiones, un mismo equipo físico puede constituir un elemento activo de la red de ordenadores y, a su vez, incorporar los elementos que gestionan el intercambio de mensajes con otros sistemas integrantes de la misma red de ordenadores. Conviene definir ambos conceptos ya que, funcionalmente, constituyen entes claramente separados.

Una *red de ordenadores* está constituida por un conjunto de sistemas informáticos autónomos, en general heterogéneos, interconectados entre sí por un sistema de comunicación (*subred de comunicaciones* o *red de comunicaciones*). La red de comunicaciones es un subsistema dentro de la red de ordenadores, necesario para poner a disposición de cada equipo informático los mensajes producidos por otros ordenadores de la red. El diseño completo de una red de ordenadores, se simplifica notablemente cuando se separan los aspectos puros de la comunicación, de los aspectos relacionados con los sistemas y las aplicaciones que deben correr en los mismos [Tanenbaum_91].

Consiguientemente, para abordar el diseño de una red de ordenadores, es necesario considerar las características de la red de comunicaciones que permite la interconexión de los sistemas que la integran. El conocimiento de las prestaciones de una determinada red de comunicaciones, es la base para definir claramente los servicios que ofrece a los usuarios y articular así el procedimiento que deben seguir los sistemas conectados en red, para coordinar el intercambio de mensajes dentro de la misma.

Para la elección de un procedimiento que permita el transporte de datos entre los ordenadores de la red, hemos de tener en cuenta que los sistemas que deben conectarse a la red de comunicaciones estarán ubicados, normalmente, en lugares de difícil accesibilidad y que, en ocasiones, será necesario un cierto grado de movilidad en los mismos. En estas circunstancias la utilización de sistemas de comunicación inalámbricos resulta muy apropiada por no estar sujetos a una infraestructura fija.

Los sistemas de comunicación de datos inalámbricos representan, hoy en día, un segmento importante y en auge dentro de la industria de las comunicaciones. En [Pahlavan_94] se presenta un completo estudio sobre este campo, cada vez más diverso, identificando dos corrientes principales en función de los requerimientos de las redes de ordenadores que utilizan este procedimiento de interconexión. Una de estas corrientes está ocupada por las *Redes de Área Local Inalámbricas* (WLAN), que requieren una alta velocidad de transmisión (de 1 a 20 Mbps), se despliegan en áreas relativamente restringidas (edificios, naves industriales, etc.) y soportan aplicaciones donde deben transferirse ficheros de datos voluminosos y tareas de impresión, fundamentalmente.

La otra corriente comprende lo que, en la literatura sobre el tema, se conoce genéricamente como *Redes de Datos Móviles*. Se distinguen por utilizar velocidades de transmisión más bajas (≤19.200 bps), abarcar áreas geográficas extensas y soportar una diversidad de aplicaciones caracterizadas por el envío de mensajes cortos. La palabra *móviles* en este contexto hace referencia a la facilidad de cambio de ubicación de estos sistemas, no a que los sistemas estén en movimiento. No obstante, la posibilidad de sistemas en movimiento también queda englobada si se instalan en vehículos que pueden estar desplazándose durante las comunicaciones.

Para hacer efectiva la transmisión de datos, estas redes utilizan estaciones de radio o teléfonos móviles, controlados por un sistema capaz de convertir las señales analógicas que se propagan en el medio físico en información binaria inteligible por un ordenador.

Existe en la actualidad un amplio grupo de redes de datos móviles dentro de la industria de las Telecomunicaciones. En [Pahlavan_94] se analizan algunas de las más representativas, como las redes *ARDIS* y *MOBITEX*, presentando sus características principales, las aplicaciones a las que están destinados y planteando las perspectivas de desarrollo de estas tecnologías.

Un aspecto que caracteriza a este tipo de redes es el hecho de tener que compartir un único medio de transmisión por todos los interlocutores (canal de difusión), por lo que uno de los principales problemas de su diseño se centra en determinar un método de acceso adecuado para obtener una compartición eficiente del canal. En este sentido el desarrollo del sistema *ALOHA*, en la década de los 70, llevado a cabo en la Universidad de Hawai por Abramson, constituye un importante punto de referencia. A partir de los estudios de Abramson, se han desarrollado múltiples métodos para resolver el problema de la asignación del canal que soporta las comunicaciones [Abramson_85].

Otros trabajos, dentro del campo de las redes de datos móviles, se han dedicado al estudio de las características de propagación en el espectro radioeléctrico y al análisis de modelos de tráfico, como temas más relevantes. En [Abramson_94], [Bertoni_94], [Carse_94], [Everitt_94], [Fuhrmann_94], [Katz_94] y [Pahlavan_94], pueden encontrarse algunas de las más recientes e interesantes investigaciones sobre estos temas.

Hemos mencionado la existencia de dos elementos posibles para materializar las comunicaciones en una red inalámbrica: las estaciones de radio y los teléfonos móviles. En [Tanenbaum_91] se presenta un modelo conceptual correspondiente a un modelo de difusión de paquetes[1] por radio. En el mismo se establecen tres situaciones en las cuales este procedimiento es aconsejable para la transmisión de datos:

[1] *Paquete:* Unidad de información estructurada que se intercambia entre los sistemas conectados a la red de comunicaciones.

1. Estaciones localizadas en lugares sin infraestructura telefónica.
2. Estaciones en movimiento (ubicadas en barcos, automóviles, etc.)
3. Estaciones con una relación de tráfico elevado, entre el pico y el promedio, o con una tasa muy baja de datos.

Hoy en día las dos primeras situaciones pueden soslayarse también con teléfonos móviles, pero un análisis de las características del tráfico que se va a generar en nuestra red, y que concuerda con la tercera situación, aconseja descartar este último elemento por el excesivo coste que supondría mantener una línea dedicada. Aunque la utilización de líneas telefónicas conmutadas reducen el coste, éste sigue siendo mucho mayor que el necesario para mantener las estaciones de radio y, además, implica una mayor lentitud en formalización de cada enlace entre los ordenadores de la red, pues debe efectuarse previamente una llamada telefónica para establecer el circuito de datos.

Del análisis de los trabajos de referencia citados en esta introducción y teniendo en cuenta las aplicaciones a los cuales se orienta la red de ordenadores, la utilización de una red digital de radio para el soporte de las comunicaciones se revela como el procedimiento más adecuado. Esto permitirá contar con la ventaja de no depender de ningún tipo de cableado, una facilidad de movilidad en las estaciones y un bajo coste de mantenimiento. El principal inconveniente es la limitación de la velocidad de transmisión (máximo 19200), que es bastante menor que la que proporcionan otros sistemas que utilizan cables coaxiales, fibras ópticas o enlaces por satélite. Esta velocidad puede estar además acotada por el ancho de banda que concede la Administración cuando se asigna un canal de radiofrecuencia. Aunque evidentemente una mayor velocidad implica mayores prestaciones, el hecho de tener que trabajar con velocidades de transmisión de 19200 ó 9600 bps no es un handicap considerable para este caso, ya que estamos tratando con velocidades que son suficientes para un correcto funcionamiento de las aplicaciones de la red.

3.2 LA RED DIGITAL DE RADIO

3.2.1 PRUEBAS INICIALES

Una vez concluida la elección de una red digital de radio como soporte de las comunicaciones, y teniendo en cuenta la finalidad de llevar a cabo una implementación práctica de la red de ordenadores diseñada, se estimó la conveniencia de realizar una serie de pruebas con estaciones de radio para determinar las dificultades reales que podían encontrarse. Estas pruebas servirían también de base para buscar soluciones, ya desarrolladas, que podían ajustarse a los servicios que demanda esta red.

Inicialmente se procedió a conectar dos ordenadores personales (PC) a través de emisoras de radio, mediante una conexión directa emisora-PC. Dado que se disponía de emisoras FM (Frecuencia Modulada) con entrada digital (normas TTL), bastaba realizar una conversión de las señales suministradas por el puerto serie (normas RS232C) del ordenador, a las señales aceptadas por la emisora y viceversa.

La figura 3.1 presenta el esquema del diseño que se realizó para hacer la conversión (RS232C-TTL). La activación de la portadora en la emisora PTT (*Push-to-Talk*) se realiza desde el propio ordenador, utilizando la señal RTS (*Request to Send*).

Para intercambiar información entre los PC se diseñó un protocolo de enlace simple, sin control de errores, con capacidad de selección del tamaño del campo de datos n, de la velocidad de transmisión v y del tiempo de activación de portadora antes de enviar información al medio T_p. Cada ordenador analiza su señal CTS (*Clear*

to Send) antes de activar RTS. Si CTS está activada indica que la emisora ha detectado portadora (puesta por la otra estación) y permanece en recepción. Si CTS no está activada indica que el canal está libre y se puede activar la señal PTT para emisión. Las tramas enviadas no se confirman. Una sencilla aplicación permitía enviar ficheros entre los dos PC, comparándose con los originales una vez transmitidos para comprobar el número de errores producidos.

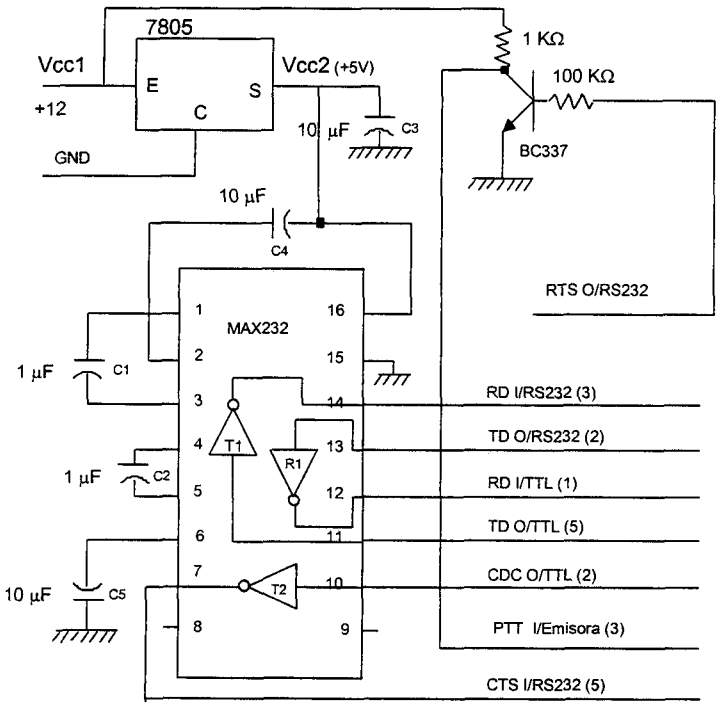

Figura 3. 1 Conexión RS232C - TTL

Gracias a la facilidad para variar los parámetros n, v, y T_p, se pudieron realizar suficientes pruebas en laboratorio para obtener la siguientes conclusiones:

- A partir de $T_p = 20\ ms$ el aumento del tiempo de portadora, antes de enviar datos al medio, no afectaba a la calidad de la transmisión. Este tiempo es necesario para que la emisora de radio desarrolle toda la potencia antes de enviar los datos, y para que la estación receptora tenga tiempo de adecuar sus circuitos para la recepción a partir de la detección de la portadora. Si T_p no es suficiente, se pierden los primeros datos de cada trama, y si es excesivamente grande se pierde un tiempo

que podría estar dedicado a la transmisión de información. Por lo tanto, el T_p óptimo depende en parte de la calidad de las emisoras y de las condiciones de recepción del medio. Con otras condiciones y otro modelo de emisoras de radio, debería buscarse el T_p ideal para no malgastar tiempo de emisión y conseguir el máximo rendimiento del canal.

- Para conseguir velocidades de transmisión de 4800 bps sin que el número de errores fuese significativo y el rendimiento del sistema decayera notablemente (ha de tenerse en cuenta que cada trama que contenga al menos un error, debe reexpedirse cuando se esté trabajando en una situación real), la información contenida en cada trama no podía sobrepasar los 20 *bytes*.
- Para enviar tramas con 128 *bytes* de información, debía reducirse la velocidad a 1200 bps para poder transmitir con unas mínimas garantías.

Teniendo en cuenta estos resultados, la conexión directa de los ordenadores que conforman la red a una emisora de radio, implementándose el control de la transmisión en los propios ordenadores, puede utilizarse con aceptables resultados si la cantidad de información intercambiada entre los mismos no es muy elevada, se organiza esta en tramas muy cortas o se reduce considerablemente la velocidad de transmisión.

Podría suponerse que con estas prestaciones de la red de comunicaciones es suficiente para atender el objetivo de controlar procesos a distancia o supervisar estaciones de adquisición de datos, pues en la mayoría de ocasiones la información que se requiere transmitir no resulta excesiva. Sin embargo si queremos dotar a este procedimiento de una potencia razonable, destinándolo al control de una amplia gama de procesos y distintos tipos de sistemas de adquisición de datos, es previsible que se presente la necesidad de tener que intercambiar ficheros de datos de un tamaño considerable, para un correcto funcionamiento del sistema.

Ante esta perspectiva, para tener un buen rendimiento en la red de ordenadores es necesario trabajar con las máximas velocidades que permita la red de

comunicaciones (19200 bps ó 9600 bps en su defecto) y que el tamaño del campo de datos de cada trama que se envía al medio sea el mayor posible.

Para que estas condiciones se cumplan hay que dotar de modems a las emisoras para reducir la tasa de error medio del circuito k (k = *nº bits erróneos/nº de bits transmitidos*). Utilizando modems comerciales, adaptados especialmente para la transmisión de datos vía radio y utilizando el mismo protocolo simple, especificado con anterioridad, conseguimos transmitir a 19.200 bps tramas de 256 *bytes* de datos con una tasa de error medio $k = 1{,}2 \cdot 10^{-6}$. En [Pahlavan_94] se establece como necesario para transmitir con garantías en las redes de datos móviles una tasa de error medio $k \leq 10^{-5}$, considerando como bueno un circuito con $k = 10^{-6}$.

Se ha de observar que aunque estas pruebas se realizaron en laboratorio se intentó simular, en la medida de lo posible, las condiciones más adversas para emisión/recepción. Para ello se limitó el tiempo de portadora T_p a 10 ms, se utilizaron resistencias de carga en las emisoras y se apantallaron parcialmente las mismas con cables entrelazados para simular una jaula de Faraday.

La tasa de error $k = 1{,}2 \cdot 10^{-6}$ es la más desfavorable de las obtenidas, observándose que utilizando antenas de lugar de resistencia de carga, con la máxima potencia de las emisoras, no se detectan errores.

Establecidos los elementos necesarios para hacer efectivo el transporte físico de la información en la red de comunicaciones, es necesario disponer de un protocolo que regule la comunicación. El protocolo diseñado para las pruebas anteriores era extremadamente sencillo, ya que su único objetivo residía en obtener información acerca de las mejores condiciones físicas del circuito de datos.

El protocolo que debe regular el tráfico en la red de comunicaciones debe cumplir los siguientes requisitos, que no se contemplaban en el caso anterior:

1) Establecer un procedimiento de acceso múltiple al canal de comunicaciones. En una situación real este canal será utilizado por varias

estaciones. En nuestras pruebas sólo disponíamos de dos estaciones y el acceso era muy simple.

2) Incluir mecanismos de transparencia para poder transmitir cualquier combinación de *bits*.

3) Incluir mecanismos de detección de errores y estrategias de retransmisión.

4) Gestionar el control del diálogo en la red, articulando reglas que determinan el turno de intervención, mecanismos de control del flujo de datos, asentimientos, etc.

5) Flexibilidad en la configuración de determinados parámetros para adecuar la red de comunicaciones a los requerimientos específicos de las diversas situaciones que puedan plantearse.

El diseño de una red de comunicaciones digitales de radio que pueda dar cobertura a las necesidades planteadas, es un trabajo considerable. Debe tenerse en cuenta que, una vez elegidos los elementos y diseñado el protocolo que regula las comunicaciones, deben realizarse pruebas de simulación, estudios de efectividad comparándola con otras redes similares y otras pruebas, necesarias para verificar su correcto funcionamiento en base a las especificaciones iniciales. Centrando nuestro esfuerzo en ésta cuestión, desviaríamos nuestro interés inicial de diseñar una red de ordenadores con una utilidad específica.

Existen, por otra parte, redes de comunicaciones de datos vía radio ya diseñadas, que pueden dar una respuesta satisfactoria a estos requerimientos. En esta tesitura se entendió más conveniente utilizar una red de comunicaciones ya confeccionada, que dispusiera de suficiente flexibilidad como para poder adaptarla a nuestras necesidades. Así el trabajo se centraría en el diseño de la red de ordenadores y especialmente en las aplicaciones de red que van a permitir la consecución del objetivo inicial.

Un estudio de los sistemas de estas características que podríamos encontrar en el mercado, nos condujo a la elección de una red de radio-paquetes gestionada por el

protocolo estándar AX25. El *software* que realiza las funciones relativas a este protocolo, se encuentra implementado en un dispositivo TNC (*Terminal Node Controller*) que lleva incorporado un módem para adaptar las señales recibidas y enviadas a la emisora. Este elemento dispone también de un puerto serie (normas RS232C), al que se puede conectar cualquier sistema informático que se adapte a los modos de operación permitidos por el TNC.

De esta forma la red de comunicaciones constituye un ente independiente, entregando a los sistemas conectados a la misma una información libre de errores, gestionando el acceso al medio y aportando otras facilidades que simplifican considerablemente el diseño del protocolo de enlace de la red de ordenadores.

3.2.2 ELEMENTOS DE LA RED SELECCIONADA

Básicamente un nodo de la red de comunicaciones está constituido por una emisora de radio y un TNC (figura 3.2). Cada nodo debe contar con aquellos elementos auxiliares necesarios para la operatividad de estos elementos (antena, cableado, alimentación eléctrica, etc.)

La emisora de radio es el elemento responsable de enviar al medio, convenientemente moduladas, los tonos recibidos del TNC (o señales digitales si la emisora está acondicionada para ello). En recepción devuelve al TNC los tonos o señales digitales procedentes de la demodulación de las ondas de radio recibidas.

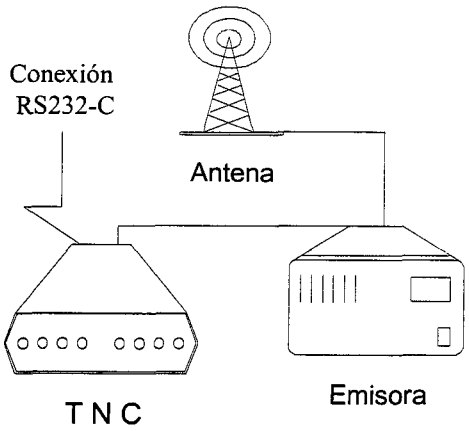

Figura 3. 2 Nodo de la red de comunicaciones

En principio, cualquier emisora de radio con unas buenas condiciones de emisión/recepción puede ser suficiente. Es necesario, no obstante, conocer algunas de las características de estas emisoras, como: tiempo de conmutación para la

emisión/recepción y tiempo necesario para que la emisora desarrolle su máxima potencia, para ajustar convenientemente el tiempo de portadora T_p definido en el apartado 3.2.1.

Hoy en día existen emisoras especialmente diseñadas para la transmisión de datos, por lo que es recomendable su utilización para obtener mejores prestaciones. Tanto para realizar las pruebas iniciales, como para comprobar la operatividad de la red de ordenadores diseñada en el ejemplo práctico, se ha utilizado una de estas emisoras modulando en frecuencia (FM) en la banda de UHF. Las características de la emisora que afectan a la transmisión de datos son las siguientes:

- ENTRADA ANALÓGICA. Para trabajar en banda estrecha (máximo 10 kHz) a velocidades $V \leq 9600$ bps.
- ENTRADA DIGITAL. Para trabajar en banda ancha a una velocidad V=19200 bps.
- Detección rápida de la portadora y de la conmutación entre la emisión y recepción (menos de 2 ms)

Estas dos últimas características permiten ajustar T_p a 10 ms (tiempo mínimo permitido por la red de comunicaciones) sin variar la tasa de error del circuito. Este ajuste solo puede darse por válido para pruebas en laboratorio y utilizando esta red con el módem que incorpora el TNC. Para una ubicación diferente de las emisoras deberá obtenerse el T_p óptimo, como ya se ha comentado.

En cada nodo disponemos de una sola emisora con una única frecuencia de portadora, por lo que el mismo canal ha de utilizarse para la emisión y recepción de la información, constituyendo un sistema *semi-duplex*.

El TNC es el elemento que controla el intercambio de información entre la red de comunicaciones y el sistema informático al cual está conectado. La comunicación con este último se realiza mediante transmisiones asíncronas bajo normas RS232C, disponiendo de distintos modos de operación. Dependiendo del

modo de operación seleccionado, la información entre los TNC se envía sin ninguna regulación, dejando el control del enlace a los sistemas informáticos o utilizando el protocolo AX25.

El TNC es programable, por lo que resulta muy flexible; tanto para definir las características de la comunicación asíncrona con el sistema informático (velocidad de transmisión, nº de *bit* de datos por palabra, paridad, control de flujo, etc.), como para ajustar los parámetros que regularán el diálogo en la red de comunicaciones (tiempos de espera, número de conexiones simultáneas que se pueda establecer, tamaño de la ventana[2], número de reintentos, tamaño del campo de datos, etc.). Permite así mismo la monitorización de las comunicaciones en la red, lo que le convierte en una herramienta útil para el análisis de incidencias.

La norma del protocolo AX25 no especifica un algoritmo de acceso al canal [Tanenbaum_91]. Para resolver este acceso se recomienda utilizar el correspondiente al ALOHA puro o al CSMA (*Carrier Sense Múltiple Accees*/Acceso Múltiple por Detección de Portadora). Una descripción completa de los procedimientos de acceso al canal mencionados pueden encontrarse en [Stallings_91] y [Tanenbaum_91]

El TNC con el que desarrollamos nuestro trabajo utiliza un procedimiento CSMA con un algoritmo de persistencia (programable) para acceder al canal de comunicación. Las técnicas CSMA p-persistentes requieren que exista un tiempo de troceado de propagación (*T-slots*) [Alabau_86]. Para ello se define un tiempo de espera igual al retardo máximo de propagación en el canal (considerando los dos puntos más alejados). En el TNC utilizado este tiempo es programable, por lo que se puede adecuar a las características de la instalación.

El algoritmo de acceso el canal funciona de la siguiente forma:

---

[2] *Ventana:* Número máximo de tramas de información que pueden enviarse sin que estas sean confirmadas

El valor de *persistencia* se programa entre 0 y 255. Cuando el TNC desea enviar información al medio y detecta que el canal esté libre genera un número aleatorio *l* entre 0 y 255. Si *l≤persistencia* el TNC activa la portadora y envía el paquete de datos al medio. Si *l>persistencia* el TNC pone en marcha el tiempo (*T-slots*) y cuando este vence vuelve a escuchar el medio reiniciándose el proceso. Un valor de *persistencia* = 255 implica que se transmite siempre que se disponga de información y el canal esté libre. Conforme disminuye el valor asignado a *persistencia* disminuye la probabilidad de acceso al medio.

El valor de *persistencia* debe ajustarse en función de la carga de la red y el número de estaciones que la integran. En [García_90] puede verse que el rendimiento de una red de comunicaciones de estas características mejora cuando al aumentar la carga se disminuye la probabilidad de acceso de cada estación, ya que de esta forma se consigue reducir el número de colisiones.

Las facilidades de programación del TNC permitirán configurar la red de comunicaciones a medida de los servicios demandados por los sistemas conectados a la misma. Estos servicios estarán condicionados, en cada caso, por las características específicas de las aplicaciones diseñadas para la red de ordenadores.

3.3 EL PROTOCOLO ESTÁNDAR AX25, V2.0

El protocolo AX25 constituye un estándar para el envío de paquetes de datos, en las redes de comunicación que utilizan sistemas de radio como soporte de las transmisiones. Propuesto por la *American Radio Relay League* [ARRL_84] está basado en el protocolo X25 del CCITT[3], aunque mantiene algunas diferencias como: la utilización de un campo de direcciones extendido, la adición de un campo *PID* (*Protocol Identifier*) en las tramas de información y la adición de tramas de *Información No Numeradas* UI (*Unnumbered Information*) [Stallings_91]. También sigue las Recomendaciones CCITT en el uso de múltiples enlaces; las Recomendaciones ISO[4] 3309, 4335 y 6256 (control de enlace de datos de alto nivel, HDLC [ECMA_81]).

El protocolo permite el enlace directo entre dos estaciones individuales o entre una estación individual y un controlador multipuerto. Si la estación lo soporta pueden establecerse varias conexiones simultáneas a nivel de enlace. Todas las estaciones pueden emitir ordenes y respuestas (estaciones combinadas) por lo que constituye un modo de operación balanceado.

3.3.1 ESTRUCTURA DE LA TRAMA

La transmisión de radio-paquetes se realiza en bloques de datos (tramas), subdividida en varios campos. Seguidamente se muestran los tres tipos básicos de tramas: No Numeradas (U), Supervisión (S) e Información (I).

---

[3] *CCITT:* Comité Consultif International Telegraphique et Telephonique
[4] *ISO:* International Organization for Standardization

Guión	Dirección	Control	CRC	Guión
01111110	112/560 Bits	8 Bits	16 Bits	01111110

Tramas U y S

Guión	Dirección	Control	PID	Info	CRC	Guión
01111110	112/560 Bits	8 Bits	8 Bits	N*8 Bits	16 Bits	01111110

Tramas I

Cada campo esta formado por un número de *bits* múltiplo de 8 realizando las siguientes funciones:

Guión. Se utiliza para delimitar las tramas, apareciendo al principio y al final de cada una de ellas. Su conformación es: "01111110" (7E hex = 126 dec). Una secuencia de seis "1" seguidos no está permitida dentro de la trama por lo que el protocolo incluye el mecanismo de transparencia típico de los protocolos orientados a *bit* [Stallings_91].

Campo de dirección. Se utiliza para identificar tanto el origen de la trama como su destino, conteniendo facilidades para la operación con repetidores de nivel 2. Si la conexión se establece de forma directa entre dos estaciones, se utilizan únicamente 112 *bits* para especificar el destino y origen de la información (el equivalente a 7 *bytes* para cada uno). Cuando la conexión se establece a través de repetidores de nivel 2, el campo de dirección se extiende 56 *bits* por cada repetidor que se utiliza (máximo 8 repetidores).

El último grupo de 8 *bits* del subcampo destino dispone de un *bit* (*Bit-H*), para indicar si el campo de dirección contiene otra dirección correspondiente a un repetidor (*Bit-H = 0*) o la siguiente dirección corresponde al origen (*Bit-H = 1*). El subcampo dirección de repetidor se codifica de la misma forma que los subcampos dirección origen y destino, el *bit-H* se usa igualmente para indicar si a continuación hay otro repetidor o no. La operación con múltiples repetidores es un método para interconectar estaciones a largas distancias sin utilizar un protocolo de nivel 3.

Campo de control. Se utiliza para identificar el tipo de trama que se envía y controlar varios atributos de la conexión a nivel 2.

Campo PID. Debe aparecer en las tramas de información (I y UI) solamente. Identifica que clase de protocolo de nivel 3, si existe, está en uso.

Campo de información. Contiene los datos de usuario. El campo I puede tener una longitud máxima de 256 *bytes* y debe contener un numero entero de *bytes*.

Secuencia de comprobación de trama (FCS o CRC). 16 *bits* redundantes para la detección de errores en la transmisión. El polinomio utilizado para el cálculo de este campo sigue las Recomendaciones ISO 3309 (HDLC).

Cualquier trama formada por menos de 136 *bits* (incluyendo los guiones de inicio y fin), no acotadas por guiones de inicio y fin, o con un numero no entero de *bytes*, se consideraran invalidas por el nivel de enlace. Si se deja de transmitir una trama prematuramente, se enviarán al menos 15 señales consecutivas correspondientes al valor lógico "*uno*". Cuando es necesario para una estación mantener la portadora mientras no se envían tramas, el tiempo entre tramas se rellena con guiones contiguos.

Una vez especificada la estructura de las tramas, analizaremos los procedimientos utilizados por el protocolo para gestionar el enlace de datos utilizando el campo de control. Seguidamente se muestra el formato básico del campo de control en los tres tipos de tramas especificados con anterioridad.

Tipo Campo	*Bits Campo Control*			
Control	*7 6 5*	*4*	*3 2 1*	*0*
Trama I	N(R)	P/F	N(S)	0
Trama S	N(R)	P/F	S S	0 1
Trama U	M M M	P/F	M M	1 1

donde:

- El *bit* 0 es el primer *bit* enviado y el *bit* 7 es el último.
- N(S) es el número de secuencia de la trama enviada
- N(R) es el número de secuencia de recepción
- Los *bits* "S" son los *bits* de la función de supervisión.
- Los *bits* "M" son los *bits* modificadores de tramas no numeradas.
- El *bit* P/F es el *bit Poll/Final*. En el modo de operación balanceado este *bit* se utiliza para forzar una respuesta a una determinada trama que se envía como comando.

Tramas de Supervisión (S). Realizan la supervisión del control de enlace, utilizándose como confirmación o petición de retransmisión de tramas (I) y para el control de la ventana del nivel de enlace.

Los valores de los *bits* "S" en las tramas de supervisión se muestran a continuación con sus significados:

Bits Campo Control		7 6 5	4	3 2	1 0
Receive Ready	RR	N(R)	P/F	0 0	0 1
Receive not Ready	RNR	N(R)	P/F	0 1	0 1
Reject	REJ	N(R)	P/F	1 0	0 1

RR (Sistema preparado para recibir). Se utiliza para:

1. Indicar que el emisor de RR esta preparado para recibir más tramas.
2. Confirmar la recepción de tramas I (hasta N(R)-1 incluida).
3. Terminar una condición ocupado anterior creada por un comando RNR que se haya enviado.

RNR (Recepción imposible). Se usa para indicar al emisor de tramas I que el receptor esta ocupado temporalmente y no puede aceptar más tramas I. Las tramas hasta N(R)-1 se confirman.

REJ (Trama rechazada). Se usa para pedir la retransmisión de tramas I empezando por N(R). Se confirman las tramas anteriores hasta N(R)-1. Solo se permite una condición de trama rechazada en cada dirección al mismo tiempo. La condición de rechazo termina con la recepción de las tramas I adecuadas.

Tramas No Numeradas (U). Se utilizan para mantener un control adicional sobre el enlace. Son las responsables de establecer y terminar las conexiones de enlace. También permiten la transmisión y recepción de información fuera del control de flujo normal. Algunas tramas U pueden contener información y campos PID. Los campos de control de las tramas no numeradas son comandos (*Bit* P/F es *bit* P) o respuestas (*Bit* P/F es *bit* F). El contenido de los *bits* "M" de estas tramas identifican los distintos comandos (Cmd) y respuestas (Res) que pueden enviarse:

Campo Control	Tipo	*Bits* Campo Control
		7 6 5 4 3 2 1 0
SABM	Cmd	0 0 1 P 1 1 1 1
DISC	Cmd	0 1 0 P 0 0 1 1
DM	Res	0 0 0 F 1 1 1 1
UA	Res	0 1 1 F 0 0 1 1
FRMR	Res	1 0 0 F 0 1 1 1
UI	Ambos	0 0 0 P/F 0 0 1 1

Comando SABM. Se usa para poner un enlace entre dos estaciones en modo asíncrono balanceado. El receptor de un comando SABM confirma la recepción y aceptación del mismo enviando una trama respuesta UA, a la primera oportunidad. Si la estación receptora no es capaz de aceptar este comando debería responder con una trama DM, si es posible.

Comando DISC. Se usa para terminar una sesión de enlace entre dos estaciones. El receptor confirma la aceptación de DISC enviando una trama respuesta UA a la primera oportunidad. La estación que envió el DISC entra en estado desconectado cuando recibe la respuesta UA.

Respuesta UA. Se envía como confirmación a la recepción y aceptación de un SABM o DISC. Un comando recibido no se procesa hasta que la trama UA se envía.

Respuesta DM. Se envía cuando una estación recibe una trama distinta de SABM o UI mientras está en modo desconectado. También se envía a una petición de conexión, para indicar que no puede aceptar una conexión en ese momento.

Tramas UI. Contienen PID y campo de información. Se usan para enviar información fuera de los controles normales. Esto permite que campos de información viajen saltándose el control de flujo. Como esta tramas no se confirman, si se deterioran no hay manera de recuperarlas.

Respuesta FRMR. Se envía para comunicar que el receptor de una trama no la puede procesar con éxito y que la condición de error no es corregible enviando de nuevo la trama errónea. Esta condición aparecerá cuando una trama sin un error CRC se ha recibido en una de las siguientes condiciones:

1. Recibir un comando o respuesta invalida o no implementada.
2. Recibir una trama I cuyo campo de información sobrepasa la longitud permitida (256 *bytes*).
3. La recepción de un N(R) no adecuado.
4. La recepción de una trama con un campo de información no permitido, o la recepción de una trama U o S de longitud incorrecta.
5. La recepción de una trama S con el *bit* F a "1", excepto durante una condición de recuperación de tiempo o como respuesta a una trama comando enviada con el *bit* P a "1".
6. La recepción de una trama respuesta UA o DM no esperada.
7. La recepción de una trama con un N(S) invalido.

Cuando una trama FRMR se envía, se añade un campo de información de 3 *bytes* a la trama con información adicional sobre donde ocurrió el problema.

Bits Campo Información

23 22 21 20	19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0 0 0 0	Z Y X W	N(R) CR N(S) 0	C. Control trama rechazada

donde:

- En los *bits* 0-7 del campo información, se especifica el campo de control de la trama rechazada
- En el segundo octeto de *bits* se especifican las variables de estado de la estación que emite la trama FRMR.
- En los *bits* W,X,Y,Z del tercer octeto se especifica la causa del rechazo:
    1. Si W es "1", el campo control recibido era inválido o no implementado.
    2. Si X es "1", la trama que causo la condición de rechazo se consideró inválida, porque era una trama U o S que tenia un campo de información lo cual no esta permitido. El *bit* W debe ser '1' además del *bit* X.
    3. Si Y es "1", la trama recibida tiene un número de *bits* no múltiplo de 8 o es excesivamente larga
    4. Si Z es "1", el campo de control recibido contiene un N(R) inválido

Tramas de Información (I). Se distinguen por tener el *bit* 0 a "0" y contienen información del nivel superior. El campo de control de las tramas de información se codifica de la siguiente forma:

Bits Campo Control

7 6 5	4	3 2 1	0
N(R)	P/F	N(S)	0

Estas tramas son secuencialmente numeradas por N(S) para mantener el control de su paso a través de la conexión del nivel de enlace. En N(R) se especifica

la trama de información que espera recibir la estación que emite la trama. Esto permite mantener una supervisión del enlace sin necesidad de utilizar tramas S. El subcampo de información dentro de estas tramas tiene una longitud variable, con un máximo de (256 x 8) *bits*.

Para mantener la integridad dela conexión AX25 de nivel 2, se utilizan los siguientes tiempos de espera:

a) Tiempo de confirmación T_1 . Se utiliza para asegurar que una estación no espera indefinidamente una respuesta después del envío de una trama. Este tiempo no puede expresarse en tiempo absoluto, puesto que el tiempo necesario para enviar tramas depende de la velocidad de transmisión usada en el nivel físico. T_1 debería ser al menos dos veces la cantidad de tiempo necesario para enviar la trama de longitud máxima a la otra estación, ya que debe dar tiempo suficiente a esta estación para realizar algún proceso antes de responder. Si se usan repetidores de nivel 2, el valor de T_1 debe ajustarse según el número de repetidores que la trama tenga que atravesar.

b) Tiempo de retraso de respuesta *T2*. Puede implementarse en la estación para especificar el máximo retraso introducido entre, el instante en que se recibe una trama I y el instante en que se envía la respuesta resultante. Este retraso puede introducirse para permitir a la estación receptora esperar un poco de tiempo para determinar si hay más de una trama para enviar. Si durante esta espera se reciben más tramas (siempre teniendo en cuenta el tamaño de ventana), la estación receptora puede confirmarlas al mismo tiempo. El uso del tiempo T_2 no es obligatorio, pero se recomienda para mejorar la eficiencia del canal.

c) Tiempo de enlace inactivo *T3*. Se usa cuando T_1 no esta en marcha, para mantener la integridad del enlace. Se recomienda que siempre que no haya tramas I pendientes de confirmación, se envíe una trama RR o RNR con el *bit* P a "1" cada T_3 unidades de tiempo para preguntar el estado de la otra estación. T_3 se define localmente y depende en gran medida del nivel 1 de operación. T_3 debe ser mayor que T_1 y puede ser muy grande en canales de alta integridad.

3.3.2 FASES PARA EL INTERCAMBIO DE INFORMACIÓN

CONEXIÓN DEL ENLACE. Cuando una estación desea conectar con otra, enviará un comando SABM a ese dispositivo y activa su T_l. Si la otra estación admite la conexión responderá con una trama UA y ambas reinicializan sus variables internas N(S) y N(R). La recepción de una trama de respuesta UA en el otro extremo, provocará la cancelación del tiempo T_l considerando la fase de establecimiento del enlace completada y pasando a la fase de transferencia de la información.

Si la otra estación no responde antes de que venza T_l el dispositivo que pide la conexión reenviará una trama SABM, comenzando T_l de nuevo. La estación realiza esta operación un número de veces que puede seleccionarse a voluntad del usuario. Cuando el número de reintentos se ha completado sin conseguir establecerse el enlace se informa al nivel superior de la eventualidad.

Si al recibir un comando SABM la estación receptora no puede entrar en el modo de operación indicado, debería enviar una trama DM. Cuando se reciba una respuesta DM, la estación que envió el SABM deberá cancelar T_l e informar de la imposibilidad de establecer el enlace al usuario. Una estación que envía un comando SABM ignorará y descartará otros campos de control en la trama excepto SABM, DISC, UA o DM.

TRANSFERENCIA DE LA INFORMACIÓN. Durante esta fase tiene lugar el intercambio de información entre las estaciones que han establecido el enlace. En esta fase las estaciones transmiten y aceptan tramas conteniendo datos del nivel superior, junto con otra información de control que permita regular el tráfico y detectar posibles anomalías.

Siempre que se envía una trama de información se pone en marcha T_l , desactivándose cuando se tiene confirmación de que la trama ha sido recibida en la otra estación. Si una trama I o S se recibe correctamente, incluso en una condición de ocupado, el N(R) de la trama recibida debería comprobarse para ver si incluye una

confirmación de envíos pendientes de tramas I. El tiempo T_1 deberá cancelarse si la trama recibida actualmente confirma tramas pendientes anteriores.

Si T_1 vence antes de recibir una confirmación, el dispositivo debe actuar según el procedimiento de retransmisión. Este procedimiento puede variar según la versión del protocolo, así en la versión 1 se reexpiden de nuevo las tramas de información desde la primera no confirmada; mientras que en la versión 2 del mismo protocolo se envía una trama de supervisión con el *bit* P activado, para obligar a la estación receptora que informe acerca de la situación del enlace.

DESCONEXIÓN DEL ENLACE. En el estado de transferencia de información cualquier estación puede solicitar una desconexión del enlace, transmitiendo un comando DISC y activando T_1. La estación que recibe un DISC válido enviará una trama respuesta UA y entrará en estado de desconexión, completándose así la liberación del enlace lógico.

Si una respuesta UA o DM no se recibe correctamente antes de que venza T_1, la trama DISC se enviará de nuevo, reiniciándose T_1. Después de repetir un número de veces el proceso, la estación entra directamente en estado desconectado con la estación a la cual dirige el comando DISC.

Existen varias situaciones de bloqueo en el nivel de enlace que son recuperables sin terminar la conexión. Estas situaciones de error pueden ocurrir como resultado de mal funcionamiento de alguna de las estaciones o si existen errores en la transmisión

Condición de estación ocupada. Cuando una estación no puede recibir temporalmente tramas I, por ejemplo cuando tiene los *buffers* de recepción llenos, enviará una trama RNR. Siempre que se recibe una trama RNR, se descartará la transmisión de tramas I hasta que la condición de ocupado termine. Esta condición termina enviando una trama UA, RR, REJ o SABM.

Recuperación de tramas con errores en CRC. Se realiza mediante la trama de supervisión REJ, enviando ésta cada vez que se detecta una trama bien conformada pero con un error en el CRC de la misma. Esta condición termina cuando la trama I requerida se reciba. Una estación que reciba el comando REJ terminará la condición reenviando todas las tramas I pendientes, comenzando con el número de trama indicada en el N(R) de la trama REJ. Si una estación recibe una trama REJ con el *bit* P, debería responder con una trama RR o una RNR con el *bit* F antes de retransmitir las tramas I pendientes.

Condición de rechazo de trama. Una condición de rechazo de trama ocurre cuando una trama sin errores CRC se recibe con una de las condiciones expuestas anteriormente en la *respuesta FRMR*. Una vez que ocurre un error de rechazo no se aceptan más tramas I (excepto para examinar el *bit* P/F) hasta que se resuelve el error.

Recuperación de T_1. Si una estación, debido a un error de transmisión, no recibe una trama I individual o la ultima trama I en una secuencia de tramas I, no detectará un error en el formato de la trama o en el CRC y por tanto no transmitirá: ni RR, ni FRMR, ni REJ . La estación que transmitió las tramas I no confirmadas, tras pasar el periodo de tiempo T_1, deberá tomar las acciones de recuperación según se ha comentado en la fase de transferencia de la información. Esta condición termina con la recepción de una confirmación para las tramas enviadas, o reinicializando el enlace.

Recuperación de T_3 . El tiempo T_3 se usa para asegurar que el enlace todavía es funcional durante periodos de poca transferencia de información, siempre que T_1 no este en marcha (no hay tramas I pendientes). Cuando vence T_3 se transmite una trama RR o RNR como un comando con el *bit* P, ejecutándose el procedimiento de espera de confirmación.

3.4 EVALUACIÓN DEL RENDIMIENTO DE LA RED DE COMUNICACIONES

Gracias a la flexibilidad de parametrización de los dispositivos TNC que tienen implementado el protocolo AX25, podemos configurar la red de comunicaciones adaptándola a las características del tráfico generado por la red de ordenadores.

Como hemos mencionado anteriormente, la norma del protocolo AX25 no especifica el algoritmo de acceso al canal, utilizándose normalmente el ALOHA puro o el CSMA (apartado 3.2.2). El TNC dispone de un procedimiento de acceso CSMA *p-persistente*, donde la persistencia y otros parámetros útiles para el correcto funcionamiento de la red, se codificarán en función de las características del tráfico que va a cursar.

Esta es una de las posibilidades de acceder al medio en la red de comunicaciones, pero no la única. Combinando adecuadamente algunos de los parámetros del TNC, podemos conseguir que el acceso al medio se produzca mediante una selección controlada por una de las estaciones de la red. De esta forma se evitarán las colisiones en el medio, pero se obtendrá una mayor rigidez en las comunicaciones, ya que éstas deben ser secuenciales aunque no exista información que transmitir. La utilización de uno u otro procedimiento vendrá determinado por las particularidades del proceso a cuyo control se destina la red de ordenadores.

Un estudio generalizado del rendimiento de esta red de comunicaciones resultaría demasiado ambiguo, pues debería restringirse antes el amplio abanico de

posibilidades a una situación más concreta. Por otra parte el rendimiento de las redes que utilizan procedimientos de acceso al medio CSMA con persistencia *p* están suficientemente estudiados; por lo que en este apartado vamos analizar las prestaciones de la red de comunicaciones, adaptada al mismo supuesto práctico al que se aplicará posteriormente la red de ordenadores.

Para que esta red proporcione un servicio adecuado para la recopilación de datos sismológicos, obtenidos por sistemas autónomos geográficamente dispersos, hemos de tener en cuenta que, cuando se produce un evento significativo (terremoto) todas las estaciones de una red sismológica registrarán el suceso, o al menos una gran parte de ellas, y los sistemas informáticos que controlan cada estación de campo desearán transmitir la información al centro de proceso prácticamente a la vez. En una situación de estas características, donde varias estaciones desean enviar al medio información de forma simultánea, la probabilidad de colisión sería elevada y para evitar gran parte de las mismas debería bajarse el valor de la *persistencia* para obtener un rendimiento adecuado. Sin embargo, disminuir el valor de la *persistencia* penalizaría el acceso de cualquier comunicación al medio; y en los casos en los que se desea transmitir ficheros entre dos estaciones exclusivamente, el rendimiento decaería notablemente.

En estas condiciones es aconsejable un procedimiento de selección para asignar el canal y solicitar así, de forma ordenada, toda la información de cada una de las estaciones. Los procedimientos de selección, si bien presentan una ventaja notable cuando todas las estaciones tienen información que transmitir, resultan lentos cuando la información es esporádica y sólo por parte de alguna de las estaciones de la red. Este sería el caso, por ejemplo, de una estación que necesita transmitir ocasionalmente una alarma o un aviso para solicitar datos necesarios para el mantenimiento de la red. Si se produce esta situación, deberá esperar a que el canal se le asigne para poder comunicar el suceso.

Para compaginar ambas situaciones se ha optado por un procedimiento mixto de asignación del canal. Así, mientras no existan mensajes circulando por la red,

cualquier estación puede acceder al medio para enviar un aviso indicando que tiene algún mensaje pendiente. Una vez que este hecho se produce la estación situada en el centro de recogida de datos se convierte en una estación principal, a efectos de control de comunicaciones en la red, y establece un turno de asignación del canal, empezando por la estación que originó el aviso.

De esta forma se obtiene agilidad en cuanto a la comunicación de sucesos en la red de ordenadores y aseguramos que, cuando existe información voluminosa que transmitir, la efectividad del canal no va a verse reducida por la existencia de colisiones entre los radio-paquetes. Este procedimiento permite mantener el medio en silencio si no existe información que intercambiar entre las estaciones de la red, ya que no es necesario seleccionarlas continuamente para asignarles el canal.

Como en la fase de transferencia de información no existen colisiones, el parámetro de persistencia de la red de comunicaciones se codifica a su valor máximo. Así, el acceso al medio será *1-persistente* y se transmitirá siempre que se encuentre el canal en silencio [Alabau_86]. Esto permite ignorar el *tiempo de slot* para calcular la eficiencia del canal, ya que el algoritmo que determina el acceso al medio dará siempre como resultado que la información sea enviada.

En la fase del establecimiento de las conexiones del enlace de datos, entre la estación principal y el resto de estaciones de la red, existe un procedimiento CSMA para asignar el canal, donde pueden producirse colisiones. El hecho de asignar a la persistencia su valor máximo podría perjudicar en esta fase, sin embargo existen suficientes mecanismos en la red de comunicaciones (escucha antes de transmitir, reintentos en las solicitudes de conexión si estas no son asentidas por el receptor) y otros que se pueden implementar en la red de ordenadores, para asegurar que el enlace de datos sea establecido siempre que sea necesario.

Una vez fijadas las características de trabajo de la red de comunicaciones se analizará el rendimiento que la misma puede proporcionar, así como qué parámetros pueden afectar a este rendimiento y en qué medida se puede actuar sobre los mismos

para obtener las mejores prestaciones. Debe entenderse que el rendimiento que se pretende evaluar hace referencia, exclusivamente, a la red de comunicaciones, es decir, se trata de determinar que capacidad tiene la red de comunicaciones para tramitar la información proporcionada por los sistemas conectados a la misma. Este rendimiento es básico para determinar el rendimiento de la red de ordenadores, pero para calcular este último es necesario considerar también el tiempo requerido para que se ejecuten los procesos definidos para la red de ordenadores.

Para cuantificar el rendimiento de la red de comunicaciones, debemos definir previamente que entendemos por ***rendimiento*** en el contexto de las comunicaciones: "*relación entre la capacidad efectiva de la red y la capacidad del canal*" [García_90].

$$R = \frac{V_e}{V_c} \qquad (3.1)$$

donde: V_e = Velocidad efectiva (nº de *bit* útiles/seg)

V_c = Velocidad del canal (nº *bit*/seg)

La velocidad del canal es un termino fijo que corresponde a la velocidad de transmisión de la red de comunicaciones. La velocidad efectiva del canal se determina como: la relación entre el número de *bit* útiles n , es decir, todos los datos suministrados por el nivel superior al nivel de enlace para ser emitidos a través de la red de comunicaciones, sin contar aquellos *bits* que el protocolo de enlace introduce para realizar sus funciones (*bit* redundantes); y el tiempo que el circuito es ocupado para enviar esta información T_0.

$$V_e = \frac{n}{T_o} \qquad (3.2)$$

Si el emisor de una trama de datos no recibe confirmación de la misma en el tiempo T_1, el propio protocolo AX25 se encarga del proceso de recuperación. Si se

recibe una trama errónea el receptor la rechaza y esta se reexpide por parte del emisor.

El tiempo T_0 empleado para enviar *n bit* de información puede descomponerse, según se aprecia en la figura 3.3, en:

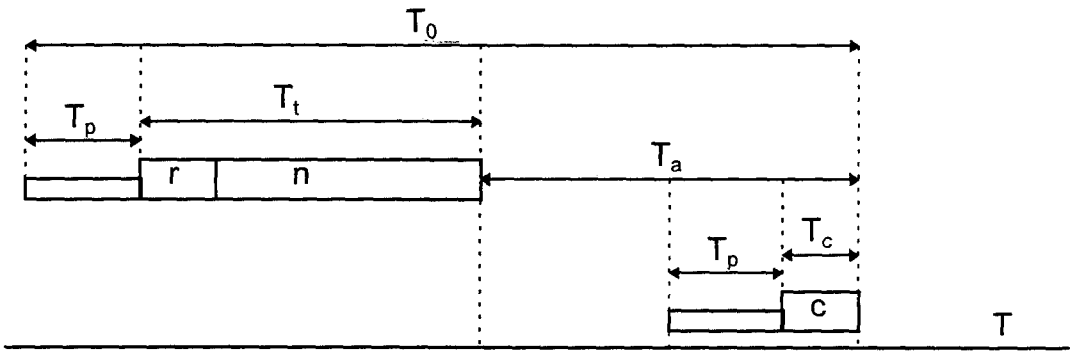

Figura 3. 3 Desglose del tiempo de ocupación del circuito

T_p = Tiempo de portadora

T_t = Tiempo de escritura de la trama en el medio. En la trama deben incluirse, además de los *n bits* de información, los *r bits* que el protocolo incluye para la gestión del enlace.

T_a = Tiempo de asentimiento. Tiempo transcurrido desde que el emisor envía el último *bit* de la trama, hasta que interpreta la confirmación o rechazo de la misma.

$$T_0 = T_p + T_t + T_a \qquad (3.3)$$

A su vez podríamos dividir el tiempo de asentimiento en: tiempo empleado en enviar la confirmación T_c (anteponiendo previamente la portadora T_p) y tiempo de respuesta del receptor T_r. Para los cálculos teóricos iniciales utilizaremos T_a englobando estos tiempos.

Existe un termino más a tener en cuenta para determinar el rendimiento. Este termino es el número de transmisiones por trama (N_t). Explicaremos en que consiste a partir de un ejemplo:

Supongamos que transmitimos 100 tramas. En un circuito ideal, sin errores, el número de transmisores por trama sería: $N_t = 1$. Considerando un caso más real, donde los errores han afectado a tres tramas que han tenido que retransmitirse, a la relación entre el número total de tramas transmitidas (103) y el número de transmisiones válidas (100) lo identificamos con el número de transmisiones por trama (N_t = 103/100 = 1,03).

Así, a efecto de cálculos temporales podemos suponer que cada trama se transmite 1,03 veces, aunque en realidad se transmiten 97 tramas una vez y 3 tramas 2 veces. El valor de N_t determina la proporción de transmisiones mas retransmisiones que le correspondería a cada trama [Chu_74].

Estadísticamente N_t viene determinado por:

$$N_t = \sum_{i=1}^{i=\infty} i \,.\, P_i \qquad (3.4)$$

donde P_i es la probabilidad de que una trama requiera i retransmisiones.

$$P_i = p^{(i-1)} \,.\, (1-p) \qquad (3.5)$$

siendo p la probabilidad de que una trama se vea afectada por errores de transmisión.

Mediante medidas experimentales en circuitos reales, se ha determinado que la probabilidad de que una trama se reciba con error, es proporcional a la longitud de la trama [Vidaller_78], siendo esta probabilidad:

$$p = k \,.\, (n + r) \qquad (3.6)$$

donde $n + r$ es el número total de *bits* en la trama (*bits* útiles mas redundantes) y k es la tasa de error medio del circuito definida en el apartado 3.2.1.

Así N_t nos queda:

$$N_t = \sum_{i=1}^{i=\infty} i \,.\, p^{(i-1)} .(1-p) = \frac{1}{1-p} \qquad (3.7)$$

Para obtener el tiempo real (T_0') empleado en transmitir los n *bit* de información, en un circuito con errores, deberá multiplicarse el tiempo utilizado para enviar una trama por N_t, así:

$$T_0' = (T_p + T_t + T_a).\, N_t \qquad (3.8)$$

como:

$$T_t = \frac{n+r}{V_c} \qquad (3.9)$$

tenemos:

$$T_0' = \left(\frac{n+r}{V_c} + T_a + T_p\right) \frac{1}{1-p} \qquad (3.10)$$

utilizando **(3.2)**:

$$V_e = \frac{n}{T_0'} = \frac{n}{\left(\frac{n+r}{V_c} + T_a + T_p\right)\frac{1}{1-p}} = \frac{(1-p).\, n\, .\, V_c}{n + r + (T_a + T_p).\, V_c} \qquad (3.11)$$

el rendimiento será:

$$R = \frac{V_e}{V_c} = \frac{(1-p).\, n}{n + r + (T_a + T_p).\, V_c} \qquad (3.12)$$

A partir de esta expresión podemos realizar una estimación del rendimiento de la red de comunicaciones configurada con las características descritas. Como el tiempo de asentimiento depende del tiempo de portadora (véase, figura 3.3), nos interesa desglosar este término, ya que el tiempo de portadora es un parámetro que puede variar, debiendo de ajustarse a las condiciones de emisión-recepción del canal

de comunicación. Por otra parte T_a no puede determinarse experimentalmente de una forma directa.

Para obviar esta situación disponemos de un parámetro en el TNC que nos puede ayudar a realizar la determinación experimental rendimiento. En el dispositivo que tiene implementado el protocolo AX25, se parametriza el T_{resp} (tiempo de espera para asentimientos). Este parámetro determina el tiempo mínimo que se impone al receptor para el reconocimiento de tramas de información. Cuando un TNC detecta una trama con un campo de control que la identifica con una trama I, se activa T_{resp} y sólo cuando este vence, se produce la confirmación o rechazo de dicha trama por el receptor. Este parámetro se codifica en intervalos de 100 ms y lógicamente para que actúe adecuadamente, este tiempo debe ser mayor que el tiempo necesario para enviar una trama de longitud máxima al medio ($T_{resp} > T_t$).

Para realizar nuestros cálculos consideraremos que el tiempo de propagación en el medio es despreciable frente a los tiempos que estamos manejando y consiguientemente podemos suponer, sin introducir un error apreciable, que cuando un *bit* se envía al medio es automáticamente detectado por el receptor. Con esta premisa y teniendo en cuenta el desglose de tiempos que se efectúa en la figura 3.4, podemos calcular el tiempo de respuesta del receptor (T_r), como la diferencia entre T_{resp} y el tiempo utilizado por el emisor para enviar el resto de la trama -a partir del campo de control- al medio (T_t').

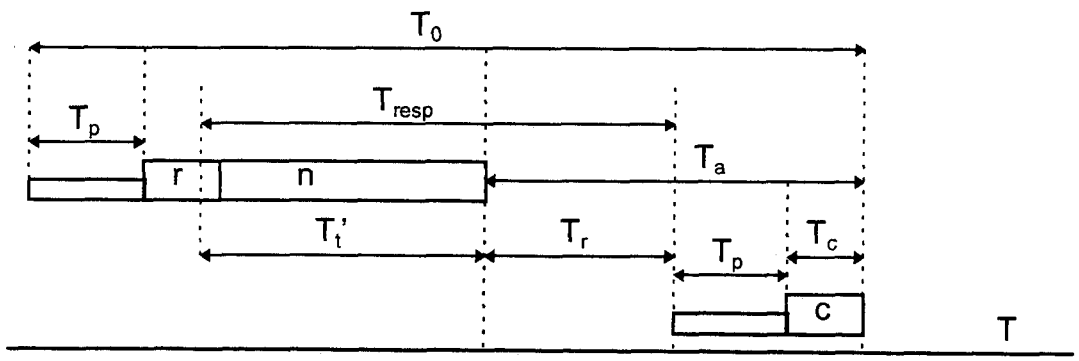

Figura 3. 4 Desglose del tiempo de ocupación del circuito

El tiempo de confirmación de la trama (T_c) se determina fácilmente porque conocemos el nº de *bit* de una trama de asentimiento (c) y la velocidad de transmisión. De esta forma podemos calcular indirectamente T_a como:

$$T_a = T_r + T_p + T_c \qquad (3.13)$$

donde:

$$T_r = T_{resp} - T_t' \qquad (3.14)$$

siendo:

$$T_t' = \frac{r' + n}{V_c} \qquad (3.15)$$

r' será el número de *bits* redundantes del protocolo enviados después del campo de control [PID, CRC y guión final (4 x 8 *bits*)], de acuerdo con la estructura de la trama AX25.

Utilizando **(3.13)**, podemos expresar **(3.12)** como sigue:

$$R = \frac{(1-p).n}{n + r + (2T_p + T_r + T_c).V_c} \qquad (3.16)$$

De la misma forma que se supone despreciable el tiempo de propagación, frente a los tiempos manejados en la expresión anterior, también debe suponerse despreciable la probabilidad de que una trama de confirmación o rechazo se vea afectada por el ruido del canal y sea rechazada. Debe tenerse en cuenta que como se ha expresado en **(3.6)**, esta probabilidad es proporcional al tamaño de la trama y dada su menor longitud (20 *bytes*) frente al tamaño de una trama de información (276 *bytes*), esta consideración resulta acertada [Alabau_86].

Seguidamente se evaluará el rendimiento que proporciona esta red de comunicaciones trabajando a velocidades de 9600 y 19200 bps. Para ello

consideramos el envío de tramas de información de longitud máxima (256 bytes de datos, $n = 2048$ *bits*). El valor de r, trabajando directamente entre dos estaciones sin repetidores de nivel 2, será: $r = 20 \times 8 = 160$ *bits*.

Así, utilizando **(3.9)** tenemos:

Para $V_c = 9600\ bps$ $\qquad T_t = 0{,}23\ s$

Para $V_c = 19200\ bps$ $\qquad T_t = 0{,}115\ s$

Como T_{resp} debe ser múltiplo de 0,1 s y mayor que T_t $\Rightarrow$

Para $V_c = 9600\ bps$ $\qquad T_{resp} = 0{,}3\ s$

Para $V_c = 19200\ bps$ $\qquad T_{resp} = 0{,}2\ s$

Utilizando **(3.15)** obtenemos:

Para $V_c = 9600\ bps$ $\qquad T_t' = 0{,}21667\ s$

Para $V_c = 19200\ bps$ $\qquad T_t' = 0{,}10833\ s$

Por lo que el tiempo de respuesta (T_r) del receptor será:

Para $V_c = 9600\ bps$ $\qquad T_r = T_{resp} - T_t' = 0{,}08333\ s$

Para $V_c = 19200\ bps$ $\qquad T_r = T_{resp} - T_t' = 0{,}09167\ s$

El tiempo de confirmación es:

Para $V_c = 9600\ bps$ $\qquad T_c = \dfrac{c}{V_c} = 0{,}01667\ s$

Para $V_c = 19200\ bps$ $\qquad T_c = \dfrac{c}{V_c} = 0{,}00833\ s$

Como T_r, T_c, n y r son parámetros fijos dentro de cada velocidad para todas las condiciones, determinaremos el valor del rendimiento de la red en función de distintas tasas de error en el circuito y parametrizando distintos tiempos de portadora.

Utilizando la expresión **(3.6)** para determinar el valor de p, los valores del rendimiento calculados mediante la expresión **(3.16)** quedan expresados en la tabla 1. Para realizar los cálculos se ha tomado como referencia valores de $k = 0$, que correspondería a un circuito sin errores; $k = 10^{-4}$ como una tasa de error no apropiada aunque la transmisión sea posible; $k = 10^{-5}$ como la tasa de error mínima recomendada; y $k = 1{,}2 \cdot 10^{-6}$ por ser la tasa de error experimental obtenida en laboratorio.

V_c (bps) → T_p (ms) ↓	CIRCUITO SIN ERRORES $k = 0$		$k = 10^{-4}$		$k = 10^{-5}$		$k = 1{,}2 \cdot 10^{-6}$	
	9600	19200	6900	19200	9600	19200	9600	19200
10	0,609	0,454	0,475	0,354	0,596	0,444	0,607	0,474
20	0,576	0,418	0,449	0,326	0,563	0,409	0,574	0,417
30	0,547	0,388	0,426	0,302	0,535	0,379	0,546	0,387
40	0,520	0,362	0,405	0,282	0,509	0,354	0,519	0,361
50	0,496	0,339	0,386	0,264	0,485	0,332	0,495	0,338
60	0,474	0,318	0,369	0,248	0,464	0,311	0,473	0,317
70	0,454	0,300	0,354	0,234	0,444	0,293	0,453	0,299
80	0,435	0,284	0,339	0,221	0,425	0,278	0,434	0,283
90	0,418	0,270	0,326	0,210	0,409	0,264	0,417	0,269
100	0,403	0,257	0,314	0,200	0,394	0,251	0,402	0,256

Tabla 1. Rendimiento de la red de comunicaciones en función de: V_c, k y T_p

Expresado en tanto por ciento el máximo rendimiento que podemos obtener en la red de comunicaciones, con los condicionantes impuestos, es de un 60,9 % si trabajamos a 9600 bps y del 45,4 % trabajando a 19200 bps.

Ambos resultados pueden considerarse buenos teniendo en cuenta que, en transmisiones vía radio, los tiempos de portadora son necesarios para el correcto funcionamiento del sistema y según puede observarse en la tabla 1, el valor de T_p influye en gran medida en el rendimiento de la red. El hecho de no disponer de dos canales simultáneos para ir confirmando por uno de ellos, mientras por el otro se envía la información, afecta en gran medida también a estos resultados.

La diferencia de rendimiento de aproximadamente un 15%, bajo las misma condiciones, que tenemos en la red trabajando a 9600 bps y 19200 bps es normal porque los tiempos que no se emplean en enviar información, pero que han de consumirse en la transmisión, deben multiplicarse por la velocidad del canal para determinar el número de *bits* que dejan de enviarse al medio. Por este motivo para los mismos tiempos, el aumento de la velocidad del canal va en detrimento de la efectividad de la red de comunicaciones.

Aunque el rendimiento aumenta con la disminución de la velocidad de transmisión, el parámetro que debe valorarse a la hora de seleccionar una u otra opción es la velocidad efectiva del canal:

$$V_e = R \cdot V_c$$

La velocidad del canal que se seleccionará será la que proporcione una mayor velocidad efectiva, siempre dentro de las limitaciones que impone el ancho de banda asignado por la Administración para el canal de radio frecuencia concedido. Así conseguimos, con los mismos elementos, la máxima velocidad efectiva.

Como la trama de longitud máxima que podemos enviar consta de 2208 *bits*, podemos transmitir información con tasas de error medio próximos a $k = 10^{-4}$. Sin embargo, no es recomendable la utilización de un circuito de estas características, ya que una pequeña variación de las condiciones del canal podría ocasionar que la transmisión por el mismo fuera imposible.

Para incidir en la tasa de error medio del circuito, además de los ajustes necesarios en las emisoras de radio y antenas, podemos utilizar el valor del tiempo de portadora T_p. El valor asignado al tiempo de portadora puede afectar a la tasa de error del circuito ya que, si este tiempo no es suficiente, los primeros *bits* de cada trama pueden perderse.

Aunque el rendimiento de dos posibles opciones con distintos T_p y tasas de error, puedan ser similares (por ejemplo, a 9600 bps, con T_p = 20 ms y $k = 10^{-4}$

obtenemos un rendimiento parecido si T_p = 70 ms y la tasa de error es $k = 10^{-5}$, véase tabla 1), siempre nos inclinaremos por aquella que tenga una tasa de error menor. Tomar esta medida es necesario para garantizar un tráfico fluido en la red.

Por lo tanto marcaremos como referencia la tasa de error medio del circuito, teniendo en cuenta que debe ser $k \leq 10^{-5}$, ya que a partir de este valor de k el rendimiento empieza a ser muy próximo al rendimiento ideal de un canal sin errores. Con esta base seleccionaremos el T_p menor posible.

De la misma forma que en laboratorio obteníamos un valor para la tasa de error medio ($k = 1{,}2 \cdot 10^{-6}$), gracias a las prestaciones de los dispositivos TNC, siempre podemos monitorizar cualquier conexión real y determinar el número de tramas rechazadas. De esta forma se obtiene un valor experimental de k y se pueden realizar los ajustes necesarios para obtener las condiciones deseadas.

Para proceder a fijar el ajuste del tiempo de portadora en los canales entre la estación de recogida de datos y las distintas estaciones de campo, operaremos como sigue:

1. Se ajustarán los equipos de radio para conseguir las mejores condiciones de emisión/recepción en cada emplazamiento.

2. Se fija el menor valor de T_p (10 ms) y se determina un valor experimental para k con estas condiciones.

3. Si el valor de k no es satisfactorio se irá incrementando el valor de T_p hasta conseguir que la tasa de error media del circuito sea: $k \leq 10^{-5}$.

CAPITULO 4

DISEÑO DE LA RED DE ORDENADORES

- ***Metodología de diseño***
- ***Estructura de la red***
- ***Aplicaciones en red***
- ***Arquitectura***
- ***Protocolos de nivel***

4.1 METODOLOGÍA DE DISEÑO

Según se estableció en el capítulo 2 para realizar un *control digital directo* sobre procesos distribuidos vamos a servirnos de las prestaciones que proporciona una red de ordenadores. Los factores diferenciales que hacen que una red de ordenadores deba tener un tratamiento formal distinto de un sistema informático constituido por un solo nodo son múltiples [García_90].

Se pueden citar, entre otros:

Conceptuales: Interconecta nodos autónomos que pueden operar de forma integrada o independiente

Estructurales: Es un macrosistema cuyos componentes son unidades independientes, en general computadores y canales de comunicación, con sus elementos lógicos asociados

Operativos: El sistema operativo para gestión integrada debe poder trabajar de forma distribuida o centralizada, sincronizando en cada caso la repartición de procesos, datos y recursos

Geográficos: Las distancias entre nodos son importantes, por lo que han de optimizarse los costes de los enlaces, diseño topológico, asignación de capacidades y conectividad

Volumen: Número de nodos y cantidad de tráfico que se genera

El propósito de cualquier red de ordenadores es hacer que un conjunto de programas, datos y equipo estén disponibles para cualquier sistema, conectado a ella, que así lo solicite [Tanenbaum_91]. Sin embargo, los objetivos que se pretenden cubrir con su utilización pueden ser diferentes. Esto explica el hecho de que en la actualidad existan funcionando un gran número de redes muy diversas en el mundo.

La *estructura*[1] de una red de ordenadores y su *arquitectura*[2] dependen, en gran medida, de la utilidad final para la que ha sido concebida [Halsall_92]. Así, por ejemplo, mientras que uno de los objetivos fundamentales de la red ARPANET era manejar la interconexión de un gran número de redes WAN[3] y LAN[4], para las redes públicas el aspecto de interconexión de redes se trató como una idea posterior y no constituye uno de los ejes centrales de su diseño. Esto explica que sus niveles de red, por ejemplo, difieran notablemente.

La finalidad de la red que se propone, quedó establecida en el capítulo 2 de esta memoria. El primer objetivo a la hora de realizar su diseño será la confección de una estructura que haga posible su materialización.

La elección de los sistemas informáticos que deben formar parte de la estructura de la red de ordenadores puede ser sencilla, pues únicamente debe cuidarse que sean suficientes para ejecutar las funciones de control requeridas y que dispongan de un módulo de comunicaciones para interactuar con la red.

El problema no es tan simple cuando se trata de diseñar la conexión de los ordenadores a la subred de comunicaciones [Schwartz_94]. En este punto es necesario responder a varias preguntas, como: ¿es conveniente dedicar una línea

---

[1] *Estructura de red:* Conjunto de sistemas destinados a ejecutar programas de usuario y sistema de interconexión que hace posible la comunicación entre los mismos.

[2] *Arquitectura de red:* Conjunto de normas que regulan la interconexión de los sistemas conectados a la red.

[3] *WAN:* Wide Area Network (Red de Área Extendida)

[4] *LAN:* Local Area Network (Red de Área Local)

permanente a cada usuario de la red?; ¿deben existir usuarios compartiendo un dispositivo de acceso?; ¿se debe confeccionar una subred con conexiones punto a punto?; ¿es más conveniente utilizar una subred de difusión?; si disponemos de un canal de datos compartido ¿el acceso va a ser determinístico o estadístico?; etc.

Las características del tráfico que va a cursar la red, así como la ubicación dispersa de los sistemas informáticos determinan las peculiaridades que debe reunir la subred de comunicaciones. Estos condicionantes han sido analizados en el capítulo 3 de la memoria, permitiendo la elección de una subred de comunicaciones que se ajusta a los requisitos planteados.

Una vez explicitada la estructura de la red y antes de proceder al diseño de su arquitectura, es necesario contemplar un aspecto más. Las aplicaciones de red que son necesarias para que los sistemas informáticos, usuarios de la misma, puedan llevar a efecto las funciones de control programadas.

Las aplicaciones de red no solo afectan al diseño de su nivel correspondiente, si no que pueden influir en la composición del resto de niveles que conforman la arquitectura y en la definición de los servicios que estos deben ofertar a su nivel usuario. Dedicaremos un apartado, dentro de este capítulo, a analizar que aplicaciones deben ejecutarse en red para que el procedimiento de control propuesto sea efectivo.

Con estas premisas estaremos en condiciones de proponer una arquitectura para la red de ordenadores. Prácticamente todas las arquitecturas de redes actuales son de carácter estructurado, organizándose en una serie de capas o niveles, con objeto de reducir la complejidad de su diseño. Cada nivel se construye sobre su predecesor de forma jerarquizada. El número de niveles, nombre y funciones que realizan varía de una red a otra. Sin embargo, en cualquier red el propósito de cada nivel es ofrecer ciertos servicios a su nivel superior, liberándolo del conocimiento detallado de como se realizan estos servicios [Tanenbaum_91].

Aunque con las arquitecturas estructuradas se simplifica la labor de diseño, identificando fácilmente lo que ocurre en cada nivel, se eleva la redundancia de datos (se ha de incluir una cabecera por nivel) y el número de procesos que se ejecutan (al menos uno por nivel). No obstante las ventajas ofrecidas son superiores a los inconvenientes que puedan producirse. Las mejoras derivadas de utilizar una arquitectura estructurada, pueden resumirse en las siguientes [García_90]:

- Distribución de funciones
- Independencia de los dispositivos
- Independencia de las comunicaciones
- Facilidades de cambios
- Simplicidad del control de la red
- Mejora de la disponibilidad del sistema
- Mejora de la eficacia del sistema

Siguiendo la metodología habitual organizaremos nuestra red en una serie de niveles, construyendo cada uno de ellos sobre su predecesor. Definir las funciones asignadas a cada nivel será nuestro siguiente objetivo. La ejecución de estas funciones determinará los servicios que cada nivel oferta a su nivel inmediato superior.

Los niveles no son bloques monolíticos, si no que están constituidos por una conjunto de *entidades*[5] de nivel, como mínimo una en cada extremo de la comunicación. Las entidades del mismo nivel ubicadas en distintas máquinas (entidades pares o iguales), cooperan entre sí para llevar a cabo las funciones asignadas a este nivel. Esa cooperación exige la existencia de un diálogo, que se

[5] *Entidades:* Conjunto de elementos activos que se encuentran en cada uno de los niveles. Pueden ser *software* (como un proceso), o *hardware* (como un *chip* inteligente de E/S) [TAN_91].

realiza bajo un conjunto de normas conformando lo que se denomina *protocolo* específico para dicho nivel [Alonso_96].

Nuestra siguiente tarea será diseñar los protocolos correspondientes a cada uno de los niveles definidos en la arquitectura. Para implementar adecuadamente estos protocolos es necesario la especificación formal de los mismos. Como conclusión del trabajo de diseño se realizará la especificación formal de los protocolos mediante máquinas de estados finitos y se efectuarán una serie de sugerencias para su implementación.

Seguidamente, y a modo de resumen, se enumeran los pasos a seguir en el diseño de la red de ordenadores:

1. Definir una estructura para la red cuyos elementos, sistemas informáticos y subred de comunicaciones, se conformen como consecuencia de la utilización que el usuario hará de la red. La subred de comunicaciones debe adecuarse al tráfico y a la dispersión geográfica de los sistemas conectados a la misma.
2. Determinar las aplicaciones de red necesarias para que los sistemas informáticos, usuarios de la red, puedan cumplir los objetivos programados.
3. Definir una arquitectura estructurada en niveles, compatible con los condicionantes derivados de los dos puntos anteriores.
4. Especificar las funciones asignadas a cada nivel, para que provea del servicio adecuado a su nivel inmediato superior.
5. Diseñar los protocolos que permiten la comunicación entre niveles pares, situados en sistemas diferentes.
6. Realizar una especificación formal de los protocolos confeccionados, para facilitar su verificación e implementación.

4.2 ESTRUCTURA DE LA RED

La red de ordenadores propuesta para realizar un control distribuido sobre plantas remotas queda representada en la figura 4.1. En ella pueden distinguirse los sistemas destinados a ejecutar los programas (ordenador central OC y sistemas informáticos remotos SIR) y el sistema de interconexión que permite el diálogo entre estos (subred de comunicaciones).

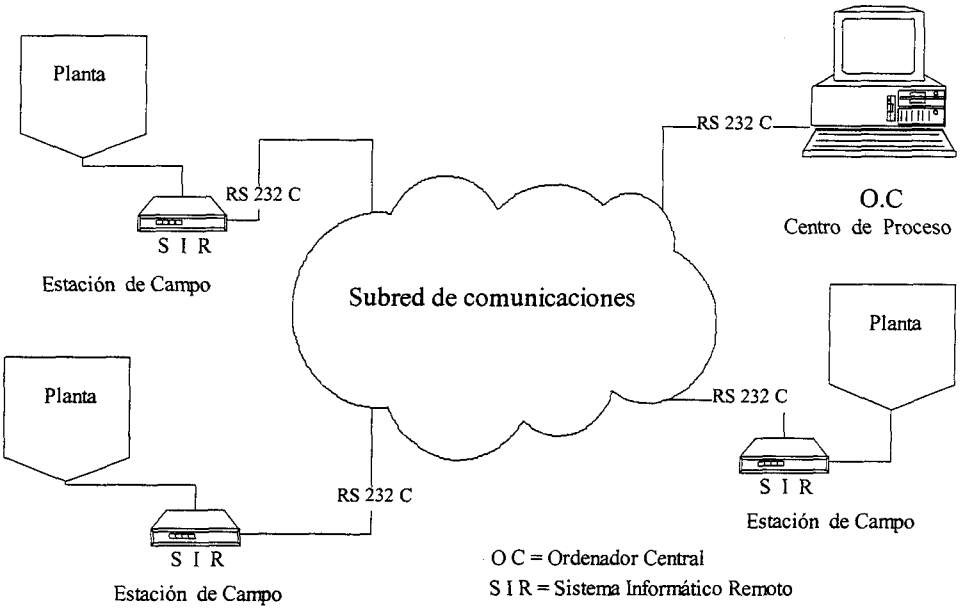

Figura 4.1 Red de Ordenadores

Una red de ordenadores es una colección *interconectada* de sistemas informáticos *autónomos*. Debemos de comprobar que el sistema propuesto cumple con estos dos requisitos para identificarlo con una red de ordenadores y proceder, consiguientemente, al diseño de su arquitectura.

El termino "*central*", asignado al sistema informático ubicado en el centro de proceso, hace referencia al elemento que coordina diversos controles directos y supervisa, de alguna forma, la integración de todo el proceso. Su nomenclatura no implica que este sistema asuma las funciones de una estación principal en la red de ordenadores y que gobierne el comportamiento de los sistemas informáticos situados en las estaciones de campo. Si esto fuera así, la relación entre los sistemas se convertiría en una típica relación maestro-esclavo, dejando de constituir una red de ordenadores.

Los sistemas informáticos remotos también son entes autónomos, siendo capaces de interactuar con el proceso y con la red de forma independiente. Su programación debe permitir realizar un control local, elaborando información depurada para enviar a cualquiera de los sistemas conectados a la red de comunicaciones.

Dos ordenadores están interconectados si son capaces de intercambiar información entre sí. La subred de comunicaciones que permite este dialogo, ajustándose a las características geográficas de ubicación de los sistemas y al tráfico previsto, fue analizada en el capítulo 3 de la memoria. Su misión es servir de soporte al envío de mensajes entre los sistemas informáticos conectados a la red.

Consiguientemente el sistema propuesto cumple con los requisitos estructurales necesarios para ser considerado una red de ordenadores. Esto nos permitirá repartir la potencia de tratamiento de la información en el espacio, y los elementos *hardware* y *software* de la red cooperarán conjuntamente para posibilitar un control distribuido de los procesos.

Para abordar con una mejor perspectiva el diseño de la arquitectura de la red de ordenadores, describiremos las funciones asignadas a los sistemas informáticos que la componen. Estas funciones hacen únicamente referencia al trabajo que realizan como integrantes de la red de ordenadores. Por encima de estas funciones estarían aquellas destinadas al control específico del proceso o procesos remotos.

FUNCIONES DEL ORDENADOR CENTRAL:

- Establecer sesiones con los SIR de las estaciones de campo para el intercambio de mensajes. Normalmente el operador de la red estará ubicado en el centro de proceso, por lo que las sesiones deben poder establecerse a solicitud del operador, por imperativo de las aplicaciones que se ejecutan en el ordenador central o a petición de cualquier SIR.
- Gestionar el control del tráfico en la red, una vez establecida la sesión con todos los SIR activos.
- Controlar el correcto funcionamiento de la red de ordenadores en su conjunto.
- Proporcionar el interfase de usuario para operar con la red.

En cuanto a las características técnicas del equipo que va desarrollar este trabajo, dependerán en buena parte de las funciones de control adicionales que debe realizar. Así, si este ordenador debe procesar gran cantidad de información y ejecutar algoritmos complejos para coordinar un control distribuido, necesitaremos un sistema con elevadas prestaciones. No obstante, el trabajo realizado en el centro de proceso también puede repartirse entre varios sistemas.

Las excelentes prestaciones que hoy en día ofrecen muchas redes de área local, permiten proponer como alternativa a un potente ordenador la siguiente configuración:

Instalar en el centro de proceso una red de área local, donde uno de los nodos de esta red asumiera las funciones descritas anteriormente para el ordenador

central. Este sistema estaría dedicado en exclusiva a interactuar con la red de ordenadores externa, facilitando la información necesaria a otros nodos de la red local para que estos realicen las labores de cálculo, almacenamiento y control del proceso. De esta forma el ordenador central haría de puente entre las dos redes y bastaría con un ordenador personal (con procesador 486 ó Pentium) para realizar este trabajo.

La elección de una u otra alternativa no afecta al diseño de la arquitectura de la red de ordenadores externa, pero conviene tenerlas presentes a la hora de realizar una implementación práctica del sistema. En cualquier caso, las características de un proceso concreto y la complejidad de su control serán los que determinen que opción es la más adecuada.

Los sistemas informáticos remotos deben realizar un control local sobre la planta y atender a las comunicaciones con la red de ordenadores. Su programación debe contemplar todas aquellas funciones de control que sean necesarios y ser congruente con la arquitectura definida para la red de ordenadores, incorporando los protocolos especificados para la misma.

Este trabajo podría ser llevado a cabo por un ordenador convencional, pero su procesador debería tener suficiente capacidad para atender todas estas funciones de forma simultánea. Por otra parte, hemos de considerar que elementos auxiliares del ordenador, como la pantalla y el teclado, no son necesarios en las estaciones de campo. Interesa que la actuación del usuario sobre el sistema informático remoto, pueda efectuarse desde la consola del ordenador central.

Lo ideal es que ambas tareas, el control y el diálogo con la red, puedan realizarse de forma independiente, al menos con dos procesadores intercomunicados. En esta memoria se propone el diseño de un sistema que cumple con estos requisitos. Su descripción técnica, operatividad y posibilidad de adaptación a las características del proceso a controlar se realizarán en el capítulo 5 de la memoria.

Los sistemas de adquisición y actuadores necesarios para realizar el control de la planta pueden estar integrados en el propio sistema informático remoto o constituir entidades separadas. En cualquiera de los casos, los programas de control residentes en el SIR se adecuarán a estas circunstancias. Por ejemplo, la programación del SIR incorporará el algoritmo de adquisición de una señal si ejecuta directamente esta función. Si la adquisición se realiza por una estación de medida autónoma, la programación del SIR debe incorporar todas las funciones que puedan realizarse con el sistema de adquisición y el protocolo específico que permite el diálogo con el mismo.

Los sistemas informáticos integrantes de la red de ordenadores, se conectan a la subred de comunicaciones mediante una interfase estándar RS232C. Esta norma fue propuesta por la Asociación de Industrias Electrónicas (EIA) y su versión internacional se encuentra incluida en la recomendación V24 del CCITT. Este estándar es el más utilizado para conectar ordenadores (Equipos Terminales de Datos, ETD) a modems (Equipos Terminales de Circuitos de Datos, ETCD). Prácticamente todos los sistemas informáticos convencionales disponen de alguna salida con estas normas y aquellos que se deben confeccionar, como el SIR, se pueden adaptar con suma facilidad a la misma.

Por otra parte, los nodos de la red de comunicaciones seleccionada (véase capítulo 3) disponen de una entrada que sigue la norma RS232C. Su utilización por parte de los sistemas informáticos resulta, por tanto, obligada para poder acceder a la red de comunicaciones.

4.3 APLICACIONES EN RED

El objetivo fundamental de la red de ordenadores propuesta es servir de plataforma para realizar un control distribuido de procesos remotos. El procedimiento de control determinará que aplicaciones deben realizarse en red para que este objetivo sea factible.

En el capítulo 2 de la memoria se analizaba la forma de operar un control distribuido. Básicamente se organizaba en distintos niveles de control de forma que el control local es supervisado por un nivel de orden superior, que coordina las actuaciones de los controles directos. En la estructura propuesta para la red de ordenadores, el nivel de control superior estaría ubicado en el centro de proceso. Esta forma de proceder determina que la mayoría de mensajes presentes en la red se intercambien entre el ordenador central y cualquiera de los sistemas informáticos remotos existentes.

En ocasiones puede ser necesario que un control local necesite de la información presente en otro nodo remoto para poder operar. En estas situaciones es aconsejable que el proceso de intercambio de información sea vigilado por el nivel de control superior, por lo que los mensajes se tramitarán a través del sistema que realiza estas funciones. Así un fichero de datos requerido por un nodo remoto será siempre solicitado al ordenador central, quien se encargará de obtenerlo a través de la red en el punto donde se encuentre y remitirlo al nodo que los solicitó.

Aunque este mecanismo es más lento que si el fichero se remite directamente desde el punto donde se encuentra, evita que parte del control escape al conocimiento

del nivel que debe supervisarlo. Por otra parte la información que, generalmente, puede requerir un nodo remoto de otro será poco voluminosa (situación de alguna de sus variables de estado) y la lentitud no puede considerarse significativa.

La comunicación directa entre nodos remotos quedará restringida a casos muy particulares y se realizará utilizando una prestación del protocolo AX25. Este permite enviar mensajes de multidifusión que son recibidos por todos los ordenadores presentes en la red y que no necesitan del establecimiento de una conexión previa. Esta utilidad puede ser interesante, por ejemplo, para que una estación remota origine un disparo y todos los nodos de la red confeccionen un fichero con los registros que están efectuando, aunque el algoritmo que regula su sistema de adquisición entienda que se trata de eventos no significativos. En cualquier caso este tipo de comunicaciones serán siempre de carácter especial y quedarían fuera de lo que se consideran aplicaciones en red.

La transferencia de archivos es una de las aplicaciones más comunes en las redes de ordenadores. Los archivos que pueden encontrarse distribuidos entre los sistemas informáticos de esta red de ordenadores pueden ser muy diversos: ficheros de datos confeccionados por las estaciones remotas, ficheros de configuración para sistemas de medida autónomo presentes en la red, programas de trabajo para las estaciones remotas que pueden ser enviados desde el centro de proceso, etc.

El volumen de estos ficheros puede ser muy variable dependiendo de las características del proceso que se está controlando. Es de prever que podamos encontrar ficheros con un considerable número de datos, que deben ser cursados por la red para hacer posible el control requerido. Necesitaremos definir una aplicación que proporcione un servicio de *transferencia de archivos* en la red, para que las aplicaciones de control puedan realizarse de forma satisfactoria.

Otro aspecto que debemos considerar es la necesidad de que se efectúen unas determinadas operaciones en cualquiera de los sistemas de la red, a solicitud de otro

sistema, porque este último necesita el resultado de la operación para trabajar adecuadamente.

Estas actuaciones pueden deberse a:

- Necesidades de mantenimiento de la propia red de ordenadores. Por ejemplo, un S.I.R. puede solicitar que se ejecute una función en la estación central para que le informe del número de estaciones activas, porque necesita este dato como entrada al algoritmo que determina su tiempo de acceso al medio.
- Necesidades de las aplicaciones de control. Por ejemplo, la estación central puede solicitar a un determinado SIR que se ejecute un algoritmo de control local concreto, en función del conocimiento global que el sistema supervisor tiene del proceso en su conjunto.
- Necesidades del operador de la red. Por ejemplo, el operador puede solicitar, desde la consola del ordenador central, que un SIR efectúe una reinialización de un dispositivo adicional de la estación de campo (un sistema de adquisición autónomo, podría ser el caso), y le transfiera el control del mismo para cambiarle algún parámetro.

La *entrada remota de trabajos* es, por tanto, una aplicación imprescindible en la red de ordenadores.

De esta forma, las aplicaciones de usuario (programas de control) utilizarán los servicios proporcionados por las aplicaciones de la red (*transferencia de ficheros* y *entrada remota de trabajos*) para cubrir el objetivo de control propuesto.

Estas mismas aplicaciones de red permitirán, además, lograr el segundo de los objetivos planteado en el inicio; realizar una supervisión a distancia de los distintos elementos que conforman el sistema remoto en su conjunto.

4.4 ARQUITECTURA

4.4.1 DEFINICIÓN DE NIVELES

El diseño de una red de ordenadores, con una arquitectura que permita a diversos sistemas comunicarse de una forma eficiente, resulta un problema complejo. El diseño debe considerar todos los eventos que puedan acontecer durante una comunicación, los efectos y causas de esos eventos, los aspectos técnico-operacionales de los sistemas que intervienen en la red, etc. Todo ello tienen que ser cuidadosamente especificado, para que los diferentes procesos de aplicación puedan cooperar entre sí para realizar las tareas distribuidas en la red [Beltrào_89].

La clave para reducir la complejidad del diseño de una red de ordenadores es proyectarla como un conjunto jerárquico de niveles (o capas), cada uno superpuesto sobre el anterior. Reduciendo el diseño global de la red al diseño de cada uno de sus niveles, se simplifica considerablemente el trabajo de desarrollo y mantenimiento [Tanenbaum_91].

El diseño de un nivel se restringe al contexto de ese nivel y supone que los problemas externos ya están debidamente resueltos. En una arquitectura jerárquica el nivel *N* solo sabe que existe un nivel *N-1,* prestador de determinados servicios y un nivel *N+1* que le va a solicitar los servicios que él realiza. Esto conlleva una independencia entre niveles, ocupándose cada nivel de realizar sus servicios independientemente del protocolo que utilice para ello. Así un nivel puede ser alterado en sus aspectos físicos o lógicos sin afectar al comportamiento de la red, siempre que no se modifiquen los servicios que presta.

Antes de que se abordara el problema de compatibilidad entre sistemas heterogéneos que se conectaban a la misma red, algunos grandes fabricantes ya tenían definidas soluciones para interconectar sus propios equipos. Estas soluciones particulares se basaban en estructuras de niveles jerarquizados como la descrita y se acuñaron con el término *arquitectura de red*, sinónimo de "*conjunto de convenios para interconectar sus sistemas*"; arquitecturas: SNA[6] de IBM y DNA[7] de Digital por citar las más significativas.

La incompatibilidad entre equipos de distintos fabricantes, fue resuelta inicialmente por medio de convertidores. Un convertidor interpreta la información originaria de uno de los sistemas y la traduce de forma inteligible para el otro sistema, antes de transferirla. Los convertidores son lentos e inadecuados para solucionar incompatibilidades a nivel de aplicaciones, ya que es necesaria su integración en el propio sistema operativo de los equipos involucrados [Beltrào_89].

En 1977 la Organización Internacional de Normalizaciones (ISO) vio la necesidad de normalizar la interconexión de sistemas heterogéneos y creó un subcomité (SC16) para estudiar el problema. A mediados de 1979 el desarrollo del modelo de arquitectura estaba terminado (*Reference Model for Open Systems Interconnection*, RM-OSI). Finalmente el RM-OSI fue oficialmente aprobado por ISO, en 1983, como una norma internacional para la interconexión de sistemas abiertos, a través del documento ISO 7498 [ISO_83].

En la figura 4.2 se presenta un modelo basado en la propuesta desarrollada por ISO. En él se muestra una estructura con 7 niveles jerarquizados. En este modelo el nivel *N* de una máquina conversa con el nivel *N* de otra utilizando un conjunto de normas "*protocolo de nivel N*". En realidad no existe una transferencia directa de datos entre los niveles *N* de cada sistema. Cada nivel pasa la información de datos y control a su nivel inferior hasta que se alcanza el nivel físico, correspondiente al nivel

[6]*SNA:* System Network Architecture
[7]*DNA:* Digital Network Architecture

más bajo de la estructura. A través del medio físico se realiza la comunicación real (indicado en la figura con líneas sólidas), mientras que la comunicación entre el resto de niveles es virtual (líneas punteadas en la figura).

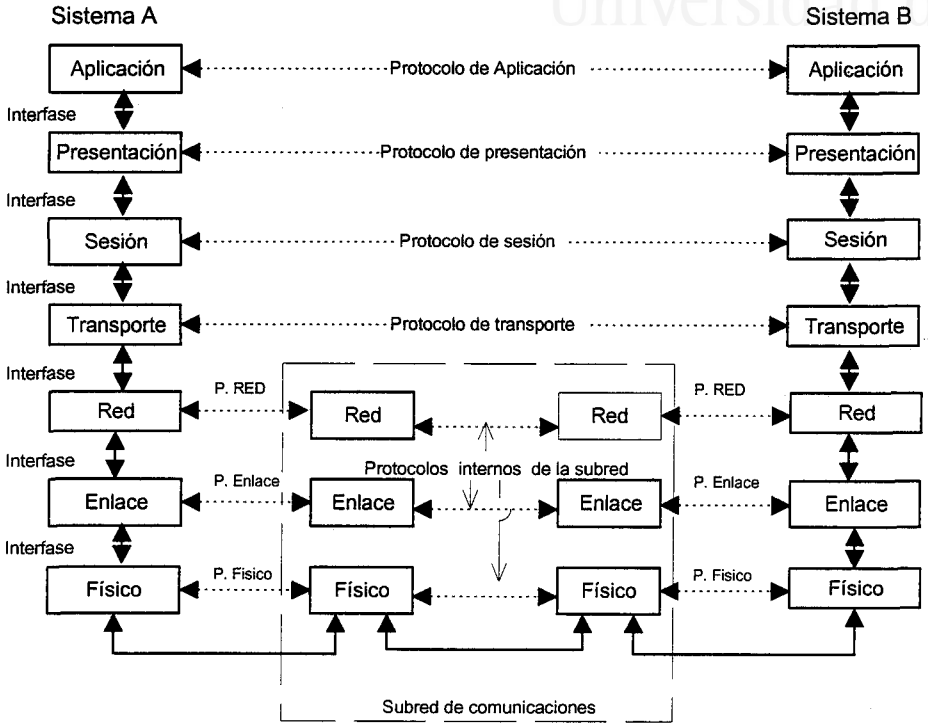

Figura 4.2 Arquitectura de red basada en el modelo OSI

Entre cada par de niveles adyacentes existe una *interfase* que permite a cada nivel acceder a los servicios que le ofrece su nivel inferior.

Al conjunto de niveles y protocolos se le denomina *arquitectura de red.* Las especificaciones de ésta deben contener información suficiente para escribir un programa o construir el *hardware* correspondiente a cada nivel, y que siga en forma correcta el protocolo apropiado. Los detalles de realización y las especificaciones de las interfase no forman parte de la arquitectura. Es más, no es necesario que las interfases de todos los sistemas conectados a la red sean iguales siempre que cada sistema utilice correctamente todos los protocolos [Tanenbaum_91].

Básicamente las funciones asignadas a los 7 niveles del modelo OSI son las siguientes:

Nivel Físico: Define las características mecánicas, eléctricas, funcionales y procedimentales para activar, mantener y desactivar conexiones físicas para la transmisión de *bits* entre entidades de enlace de datos.

N. de Enlace: Detecta, y posiblemente corrige, los errores de nivel físico proporcionando una línea libre de errores de transmisión al nivel de red. Para ello trocea los datos de entrada agrupándolos en *tramas*, los transmite secuencialmente y procesa las tramas reconocidas.

Nivel de Red: Se ocupa del control de la operación de la subred de comunicaciones. Su propósito es proporcionar una comunicación extremo a extremo a las entidades de transporte, independizándolas de todo lo relativo a como estén interconectados los sistemas de la subred. El nivel de red realiza las funciones de encaminamiento necesarios y proporciona una ruta de conexión a través de nodos intermedios, que se encargan de retransmitir la información. En redes de difusión el problema del encaminamiento es simple, por lo que este nivel es muy delgado o incluso inexistente.

N. Transporte: Proporciona una transferencia de datos transparente, extremo a extremo, entre las entidades de sesión. Es una especie de puente entre la calidad del servicio que oferta el nivel de red y el servicio requerido por el nivel de sesión. Si la conexión ofrecida por el nivel de red es fiable y económica, las funciones del nivel de transporte necesarias quedan sensiblemente reducidas. Este nivel es también responsable de fragmentar la información recibida del nivel de sesión en unidades más pequeñas, para adecuarlas al formato aceptado por la subred de comunicaciones, siempre que esto sea necesario.

Nivel de sesión: Permite que los distintos sistemas conectados a la red, puedan establecer sesiones entre ellos para realizar un transporte de datos ordinario. Una de las funciones más importantes de este nivel es gestionar el control del diálogo. Así las sesiones permiten que el tráfico vaya en ambas direcciones al mismo tiempo o sólo en una, vigilando en este último caso quien tiene el turno.

N.Presentación: Se ocupa de los aspectos sintácticos (representación de la información intercambiada). Ejemplos típicos de los servicios presentados por este nivel son la codificación de la información, la comprensión de datos y la encriptación de estos por razones de privacidad y autentificación.

N. Aplicación: Proporciona un medio para que los procesos de usuario (programas de aplicación de usuario) pueda acceder al entorno OSI

Una descripción más detallada de las funciones de cada uno de los niveles del modelo OSI, así como la forma concreta de llevar a cabo el envío de datos entre los sistemas conectados a este entorno, se encuentran debidamente documentados en una amplia bibliografía, [Alabau_86], [Alonso_96], [Antáo_89], [Halsall_92], [Stallings_91], [Schwartz_94], [Tanenbaum_91], etc.

El modelo RM-OSI, por sí mismo, no es una arquitectura de red ya que no especifica, en forma exacta, los servicios y protocolos que se deben utilizar en cada uno de los niveles [Tanenbaum_91]. Es simplemente un modelo de referencia que permite confeccionar una arquitectura concreta y que se utilizará como base para diseñar la arquitectura de la red de ordenadores propuesta. No obstante la ISO también ha generado normas para todos los niveles aunque estas, estrictamente hablando, no forman parte del modelo. Cada una de ellas se ha publicado como normas internacionales independientes.

La propia ISO afirma que es difícil justificar que son 7 el número de niveles que forman la "mejor" arquitectura posible para interconectar sistemas abiertos. De

hecho, un mismo nivel puede variar mucho de tamaño de una a otra red o incluso no ser necesaria la existencia del mismo. La presión ejercida por las grandes compañías quizás precipitó la aparición de las normas y estas no sean las más idóneas, como sostienen algunos especialistas.

En cualquier caso, resulta de gran interés la filosofía de diseño utilizada en el modelo de ISO. Los principios arquitectónicos manejados por ISO para definir su modelo de 7 niveles serán la referencia empleada en la memoria para proponer una arquitectura de la red de ordenadores, que permita estructurar adecuadamente los servicios que de ella se demandan.

Estos principios arquitectónicos se encuentran en el documento [ISO_81] y, básicamente, se pueden resumir como sigue [García_90]:

- No crear un número muy grande de niveles a fin de no dificultar el trabajo de descripción e integración de esos niveles. Pero por otra parte deben existir niveles suficientes como para que no tengan que realizarse funciones distintas, dentro del mismo nivel, sin necesidad.
- Debe crearse un nivel siempre que se requiera un nivel de abstracción diferente en el manejo de los datos.
- Cada nivel debe realizar unas funciones bien definidas, agrupándose las funciones similares en un mismo nivel. Funciones diferentes en el proceso realizado o en la tecnología que envuelven deben estar en niveles distintos.
- Debe fijarse un límite en el punto en que las descripciones de servicio puedan ser suficiente pequeñas, para que se minimice el número de interacciones entre dos niveles a través de ese límite.
- Los diferentes niveles se deben elegir de manera que permitan la definición de normas de protocolos e interfases, así como ser suficientemente flexibles para que puedan ser rediseñados y sus protocolos cambiados para aprovechar las ventajas de los nuevos avances en tecnología de la

arquitectura, *hardware* y *software*, sin tener que cambiar los servicios esperados de ellos.

- Seleccionar límites en los puntos en que la experiencia pasada haya demostrado que es interesante.
- Crear para cada nivel interfases con el nivel superior e inferior únicamente.
- Formar subniveles dentro de los niveles cuando se requieran servicios de comunicación distintos.

El modelo propuesto por ISO en base a estos principios tiene un condicionante que aún no hemos mencionado. Aunque el modelo no presupone ningún tipo de red en particular, éste fue concebido para grandes redes informáticas que utilizan nodos en la subred de comunicaciones con capacidad de encaminamiento, almacenamiento de la información y reenvío de la misma en el momento oportuno.

Pero, por ejemplo, en las redes de área local estas técnicas son poco utilizadas. Las características específicas que reúnen este tipo de redes permite eliminar funciones de alguno de los niveles del modelo OSI e incluso niveles enteros.

Las topologías de la subred de comunicaciones utilizadas en las redes de área local (bus o anillo, las más comunes) condicionan que las entidades de nivel de enlace comuniquen de forma directa (extremo a extremo). Así en un medio de difusión, como el bus, la transmisión es captada por todos los sistemas conectados a la red. En un anillo la transmisión de una estación recorre todo el anillo hasta que llega a la estación emisora, siendo de esta forma captada, eventualmente, por todas las interfases de la red. Estas topologías eliminan las necesidades de enrutamiento presentes en el nivel de red del modelo OSI.

Por otra parte, el protocolo que debe regular los accesos al medio de comunicación (dispuesto en bus o en anillo) tiene que realizar el control de la

congestión en la subred de comunicación. Esta es otra función del nivel de red del modelo OSI.

En vista de esto el comité 802 del IEEE (*Institute of Electrical and Electronic Engineers*), responsable de emitir normas para las redes de área local buscando la compatibilidad con el modelo OSI, limitó la propuesta de normas a los niveles 1 y 2 (físico y enlace) del RM-OSI, dejando vacío el nivel de red.

En la figura 4.3 se presenta la correspondencia entre el modelo de referencia OSI y el modelo IEEE 802 para redes locales.

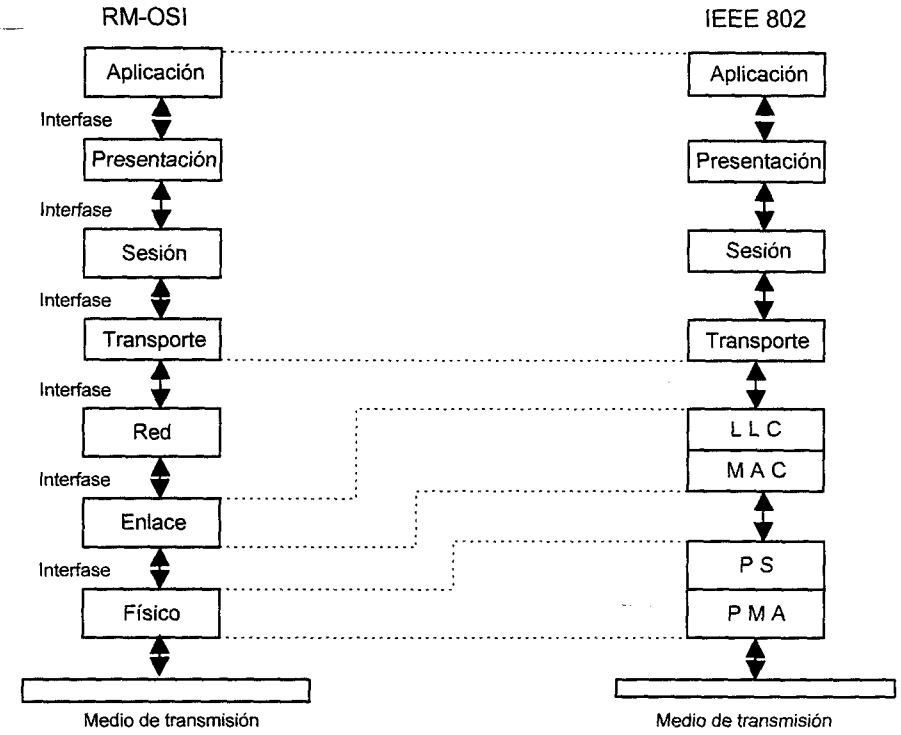

Figura 4.3 Correspondencia entre RM-OSI y el modelo IEEE 802

Para las redes locales ha sido necesario readaptar el modelo OSI en dos aspectos fundamentales [Alabau_86]:

1) En las redes locales los sistemas se comunican extremo a extremo a nivel de enlace (nivel 2). El protocolo de nivel más bajo en el modelo OSI con comunicación extremo a extremo es el protocolo de transporte (nivel 4).

2) El nivel de enlace en las redes locales se subdivide en dos subniveles:

- *Control de enlace lógico*. LLC (*Logical Link Control*)
- *Control de acceso al medio*. MAC (*Medium Access Control*)

Los objetivos que subyacen en esta decisión son conseguir que el primer nivel extremo a extremo (LLC) sea independiente de la topología de la red local, del medio físico y del método de acceso el mismo. Así, los posibles cambios en la tecnología del medio no implican modificaciones en el protocolo de control del enlace.

De la misma forma, en las redes de área local se subdivide el nivel físico. El primer subnivel *señalización física* (*Physical Signalling*, PS) es un bloque de proceso que se encarga de la codificación y decodificación de la información y de las funciones de control del medio. El segundo subnivel *conexión a los medios físicos* (*Physical Media Attachment*, PMA), se encarga de imprimir y leer la información del medio, pudiendo estar capacitado para realizar otras funciones como detectar el estado del medio (ocupado o en silencio) y percibir colisiones en el mismo.

Los niveles superiores del modelo OSI (4 a 7) son independientes de las características de la subred de comunicaciones y permanecen inalterados al adaptarlos a las redes locales. Lógicamente para seleccionar la *clase*[8] específica de protocolo de transporte ISO utilizado en este nivel, debe tenerse en cuenta las características de la subred de comunicaciones, los requisitos de la aplicación y las características del usuario [Beltrào_89].

La subred de comunicaciones descrita en el capítulo 3 de la memoria podría considerarse un modelo híbrido entre una red de área extendida y una red local. El primer protocolo extremo a extremo es un protocolo de enlace y la comunicación se

---

[8] *Clase de protocolo*: ISO ha definido cinco clases en su especificación de protocolo de transporte, en función de los servicios demandados a este nivel y los servicios proporcionados por su nivel inmediato inferior (red o LLC)

realiza a través de un medio de difusión, como ocurre en las redes locales. Sin embargo, el control del acceso a ese medio de difusión no lo tienen los sistemas informáticos conectados en la red de ordenadores, si no que es una función realizada por los nodos de la subred de comunicaciones. El enlace entre puntos terminales puede, además, pasar por varios nodos intermedios de la subred, que es la responsable de establecer el camino lógico entre los puntos extremos.

Esto obliga a replantearse la confección de los niveles inferiores de la arquitectura, para hacer compatible la utilización de esta subred de comunicaciones con el entorno OSI. Teniendo en cuenta que el primer nivel extremo a extremo es un nivel de enlace, se propone un modelo, figura 4.4, similar al utilizado en las redes de área local, pero con ciertas matizaciones derivadas de la utilización de una subred de comunicaciones no integrada en los sistemas informáticos que se interconectan.

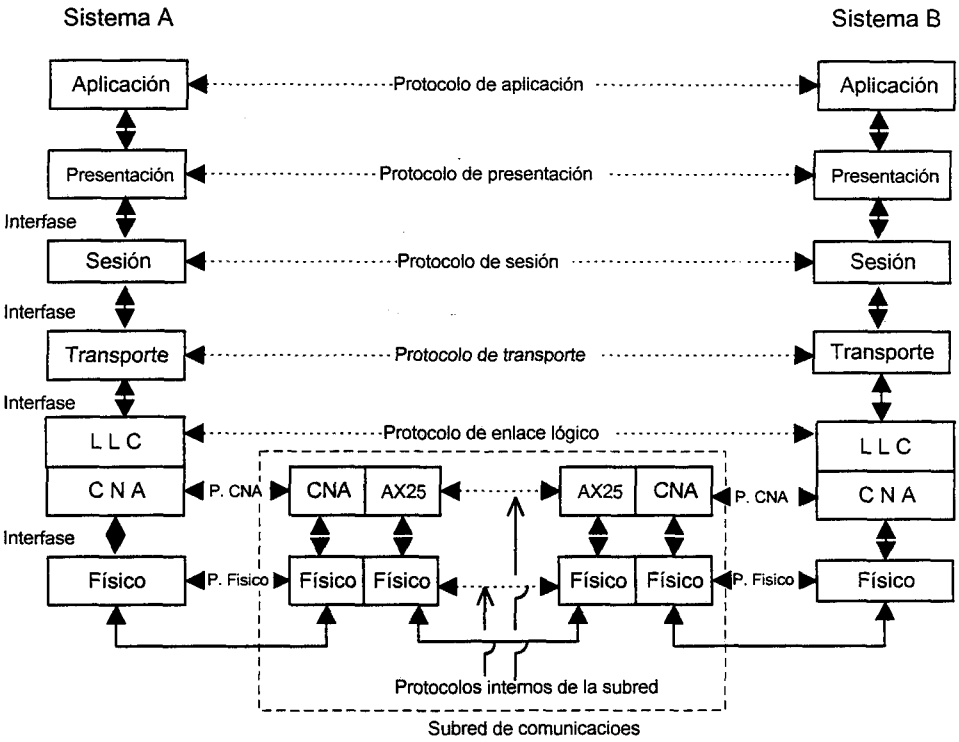

Figura 4.4 Adaptación de la subred al entorno OSI

De la misma forma que en las redes de área local el nivel de enlace se divide en dos subniveles: *Control de enlace lógico* LLC (*Logical Link Control*) y *Acceso a*

la red de comunicaciones NCA (*Network Communications Access*). El primero de ellos se encargará de la gestión del diálogo a nivel de enlace, entre puntos extremos, mientras que el segundo adecuará los datos proporcionados por el LLC a un formato de trama entendible por el nodo de la red de comunicaciones.

Aunque el nombre del LLC coincide con el asignado a las redes locales y las funciones básicas que realiza de cara a los niveles superiores son las mismas, varia la forma de realizar estas funciones por las diferentes características de la subred de comunicaciones. Así, por ejemplo, el direccionamiento al sistema destino es una función adjudicada al LLC en las redes locales. Para realizar esta función, el LLC incluye un campo de dirección con la identificación del origen y destino de la información.

Para este caso también debe tenerse en cuenta que el enlace entre dos sistemas conectados a la red de comunicaciones puede llevarse a cabo a través de más de dos nodos. Así mismo, la subred requiere que exista una conexión de enlace previa al envío de datos. Esto obliga a elegir una *clase* de LLC que soporte operaciones orientadas a conexión y que en la solicitud de conexión se incluya el origen y destino de la información, y además, se especifique en el campo de dirección los identificativos de los nodos intermedios de la subred, si existen.

Una vez establecida una conexión a nivel de enlace el LLC debe asignar un canal de comunicación a este, de forma que todos los datos son enviados por dicho canal a su destino sin necesidad de especificar, en las tramas de datos, el campo de dirección. Se trata de una forma diferente de direccionar la información y esto debe ser tenido en cuenta a la hora de confeccionar el protocolo del LLC.

Otro aspecto que debe considerarse es el hecho de que la subred de comunicaciones entrega una información libre de errores físicos. El protocolo de enlace interno de la subred (AX25) dispone de mecanismos de detección de errores y estrategias de retransmisión para recuperación de los mismos. Esto hace innecesario que esta función sea realizada nuevamente por el LLC. Aunque es muy improbable

que se produzca un error físico entre el nodo de comunicaciones y el ordenador conectado al mismo, siempre puede utilizarse como medida de precaución una *clase* de protocolo de transporte, por encima del LLC, que proporcione un servicio de detección y corrección de errores para hacer más fiable la comunicación entre los entes de transporte.

Dependiendo de las necesidades del nivel superior se puede optar por un LLC que proporcione un servicio de datos orientado a conexión, es decir, las tramas de datos serán confirmadas a nivel de enlace lógico, estableciéndose estrategias de retransmisión en este nivel. O bien puede seleccionarse, como también se hace en las redes de área local, un servicio de datos para el LLC no orientado a conexión. En este último caso entre los sistemas conectados a la red se intercambian tramas de datos no numeradas, no existiendo reconocimiento de las mismas, ni control de flujo, ni estrategias para recuperar la información en este nivel. Esta opción es la más recomendable ya que el protocolo es mucho más sencillo y además es suficiente porque la subred de comunicaciones proporciona datos fiables.

El segundo subnivel, dentro del nivel de enlace lógico, no puede tener las mismas funciones que el MAC de las redes locales. Aquí el acceso al medio es gestionado por los nodos de la subred. La principal misión del subnivel MAC queda aquí delegada y el subnivel que queda por debajo del LLC únicamente debe ocuparse de adaptar la información suministrada por su nivel superior a un formato entendible por el nodo de la subred. Este subnivel, denominado *acceso a la red de comunicaciones*, se encargará de poner los delimitadores a las tramas e incluir un campo específico (número de puerto) requeridos por los nodos de la subred para operar adecuadamente con las tramas recibidas. También será misión de este subnivel incluir la *transparencia* a la información, de forma que cualquier combinación de *bits* pueda enviarse como dato dentro de una trama de enlace.

El nivel físico no es necesario subdividirlo pues, como se ha mencionado, ahora no existe un control directo sobre medio de comunicación. Sus funciones son iguales a las descritas por el modelo ISO para este nivel, utilizándose un protocolo

estándar para regular el intercambio de *bits* entre los sistemas informáticos y la subred de comunicaciones.

Con el modelo propuesto podríamos emplear la subred de comunicaciones descrita en esta memoria para interconectar sistemas abiertos, conforme a las especificaciones ISO, utilizando protocolos normalizados a partir del nivel de transporte. Esta sería una buena opción si, en cada nodo de la red de ordenadores que debe realizar un control local, dispusiéramos de un ordenador con suficiente capacidad de proceso como para soportar los protocolos normalizados de los niveles superiores, los protocolos que deben diseñarse para los niveles (1 y 2) y los programas de usuario que se utilizan en el control local.

Sin embargo, esta no es la situación más habitual que podemos encontrar en los procesos que interesa controlar. Por las características de ubicación de las estaciones de campo se necesita que el sistema informático que la controla sea lo más sencillo posible, enfocado fundamentalmente al control local y que el número de procesos que ejecute para atender a la red de ordenadores se minimice. En consecuencia, debe simplificarse al máximo la arquitectura que posibilita la interconexión entre los sistemas conectados a la red de ordenadores.

Otros hechos que contribuyen a utilizar una arquitectura más sencilla son el número y la dificultad de las aplicaciones que deben ejecutarse en red. Esta red no tiene que soportar aplicaciones de correo electrónico, ni proporcionar servicios de valor añadido como teletex o videotex. Los ficheros van a intercambiarse entre los sistemas, pero un fichero no podrá ser abierto por un sistema externo ni existirá un acceso múltiple a un mismo fichero, es decir, no será necesaria una manipulación de los ficheros desde sistemas distribuidos. Todo esto condiciona que los servicios necesarios para que las aplicaciones en red puedan llevarse a cabo, y que deben proporcionar los niveles inferiores al de aplicación, sean más reducidos.

La misión fundamental de la red de ordenadores propuesta es servir de soporte a sistemas de control distribuido, por tanto, las normas de interconexión entre

los ordenadores utilizados en el control deben articularse pensando en lograr la máxima productividad en el proceso de control. El modelo de referencia ISO es adecuado para interconectar sistemas que realizan funciones muy distintas a los ordenadores de esta red. Su utilización, además de necesitar sistemas informáticos más complejos, requiere una mayor inversión en tiempos de comunicación pudiendo esto afectar a la eficiencia del control.

Atendiendo a estas consideraciones, se propone la estructura de niveles presentados en la figura 4.5 como modelo de interconexión en una red de ordenadores dedicada al control distribuido de procesos remotos.

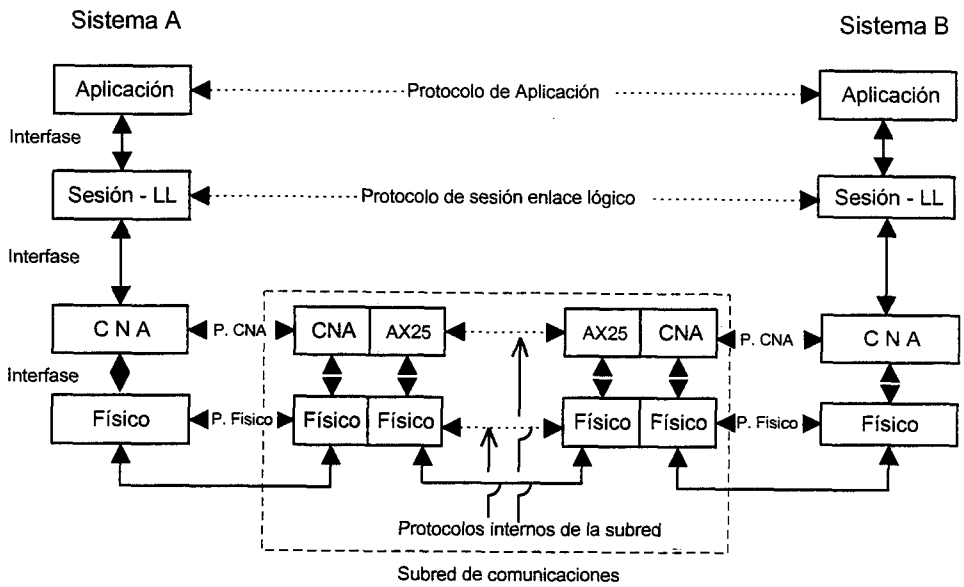

Figura 4.5 estructura de niveles de la red de ordenadores

Entendemos que esta estructura de niveles es la más sencilla que se puede plantear y que es suficiente para cubrir los objetivos propuestos. Para su formulación, además de las características específicas de la red de ordenadores, se han tenido en cuenta los principios arquitectónicos que ISO utilizó para formular su modelo OSI. En esta memoria utilizaremos también la terminología OSI para describir la interacción entre los niveles.

Antes de enunciar las funciones que debe realizar cada uno de estos niveles, determinando así el servicio que presta a su nivel superior, se analizará que se entiende por un *servicio* en el contexto OSI, los *tipos* de servicio que existen y el procedimiento para que un determinado nivel demande este servicio a su nivel inferior.

4.4.2 SERVICIOS. PRIMITIVAS DE SERVICIO

La verdadera función de cada uno de los niveles consiste en proporcionar servicios a los niveles superiores. Así las entidades del nivel *N* desarrollan un servicio que utiliza el nivel *N + 1*. En este caso al nivel *N* se le denomina *proveedor* del servicio y al nivel *N + 1 usuario* del servicio. El nivel *N* puede, a su vez, utilizar los servicios del nivel *N -1* con objeto de proporcionar su servicio.

Los servicios se encuentran disponibles en el *punto de acceso al servicio* SAP (*Service Access Point)* . Los SAP del nivel *N* son los lugares en donde el nivel *N + 1* puede acceder a los servicios que se ofrecen. Cada uno de los SAP tiene una dirección que lo identifica de forma particular. Por ejemplo, los SAP en el sistema telefónico son los enchufes en los que se conectan los teléfonos y las direcciones de los SAP son los números telefónicos correspondientes a dichos enchufes. Para poder llamar a alguien se debe conocer la dirección de su SAP. En el sistema postal, de manera análoga, las direcciones de los SAP son las direcciones de las calles y de las oficinas postales. Para poder enviar una carta se debe conocer la dirección del SAP destinatario [Tanenbaum_91].

Para que se lleve a cabo el intercambio de información entre dos niveles deberá existir un acuerdo sobre un conjunto de reglas acerca de la interfase. En una interfase típica la entidad del nivel *N + 1* pasa una *unidad de datos de la interfase* IDU (*Interfase Data Unit)* a la entidad del nivel *N*, a través del SAP, como se muestra en la figura 4.6. La IDU está formada por una *unidad de datos del servicio* SDU (*Service*

Data Unit) y de alguna *información para el control de la interfase* ICI (*Interfase Control Information*) . La información para el control del interfase desaparece una vez cumplido su objetivo.

La SDU recibida en el nivel *N* puede ser fragmentada, si ello es necesario, de forma que cada una de las partes se completa con la *cabecera* propia de este nivel, conformando una *unidad de datos de protocolo* PDU (*Protocol Data Unit*). En la figura 4.6 se ha supuesto que la SDU no se fragmenta por simplicidad. La PDU se intercambia con su entidad *par* en otro sistema para que el protocolo de este nivel pueda realizarse. Una PDU puede conformarse con datos del nivel superior (SDU recibida) y la cabecera del nivel *N*, o bien, puede generarse en el propio nivel conteniendo entonces información de control para el protocolo de nivel *N*.

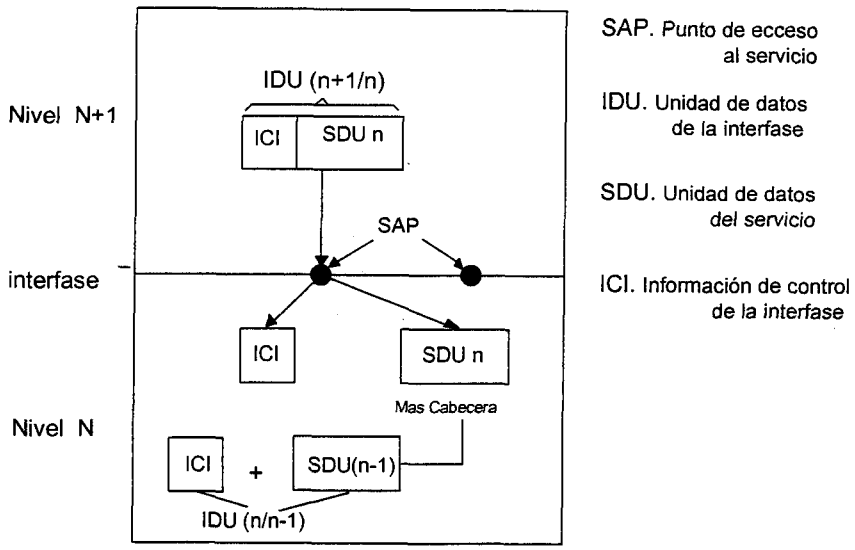

Figura 4.6 Relación entre niveles en una interfase

Con cada PDU el nivel *N* se conforma una nueva SDU para el nivel *N-1* que transmitirá por su interfase correspondiente y por el mismo procedimiento anterior. Debe destacarse que la información de control ICI es necesaria porque ayuda a que los niveles inferiores realicen su trabajo, pero no forma parte de los datos.

Los distintos niveles que conforman la arquitectura pueden ofrecer dos tipos diferentes de servicios: orientado a conexión y sin conexión. El servicio orientado a conexión está modelado basándose en el sistema telefónico. Para poder hablar con alguien se debe tomar el teléfono, marcar el número, hablar y colgar. Similarmente, para utilizar un nivel que proporciona un servicio orientado a conexión, el usuario del servicio establece primero una conexión, la utiliza y después termina la conexión. El aspecto fundamental de la conexión es que actúa en forma parecida a la de un tubo; el que envía introduce objetos por un extremo y el receptor los recoge, en el mismo orden, por el otro extremo [Tanenbaum_91].

A diferencia de esto, el servicio sin conexión se modela basándose en el sistema postal. Cada mensaje (carta) lleva consigo la dirección completa de destino y cada uno de ellos se encamina, en forma independiente, a través del sistema que interconecta el origen y el destino del mensaje. Normalmente, cuando dos mensajes se envían al mismo destino el primero que se envíe será el primero en llegar. Es posible, sin embargo, que el primero que se envíe sufra una retardo y llegue antes el que se envió en segundo lugar. Con un servicio orientado a conexión es imposible que suceda esto [Tanenbaum_91].

Cada servicio se caracteriza por la calidad del mismo; algunos de ellos se consideran *fiables* en la medida que nunca pierden la información que transportan. Por lo general, un servicio fiable se realiza haciendo que el receptor notifique la recepción de cada mensaje para que el transmisor esté seguro de que su mensaje llegó al destino. El proceso de notificación introduce un exceso de tráfico y retardos. Una aplicación definida para esta red, la *transferencia de archivos,* es una situación típica en la que es deseable y apropiado tener un servicio orientado a conexión fiable. Para las aplicaciones de control es imprescindible que todos los *bits* lleguen correctamente y en el mismo orden en que se enviaron.

No todas las aplicaciones necesitan conexiones; por ejemplo *el correo electrónico*, donde los mensajes no requieren una contestación inmediata, puede prescindir de servicios en los que se necesita hacer una conexión previa con el sistema

destino. Tampoco es esencial tener un envío 100% fiable, especialmente si eleva su costo. Todo lo que se necesita, a fin de cuentas, es un medio de envío de mensajes sencillo que tenga una alta probabilidad, pero no una garantía de alcanzar su destino. Un servicio sin conexión que no es fiable (es decir, que no tenga asentimientos) se conoce con frecuencia como servicio datagrama, por analogía con el servicio de telegramas, el cual tampoco proporciona acuse de recibo de la información al emisor [Tanenbaum_91].

En algunas situaciones convendría no tener que establecer una conexión para enviar un mensaje pequeño, pero sí sería fundamental que el proceso sea fiable. Para estas aplicaciones se puede establecer un servicio de datagramas con asentimientos de información. Este servicio es parecido al proceso de enviar una carta certificada y solicitar un acuse de recibo. Cuando éste regresa, el emisor está completamente seguro de que la carta se entregó a la persona interesada.

Un servicio, proporcionado por un nivel, está formalmente especificado por un conjunto de *primitivas* (operaciones). Estas primitivas le indican al servicio que debe efectuar una acción o notifican la acción tomada por una entidad par. Como se muestra en la figura 4.7, las primitivas de servicio en el modelo OSI pueden dividirse en cuatro clases .

PRIMITIVA	ACCIÓN
SOLICITUD	Una entidad desea que el servicio realice un trabajo
INDICACION	Una entidad es informada acerca de un evento
RESPUESTA	Una entidad responde a un evento
CONFIRMACION	Una entidad es informada acerca de su solicitud

Figura 4.7 Clases de primitivas de servicio

La primera clase de primitiva es la primitiva petición o solicitud (*request*). Se utiliza para que un trabajo se realice, por ejemplo, establecer una conexión o enviar

datos. Una vez que se ha efectuado el trabajo se le avisa a la entidad par, ubicada en el sistema destino, mediante la primitiva indicación (*indication*). Por ejemplo, después de una petición de conexión CONNECT.*request*, la entidad que se está direccionando obtiene una indicación de conexión CONNECT.*indication*, con la cual se le anuncia que alguien desea establecer una conexión con ella. La entidad que recibió la CONNECT.*indication* utiliza entonces la primitiva respuesta de conexión CONNECT.*response* para decir si acepta o rechaza la conexión propuesta. En cualquier caso, la entidad que emite la CONNECT.*request* inicial sabe lo que pasó a través de la primitiva confirmación de la conexión CONNECT.*confirm* [Tanenbaum_91].

Las primitivas normalmente van acompañadas de parámetros. Si por ejemplo se solicita una conexión, los parámetros pueden especificar la máquina a la que se va a conectar, el tipo de servicio que se desea, el tamaño máximo del mensaje utilizado en la conexión, etc.

Los distintos servicios pueden ser confirmados o sin confirmar. En un servicio confirmado, hay una *petición*, una *indicación*, una *respuesta* y una *confirmación*. En un servicio sin confirmar, solamente hay una *petición* y una *indicación*. La expresión "CONNECT" siempre debe ser un servicio confirmado porque el corresponsal remoto deberá estar de acuerdo en establecer una conexión. Por otra parte, la transferencia de datos puede ser confirmada o sin confirmar, dependiendo de si el emisor necesita o no tener un acuse de recibo de la información. En las redes se utilizan los dos tipos de servicio [Tanenbaum_91].

En la figura 4.8 se muestra un ejemplo de transferencia de información entre dos niveles $N+1$ situados en ordenadores diferentes. En cada paso se toma en cuenta una interacción entre dos capas en cada uno de los ordenadores. Cada petición o respuesta genera una indicación o confirmación en el otro extremo un instante después. Los usuarios del servicio están en el nivel $N + 1$ y el proveedor del servicio está en el nivel N. Debe observarse que no es necesario que el servicio DATA sea confirmado, el nivel N puede encargase solamente de transportar la información hasta el otro extremo sin confirmar que esta llega. La interpretación de los datos que se intercambian

corresponde al nivel *N+1* y los asentimientos a estos datos, pueden realizarse por este nivel emitiendo nuevos datos con el mismo formato desde el punto de vista del nivel *N* [Rosa_92].

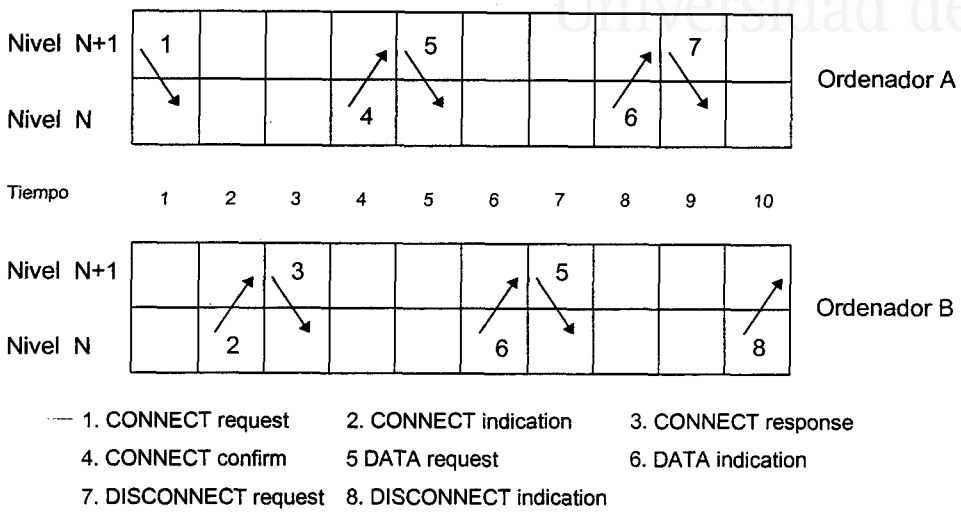

Figura 4.8 Esquema de interacción entre niveles

En la figura 4.8 se puede ver claramente los pasos que sigue la comunicación. El nivel *N+1* de "A" solicita el servicio (CONNECT) a su nivel *N* por medio de la primitiva *request*, esto genera en el nivel *N* del sistema "B" una indicación (primitiva *indication*) hacia su nivel *N+1* de petición de conexión. El nivel *N+1* de "B" acepta la llamada y pide a su nivel *N* que de curso a la conexión por medio de la primitiva *response*, lo que genera en el nivel *N* de "A" una confirmación de la conexión (primitiva *confirm.*) que envía a su nivel *N+1* completándose así el servicio solicitado. Una vez establecida la conexión, los únicos servicios solicitados son el envío de datos (DATA) utilizándose la primitiva *request*, dirigida siempre desde el nivel *N+1* al *N* para solicitar el servicio, y generando una primitiva *indication* en nivel *N* del otro sistema, que dirigirá a su nivel *N+1*. El proceso continuará, utilizando ambos interlocutores el servicio de envío de datos, hasta que uno de ellos pida la desconexión.

Para finalizar con este apartado mencionaremos que, en muchas ocasiones, tienden a confundirse los conceptos de *servicio* y *protocolo*, por lo que nos parece adecuado hacer énfasis en su distinto significado, incluyendo la diferenciación que A. Tanenbaum realiza en su libro "Redes de Ordenadores" [Tanenbaum_91].

*"Un **servicio** es un conjunto de primitivas (operaciones), que una capa proporciona a la capa superior. El servicio define las operaciones que la capa efectuará en beneficio de sus usuarios, pero no dice nada con respecto a como se realizan dichas operaciones. Un servicio se refiere a una interfase entre dos capas, siendo la capa inferior la que provee el servicio y la capa superior la que utiliza el servicio".*

*"Un **protocolo**, a diferencia del concepto de servicio, es un conjunto de reglas que gobiernan el formato y el significado de las tramas, paquetes o mensajes que son intercambiados por las entidades equivalentes (pares) dentro de un nivel o capa. Las entidades utilizan protocolos para realizar sus definiciones de servicio, teniendo libertad para cambiar el protocolo, pero asegurándose de no modificar el servicio visible a los usuarios".*

4.4.3 FUNCIONES DE LOS NIVELES

Para poder diseñar el protocolo de un nivel debe describirse, previamente, que funciones debe realizar ese nivel y que servicios oferta. En [ISO_81a] se presentan una serie de directrices acerca de como debe describirse un nivel. Esta descripción debe constar de:

- Una exposición general de los objetivos del nivel.
- Una especificación del servicio o servicios ofrecido por el nivel.
- Una especificación del servicio suministrado por su nivel inferior.

De acuerdo con estas directrices se procederá al análisis de las funciones asignadas a los distintos niveles que conforman la arquitectura propuesta, determinando los servicios que prestan a sus niveles usuarios.

NIVEL FÍSICO. La función asignada a este nivel es la transmisión de un conjunto de *bits* a través de un canal de comunicación. Para ello debe activar y desactivar la conexión física, transmitir las unidades de datos de servicio físico PSDU (*Physical Service Data Unit*) y gestionar los problemas concernientes al medio físico con el cual opera.

Este nivel debe de proporcionar los medios mecánicos, eléctricos, funcionales y procedimentales, para permitir una comunicación entre las entidades del enlace de datos. Debe resolver problemas como determinar los niveles de voltaje que se emplean para representar un "0" y un "1" lógicos, como se inicia y termina una conexión física, que modo de transmisión va a utilizarse (serie o paralela), etc. En definitiva, este nivel debe garantizar que los *bits* se envíen con arreglo a unas normas reconocibles por su entidad física par y en el mismo orden que fueron ofrecidos por la entidad de enlace de datos para su transmisión.

El nivel físico proporciona un servicio DATA (envío y recepción de datos) no confirmado a su nivel superior. Así, cuando reciba una primitiva DATA.*request* de su nivel superior, adecuará la SDU proporcionada a las características de transmisión del medio físico y la enviará. De la misma forma, si el nivel físico recibe una información a través del medio, adaptará esta información a un formato inteligible por su nivel superior y la enviará a este por medio de una primitiva DATA.*indication*.

Al ser el nivel físico el más bajo de la arquitectura no procede especificar el servicio suministrado por el nivel inferior. No obstante es conveniente resaltar que, para el caso concreto de la red objeto de estudio, la comunicación a nivel físico se produce entre los sistemas informáticos conectados en red y los nodos de la subred de comunicaciones. Por necesidades de estos nodos la transmisión debe ser serie y asíncrona, por lo que el nivel físico deberá arropar cada *byte* de datos, suministrado por el nivel superior, con un *bit* de inicio, un *bit* de paridad (si se requiere) y los *bits* de parada que determine el procedimiento asíncrono que se selecciona.

La forma de realizar las funciones asignadas a este nivel es competencia del protocolo, por lo que se estudiará en su apartado correspondiente.

NIVEL NCA (Acceso a la red de comunicaciones). Antes de abordar la descripción de este nivel vamos a realizar un análisis de las funciones que realiza un nivel de enlace convencional. Este análisis es recomendable porque, atendiendo a uno de los principios arquitectónicos de ISO "*formar subniveles dentro de los niveles cuando se requieran servicios de comunicación distintos*", se ha dividido el nivel de enlace en los subniveles NCA (acceso a la red de comunicaciones) y LL (enlace lógico). Este último se ha unido al nivel de sesión para simplificar la arquitectura.

Para realizar esta subdivisión se han seguido los mismos criterios que en las redes de área local, independizando una comunicación extremo a extremo, como es el enlace lógico, del enlace con la subred de comunicaciones. Consiguientemente, revisaremos que funciones realiza un nivel de enlace en su conjunto para ver cuales deben realizar cada uno de los subniveles de la arquitectura.

Según la Asociación Europea de Fabricantes de Ordenadores (ECMA) un enlace de datos es "*un conjunto de dos o más estaciones terminales y un circuito de interconexión, trabajando con un método particular que permite el intercambio de información*" [ECMA_79]. En este contexto el termino "*estación terminal*" no incluye el origen y el fin de los datos.

La misión fundamental de un nivel de enlace es proporcionar una gestión eficiente del circuito de datos, para suministrar a su nivel superior una línea libre de errores de transmisión. Los protocolos que deben ejecutar esta función deben definir reglas para establecer y liberar un enlace de datos, controlar la correcta transferencia de información y recuperar las anomalías que puedan producirse.

Para ello, un nivel de enlace debe realizar las siguientes funciones básicas:

- Sincronizar las tramas[9] de datos, estableciendo la delimitación de los mensajes para poder recuperarlos a partir de la secuencia de *bits* o caracteres recibidos por el circuito físico.
- Transparencia de la información, propiciando que cualquier secuencia de *bits* o cualquier carácter pueda ir incluido como dato dentro de una trama
- Control del flujo de datos, mediante reglas que determinan el turno de intervención a través del enlace.
- Control de errores, incluyendo algún mecanismo que permita detectar si una trama recibida es correcta o no.
- Recuperación de anomalías, incorporando estrategias que le permitan salir de situaciones como recepción de tramas erróneas, ausencias de respuesta, etc....

Ya mencionamos que una de estas funciones, controlar y recuperar los errores producidos en el medio físico, no tendrá que realizarse por los subniveles de enlace. La subred de comunicaciones proporciona (mediante su protocolo de enlace interno) una línea libre de errores y, consecuentemente, partiremos de este supuesto a la hora de definir las funciones relativas a estos subniveles y al diseñar los protocolos que las llevan a cabo.

Del resto de funciones que realiza un nivel de enlace, se asigna al subnivel NCA la misión de confeccionar el formato de las tramas de datos que deben intercambiarse con el nodo de la subred. Esto incluye las funciones de sincronismo de las tramas recibidas y la inclusión del mecanismo de transparencia que requiere la subred de comunicaciones utilizada.

El nivel NCA oferta al nivel sesión-LL un servicio de envío y recepción de datos no confirmados. Cuando reciba una primitiva DATA.*request* aplicará la

[9] *Trama*: Unidad de información estructurada que se intercambia entre las entidades de enlace.

transparencia a la SDU suministrada, confeccionará una trama de datos con un formato entendible por el nodo de la subred de comunicaciones, pondrá los delimitadores de la trama y, a través de la interfase NCA-nivel Físico, la enviará a este último para que la imprima en el medio. En la primitiva enviada por el nivel sesión-LL deberán incluirse como parámetros el campo de control del enlace lógico y el canal por el cual se desea enviar la información, para que el nivel NCA los incluya en su lugar correspondiente dentro de la trama de enlace.

Cuando el nivel NCA reciba del nivel físico una primitiva DATA.*indication* sincronizará la llegada de la trama de enlace a través de su delimitador, eliminará este, el delimitador final y los datos de cabecera que su nivel par NCA haya añadido, aplicará el mecanismo para quitar la transparencia y entregará la información relevante a su nivel superior mediante una primitiva DATA.*indication*. Esta última primitiva debe incorporar como parámetros el canal por el cual se ha recibido la información y el campo de control de la trama.

Para poder llevar a cabo esta función el nivel NCA hace uso del servicio proporcionado por el nivel físico, especificado con anterioridad.

NIVEL DE SESIÓN-LL. Debido a la utilización de nodos con capacidad de proceso en la subred de comunicaciones, y dadas las características de trabajo de estos dispositivos, las funciones propias de un nivel de enlace se simplifican notablemente.

Además de no necesitar la implementación de un control de errores, tampoco será necesario introducir un control de flujo que arbitre el ritmo de generación de la información, ni una función de recuperación de situaciones anómalas (por ejemplo, ausencia de respuestas). Los nodos de la subred de comunicaciones disponen de un protocolo de enlace interno, con una coordinación de la comunicación que incluye el asentimiento de tramas de enlace y una estrategia de retransmisión para recuperar situaciones de bloqueo que puedan producirse.

Esto garantiza que los datos suministrados a un nodo de la subred son entregados a su destinatario en el otro extremo del enlace. Sería excesivo volver a establecer estos controles, que corresponderían al subnivel LL (enlace lógico), en los sistemas informáticos de la red de ordenadores. Máxime si se tiene en cuenta que las aplicaciones que deben operar en red tienen que arbitrar mecanismos para asegurarse que los datos, suministrados por las entidades de aplicación, son recibidos por su entidades pares en otros sistemas.

Teniendo en cuenta las funciones de enlace que ya realiza el nivel NCA, la única misión que le quedaría a un subnivel de enlace lógico sería regular los campos de control y dirección en la trama que conforma el nivel NCA. Esto permite llevar a cabo un diálogo extremo a extremo, a nivel de enlace, entre los sistemas conectados en red. Es una función similar a la que realiza el subnivel LLC en las redes de área local, aunque con un sistema de direccionamiento distinto.

Un nivel LL debe existir, pues es necesario independizar la comunicación extremo a extremo a nivel de enlace de la comunicación, también a nivel de enlace, que se produce con los nodos de la subred. Esto, unido al hecho de que los enlaces entre la estación central y cualquier estación de campo sólo se establecen cuando se solicita una sesión, liberándose el enlace cuando la sesión termina, aconsejan agrupar los niveles *enlace lógico* y *sesión* en uno solo. Esta solución permite simplificar al máximo la arquitectura de la red, eliminado interfases innecesarios y reduciendo el tiempo de comunicación.

En base a las consideraciones expuestas se propone un nivel *sesión-LL* con las siguientes funciones:

- Establecer y liberar *sesiones-conexiones de enlace* para que las entidades de aplicación puedan llevar a cabo un transporte de datos ordinario.
- Direccionar los datos recibidos del nivel de aplicación a través del canal por el que estableció el enlace con la entidad destino.

Para realizar estas funciones utilizará los campos de control y canal, presentes en la trama de enlace que se intercambian con el nodo de la subred. El protocolo que se diseñe para este nivel especificará la forma de realizar estas funciones.

Dentro del mecanismo que se arbitre por este nivel para el establecimiento de las sesiones-enlace deberá tenerse en cuenta el procedimiento de acceso al canal de comunicaciones. En el apartado 3.4 de la memoria se analizaba la conveniencia de utilizar un procedimiento mixto para asignar el canal. Así, mientras en una fase de transferencia de datos el nivel de aplicación de la estación central establece el turno de intervención, cuando no existen mensajes circulando por la red y todas la sesiones está liberadas cualquier sistema conectado en red puede acceder al medio. En esta situación pueden producirse colisiones en el establecimiento de una sesión-enlace y esta no llegar a conseguirse.

El nivel sesión-LL, a la hora de establecer una sesión-enlace solicitada por el nivel de aplicación, deberá elaborar un procedimiento para controlar el establecimiento de estas sesiones considerando que las solicitudes, que realiza a su nivel inferior, pueden ser rechazadas por colisiones en el medio. Este procedimiento informará acerca del éxito en el establecimiento de la sesión ó de si esta tiene algún impedimento para completarse, a fin de que el nivel de aplicación tome las medidas oportunas.

Mediante estas funciones el nivel sesión-LL proporciona los siguientes servicios al nivel de aplicación:

- Servicio confirmado (CONNECT), que le permitirá a las entidades de aplicación establecer sesiones con sus entidades pares en otras máquinas.
- Servicio no confirmado de envío y recepción de datos (DATA).
- Servicio no confirmado de liberación de sesiones (DISCONNECT).

La solicitud de los servicios se realizará por medio de primitivas, tabla 4.1. Y, al igual que hacen otros niveles, se apoyará en los servicios prestados por su nivel inferior

(Nivel NCA, que construye tramas en un formato inteligible por el nodo de la subred) para llevar a cabo sus servicios.

PRIMITIVA	DIRECCIÓN	PARÁMETROS
CONNECT.*request*	N. Aplicación a N. Sesión-LL	Estación destino de la conexión
CONNECT.*indication*	N. Sesión-LL a N. Aplicación	Estación que solicita la conexión
CONNECT.*response*	N. Aplicación a N. Sesión-LL	Estación que solicita la conexión
CONNECT.*confirm*	N. Sesión-LL a N. Aplicación	Estación destino de la conexión
DISCONNECT.*request*	N. Aplicación a N. Sesión-LL	Estación destino de la desconexión
DISCONNECT.*indication*	N. Sesión-LL a N. Aplicación	Estación origen de la desconexión
DATA.*request*	N. Aplicación a N. Sesión-LL	Estación destino de la información
DATA.*indication*	N. Sesión-LL a N. Aplicación	Estación origen de la información
DATA.*request*	N. Sesión-LL a N. NCA	Campo Control, Campo Canal
DATA.*indication*	N. NCA a N. Sesión-LL	Campo Control, Campo Canal

TABLA 4.1 Primitivas intercambiadas entre el nivel de sesión-LL y sus niveles adyacentes

El nivel de sesión-LL tendrá conocimiento de la distribución de los nodos de la subred de comunicaciones de forma que, con unas tablas de enrutamiento fijo, será capaz de confeccionar el camino a través del cual debe producirse la conexión con la estación destino. Una vez establecido el enlace asignará un canal específico al mismo. A partir de este momento utilizará el parámetro "estación destino", suministrado por la aplicación, para saber por qué canal debe enviarse una determinada información.

Este nivel mantendrá en todo momento una relación de las sesiones-enlaces que tiene establecidas, utilizando una tabla dinámica con la correspondencia entre estaciones y canales. Cuando reciba una primitiva (por ejemplo DATA.*indication*) del nivel inferior, utilizará esta tabla para informar al nivel de aplicación (mediante otra primitiva DATA.*indication)* sobre el origen de la información, en función del canal por el cual esta ha sido recibida.

NIVEL DE APLICACIÓN. La estructura de las entidades de un nivel de aplicación son más comprensibles si definimos previamente ciertos términos y conceptos relativos a este nivel.

Por *aplicación* se entiende un "*conjunto de requisitos de un usuario para el procesamiento de la información*". La actualización de un banco de datos, la transferencia de archivos o el cálculo de determinadas magnitudes a partir de unos datos de entrada, son ejemplos de aplicaciones.

Un *proceso de aplicación* es "un elemento lógico de un determinado sistema que procesa la información para una aplicación". Por ejemplo, un proceso que accede y permite manipular los datos de un fichero, residente en un sistema servidor, es un proceso de aplicación para la administración de archivos.

Una aplicación se dice que es *distribuida* cuando los procesos relacionados con ella residen en sistemas distintos [Beltrào_89]. La transferencia de archivos a través de una red es un ejemplo de aplicación distribuida: en un sistema se ejecuta el proceso emisor de los bloques que conforman un archivo y en otro sistema el proceso receptor de estos bloques. Esta aplicación presta un servicio específico a las aplicaciones de usuarios residentes en cada sistema, para que estas desarrollen las tareas que tienen programadas.

La ISO estuvo trabajando en el desarrollo de aplicaciones que prestaran servicios específicos a los sistemas conectados en red: transferencia y manipulación de archivos TMA, sistemas de tratamientos de mensajes STM, definición de terminales virtuales TV, entrada remota de tareas ERT, etc.

Casi todas estas aplicaciones específicas tienen bastantes aspectos en común. Así, por ejemplo, casi todas necesitan manejar conexiones y muchas tienen que coordinar actividades entre varios participantes. Para evitar que cada una de estas aplicaciones, u otras nuevas que pudieran surgir, tengan que resolver estos problemas, la ISO decidió subdividir el nivel de aplicación definiendo un subnivel de servicios comunes que las aplicaciones específicas pudieran utilizar como soporte.

La figura 4.9 representa la estructura lógica del nivel de aplicación del modelo de referencia OSI (*Open Systems Interconnection*) de la ISO. Las aplicaciones de usuario están fuera del alcance del RM-OSI, pero el nivel de aplicación debe de facilitar medios para conectarlas a sus servicios específicos y/o comunes.

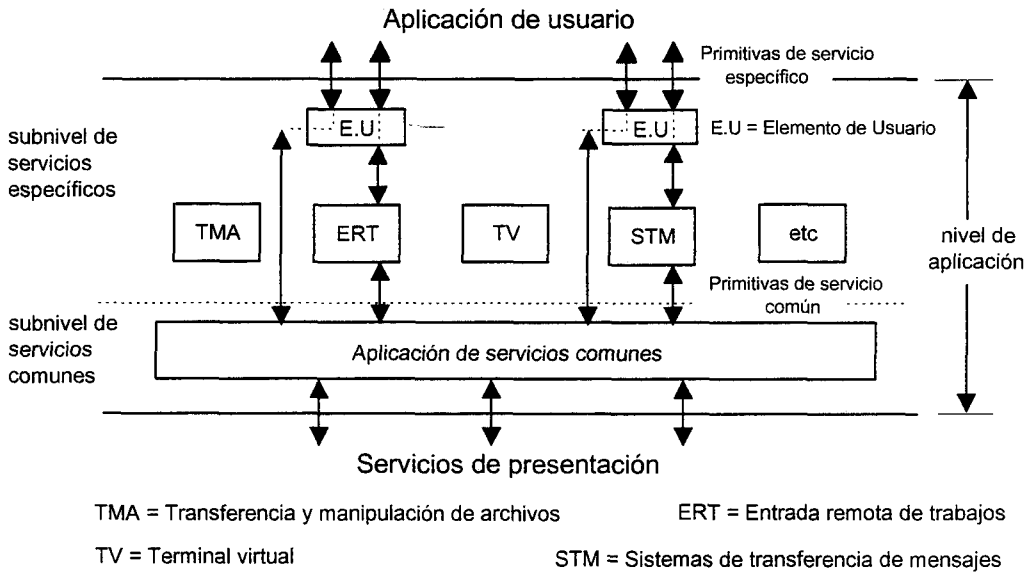

Figura 4.9 Estructura lógica del nivel de aplicaciones del RM-OSI de la ISO

Para comunicar las aplicaciones de usuario con el nivel de aplicación se define una interfase denominada *elemento de usuario*, de forma que éste determina la invocación de primitivas de servicios específicos o directamente las primitivas de servicio común, en función de las necesidades de las aplicaciones de usuario.

Debe destacarse el papel que juega el elemento de usuario en esta estructura. No se trata de un subnivel más, ya que no recibe y envía primitivas generadas por el mismo. Su labor es la misma que la de cualquier interfase entre niveles, pero con la particularidad de que, en función de la solicitud recibida de la aplicación de usuario, puede bifurcar el servicio demandado a una entidad de aplicación específica o a una entidad situada en un subnivel inferior.

Uno de los principios arquitectónicos utilizados por ISO para definir su modelo de interconexión dice que "*se debe crear para cada nivel interfases con el nivel*

superior e inferior únicamente". Esta afirmación debería servir también cuando un nivel debe desdoblarse. De acuerdo con este principio no se podría comunicar directamente la aplicación de usuario con el subnivel de servicios comunes, saltándose un subnivel intermedio como es el de servicios específicos. No obstante esta comunicación es necesaria para mejorar las prestaciones en la interconexión de sistemas abiertos.

La inclusión del elemento de usuario en el nivel de aplicación de la red, por encima de las entidades de aplicación específicas (véase figura 4.9), permite salvar esta situación. De esta forma, la aplicación o aplicaciones de usuario comunican siempre con el subnivel más alto del nivel de aplicación (donde de encuentra el elemento de usuario) y la comunicación del subnivel de servicios comunes se produce siempre con su nivel inmediato superior (subnivel de servicios específicos), independientemente de que esta comunicación la realice con una entidad de aplicación o con el elemento de usuario.

En esta red de ordenadores disponemos de dos aplicaciones, *transferencia de archivos* y *entrada remota de trabajos,* que pueden ser consideradas aplicaciones de servicios específicos para la red. Ambas aplicaciones necesitan de una conexión para poder ejecutarse, por lo que podríamos aislar un subnivel, dentro del nivel de aplicación, que se encargará de prestar este servicio común. Sin embargo, este servicio sería el único que proporcionaría este subnivel y un servicio de estas características ya se presta por el nivel de sesión-LL.

Como ya se ha comentado, el interés se centra en reducir al máximo el número de interfases entre niveles. Si se incorpora un subnivel de servicios comunes, deberán definirse nuevas primitivas de servicio para posibilitar el acceso de las aplicaciones específicas de red y del elemento de usuario a este subnivel.

El nivel de aplicación de esta red de ordenadores tendrá una estructura lógica (figura 4.10) similar a la del modelo OSI, pero prescindiendo del subnivel de servicios comunes. El elemento de usuario recibirá las primitivas de servicio, generadas por las aplicaciones de usuario, y en función del tipo de primitiva la direccionará a un punto de

acceso al servicio (SAP), perteneciente a una aplicación específica o directamente a un SAP del nivel de Sesión-LL.

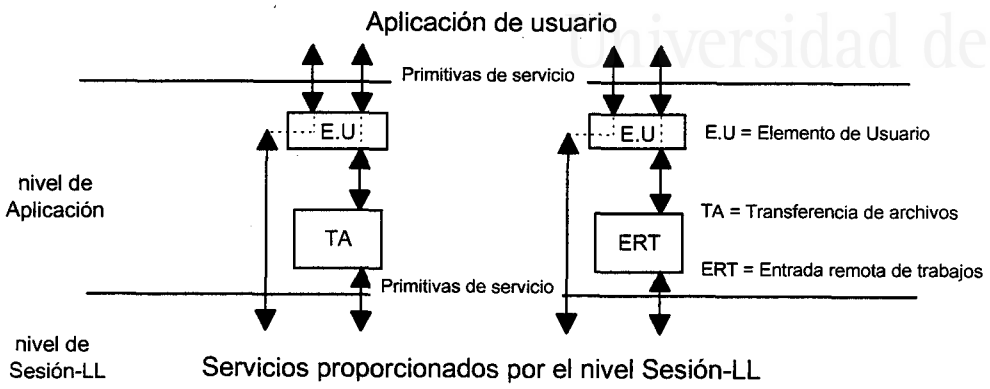

Figura 4.10 Estructura lógica del nivel de aplicaciones

Teniendo en cuenta que, para transferir un fichero o para pedir a un sistema remoto que realice un determinado trabajo, necesitamos establecer una sesión previa entre las dos estaciones, podemos diseñar las aplicaciones de usuario de forma que no emitan ninguna primitiva de solicitud de un servicio específico si previamente no existe una sesión entre los sistemas que desean dialogar. Si procedemos así, cuando se soliciten los servicios proporcionados por una aplicación específica ésta no necesitará realizar la conexión previa y se minimizará el número de primitivas que se intercambia entre las entidades de aplicación específicas y el nivel sesión-LL. Así, los servicios CONNECT y DISCONNECT demandados por la aplicación de usuario al nivel de aplicación de la red, serán siempre desviados por el elemento de usuario al nivel de sesión-LL que proporciona este servicio.

El resto de servicios solicitados por la aplicación de usuario son encaminados, por el elemento de usuario, a las entidades de aplicación que prestan servicios específicos. Las aplicaciones específicas de la red utilizarán, únicamente, el servicio DATA proporcionado por el nivel de sesión-LL para cumplir su objetivo. Procediendo de esta forma, las primitivas que se intercambian entre las aplicaciones de usuario y el nivel de aplicación de la red serían las siguientes.

- CONNECT.*request.* (parámetro = dirección de la estación destino). Esta solicitud será bifurcada por el elemento de usuario, presente en el nivel de aplicación, directamente al SAP del nivel sesión-LL, que se encarga de prestar el servicio de establecimiento de sesiones.

- CONNECT.*indication* (parámetro = dirección de origen). Informa a la aplicación de usuario, a través del elemento de usuario, que una determinada estación desea establecer una sesión con ella para un intercambio de datos.

- CONNECT.*response* (parámetro = dirección origen). La aplicación de usuario llamada informa acerca de su disposición al establecimiento de una sesión.

- CONNECT.*confirm* (parámetro = dirección destino). El nivel de sesión-LL del sistema que originó la solicitud de sesión informa a la aplicación de usuario, a través del elemento de usuario, acerca de su solicitud

- DISCONNECT.*request* (parámetro = dirección estación con la que se solicita la desconexión). El elemento de usuario bifurca igualmente esta solicitud al SAP del nivel sesión-LL que proporciona este servicio. Se trata de un servicio no confirmado por lo que sólo consta de esta primitiva y de la primitiva DISCONNECT.*indication*, dirigida desde el nivel sesión-LL al elemento de usuario de la aplicación (parámetro = dirección de la estación que solicitó la desconexión).

- E_FICH.*request* (parámetros = dirección destinataria del fichero, un identificativo que permita seleccionar la información que se desea enviar y otros necesarios para el control del fichero en el extremo receptor). Solicitud de enviar fichero que el elemento de usuario bifurcará a la aplicación específica *transferencia de archivos* (TA). Esta aplicación debe proporcionar un servicio fiable de forma que, el protocolo que se diseñe para comunicar dos entidades responsables de la transferencia de un fichero, debe arbitrar mecanismos para detectar posibles errores y recuperar situaciones anómalas garantizando la integridad de la información.

- R_FICH.*indication* (parámetro = estación origen del fichero). La aplicación específica TA informa a la aplicación de usuario que se encuentra ocupada en un proceso de

recepción de un fichero. No es necesario enviar ningún parámetro adicional, pues solo se remitirá a la aplicación de usuario el fichero recibido si la operación se completa con éxito.

- FIN_EFICH.*indication* (parámetro = estación destino del fichero). La aplicación específica TA informa a la aplicación de usuario que el proceso de envio de fichero solicitado ha finalizado con éxito.

- FIN_RFICH.*indication* (Parámetros = estación origen del fichero, un identificativo que permita localizar la información del fichero recibida y otros necesarios para el control del fichero remitidos por la otra estación). La aplicación específica TA informa a la aplicación de usuario que el proceso de recepción de un fichero ha finalizado con éxito, remitiendole la información necesaria para que pueda recuperarlo.

- ABORT_TA.*request* (parámetro = estación origen o destino del fichero). Solicitud de la aplicación de usuario a la aplicación específica TA para que corte el proceso de envío o recepción del fichero que mantiene con una determinada estación. Esta situación puede producirse si la sesión con la estación que se está intercambiando el archivo se interrumpe. La aplicación de usuario será avisada por su nivel de sesión-LL mediante una primitiva DISCONNECT.*indicatión*, pero la aplicación específica TA desconoce esta situación porque dicha primitiva se remite directamente al elemento de usuario para que la bifurque a la aplicación de usuario. Es esta última quien debe advertir a la aplicación TA que la operación no puede continuar.

- ABORT_TA.*indication* (parámetros = estación origen o destino del fichero y otro parámetro indicando la causa de la interrupción del proceso). La aplicación específica TA informa a la aplicación de usuario que el proceso de envío o recepción de un fichero se ha interrumpido por un error que no se puede recuperar en este nivel. Esta situación se producirá cuando una entidad de la aplicación TA agote los reintentos para recuperar situaciones anómalas que puedan ocurrir durante el proceso.

- JOB.*request* (parámetros = dirección de la estación que debe efectuar el trabajo y un identificativo que permita seleccionar el tipo de trabajo que se desea). Solicitud para que se efectúe un trabajo en un sistema remoto. La solicitud se realiza por la aplicación de usuario al elemento de usuario, que la bifurca a la aplicación específica *entrada remota de trabajos*. Esta solicitud puede contener también el resultado de una operación efectuada en un sistema que se desea enviar a otro. Esto puede identificarse en la aplicación específica por el parámetro "*tipo de trabajo*". Dependiendo del tipo de trabajo solicitado se acompañará la información necesaria para que el trabajo se efectúe, o la información que la aplicación de usuario desea que conozca el otro sistema. No se requiere que esta aplicación proporcione un servicio fiable, limitándose su función a tramitar la información recibida por la aplicación de usuario. Esta última será la responsable de arbitrar los medios para solucionar situaciones anómalas, por ejemplo, si ha solicitado un determinado trabajo a otra estación y la respuesta al mismo no se efectúa en el tiempo previsto, la aplicación de usuario será la responsable de solventar el problema.

- JOB.*indication* (parámetros = dirección de la estación que origina la solicitud o información, y un identificativo que permita reconocer el tipo trabajo o evento que se le comunica). Primitiva enviada por el servicio de aplicación específico *entrada remota de trabajos* a la aplicación de usuario, a través del elemento de usuario, para informar a esta que debe efectuar un determinado trabajo o que se le envía una información procedente de un trabajo realizado por otra estación. Además de estos parámetros, la aplicación de usuario recibirá la información necesaria para efectuar el trabajo o la información que se desea proporcionar.

Para prestar los servicios específicos las entidades de aplicación *transferencia de archivos* y *entrada remota de trabajos* utilizarán el servicio DATA (envío y recepción de datos) no confirmado, que proporciona el nivel de sesión_LL. Por el contenido de la cabecera de la APDU (unidad de datos de protocolo de las entidades de aplicación) podrá identificarse si una DATA.*indication*, recibida en el nivel de aplicación desde el nivel de sesión-LL, debe ser atendida por una u otra entidad.

Dentro de esta estructura no se ha considerado un nivel de presentación porque, en principio, los servicios que puede prestar este nivel pueden considerarse útiles, pero no esenciales. No obstante, pueden producirse situaciones en las cuales los datos que van a viajar por el medio físico sean confidenciales; debe tenerse en cuenta que se trata de un medio de difusión y pueden ser fácilmente capturados. O bien los ficheros que deben intercambiarse entre los sistemas conectados a la red sean excesivamente grandes y sea conveniente realizar un proceso de compresión de datos para reducir el tráfico de información.

Ante una situación de estas características podría implementarse un nivel de presentación, entre los niveles de aplicación y sesión-LL, que se responsabilizará de realizar estas tareas. Esto supone una ampliación de la estructura de niveles propuesta y, aunque no presenta excesiva dificultad por la jerarquización que se ha realizado en los niveles de la arquitectura, implicaría una nueva definición de primitivas entre las interfases implicadas.

Las funciones de *encriptar* o *comprimir* la información han sido colocadas, dentro del modelo ISO, en el nivel de presentación, pero esta no es la única alternativa. De hecho, hoy en día, muchos modems realizan comprensión de datos a nivel de enlace lógico, siendo transparente esta operación a los sistemas conectados a ellos. La *encriptación* de la información, al igual que la *comprensión,* puede llevarse a cabo en cualquiera de los niveles siempre que la entidad par, en el sistema receptor de la información, opere de la misma forma.

Si es necesario que alguna de estas funciones se lleven a cabo para desarrollar correctamente un control distribuido, el proceso que se encarga de ejecutarlas debe estar implementado en la propia aplicación de usuario. De esta forma no se altera la arquitectura y además se puede obtener un rendimiento mayor de la subred de comunicaciones.

La entidad de aplicación que presta el servicio específico de transferencia de archivos necesita, para hacer su servicio fiable, fragmentar la información para poder recuperar los fragmentos dañados. Por otra parte, la subred de comunicaciones solo

admite un determinado número de *bytes* como datos en una trama de enlace. Con la estructura de niveles propuesta podemos diseñar los protocolos de los niveles de forma que, los niveles de aplicación de la red que prestan servicios específicos sean los únicos que realicen una fragmentación de la información. Esto puede hacerse siempre que esos fragmentos, más las cabeceras de los niveles que deben atravesarse hasta confeccionar la trama de enlace, no sobrepasen el tamaño máximo que el nodo de la subred admite para el campo de datos en estas tramas. Con esto conseguimos que ningún otro nivel tenga que fragmentar la información y obtener el máximo rendimiento de la subred cuando se intercambian ficheros, pues siempre confeccionaríamos, excepto en el último fragmento, tramas con el mayor campo de datos posible.

Supongamos que una vez que el nivel de aplicación hace los fragmentos, sometemos estos a un proceso, como la encriptación o comprensión de datos, que puede alterar el número de *bytes* de dichos fragmentos. En este caso necesitaríamos realizar una nueva fragmentación, en alguno de los niveles que están por debajo de las entidades que ejecutan estos procesos, para adecuar nuevamente el campo de datos de la trama de enlace al tamaño máximo permitido por la subred.

Después de realizar un proceso de encriptación, el número de *bytes* del fragmento seleccionado por el nivel de aplicación podría ser mayor que el campo de datos admitido por la trama de enlace, por lo que la nueva fragmentación sería obligada para poder enviar estas tramas. Si realizamos un proceso de comprensión de datos con las SDU proporcionadas por el nivel de aplicación, el número de *bytes* resultante será menor y deberían reagruparse los datos para fragmentar nuevamente la información con arreglo al campo de datos máximo admitido por la trama de enlace. Si en este último caso no se fragmenta de nuevo, infrautilizamos las prestaciones de la subred de comunicaciones al tener que tramitar tramas de enlace con un contenido de información útil menor al que soporta la red.

Una nueva fragmentación puede realizarse en el nivel sesión-LL o en el NCA, pero esto complicaría el protocolo del nivel que realiza este proceso sin necesidad. Procesos como la encriptación, la comprensión y cualquier otro proceso que tenga que

ver con el tratamiento de datos, puede ser realizado por la aplicación de usuario, liberando al sistema de interconexión de proporcionar estos servicios.

Con estas precisiones podemos considerar la estructura de niveles propuesta para esta red de ordenadores, como una plataforma suficiente para que los procesos de control distribuido puedan desarrollarse de una forma eficiente. Lógicamente solo sistemas que se interconecten utilizando esta arquitectura pueden dialogar entre sí. Debe tenerse en cuenta que el acceso a nodos de control remotos únicamente es de interés para el proceso de control y que, por lo tanto, no representa ningún problema que sistemas en general abiertos no puedan tener acceso a los mismos.

No obstante, puede ser posible el acceso a nodos de control remoto o al ordenador central desde sistemas que no están interconectados a esta red. Teniendo en cuenta que el interfase para operar con la red se encuentra situado en el ordenador central, ubicado en el centro de proceso, bastaría con que este sistema (u otro que estuviera en red local con el ordenador central) hicieran de puente con una red extendida como, por ejemplo, *internet*. De esta forma la consola para operar con la red podría transferirse a cualquier otro sistema conectado a *internet*. Lógicamente este acceso debería estar restringido al operador u operadores de la red de control distribuido, por lo que deberían arbitrarse los sistemas de seguridad suficientes para que cualquier usuario no pueda entrar y manipular el proceso.

Así mismo, si determinada información resultante del proceso que se está controlando es de interés público, disponiendo de una conexión en el centro de proceso a una red de área extendida, puede depositarse esta información en una base de datos para su consulta o puede distribuirse a determinados organismos que la necesiten.

4.5 PROTOCOLOS DE NIVEL

En la especificación de los servicios suministrados por un determinado nivel *N* no se define "*como*" se realizan dichos servicios. Es el protocolo de nivel *N* el responsable de determinar como se ejecutan las funciones adjudicadas al nivel *N* para que el servicio pueda ser prestado. Con la especificación de los protocolos de cada uno de los niveles propuestos, completaremos el trabajo de diseño de la Arquitectura de la red.

Para especificar los protocolos que regulan la comunicación entre entidades pertenecientes a un mismo nivel y ubicados en sistemas distintos (entidades pares), seguiremos las directrices propuestas por ISO. Según estas [ISO_81a], la descripción del (los) protocolo (s) entre entidades *N* debe incluir:

- Una descripción general e informal de la operación entre las entidades
- Una especificación del protocolo que incluye:
    a) Una lista de tipos y formatos de los mensajes transferidos entre las entidades *N*.
    b) Las reglas que gobiernan la reacción de cada entidad *N* a las órdenes recibidas a través de las interfases de otras entidades y a sucesos internos.
- Detalles adicionales, tales como consideraciones para mejorar las prestaciones, sugerencias de implementación o una descripción pormenorizada que se acerque a la implementación.

Debe observarse que la cuestión de como implementar una determinada entidad, es dejada abierta para dar libertad a la hora de escoger los métodos de implementación.

4.5.1 ESPECIFICACIÓN INFORMAL DE LOS PROTOCOLOS

PROTOCOLO DE NIVEL FÍSICO. Este protocolo debe definir el procedimiento para el intercambio de información entre las entidades de nivel físico, ubicadas en los sistemas conectados en red y los nodos de la subred de comunicaciones. El hecho de utilizar una subred de comunicaciones ya confeccionada, condiciona la utilización de un determinado protocolo de nivel físico.

La conexión física de sistemas informáticos a la subred de comunicaciones utilizada en esta memoria debe realizarse bajo la norma RS232C. Esta norma, propuesta por la Asociación de Industrias Electrónicas EIA (*Electronic Industries Association*), corresponde a la cuarta versión revisada de la original RS232 y se encuentra englobada en la recomendación V24/V28[10] del CCITT. A menudo estas dos normas se confunden, confusión que parece lógica ya que la norma V24 incorpora algunos circuitos más, pero que muy raramente se utilizan por lo que sus diferencias no son substanciales. En cierto sentido RS232-C puede considerase un subconjunto de V24.

La norma RS232-C especifica, detalladamente, los aspectos mecánico, eléctrico, funcional y de procedimiento para establecer un circuito de datos, que servirá para transportar la información que se le entrega en forma de señales digitales.

---

[10] *SERIE "V" del CCITT*: La serie "V" del Comité Consultivo Internacional Telegráfico y Telefónico hace referencia a las recomendaciones para los estándares destinados a la transmisión de datos a través de ETCD's (Equipos terminales de circuitos de datos), por ejemplo modems.

La especificación mecánica considera un conector de 25 patillas, con la descripción de medidas y forma de numerar estas patillas. Las especificaciones eléctricas consideran los niveles de voltaje en el emisor y receptor para identificarlos con el "0" y "1" lógicos, la impedancia de entrada del receptor , la capacidad de carga en el receptor, etc.

A nivel funcional la norma específica los circuitos conectados a cada patilla así como el significado de cada uno de ellos. En la tabla 4.1 pueden verse la relación entre las principales señales de las normas V24 y RS232-C, sus respectivos circuitos y su asignación a las patillas del conector.

CCITT V24	EIA RS232-C	Patilla conector	Abrev.	Dirección	Significado
101	AA	1	-	----------	Tierra de protección
103	BA	2	TD	ETD a ETCD	Transmisión de datos
104	BB	3	RD	ETCD a ETD	Recepción de datos
105	CA	4	RTS	ETD a ETCD	Petición de emisión
106	CB	5	CTS	ETCD a ETD	Preparado para emitir
107	CC	6	DSR	ETCD a ETD	ETCD preparado para recibir
102	AB	7	-	----------	Circuito de retorno
109	CF	8	CD	ETCD a ETD	Detección de portadora
114	DB	15	-	ETCD a ETD	Reloj de emisión interno
115	DD	17	-	ETCD a ETD	Reloj de recepción interno
108.2	CD	20	DTR	ETD a ETCD	ETD preparado para recibir
125	CE	22	RI	ETCD a ETD	Indicador de llamada
111	CH	23	-	ETD a ETCD	Selector de velocidad
113	DA	24	-	ETD a ETCD	Reloj de emisión externo

TABLA 4.1 Interfases CCITT (V24) / EIA (RS232C)

Los circuitos representados en la tabla 4.1 son los más comunes, pero normalmente no se utilizan todos ellos en la mayoría de comunicaciones serie. De hecho muchos dispositivos tienen un conector de 9 patillas, en lugar del estándar de 25, con las conexiones (CD, RD, TD, DTR, circuito de retorno, DSR, RTS, CTS y RI). Muchas comunicaciones se establecen con solo cinco hilos (TD, RD, RTS, CTS y circuito de retorno) y se puede llevar a cabo una comunicación, únicamente, con tres hilos (TD, RD y circuito de retorno), siempre que se establezca un control de flujo *software* a nivel físico (*X_ON / X_OFF*).

A nivel de procedimiento en la norma RS232C se establece la secuencia de señales que deben producirse para llevar a cabo una comunicación entre un ETD (equipo terminal de datos) y un ETCD (equipo terminal de circuito de datos). El protocolo está basado en pares acción reacción, por ejemplo, cuando el ETD envía un RTS (petición de emisión) el ETCD le contesta con un CTS (preparado para emitir), siempre que esté en disposición de hacerlo.

Dado que el protocolo de nivel físico se encuentra normalizado y su diseño no se realiza en esta memoria, no haremos más consideraciones sobre el mismo. Una información más detallada sobre la norma RS232C puede encontrarse en [Alabau_86], [Black_87], [García_90], [Halsall_92], [Purser_89] y [Tanenbaum_91] , entre otros.

La subred de comunicaciones utilizada requiere que la comunicación a nivel físico con el nodo de la subred se realice de forma asíncrona. Los sistemas informáticos más usuales disponen de conexiones (puertos serie) que se atienen a la norma RS232C y los dispositivos *hardware* que controlan las conexiones, están preparados para codificar en forma asíncrona cada carácter o palabra recibido para enviarlo por el medio físico. Esta codificación se realiza de acuerdo con las características de transmisión asíncrona definidas previamente para el puerto serie. Estos dispositivos están cualificados también para detectar la información asíncrona presente en el medio y entregar a su nivel superior (enlace), los caracteres o palabras libres de la información necesaria para su sincronización.

La mayoría de entornos de programación disponen de facilidades para configurar, leer y escribir en estos puertos. Esto permite que la función de arropar a cada carácter o palabra, suministrada por el nivel de enlace, con la información necesaria para poder realizar una transmisión asíncrona, recaiga directamente sobre el nivel físico. Así el nivel de enlace ignorará el modo de transmisión en el canal, entregando únicamente al nivel físico caracteres o palabras para que estas las imprima en el medio. El nivel de enlace par recibirá solo esta información siendo transparente, a este nivel, el servicio proporcionado por el nivel físico.

PROTOCOLO DE ACCESO A LA SUBRED DE COMUNICACIONES (Protocolo NCA). La misión fundamental del nivel NCA es la confección de tramas de nivel de enlace para intercambiar información con su nivel par del nodo de la subred de comunicaciones y arbitrar el mecanismo de transparencia, necesario para que cualquier combinación de *bits* pueda ser enviado como dato dentro de la trama de enlace.

Para definir como debe llevar a cabo sus funciones el protocolo de este nivel, necesitamos conocer el formato de la trama y el mecanismo de transparencia que operan en la subred de comunicaciones.

FORMATO DE LA TRAMA:

La trama de este nivel debe adaptarse a un formato entendible por el nodo de la subred de comunicaciones. Dentro de los distintos modos de operación a nivel de enlace, que poseen los nodos utilizados, seleccionamos el *modo Host*. Su composición de campos en la trama permitirá llevar a cabo las funciones definidas para este nivel.

En la figura 4.11 se muestra el formato de la trama que debe utilizarse, para que este sea compatible con el formato en *modo Host* del nodo de la subred de comunicaciones.

Guión	Control	Puerto	Canal	Datos	Guión

Figura 4.11 Formato de la trama de enlace

En esta trama pueden distinguirse los siguientes campos:

- GUIÓN: Cada trama se encuentra delimitada por dos guiones, definidos por el carácter **FEND** (C0 hexadecimal, 192 decimal)

- CONTROL: Este campo será solamente relevante para el nivel superior (sesión-LL) y permite un diálogo a nivel de enlace entre puntos extremos a través de la subred comunicaciones. El contenido de este campo será proporcionado por el nivel sesión-LL al nivel NCA, para que este último pueda confeccionar la trama.

- PUERTO: Este campo no es necesario para la operación de la red de ordenadores, pero debe especificarse a requerimiento del *modo Host*. Con este campo se indica el dispositivo TNC, que supervisa el nodo de la subred de comunicaciones, el puerto por el cual debe enviar la información. El contenido de este campo será "1" o "2" si el TNC está en operación "puerto dual". Aunque estuviera en modo operación "puerto único" este campo debe rellenarse obligatoriamente con un "1".

- CANAL: Determina porque canal se envían o reciben datos. El TNC que controla un nodo de la subred de comunicaciones puede establecer hasta 26 conexiones simultáneas (canales A→Z). Cada conexión con una determinada estación se establece por un canal de forma que, una vez establecida y hasta su liberación, todos los datos intercambiados con esta estación llevarán el mismo identificativo de canal. El control de este campo corresponde al nivel de sesión-LL, quien lo suministrará al nivel NCA para que éste configure la trama.

- DATOS: Campo que contienen datos del nivel superior y se elabora con la SDU proporcionada por este.

La confección de esta trama permite al protocolo de nivel NCA realizar una de las funciones que tiene asignadas este nivel, el sincronismo de las tramas de enlace. Los guiones son la referencia que determinan el límite de las tramas. El nivel NCA recibirá secuencialmente una serie de *bytes*, proporcionados por el nivel físico, donde el carácter FEND determinará el origen de una trama y el siguiente FEND su final.

El nodo de la subred de comunicaciones entrega tramas completas a los sistemas informáticos delimitadas por FEND, por lo que la recepción de un conjunto de *bytes* que no comiencen con el carácter FEND es prácticamente nula, salvo un error grave de *hardware* que imposibilitaría cualquier transmisión.

TRANSPARENCIA:

Las tramas de enlace que se intercambian entre un nodo de la subred y el sistema conectado al mismo, están delimitadas por un carácter que determina el principio y fin de una trama (*Guión*). Este carácter no puede estar contenido dentro de la trama, pues esto se entendería como final de trama dando lugar a conformaciones de trama erróneas. Los datos que van a intercambiarse serán, en la mayoría de los casos, ficheros binarios por lo que cualquier conformación de *bits* está permitida. Esto obliga a incluir un sistema de transparencia, que se gestiona de la siguiente forma:

Se utiliza el carácter **FESC** (DB hex.) como carácter de transparencia, por lo que éste tampoco puede aparecer como dato dentro de la trama.

Para implementar la transparencia, el sistema que envía la información debe analizar cada *byte* a transmitir:

1. Si es un **FEND** lo transforma en **FESC+TFEND** (TFEND = DC hex)
2. Si es un **FESC** lo transforma en **FESC+TFESC** (TFESC = DD hex)

Para eliminar la transparencia el sistema que recibe la información debe operar como sigue:

El receptor analiza cada carácter considerándolo como dato hasta la recepción de un FESC. Cuando reciba un carácter FESC tendrá en cuenta el siguiente carácter, considerando ambos como un único *byte* de datos según la siguiente regla:

1. Si recibe **FESC+TFEND** lo transformará en un dato equivalente a **FEND**
2. Si recibe **FESC+TFESC** lo transformará en un dato equivalente a **FESC**

Operación del protocolo NCA. El nivel NCA proporciona un servicio DATA no confirmado. Así, cuando este nivel reciba una primitiva DATA.*request* del nivel de sesión-LL (con parámetros, *control y canal*) tomará la SDU (unidad de datos del servicio), proporcionada por el nivel de sesión-LL, le aplicará el mecanismo de

transparencia y la incluirá en la trama definida, dentro del campo de datos. Seguidamente compondrá el resto de la trama con los parámetros *control* y *canal* proporcionados y el *puerto* que tenga especificado en el TNC. Finalmente pondrá los guiones a la trama conformando una NPDU (Unidad de datos de protocolo del nivel NCA). Esta NPDU será la SDU que envíe, con una primitiva DATA.*request,* al nivel físico para que este la transmita a su entidad *par* situada en el nodo de la subred de comunicaciones.

Los campos: control, puerto y canal que incluye en la trama, no es necesario que sean sometidos al mecanismo de transparencia porque, por su contenido, no puede coincidir nunca con los caracteres FEND o FESC

Cuando el nivel NCA reciba una primitiva DATA.*indication* del nivel físico, sincronizará la trama recibida por la llegada del carácter FEND, eliminará la transparencia del campo de datos de la trama y entregará al contenido de este campo, como una SDU, al nivel de sesión-LL mediante una primitiva DATA.*indication* a la que acompañará como parámetros los contenidos de los campos de control y canal presentes en la trama.

PROTOCOLO DEL NIVEL DE SESIÓN-ENLACE LÓGICO. Para prestar los servicios definidos para este nivel se utilizarán, por parte de su protocolo, los campos de control y canal presentes en la trama de enlace.

Dentro del campo de control pueden existir los siguientes contenidos:

C .- Indica que envía un comando al nodo de la subred de comunicaciones. El comando se especifica dentro del campo de datos de la trama de enlace, cuyo contenido (sin transparencia) también confecciona este nivel. Esto permite dialogar con el nodo de la subred de comunicaciones, al cual esta conectado el sistema informático, y parametrizar y configurar el dispositivo que lo controla (TNC). Pero esto no es un servicio que necesita la red de ordenadores, ni se produce una comunicación con

ningún otro sistema conectado a la red. Es más bien una utilidad que permite a una aplicación de usuario disponer de un mecanismo de control sobre el dispositivo inteligente de la subred de comunicaciones al cual está conectado.

Ahora bien, si el contenido de ese comando es un CONNECT, seguido del identificativo de la estación con la cual se desea establecer una conexión y del camino, a través de la subred, que debe seguirse hasta la estación destino; entonces, el nodo de la subred de comunicaciones tramitará esta solicitud hasta el sistema final y será este quien acepte o no la conexión. Esto si que se trata de un diálogo extremo a extremo entre dos sistemas conectados en red y, además, es el servicio de conexión confirmado que debe prestar el nivel de sesión-LL.

Como hemos mencionado este nivel debe conocer las rutas para establecer la conexión con cualquier sistema de la red, para prestar este servicio. También debe tener conocimiento del estado de sus canales (disponible u ocupado), para asignar un canal libre a cualquier conexión solicitada. Una vez que una conexión ha sido aceptada, al tener asignado un canal específico cualquier comunicación con la estación que aceptó la conexión llevará el mismo identificativo de canal. Esto es una considerable ventaja porque no es necesario volver a especificar origen, destino y camino que debe seguir la información, en cada trama que se envía a la subred. Únicamente con el contenido del campo *canal*, el nodo de la subred que la recibe sabe el sistema al cual debe entregarla. De la misma forma, cuando un nivel de sesión-LL recibe una información del nivel NCA, conoce, por el parámetro *canal*, el origen de la información que se le remite.

Con un campo de control **C** y contenido "DISCONNECT" en el campo de datos, también se solicitará la desconexión establecida por un determinado canal (el canal se libera a partir de ese momento). Este servicio que debe proporcionar el nivel de sesión-LL se trata de un servicio no confirmado y únicamente se informará al sistema destino de la desconexión.

D.- Si el contenido del campo de control es una "**D**" (Datos), el protocolo específica que el contenido del campo de datos, incluido en esa trama, corresponde a una SDU proporcionada por el nivel superior (nivel de aplicación).

El protocolo de sesión-LL puede recibir, como parámetro proporcionado por el nivel inferior NCA, un campo de control con los siguientes contenidos:

R.- Solicitud de una conexión que no ha podido establecerse, por estar los canales ocupados o no aceptar el nodo de la subred conexiones de enlace de otros sistemas. En el campo de datos se especifica origen y la ruta de la solicitud. La aplicación de usuario será informada de este evento y será esta quien solicite una sesión al sistema que le ha proporcionado el aviso.

D.- El campo de datos de la trama de enlace contiene información para el nivel superior.

S.- Datos de sistema proporcionados directamente por la subred de comunicaciones, informando acerca de un cambio de estado en el enlace lógico entre los sistemas finales (CONNECT o DISCONNECT) por un determinado canal.

Mediante la conformación de los campos de *control* y *canal* el protocolo de sesión-LL, proporciona los servicios CONNECT, DATA y DISCONNECT adjudicados a este nivel. En el caso de los servicios CONNECT y DISCONNECT el protocolo confecciona además el campo de datos de la trama de enlace, siendo esta información relevante, únicamente, a efectos de operación del protocolo. Siempre que se solicite un servicio DATA por el nivel superior, el protocolo de sesión-LL confeccionará el campo de control con una "**D**" y el campo de datos de la trama de enlace contendrá información del nivel que solicitó el servicio.

Debemos tener en cuenta que, cuando se le solicita a este nivel el establecimiento de una sesión para las entidades de aplicación, el acceso al medio físico no está controlado por ninguna de las estaciones de la red y las tramas que solicitan el establecimiento de un enlace entre dos sistemas, pueden sufrir colisiones y no llegar a su destino.

La posibilidad de que una colisión suceda es muy pequeña, pues debe tenerse en cuenta que cada TNC escucha el medio y solo si no detecta actividad pone portadora

y envía la información. Por otra parte se puede programar en el TNC el número de reintentos que debe realizar para que una trama que necesita confirmación (como una solicitud de conexión) a nivel de AX25, vuelva a reexpedirse si esta confirmación no llega.

Aunque con estas prestaciones del TNC prácticamente se asegura la llegada de las solicitudes de conexión a las estaciones remotas, cabría la posibilidad que 2 o más TNC, con el mismo número de reintentos, encuentren vacío el medio y emitan al mismo tiempo. Debe tenerse en cuenta que los reintentos no son aleatorios pues, según se estableció en el capítulo 3, se seleccionó un procedimiento de acceso al medio "1-persistente" para obtener el máximo rendimiento en el canal a la hora de realizar la transferencia de información.

Para solventar una posible colisión de este tipo de tramas, cada vez que el nivel de sesión-LL envíe una trama que contenga una solicitud de sesión-enlace lógico, activará un tiempo de espera T_C (tiempo de espera para la conexión del enlace lógico), común para todas las estaciones en red. Este tiempo debe ser mayor que el tiempo necesario para que el TNC realice todos los reintentos que tiene programados (el mismo nº para todos los nodos de la subred). Si T_C vence el protocolo de sesión-LL entenderá que su trama ha colisionado e iniciará un procedimiento de envío de una nueva trama que contenga una solicitud de conexión con el siguiente algoritmo:

$$T_{AC} = (2^{i} - 1)\, m\, T_R$$

donde:

T_{AC}, es el tiempo para emitir la nueva trama

i, es el número de intentos ($i = 1, 2, 3 \ldots$)

m, es la prioridad de la estación ($m = 1, 2, 3 \ldots$)

T_R, es un tiempo de referencia

Bastaría que T_R fuera ligeramente superior al tiempo de propagación de la señal física entre puntos extremos de la red, para que la estación que tienen menos prioridad encuentre el medio ocupado en el primer intento y no emita su información evitando la

nueva colisión. No obstante es aconsejable que T_R sea como mínimo de 100 ms, margen suficiente para que la conexión solicitada en primer lugar se complete. Así se evita que la solicitud de la segunda conexión pudiera colisionar con la confirmación de la primera.

Es conveniente que el número de intentos (i) se limite, al menos en la estación central que es el sistema que proporciona el interfase de usuario. Si (i) se agota la aplicación de usuario será informada, mediante una primitiva CONNECT.*confirm* (conexión no aceptada), de la situación pues algún error, posiblemente de tipo *hardware*, puede estar teniendo lugar.

Por su parte, la aplicación de usuario que solicita el establecimiento de una sesión-enlace, puede establecer también un tiempo de espera para la recepción de una primitiva CONNECT.*confirm* que le informe sobre su solicitud. Si este tiempo de espera vence o la conexión no es aceptada procederá conforme determine su programación específica.

La reacción detallada de las entidades de sesión-LL a las órdenes recibidas a través de las interfases de otras entidades y a sus sucesos internos, será desarrollada en la especificación formal de este protocolo. El formato de los mensajes transferidos entre las entidades de sesión-LL sería:

Campo de control	Canal	Campo de información

Si *campo de control* = C, R ó S; el campo de información contiene datos para la gestión del protocolo.

Si *campo de control* = D; el campo de información contiene datos del nivel superior.

PROTOCOLOS DE APLICACIÓN.- Dentro del nivel de aplicación de la red debemos especificar los protocolos responsables de que los servicios ofertados por las entidades de aplicaciones específicas, *transferencia de archivos* y *entrada remota de trabajos*, puedan llevarse a cabo.

El formato de los APDU (unidades de datos de protocolos de aplicación) intercambiadas entre las entidades de aplicación, será el mismo para ambas aplicaciones específicas. Constará de un campo de *comandos para el control de la aplicación* (**CCA**) y un campo de *datos para la aplicación* (**DA**).

El **CCA** permitirá la gestión de los protocolos definidos para la transferencia de archivos y la entrada remota de trabajos. Consta de un solo *byte*, que permitirá confeccionar 2^8 comandos distintos para el control de las aplicaciones.

El campo **DA** contiene información específica para las aplicaciones en red, siendo dependiente del contenido del campo **CCA**. El nivel de aplicación era el único que se encargaba de fragmentar la información en la red, adaptando el tamaño de estos fragmentos al campo de datos máximo que admite la subred de comunicaciones en la trama de enlace. Este tamaño máximo es de 256 *bytes* para la subred utilizada, por lo que el campo **DA** será como máximo de 255 *bytes*.

Protocolo de la aplicación *transferencia de archivos*. El servicio proporcionado por la aplicación *transferencia de archivos* debe de ser fiable. El protocolo que regula el intercambio de ficheros entre dos sistemas, debe incorporar un mecanismo que le permita detectar si la información recibida es correcta y un procedimiento para asentir o rechazar los bloque del fichero que se van recibiendo.

Cada vez que se solicite, por parte de la aplicación de usuario, el envío de un fichero a la aplicación que presta este servicio específico, puede enviarse un fichero completo o parte de un fichero. Debe tenerse en cuenta que los sistemas informáticos remotos (SIR) que supervisan las estaciones de campo no tienen porque disponer, en principio, de una gran capacidad de almacenamiento. Un fichero, por ejemplo de 30

kbytes podría estar almacenado, en una estación remota, en un elemento auxiliar del SIR y solo parte de este fichero, por ejemplo 7 *kbytes*, en la memoria del SIR y en disposición de ser enviado.

Estos pormenores serán conocidos por las aplicaciones de control residentes en cada sistema conectado en red. Para la aplicación específica de *transferencia de archivos* será desconocido el hecho de si está enviando un fichero completo o parte del mismo. Las aplicaciones de usuario serán las responsables de controlar estas situaciones y cuando soliciten el envío de información de un fichero a la aplicación *transferencia de archivos*, mediante una primitiva E_FICH.*request*, incluirán como parámetros información específica para que esta aplicación pueda localizar los datos que debe enviar, el destino de estos datos e información complementaria, como:

LON_FICH (longitud total del fichero que se envía).

LON_MEM (longitud de la información que se envía en una determinada solicitud).

La información de LON_FICH no es relevante para la aplicación específica *transferencia de archivos*. Su contenido (2 *bytes*) se incluirá en uno de los bloques de información, según se especificará más adelante, y servirá para el control de las aplicaciones de usuario.

La información de LON_MEM será utilizada por la aplicación específica *transferencia de archivos* para determinar el número de bloques en los que debe dividir la información. Este dato será suministrado a la aplicación de usuario destinataria para que esta tenga conocimiento de que parte de fichero ha recibido y opere en consecuencia.

Ante la recepción de una primitiva E_FICH.*request*, enviada por la aplicación de usuario, con parámetros:

- Estación destino de la información

- Identificativo para localizar la ubicación de los datos a enviar (puede ser una dirección de memoria)
- LON_MEM: longitud de los datos que se deben enviar
- LON_FICH: longitud total del fichero del que forman parte esos datos

La aplicación específica *transferencia de archivos* operará de la siguiente forma:

1º) Con el valor de LON_MEM determinará el número de bloques que debe confeccionar, teniendo en cuenta que en el primero de ellos puede incluir hasta 251 *bytes* del fichero y hasta 255 en el resto. Este dato será almacenado en una variable *número de bloques del fichero* (NBF)

2º) Compondrá el campo de control de la aplicación (CCA), para que se identifique como *datos de fichero* (DAT_FI) por su entidad par.

3º) En los *bytes* 1 y 2 del campo de datos para la aplicación (DA), indicará la longitud de la información que se envía "LON_MEM".

El nº de *bytes* que se envían puede determinarse como:

(valor decimal del byte 2) x 256 + valor decimal del byte 1

Esta operación permite definir hasta 65.535 *bytes*, aunque el número de *bytes* que se puede mandar en cada solicitud, estará acotado por la capacidad de almacenamiento de los SIR.

4º) En los *bytes* 3 y 4 del campo DA indicará la longitud total del fichero que se envía "LON_FICH":

(valor decimal de byte 4) x 256 + valor decimal de byte 3

Esto permite enviar ficheros con una longitud máxima de 65.535 *bytes*, que normalmente es suficiente para los requerimientos de las aplicaciones de control. No obstante si tuvieran que manejarse ficheros de mayor longitud,

siempre podría retocarse el protocolo en este aspecto, añadiendo 1 *byte* más para especificar la longitud del fichero.

Si , LON_MEM = LON_FICH el fichero completo se envía en una sola solicitud.

5º) Seguirá completando el campo DA con los datos secuenciales del fichero que se envía, hasta que: DA = 255 *bytes*.

6º) Todos los datos contenidos en el campo DA, incluidos LON_MEM y LON_FICH, son sometidos a un algoritmo de *secuencia de verificación de bloque* (SVB). El mecanismo de la SVB puede ser sencillo, puesto que los errores de nivel físico ya son detectados y recuperados por la red de comunicaciones. El hecho de incluir en este nivel un procedimiento para detectar errores es una medida de precaución, necesaria para hacer fiable el servicio que presta esta aplicación específica. En nuestro caso se ha optado por confeccionar la SVB con una suma de los *bytes* que van incluyéndose en el campo DA, de la que sólo se almacena el valor de los 8 *bits* de menor peso.

7º) Una vez confeccionado el 1er bloque (256 *bytes* total), se envía como una SDU al nivel de sesión-LL con una primitiva DATA.*request* a la que se acompaña como parámetro el identificativo de la estación destino de la información. El *número de bloque enviados* se irá actualizando en una variable (NBE).

8º) Continuará confeccionando bloques de información mientras (NBE<NBF) e incrementando la variable NBE, cada vez que un bloque es enviado a su nivel inferior para que lo tramite a su entidad *par*. La SVB de cada nuevo bloque irá componiéndose con la resultante del bloque anterior.

9º) Cada cuatro bloques enviados, o un número menor si se trata de los últimos que completan LON_MEM, se envía la SVB con un campo CCA que identifique esta función y un campo DA con el contenido de SVB. Esto permite verificar a su entidad *par* la integridad de la información recibida. La entidad emisora quedará en espera de la recepción de un ACK_FI (información correcta, se

continúa el envío) ó NACK_FI (información errónea, se reexpiden los bloques dañados).

Los comandos para el control de la aplicación ACK_FI y NACK_FI, se codifican en el *byte* del campo CCA. Como estos bloques se conforman únicamente para la gestión del protocolo y no llevan campo DA, tampoco están protegidos por la SVB. Como medida de protección de estos comandos se duplica el campo CCA. El receptor verificará que los comandos de aplicación que no llevan campo DA, están duplicados para entenderlos como correctos.

Si la entidad emisora que espera una confirmación de la información suministrada, recibe una APDU (unidad de datos del protocolo de aplicación) mal conformada, no esperada o no recibe contestación, lo entenderá como un NACK_FI, reexpidiendo los bloques afectados, es decir, aquellos que se enviaron después del último ACK_FI recibido. Esta situación es muy difícil que se produzca, y de hecho, en las pruebas experimentales nunca ha llegado a suceder, pero el protocolo debe prever esta contingencia.

10º)Después de enviar la SVB se activará un tiempo de espera (*Tiempo de espera para una contestación de la entidad par* T_{EC}) . Si T_{EC} vence se incrementará un contador de reintentos C_R reexpidiéndose toda la información remitida desde la última comunicación con la entidad par. El contador C_R será puesto a cero cada vez que se reciba una información de la entidad par. Debe establecerse un límite para C_R de forma que, si este llega a su valor máximo la aplicación TA abortará el proceso de envío del fichero, comunicándoselo a su aplicación de usuario mediante una primitiva ABORT_TA.*indication.*

11º) La entidad que envía el fichero continuará con su operación hasta mandar todos los bloques que componen LON_MEM, siguiendo el mismo procedimiento. Una vez recibido el último ACK_FI devolverá el control de la aplicación a la entidad receptora, con una CCA = (C_MEDIO), para que esta entienda que el proceso ha finalizado y comunique la recepción del fichero a su aplicación de usuario. Esta última determinará, en el caso de haber recibido un fichero

incompleto, si continua solicitando el envío de sucesivas partes o si, por conveniencia de la aplicación de control, realiza otras acciones.

La entidad que envía el fichero podrá conformar los siguientes comandos para el control de esta aplicación específica: DAT_FI, SVB y C_MEDIO. Y podrá recibir: ACK_FI o NACK_FI. Todos ellos codificados en el único *byte* que define el CCA.

La decisión de confirmar la información cada cuatro bloques enviados puede considerarse arbitraria, pero existe una justificación para la misma. El envío de una SVB después de cada bloque relentiza en exceso el proceso, pues debe tenerse en cuenta que gran parte del tiempo consumido en el envío de información se utiliza en la gestión de la comunicación en la subred.

Así el tiempo de ocupación del circuito, para enviar una trama de enlace que contenga 256 *bytes* en el campo de datos, teniendo en cuenta que esta trama se confirma a nivel de AX25 (véase capítulo 3), sería:

$T_o \cong 0{,}37\ s$ para $V_e = 9600\ bps$ y $T_p = 20\ ms$

$T_o \cong 0{,}25\ s$ para $V_c = 19200\ bps$ y $T_p = 20\ ms$

Si esta trama de enlace contiene un campo de datos de 2 *bytes*, como es el caso de las tramas que contienen la SVB, el tiempo de ocupación del circuito sería:

$T'_o \cong 0{,}16\ s$ para $V_c = 9600\ bps$ y $T_p = 20\ ms$

$T'_o \cong 0{,}15\ s$ para $V_c = 19200\ bps$ y $T_p = 20\ ms$

Enviando una SVB después de cada bloque se reduciría la eficacia de la aplicación en un 30% trabajando a 9600 bps y casi en un 40% en el caso de utilizar una velocidad de 19.200 bps; considerando estos porcentajes sobre su máximo rendimiento posible. Los porcentajes aumentarían si debe utilizarse un T_p mayor en el enlace.

El envío de una única SVB al final de cada fichero podría resultar desastroso, pues el error de un solo *bit* provocaría que el fichero completo tuviera que volver a

enviarse. Conociendo la tasa de errores que pueden producirse desde que la información es entregada al nivel físico de un sistema, hasta que el nivel de aplicación puede detectarlos, podría calcularse cual sería el nº de bloques óptimos que pueden enviarse sin confirmación para obtener el mejor rendimiento.

Pero esta tasa de error depende de muchos factores: tipo y calidad del microcontrolador que gestiona el puerto serie, capacidad de proceso del sistema, entorno de programación en el cual están desarrolladas las aplicaciones, etc...

Tampoco se han podido establecer medidas experimentales fiables para aventurar un valor para esta tasa de error. Las únicas veces en los que se ha producido un error de SVB en la recepción de un fichero, enviado con este procedimiento, ha sido trabajando con sistemas como ordenadores personales, con procesador 386 y en entorno *Windows*, produciéndose de forma totalmente aleatoria. El número de tipo de aplicaciones abiertos simultáneamente afecta a posibles errores en la recepción del puerto serie con estos sistemas, pero ha sido imposible cuantificar en que medida.

La elección de enviar una SVB después de, aproximadamente, 1 k*byte* de fichero (1020 *bytes*), es estimativa. Pero esto permite obtener fiabilidad en el servicio prestado por la aplicación específica *transferencia de archivos*, reduciendo en aproximadamente un 10% el rendimiento máximo de la aplicación, si no existieran nunca errores. En el caso de producirse algún error la cantidad de información que debe reexpedirse no es excesiva, como para hacer decrecer su rendimiento de una forma apreciable.

Cuando el nivel de aplicación recibe de su nivel inferior una primitiva DATA.*indication* con parámetro (estación origen de la información), que contenga un campo de control de la aplicación CCA = DAT_FI, será atendida por la aplicación específica de transferencia de archivos que operará de la siguiente forma:

1º) Informará a su aplicación de usuario, mediante una primitiva R_FICH.*indication*, que se encuentra en un proceso de recepción de un fichero remitido por una determinada estación.

2º) Eliminará el campo CCA.

3º) Almacenará los *bytes* 1º y 2º (LON_MEM); y los *bytes* 3º y 4º (LON_FICH), para enviarlos como parámetros a su aplicación de usuario.

4º) Almacenará el resto de la información del campo DA en un buffer temporal, que irá completando con el resto de bloques enviados hasta que el proceso de recibir fichero concluya. Cada vez que recibe un bloque activa el T_{EC} en espera del siguiente. Cuando el siguiente bloque es recibido se desactiva T_{EC} y después de realizar el proceso de almacenamiento del nuevo bloque, se activa de nuevo T_{EC}. Si T_{EC} vence se procede igual que en el caso de la emisión, incrementando C_R. El contador C_R será puesto a cero cada vez que se reciba una información de la entidad par. Si se sobrepasa el número de reintentos sin recibir ninguna información de la entidad par, la aplicación TA abortará el proceso de recepción del fichero comunicándoselo a su aplicación de usuario mediante una primitiva ABORT_TA.*indication.*

Las tres primeras operaciones se realizan, únicamente, con el primer bloque DAT_FI recibido. En el resto de bloques, hasta recibir un CCA = C_MEDIO que le indicará que el proceso ha finalizado, solo eliminará la cabecera de la APDU recibida y almacenará todo el contenido del campo DA en el buffer temporal de forma secuencial.

5º) Someterá el campo DA de cada bloque recibido al algoritmo que verifica su integridad, almacenando la SVB calculada.

6º) Cada vez que reciba una APDU con un CCA = SVB, comprobará si esta coincide con la SVB que tiene almacenada.

- Si coinciden, conformará una APDU con un CCA = ACK_FI y emitirá una primitiva DATA.*request* a su nivel inferior para que envíe esta APDU a su entidad par (parámetro = estación destino de la información)
- Si no coinciden, eliminará del buffer la información recibida desde la última SVB correcta, conformará una APDU con CCA = NACK_FI y

emitirá una primitiva DATA.*request* a su nivel inferior para que la tramite.

7º) Cada vez que envíe un ACK_FI o un NACK_FI, activará el T_{EC} en espera de una contestación de su entidad par, operando de la misma forma que en el punto 4 si T_{EC} vence.

8º) Continuará el proceso hasta recibir una APDU con CCA = C_MEDIO; en cuyo caso, toda la información almacenada en el buffer temporal será enviado a su aplicación de usuario como datos de fichero. Su envío se realizará mediante una primitiva FIN_RFICH.*indication*, a la que se acompañarán como parámetros: estación origen del fichero, LON_MEM y LON_FICH

Si en todo este proceso la entidad receptora recibe un bloque mal conformado, por ejemplo, un campo DA que sobrepase los 255 *bytes*, interpretará "información errónea". Consiguientemente, rechazará la misma enviando un NACK_FI como respuesta al conjunto de bloques donde se encuentra aquel que está mal configurado.

Debe resaltarse que, en el envío de ficheros, no es necesario que se especifiquen más parámetros de los expuestos, para que el usuario receptor del fichero identifique correctamente el mismo. Los ficheros que deben intercambiarse se diseñan por las aplicaciones de control residentes en cada sistema interconectado a la red. Todos ellos se pueden confeccionar con el mismo formato, diseñándose una cabecera donde consten: nombre del fichero, fecha de creación, tipo de fichero, etc. Procediendo de esta forma un fichero recepcionado, aunque haya sido creado por otro sistema, puede ser perfectamente analizado por la aplicación de usuario receptora.

Es más, si estas cabeceras se confeccionan de forma que la longitud total del fichero, figure dentro de la cabecera en una posición determinada, podría prescindirse del parámetro "LON_FICH" que se acompaña a las primitivas que se intercambian entre la aplicación de usuario y la aplicación específica *transferencia de archivos*. Las aplicaciones de usuario, receptores de un fichero, sabrían en que posición del mismo

encontrar este dato para operar con él. Sería recomendable que el dato figurara en los primeros lugares de la cabecera del fichero para disponer con prontitud del mismo.

Protocolo de la aplicación *entrada remota de trabajos* (ERT).- Las entidades de esta aplicación específica deben de poner a disposición de las aplicaciones de usuario un mecanismo que permita a un sistema, conectado en red, solicitar a otro la ejecución de una determinada tarea.

Con el fin de no incrementar el número de aplicaciones específicas en la red, la aplicación ERT se utilizará también para transferir mensajes entre los sistemas conectados en red. Debe tenerse en cuenta que, según las funciones asignadas a estas entidades de aplicación, el servicio de transferencia de datos no confirmados que proporciona a la aplicación de usuario, sería similar al que se demandaría a una entidad que se ocupara de un *servicio de transferencia de mensajes*.

Por otra parte disponemos en los APDU de un campo para el control de las aplicaciones (CCA), en el que se puede codificar hasta 2^8 comandos diferentes. Basta identificar correctamente este campo por parte de la aplicación específica ERT, para que la aplicación de usuario que reciba una información de ella, determine, en función del parámetro obtenido a partir del CAA, si:

- debe realizar un determinado trabajo para entregar el resultado a otro sistema
- o recibe el resultado de un trabajo efectuado por otro sistema
- o un mensaje proporcionado por otro sistema

Si una entidad ERT recibe de la aplicación de usuario una primitiva JOB.*request*, con parámetros:

- Estación destino de la información "ED"

- Identificación de la función "IF" que se desea transmitir a la aplicación de usuario de la estación destino. Esta puede ser:
  - solicitud para que efectúe un determinado trabajo
  - o la respuesta a un trabajo efectuado
  - o el envío de un mensaje (generalmente avisos o alertas).

Y una SDU (unidad de datos de servicio), conteniendo:

- información complementaria para que el trabajo se efectúe
- o información resultante de un trabajo efectuado
- o el texto de un mensaje enviado.

(el contenido de esta SDU depende directamente del parámetro "IF" proporcionado)

La entidad ERT operará de la siguiente forma:

1º) Con el parámetro "IF" compondrá el campo CCA de la aplicación específica ERT. Para ello cada sistema conectado en red deberá disponer de una tabla de equivalencias, común, entre los identificativos de las posibles funciones que se pueden realizar en red y los comandos de la aplicación ERT asociados.

2º) Con la SDU proporcionada compondrá el campo DA de la APDU. No es necesario que las entidades responsables de este servicio verifiquen que el contenido del campo DA no supere los 255 *bytes*, pues los SDU proporcionados por la aplicación de usuario no sobrepasarán este valor. Sólo en el caso que la ejecución de un trabajo proporcione como resultado la confección de un fichero se puede superar el tamaño máximo de los fragmentos, pero en este caso, la aplicación de usuario llamaría a la aplicación específica *transferencia de archivos* para enviar la respuesta.

3º) Conformada la APDU (unidad de datos de protocolo de la aplicación), enviaría esta como una SDU (unidad de datos de servicio) al nivel de sesión-

LL, mediante una primitiva DATA.*request* (parámetro = estación destino), para que la envíe a su entidad par.

El servicio de transferencia de datos que proporcionan las entidades ERT es no confirmado. Si una aplicación de usuario ha solicitado la ejecución de un trabajo, a otro sistema de la red, corresponde a esta determinar si espera una contestación inmediata o continúa con otro tipo de operaciones. La entidad ERT se limita a tramitar la solicitud del nivel superior, a través de la red, para entregar esta a la aplicación de usuario destinataria.

Como en el caso de la aplicación específica TA, si la aplicación ERT confecciona un campo de control de la aplicación CCA, que contienen un campo DA vacío, se duplicará el campo CCA en la APDU como medida de protección de estos comandos.

Cuando el nivel de aplicación recibe una primitiva DATA.*indication* (parámetro = estación origen) del nivel de sesión-LL, identifica el CCA de la APDU proporcionada y, si el contenido de esta no corresponde a un comando que debe atender la aplicación TA, será siempre atendida por la aplicación específica ERT. Esta la enviará a su aplicación de usuario con el siguiente procedimiento:

1º) Utilizará el contenido del campo CCA y la tabla de funciones asociadas, para determinar el parámetro "IF" que debe enviar a la aplicación de usuario. Si el contenido del campo CCA no estuviera en dicha tabla le enviará un parámetro "IF" que la aplicación de usuario identificará como función desconocida.

2º) Enviará una primitiva JOB.*indication* a su aplicación de usuario, con parámetros (estación origen, IF) y una SDU que compondrá con el campo DA recibido.

Tanto si "IF" es desconocido para la aplicación de usuario, como si la SDU no es correcta, por ejemplo, no lleva la duplicidad de "IF" como mínimo; será la

aplicación de usuario la que determine si solicita que la última información suministrada se repita, porque entiende que ha podido existir un error, o toma otra determinación. Para solicitar la reexpedición de una información la aplicación de usuario demandará el servicio a la aplicación ERT, con un identificativo "IF" que permita reconocer a la aplicación de usuario destinataria que debe ejecutar esta función.

4.5.2 ESPECIFICACIÓN FORMAL DE LOS PROTOCOLOS

Para realizar la especificación formal del protocolo de un determinado nivel pueden utilizarse diversas técnicas. Estas técnicas se clasifican en tres categorías: *modelos de transición, lenguajes de programación* y *modelos mixtos* (estos últimos mezclan las dos técnicas anteriores) [Beltrào_89].

Los modelos de transición proporcionan una visión más sencilla e ilustrativa de la operatividad del protocolo. No obstante, la especificación de un protocolo utilizando únicamente modelos de transición no siempre es viable, especialmente si este presenta cierta complejidad. En este caso debe recurrirse a modelos que utilizan lenguajes de programación.

Dentro de las técnicas que utilizan modelos de transición pueden usarse: gráficos de UCLA [Postel_74], redes de Petri [Merlin_76], lenguajes formales [Teng_78] y modelos de máquinas de estados finitos (MEF) [Danthine_80], entre los más usuales.

Los métodos más utilizados son las redes de Petri y las MEF. En [Danthine_80] se afirma que, probablemente, las redes de Petri son más indicadas en la etapa inicial de especificación de un protocolo. Esto se debe a que están más próximas a un lenguaje natural y facilitan la identificación de los distintos estados por los que transcurre un protocolo, en respuesta a las órdenes recibidas a través de sus interfases y a la aparición de sucesos internos (por ejemplo vencimiento de un temporizador). No obstante, según

Danthine, el formalismo que se obtiene mediante MEF es más compacto y permite una implementación más cómoda del protocolo así especificado.

Dado que las operaciones realizadas por el protocolo de un determinado nivel implican, necesariamente, una comunicación real con los niveles adyacentes a él, se necesita utilizar un modelo de autómata finito con salida. Una máquina de estados finitos con salida se define formalmente [Booch_83], como:

$$(Q, \Sigma, \Delta, \delta, \lambda, Q_0)$$

donde:

- Q es un conjunto finito de estados.
- Σ es un conjunto finito de posibles entradas (alfabeto de entrada).
- Δ es un conjunto finito de posibles salidas (alfabeto de salida).
- δ es la función de transición de estado.
- λ es la función de salida.
- $Q_0 \in Q$ es el estado inicial.

La función de transición de estados asigna a la pareja (entrada, estado) actuales, el próximo estado.

$$\delta : \Sigma \, x \, Q \rightarrow Q$$

Una MEF se encuentra en un *estado* o en transición hacia el *próximo estado*. Cada entrada provoca una transición, donde el *próximo estado* vienen determinado por la función δ. La salida suministrada por la MEF viene determinada por la función λ, y existen dos alternativas a la hora de formalizar la operación de salida que se produce en un cambio de estado. Así, la salida puede asociarse a la entrada en el próximo estado (máquina de Moore), o bien puede asociarse a la transición entre estados (máquina de Mealy) [Hopcroft_79].

La función de salida para la máquina de Moore es:

$$\lambda : Q \rightarrow \Delta$$

La función de salida para la máquina de Mealy es:

$$\lambda : \Sigma \, x \, Q \rightarrow \Delta$$

De la misma forma que una MEF, un protocolo (de nivel *N*) comienza en un estado inicial Q_0'. Permanece en un *estado* Q' hasta que se produzca una condición *C* (recepción de una primitiva a través de uno de sus interfases, vencimiento de algún temporizador interno, etc.). El protocolo responde a la condición realizando algún tipo de proceso (que se puede modelar por la transición de una MEF), que produce una salida (determinada por una función de salida λ') y cambia a otro estado (determinado por una función de transición de estado δ'). Las salidas (solicitud de un servicio al nivel *N-1*, indicación de un servicio al nivel *N+1*, inicialización de algún temporizador, etc.) que se producen a raíz de las operaciones realizadas por el protocolo, son consecuencia de una entrada en un estado específico.

La forma de operar de un protocolo y una máquina de Mealy son equivalentes. Las salidas son consecuencia de la pareja actual (estado, condición) y se producen durante la transición entre estados. Consiguientemente la construcción de una máquina de Mealy correspondiente a un protocolo (identificando $Q_0=Q_0'$, $Q=Q'$, $\Sigma=C$, $\Delta=S$, $\delta=\delta'$ y $\lambda=\lambda'$), es una manera de especificar formalmente dicho protocolo.

Debe tenerse en cuenta que una entidad de nivel puede actuar como emisora o receptora de la información. Si el protocolo definido para la entidad es muy sofisticado, identificándose muchos estados diferentes en la parte emisora y receptora, se puede producir lo que se conoce como *explosión del espacio de estados* (el conjunto finito de estados de la MEF será el producto cartesiano de los posibles estados de la parte emisora y de la parte receptora). En estos casos la especificación del protocolo mediante MEF puede resultar difícil y engorrosa.

Para reducir el grado de dificultad originado por la posible explosión del número de estados, sin necesidad de recurrir a otras técnicas de especificación que utilizan lenguajes de programación, existen algunos métodos [Bochmann_80]. Pero las simplificaciones que conllevan los métodos propuestos en [Bochmann_80] tienen como consecuencia una especificación parcial del protocolo, exponiendo sólamente ciertos aspectos del mismo con respecto a las transiciones entre estados principales. Estos métodos no serán utilizados en la memoria ya que interesa que los protocolos propuestos queden completamente especificados.

Si la complejidad de un protocolo implica que su especificación completa mediante una MEF resulta poco clara por la abundancia de estados, se utilizará un modelo mixto para su especificación formal.

En un modelo de transición puro las variables de estado condicionan la aparición de nuevos estados. Por el contrario, en un modelo mixto se define una MEF con pocos estados, y se amplia con variables adicionales (V) y rutinas de procesamiento de estas variables (máquina de estados finitos ampliada, MEFA). Esto se lleva a cabo asociando a cada transición de la MEF un proceso, que puede probar y manipular las variables adicionales sin que estas tengan que ser incluidas en la definición de los estados de la MEF. De esta forma se reduce la complejidad de la MEF [Beltrào_89].

Este método puede presentar un problema. Una transición entre estados se produce cuando ocurre un suceso, pero este suceso debe incluir, en general, valores para las variables definidas para el protocolo. Si queremos especificar todas las situaciones posibles, en función del valor que toman cada una de las variables, podemos tener un excesivo número de elementos para el alfabeto de entrada (Σ).

Por otra parte, la definición de una MEFA implica que los procesos que tienen lugar en las transiciones de la MEF (ejecución de las funciones δ y λ), deben tener en cuenta el valor de las variables del protocolo no incluidas en la definición de la MEF, además del valor de la pareja (estado, entrada) actuales para proporcionar su resultado.

Una MEFA no se comporta como un autómata, por lo que presenta mayores dificultades para su implementación.

Para dar respuesta a estas dos situaciones se propone el siguiente procedimiento de especificación formal del protocolo:

- Modelizar los aspectos de control del protocolo (nucleo del protocolo) con una MEF conforme a su definición formal. Concretamente se utilizará una MM por su manera de formalizar la operación de salida. La MEF no tendrá en cuenta el valor de las variables del protocolo (V), no incluidas en su definición, para determinar su nuevo estado y su salida; aunque los procesos ejecutados durante las transiciones de la MEF podrán actualizar el valor de dichas variables en función de la entrada y estado actuales.

- Para considerar el estado de las variables (V) se definen dos procesos independientes de la MEF:

  1.- *Proceso generador de entradas* PGE. Determina el alfabeto de entrada (Σ) de la MEF que modela el núcleo del protocolo, en función de las condiciones del protocolo (C) y el estado del conjunto de variables (V).

$$PGE : C \, x \, V \rightarrow \Sigma$$

  2.- *Proceso generador de salidas* PGS. Determina el conjunto finito de salidas del protocolo (S), en función de las salidas de la MEF (Δ) y el estado del conjunto de variables (V).

$$PGS : \Delta \, x \, V \rightarrow S$$

Así, el protocolo quedaría definido formalmente, por:

$$(Q, C, \Sigma, \Delta, S, V, PGE, \delta, \lambda, PGS, Q_0)$$

En la figura 4.12 puede verse un esquema de la modelización de un protocolo mediante este procedimiento. Las variables del protocolo (*V*) serán manipuladas por los procesos que se ejecutan durante las transiciones de la MEF pero, únicamente, para actualizar su valor. El estado de estas variables será probado por el PGE y/o el PGS para tomar sus decisiones.

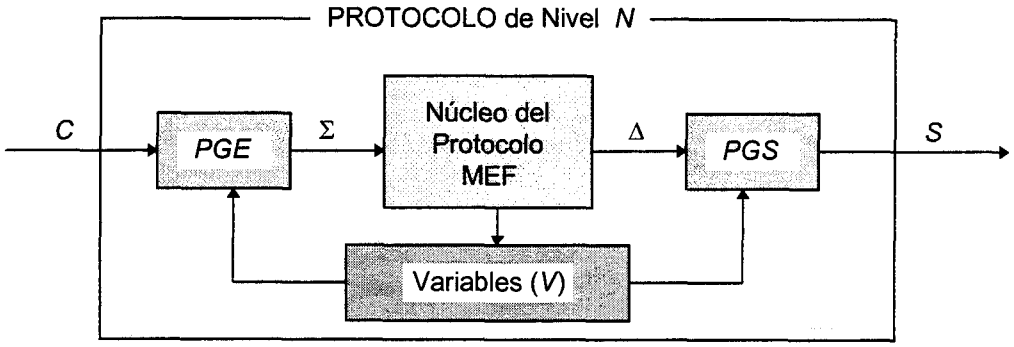

Figura 4.12 Modelización del protocolo de nivel *N*

El *proceso generador de entradas* atenderá todas las condiciones relevantes para el protocolo *C*: condiciones externas (recepción de primitivas) y condiciones internas (vencimiento de temporizadores), evaluará estas condiciones en función del estado de las variables del protocolo y generará las entradas a la MEF.

Una de las grandes ventajas de utilizar el PGE, además de no necesitar una MEFA para la especificación del protocolo, es que permite simplificar la MEF que modela su núcleo. Esto es así por dos razones:

1. Al reducirse el conjunto de entradas (Σ) se pueden combinar dos o más transiciones de la MEF en una sola. Por ejemplo, la condición "*recepción de una primitiva* DATA.*indication en la que se informa que la última secuencia de datos enviada es errónea*" y la condición "*vencimiento de un temporizador por ausencia de respuesta (sin el número de reintentos vencido)*", son tratados previamente por el PGE, suministrando la misma entrada a la MEF "*repetir envío*". Esto implica un sola transición de la MEF para dos condiciones distintas del protocolo

2.- El estado de determinadas variables del protocolo es controlado únicamente por PGE para tomar decisiones. Esto permite reducir el número de estados de la MEF que modela su núcleo. Por ejemplo, un contador de reintentos C_R puede ser gestionado por el PGE, el cual generará una única entrada "*C_R vencido*" cuando se agoten los reintentos. La única actuación de los procesos ejecutados durante las transiciones de la MEF sobre esta variable será para su actualización: *puesta a cero* $C_R \leftarrow 0$ o *incremento* $C_R \leftarrow C_R + 1$, cuando el estado y entrada actuales así lo requieran. Esto permite simplificar la MEF que modela el núcleo del protocolo, de manera que se puede definir para la misma un estado "*espera respuesta*", en lugar los que corresponderían si se tuviera en cuenta el estado del contador "*espera respuesta con $C_R = 0$*", "*espera respuesta con $C_R = 1$*", "*espera respuesta con $C_R = 2$*", etc.

La existencia del PGS permite que el valor otras variables del protocolo sean tenidas en cuenta para determinar la salida del mismo y que este valor sea ignorado por la MEF para proporcionar su salida. Por ejemplo, el número de bloques pendientes de enviar (*NBF*) en el envío de un fichero, necesariamente ha de ser tenido en cuenta para generar las salidas del protocolo (por ejemplo, no es lo mismo enviar el bloque 3 al nivel inferior, que enviar el bloque 5). Con el PGS se puede definir una única salida para la MEF cuando esta envía información de un fichero al nivel inferior "DATA.*request* con una SDU". El contenido de la SDU (nº de bloque que se envía) se determinará por el PGS en función del estado de la variable *NBF*. Esta variable únicamente será actualizada por los procesos ejecutados durante las transiciones de la MEF, en función de las distintas entradas y del estado en que se encuentre.

Para aportar uniformidad se utilizará este procedimiento para especificar todos los protocolos de nivel definidos, aunque en el caso de protocolos sencillos se cumplirá que el conjunto de variables no incluido en la definición de estados de la MEF será vacío. Consiguientemente, $C=\Sigma$, $S=\Delta$ y los procesos PGE y PGS no será necesario determinarlos.

Conceptualmente se han independizado los procesos que tienen lugar en las transiciones de la MEF y los procesos PGE y PGS, en función del tipo de actuación de los mismos sobre las variables del protocolo no incluidas en la definición de la MEF. Sin embargo, con el fin de facilitar la tarea de implementar un protocolo, conviene asociar a cada salida de la MEF la salida correspondiente del protocolo proporcionada por el PGS. Por este motivo se define una *"**acción**"* (*A*) para cada transición de la MEF que incluirá:

- El proceso ejecutado por la MEF para proporcionar un nuevo estado y una salida (ejecución de las funciones δ y λ).

$$\delta : \Sigma \, x \, Q \rightarrow Q$$

$$\lambda : \Sigma \, x \, Q \rightarrow \Delta$$

- La parte del PGS que se encarga de tratar la salida de la MEF para proporcionar la salida del protocolo.

$$PGS : \Delta \, x \, V \rightarrow S$$

Así,

$$A : \Sigma \, x \, Q \, x \, V \rightarrow Q \, , \, S$$

De esta forma el PGS se encontrará distribuido entre las acciones definidas para el protocolo y, en realidad, solo será necesaria su implementación cuando el estado de una determinada variable condicione el contenido específico de la salida de la MEF.

Para formalizar la especificación de cada protocolo deben definirse (Q, C, Σ, V, Δ, S y Q_0) y especificarse las funciones (*PGE*, δ , λ y *PGS*) mediante un programa (con la simplificación realizada, *PGE* y *A*). La forma de llevar esto a cabo será la siguiente:

1. Se diseñará el diagrama de estados del autómata que se encarga de modelar el protocolo o su núcleo. En su ilustración se utilizará la nomenclatura típica para describir MEF [Booch_83]:

Un estado se representará mediante un círculo. Una transición entre estados será una flecha que une los dos estados (*origen y destino*). La transición llevará asociada la entrada que la provoca (Σ_i) y la salida que produce (Δ_i), en la forma: Σ_i/Δ_i. El estado inicial se representará por un círculo al que le llega una transición sin origen y sin etiqueta.

2.- Se acompañará una tabla de: estados, entradas a la MEF, numeración de las acciones del protocolo, nuevos estados y salidas del protocolo, que facilitará la comprensión de su funcionamiento, ampliando detalles de su operatividad.

3.- Se especificarán en un programa el PGE y las acciones A_i numeradas en la tabla anterior.

En realidad bastaría con los puntos 1 y 3 para llevar a cabo la especificación. La tabla que se confecciona en el punto 2 se realiza con el ánimo de recoger los aspectos del protocolo que aportan la información de más relieve para comprender su forma de operar y tener recogidos estos datos de una forma simple y clara.

Para no condicionar la implementación en un lenguaje de programación determinado, se utilizará pseudocódigo para definir las operaciones que se realizan en los distintos procesos.

PROTOCOLO DE NIVEL FÍSICO. Debido a las características de los dispositivos *hardware* que controlan los puertos serie, capaces de atender simultáneamente la recepción y envío de *bytes* al medio físico, especificaremos formalmente el protocolo de nivel físico mediante dos MM. Una estaría dedicada al envío (figura 4.13) y otra a la recepción (figura 4.14).

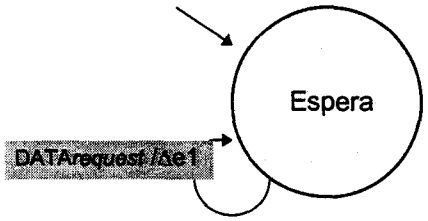

Figura 4.13 Diagrama de estados de la MEF del nivel físico. Entidad emisora

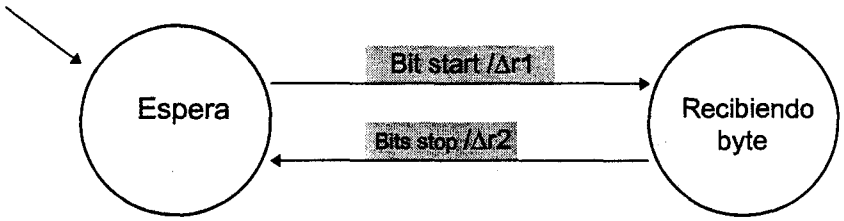

Figura 4.14 Diagrama de estados de la MEF del nivel físico. Entidad receptora

En este caso la salida de la MM coincide con la salida del protocolo. Los estados, entradas a la MM, acciones del protocolo, nuevos estados y salidas del protocolo pueden verse en la tabla 4.2 y 4.3.

PROTOCOLO DE NIVEL FÍSICO (Emisor)				
Estado	**Entrada**	**Acción**	**Nuevo estado**	**Salida**
Espera	Σe1: DATA.*request*	Ae1	Espera	Se1: DATA.*request*

Tabla 4.2

PROTOCOLO DE NIVEL FÍSICO (Receptor)				
Estado	**Entrada**	**Acción**	**Nuevo estado**	**Salida del protocolo**
Espera	Σr1: Bit start	Ar1	Recibiendo byte	Sr1: ∅
Recibiendo byte	Σr2: Bits stop	A r2	Espera	Sr2: DATA.*indication*

Tabla 4.3

Especificación de las acciones :

Ar1. Salida = ∅

Nuevo estado = Recibiendo byte

Ar2. Recuperar bits de datos del byte recibido

[considerando características configuración puerto serie]

[bit start, bit datos, bit paridad (si existe), bits de stop (1, 1,5 ó 2)]

Salida = Enviar DATA.*indication* al nivel NCA con bits de datos

Nuevo estado = Espera byte

Ae1. Encapsular byte recibido con características configuración puerto serie:

[bit start, bit datos, bit paridad (si existe), bits de stop (1, 1,5 ó 2)]

Salida = Enviar DATA.*request* al medio físico con byte encapsulado

Nuevo estado = Espera

PROTOCOLO DE ACCESO A LA SUBRED DE COMUNICACIONES (Protocolo NCA). La MM que modela este protocolo se compone de un único estado, figura 4.15.

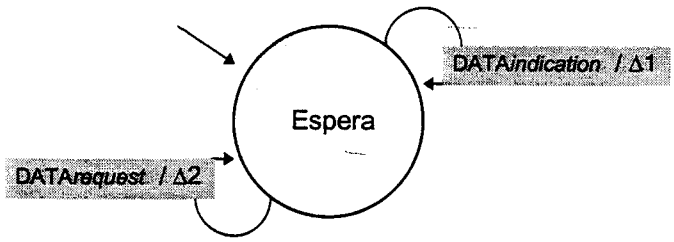

Figura 4.15 Diagrama de estados de la MEF del nivel NCA

Las salida de la MM coincide con la salida del protocolo. Los estados, entradas a la MM, acciones del protocolo, nuevos estados y salidas del protocolo pueden verse en la tabla 4.4.

PROTOCOLO DE NIVEL NCA				
Estado	**Entrada**	**Acción**	**Nuevo estado**	**Salida del protocolo**
Espera	Σ1: DATA.*indication*	A1	Espera	S1: DATA.*indication*
	Σ2: DATA.*request*	A2	Espera	S2: Data.*request*

Tabla 4.4

Especificación de las acciones:

A1. Leer Serie

Sincronizar trama *[determinar principio de trama por carácter FEND]*

Conformar cabecera de la trama de enlace *[utilizando los tres siguientes bytes leídos]*

Conformar campo de datos de la trama de enlace quitando la transparencia

[utilizando los siguientes bytes leídos hasta nuevo FEND]

Determinar final trama *[por siguiente caracter FEND]*

Salida = Enviar DATA.*indication* al nivel Sesión-LL

[parámetros (CC, canal); SDU = campo de datos de la trama de enlace]

Nuevo estado = Espera

A2. Poner FEND *principio de trama*

Conformar cabecera de la trama de enlace

[con parámetros (CC, canal) recibidos y puerto (dato conocido por el nivel NCA)]

Conformar campo de datos de la trama de enlace aplicando la transparencia

[con la SDU recibida]

Poner FEND *final de trama*

Salida = Enviar DATA.*request* al nivel Físico *[SDU = bytes de la trama de enlace]*

Nuevo estado = Espera

PROTOCOLO DEL NIVEL DE SESIÓN - ENLACE LÓGICO. La especificación de este protocolo utilizando únicamente una MM resultaría dificultosa. Debe tenerse en cuenta que existe una variable (i = intentos de conexión del enlace), que implica la aparición de numerosos estados.

Así, modelaremos el protocolo con una MM ampliada con una variable i y un *proceso generador de entradas* (PGE), que valorará el estado de esta variable para tomar sus decisiones. La variable será actualizada por alguno de los procesos que tienen lugar durante las transiciones de la MM. En este caso no es necesario diseñar un *proceso generador de salidas* (PGS) de la MM, para definir las salidas del protocolo, ya que no existe ninguna variable que pueda condicionar estas últimas ($S = \Delta$).

Para simplificar la comprensión del modo de operar de este protocolo diseñaremos una MM para el sistema emisor de una solicitud de sesión y otro para el receptor. Así, el conjunto finito de posibles parejas *(estado, entrada)* de la MM que modela el protocolo completo será el producto cartesiano de las parejas del emisor y receptor. Todas las posibles combinaciones *(estado, entrada)* quedan englobadas en las tablas de estados, entradas a la MM, acciones del protocolo, nuevos estados y salidas del protocolo, confeccionadas para el sistema emisor y receptor. En el estado inicial de ambas tablas se han previsto todas las combinaciones válidas.

Este protocolo maneja dos temporizadores T_C (tiempo de espera de conexión del enlace) y T_{AC} (Tiempo para nuevo envío de una solicitud de conexión). Al estado inicial se accede con ambos temporizadores desactivados.

ESPECIFICACIÓN DEL PGE.

Se definen i_{max} (máximo valor de intentos para establecer un enlace lógico entre dos nodos de la red) como una constante

El PGE podrá evaluar el estado de la variable:

(i , intentos de conexión del enlace)

Esta variable es actualizada por los procesos que tienen lugar en la transición entre estados de la MM.

PGE:

1. Espera C
   C = T_C vencido
   - Si **i** > i_{max} — Σ = **i** vencido, Ir a 1
   - Si **i** ≤ i_{max} — Σ = T_C vencido, Ir a 1
   - Resto de condiciones — Σ = C, Ir a 1

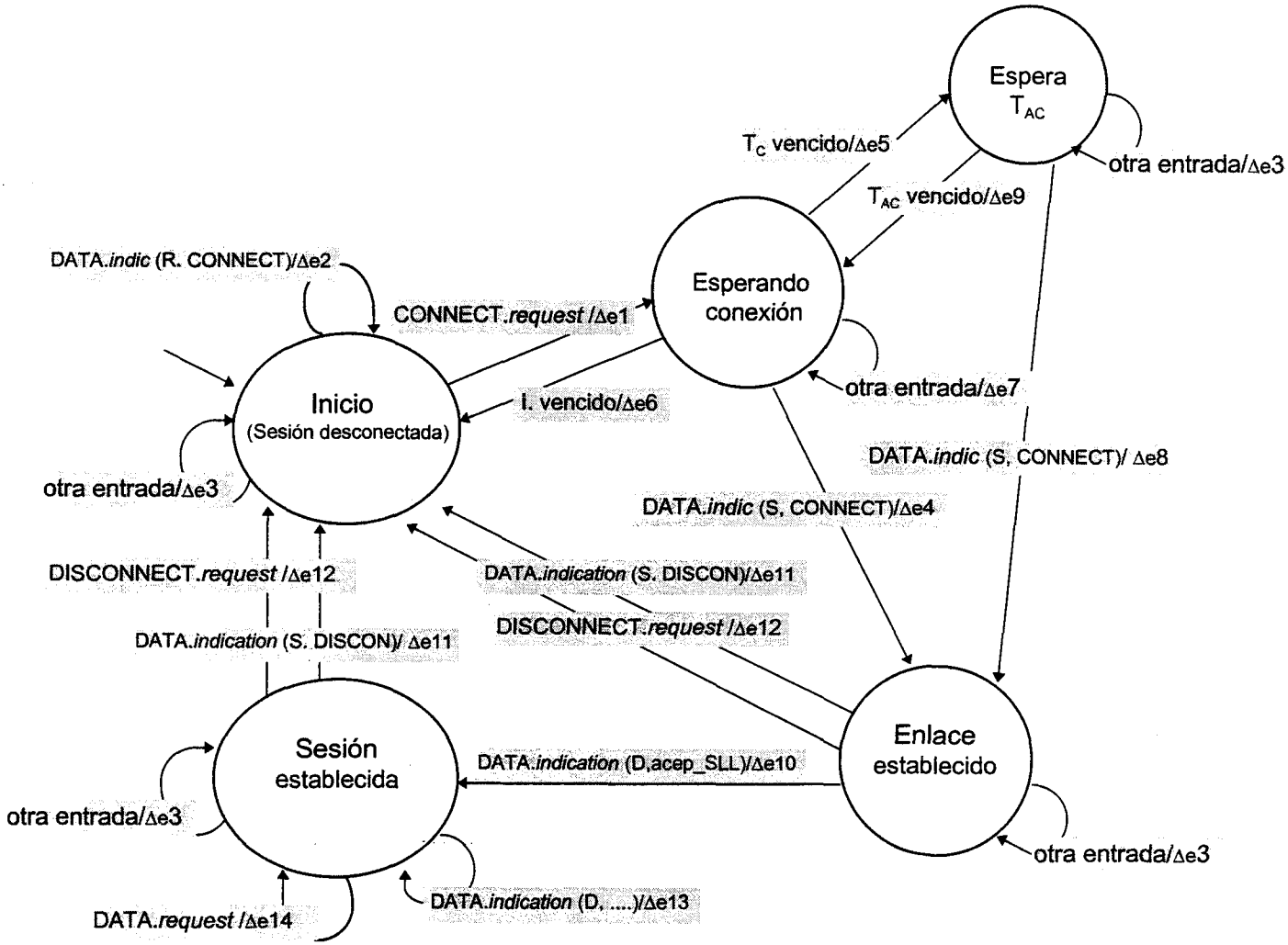

Figura 4.16 Diagrama de estados de la MEF del nivel Sesión-Enlace Lógico

Sistema emisor de la solicitud de Sesión-LL

La figura 4.16 muestra el diagrama de estados de la MM que modela el núcleo del protocolo de Sesión-LL para el sistema emisor. La tabla 4.5 recoge sus estados, entradas a la MM, acciones del protocolo, nuevos estados y salidas del protocolo.

PROTOCOLO *SESIÓN - ENLACE LÓGICO*. ENTIDAD QUE SOLICITA LA SESIÓN				
Estado	**Entrada**	**Acción**	**Nuevo estado**	**Salida del protocolo**
Inicio	Σe1: CONNECT.*request*	Ae1	Esperando conexión	Se1: i. , activa T_C DATA.*request* (C,CONNECT)
	Σe2 DATA.*indic* (S, CONNECT)	La entidad actúa como receptora de una sesión (*véase tabla correspondiente)*		
	Σe3: DATA.*indic* (R, CONNECT)	Ae2	Inicio	Se2: CONNET.*indication* (sin enlace)
	Otra entrada	Ae3	Inicio	Se3: ∅
Esperando conexión	Σe4: DATA.*indication* (S, CONNECT)	Ae4	Enlace establecido	Se4: Desactiva T_C
	Σe5: T_C vencido	Ae5	Espera T_{AC}	Se5: i. activa T_{AC}
	Σe6: i. vencido	Ae6	Inicio	Se6: CONNET.*confirm* (NO sesión-LL)
	Otra entrada	Ae7	Esperando conexión	Se7: i. activa T_C
Espera T_{AC}	Σe4: DATA.*indication* (S, CONNECT)	Ae8	Enlace establecido	Se8: Desactiva T_{AC}
	Σe7: T_{AC} vencido	Ae9	Esperando conexión	Se9: DATA.*request* (C,CONNECT) activa T_C
	Otra entrada	Ae10	Espera T_{AC}	Se3: ∅
Enlace establecido	Σe8: DATA.*indic* (D,acep_SLL)	Ae11	Sesión establecida	Se10: CONNECT.*confirm* (SI sesión-LL)
	Σe9: DATA.*indic* (S, DISCON)	Ae12	Inicio	Se11: DISCONNET.*indication*
	Σe10: DISCON.*request*	Ae13	Inicio	Se12: DATA. *request* (C, DISCON)
	Otra entrada	Ae14	Enlace establecido	Se3: ∅
Sesión establecida	Σe11: DATA.*indic* (D, ...)	Ae15	Sesión establecida	Se13: DATA.*indication*
	Σe12: DATA.*request*	Ae16	Sesión establecida	Se14: DATA.*request* (D, ...)
	Σe9 : DATA.*indic* (S, DISCON)	Ae12	Inicio	Se11: DISCONNECT.*indication*
	Σe10: DISCON.*request*	Ae13	Inicio	Se12: DATA. *request* (C, DISCON)
	Otra entrada	Ae17	Sesión establecida	Se3: ∅

Tabla 4.5

ESPECIFICACIÓN DE LAS ACCIONES. Acciones que se ejecutan durante las transiciones de la MM que modela el núcleo del protocolo de Sesion-LL.(S.emisor)

Variable del protocolo que manipulan los procesos:

(*i, intentos de conexión del enlace*)
(T_{AC} , *Tiempo para enviar una nueva solicitud de enlace)*

Se definen como constantes:

(m = prioridad de la estación (m = 1, 2, 3 ...), específico para cada estación de la red)
(T_R = Tiempo de referencia) véase apartado 4.5.1
(T_C = Tiempo espera de conexión del enlace)

Ae1. i = 0

Enviar DATA.*request* (C, canal, CONNECT) al nivel NCA

Activar T_C

Nuevo estado = Esperando conexión

Ae2. Enviar CONNECT.*indication* (sin enlace) al nivel de aplicación.

(*La aplicación de usuario interpretará esta primitiva como: Un sistema le solicita una sesión estando sus canales de enlace ocupados. Cuando la aplicación de usuario lo determine, será esta quien solicite una sesión al sistema que le ha proporcionado el aviso)*

Nuevo estado = Inicio

Ae3. Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:

Fecha, Hora, Nivel, Estado, Entrada

Nuevo estado = Inicio

Ae4. Desactivar T_C

Nuevo estado = Enlace establecido

Ae5. i = i + 1

Determinar T_{AC} ; $T_{AC} = (2^i - 1)\, m\, T_R$

Activar T_{AC}

Nuevo estado = Espera T_{AC}

Ae6. Enviar CONNECT.*confirm* (No Sesión-LL) al nivel de aplicación (*se informa acerca de la imposibilidad de establecer una sesión con el sistema destino*)

Nuevo estado = Inicio

Ae7. Desactiva T_C

i = i + 1

Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:

Fecha, Hora, Nivel, Estado, Entrada

Activar T_C

Nuevo estado = Esperando conexión

Ae8. Desactivar T_{AC}

Nuevo estado = Enlace establecido

Ae9. Enviar DATA.*request* (C, canal, CONNECT) al nivel NCA

Activar T_C

Nuevo estado = Esperando conexión

Ae10. Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:

Fecha, Hora, Nivel, Estado, Entrada

Nuevo estado = Espera T_{AC}

Ae11. Enviar CONNECT.*confirm* (Si Sesión-LL) al nivel de aplicación

Nuevo estado = Sesión establecida

Ae12. Enviar DISCONNECT.*indication* al nivel de aplicación

Nuevo estado = Inicio

Ae13. Enviar DATA.*request* (C, canal, DISCON) al nivel NCA

Nuevo estado = Inicio

Ae14. Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:

Fecha, Hora, Nivel, Estado, Entrada

Nuevo estado = Enlace establecido

Ae15. Enviar DATA.*indication* (CCA, DA) al nivel de aplicación

Nuevo estado = Sesión establecida

Ae16. Enviar DATA.*request* (D, canal, CD) al nivel NCA

Nuevo estado = Sesión establecida

Ae17. Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:

Fecha, Hora, Nivel, Estado, Entrada

Nuevo estado = Sesión establecida

La figura 4.17 muestra el diagrama de estados de la MM que modela el núcleo del protocolo de Sesión-LL para el sistema receptor. La tabla 4.6 recoge sus estados, entradas a la MM, acciones del protocolo, nuevos estados y salidas del protocolo.

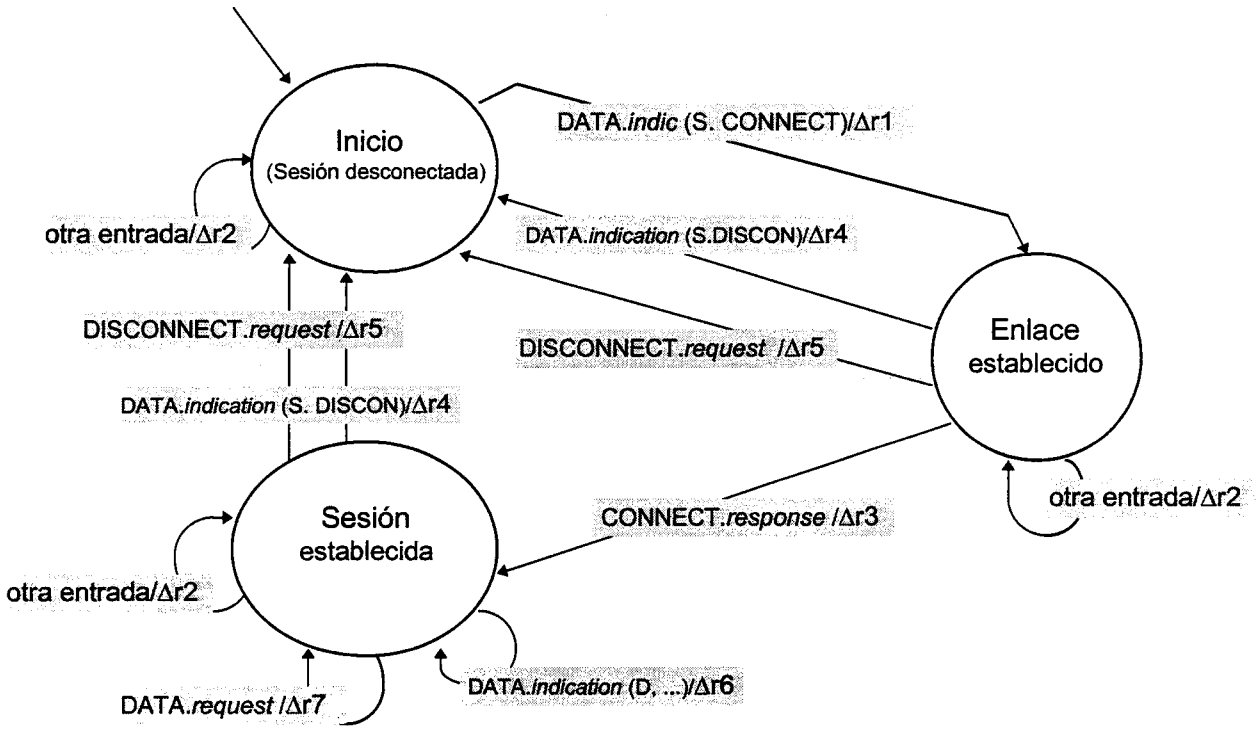

Figura 4.17 Diagrama de estados de la MEF del nivel Sesión-Enlace Lógico.
Sistema receptor de la solicitud de sesión

PROTOCOLO *SESIÓN - ENLACE LÓGICO*. ENTIDAD RECEPTORA DE LA SOLICITUD DE SESIÓN				
Estado	**Entrada**	**Acción**	**Nuevo estado**	**Salida del protocolo**
Inicio	Σr1: CONNECT.*request*	La entidad actúa como emisora de una solicitud de sesión (*véase tabla correspondiente)*		
	Σr2 DATA.*indication* (S, CONNECT)	Ar1	Enlace establecido	Sr1 CONNET.*indication*
	Σr3: DATA.*indication* (R, CONNECT)	La entidad actúa como emisora de una solicitud de sesión (*véase tabla correspondiente)*		
	Otra entrada	Ar2	Inicio	Sr2: ∅
Enlace establecido	Σr4: CONNECT.*respons*	Ar3	Sesión establecida	Sr3: DATA.*request* (D,acep_SLL)
	Σr5: DATA.*indication* (S, DISCON)	Ar4	Inicio	Sr4: DISCONNET.*indication*
	Σr6: DISCONNECT. *request*	Ar5	Inicio	Sr5: DATA. *request* (C, DISCON)
	Otra entrada	Ar6	Enlace establecido	Sr2: ∅
Sesión establecida	Σr7: DATA.*indication* (D, ...)	Ar7	Sesión establecida	Sr6: DATA.*indication*
	Σr8: DATA.*request*	Ar8	Sesión establecida	Sr7: DATA.*request* (D, ...)
	Σr5: DATA.*indication* (S, DISCON)	Ar4	Inicio	Sr4: DISCONNET.*indication*
	Σr6: DISCONNECT. *request*	Ar5	Inicio	Sr5: DATA. *request* (C, DISCON)
	Otra entrada	Ar9	Sesión establecida	Sr2: ∅

Tabla 4.6

ESPECIFICACIÓN DE LAS ACCIONES. Acciones que se ejecutan durante las transiciones de la MM que modela el núcleo del protocolo de Sesion-LL. (S. receptor).

Cuando la entidad del protocolo Sesión-LL actúa como receptora de una sesión no maneja las variables i y T_{AC}. Al recibir una primitiva DATA.*indication* (C, canal, Connect), directamente pasa al estado *Enlace establecido*. Este último estado es el mismo tanto si el sistema emite la solicitud de sesión, como si la recibe.

Las únicas acciones que no están contempladas en la tabla 4.5 y aparecen en la tabla 4.6, serían:

Ar1. Enviar CONNECT.*indication* al nivel de aplicación
Nuevo estado = Enlace establecido S. Receptor

Ar3. Enviar DATA.*request* (D,acep_SLL) al nivel NCA
Nuevo estado = Sesión establecida

Resto de acciones:

Ar2 = Ae3, Ar4 = Ae12, Ar5 = Ae13, Ar6 = Ae14, Ar7 = Ae15, Ar8 = Ae16, Ar9 = Ae17

PROTOCOLO DE LA APLICACIÓN ESPECÍFICA *TRANSFERENCIA DE ARCHIVOS.* Especificaremos este protocolo con una MM, que modelará su núcleo, ampliada con las variables C_R (contador de reintentos), *C_SVB* (secuencia de verificación de bloque calculada en el receptor) y *NBF* (número de bloques pendientes de enviar), un *proceso generador de entradas* (PGE) que valorará el estado de estas variables para tomar sus decisiones y un *proceso generador de salidas* (incluido en las acciones), que tendrá en cuenta el estado de la variable *NBF* para proporcionar la salida del protocolo. Las variables serán actualizadas por alguno de los procesos que tienen lugar durante las transiciones de la MM.

De la misma forma que en el caso de el protocolo de Sesión-LL, para simplificar la comprensión del modo de operar de este protocolo, se diseña una MM para el sistema emisor de un fichero y otro para el receptor. Todas las posibles combinaciones (*estado, entrada)* quedan englobadas en las tablas de estados, entradas a la MM, acciones del protocolo, nuevos estados y salidas del protocolo, confeccionadas para las entidades que actúan como emisora y receptora del fichero. En el estado inicial de ambas tablas se han previsto todas las combinaciones válidas.

Este protocolo maneja un temporizador T_{EC} (tiempo de espera para una contestación de la entidad par). Al estado inicial se accede con el temporizador desactivado.

ESPECIFICACIÓN DEL PGE. El PGE maneja dos variables binarias (*E/R y ARB*).

Si E/R = 0 La entidad actúa como receptora
Si E/R = 1 La entidad actúa como emisora

Si ARB = 0 Ningún bloque recibido con SVB correcta
Si ARB = 1 Algún bloque recibido con SVB correcta

Se define $C_{R\,max}$ (máximo valor de C_R) como una constante

El PGE podrá evaluar el estado de las variables:

(C_R, Contador de reintentos)
(C_SVB, Secuencia de verificación de bloque calculada en el receptor)
(NBF, Número de bloques pendientes de enviar)

PGE:

1. E/R = 0
   ABR = 0

2. Espera C
   C = E_FICH.*request*
   E/R = 1
   Σ = E_FICH.*request*
   Ir a 2
   C = DATA.*indication* (ACK)
   Si NBF > 0 Σ = ACK
   Ir a 2
   Si NBF = 0 Σ = Ultimo ACK
   Ir a 1
   C = DATA.*indication* (NACK)
   Si $C_R > C_{R\,max}$ Σ = C_R vencido
   Ir a 1
   Si $C_R \leq C_{R\,max}$ Σ = Repetir envío
   Ir a 2
   C = DATA.*indication* (SVB)
   Si C_SVB = SVB ARB = 1
   Σ = SVB correcta
   Ir a 2
   Si C_SVB ≠ SVB
   Si ARB = 0 Σ = Iniciar recepción
   Ir a 2
   Si ARB = 1 Σ = SVB errónea
   Ir a 2

C = T_{EC} vencido			
	Si $C_R > C_{R\,max}$		$\Sigma = C_R$ vencido Ir a 1
	Si $C_R \leq C_{R\,max}$	Si E/R = 0	$\Sigma = T_{EC}$ vencido Ir a 2
		Si E/R = 1	Σ = Repetir envío Ir a 2
C = ABORT_TA.*request*			
	Σ = ABORT_TA.*request* Ir a 1		
C = DATA.*indication* (C_MEDIO)			
	Σ = Fin proceso Ir a 1		
Resto de condiciones	Σ = C Ir a 2		

La figura 4.18 muestra el diagrama de estados de la MM que modela el núcleo del protocolo de la aplicación específica *transferencia de archivos* para la entidad emisora. La tabla 4.7 recoge los estados, entradas a la MM, acciones del protocolo, nuevos estados y salidas del protocolo.

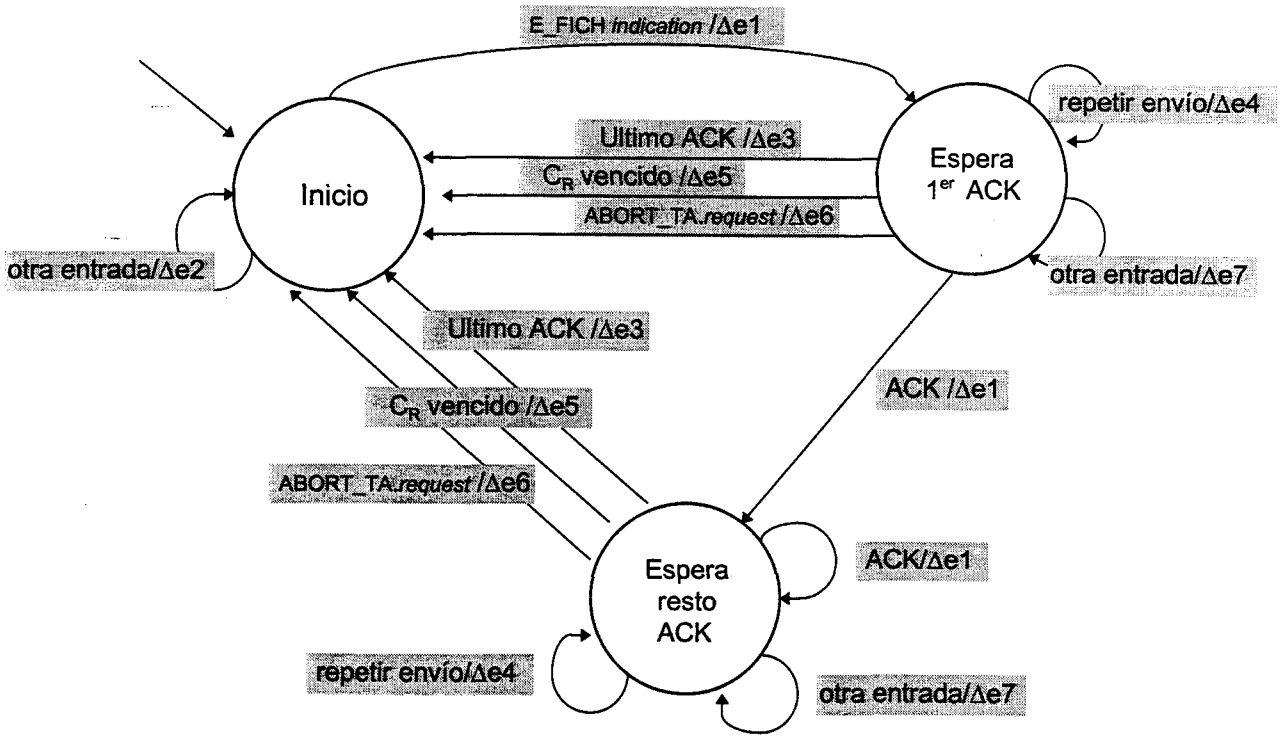

Figura 4.18 Diagrama de estados de la MEF de la aplicación "transferencia de archivos".
Entidad emisora

PROTOCOLO *TRANSFERENCIA DE ARCHIVOS* Entidad emisora				
Estado	**Entrada**	**Acción**	**Nuevo estado**	**Salida del protocolo**
Inicio	Σe0: DATA.*indication* (*DAT_FI*)	La entidad actúa como receptora (*véase tabla correspondiente*)		
	Σe1: E_FICH.*request*	Ae1	Espera 1er ACK	Se1: NBF, C_R, n DATA.*request* (bloques) n≤4 , DATA.*request* (SVB), activa T_{EC}
	Otra entrada	Ae2	Inicio	Se2: ∅
Espera 1er ACK	Σe2: ACK	Ae3	Espera resto ACK	Se1: NBF, C_R, n DATA.*request* (bloques) n≤4 , DATA.*request* (SVB), activa T_{EC}
	Σe3: Ultimo ACK	Ae4	Inicio	Se3: C_R, desactiva T_{EC} , DATA.*request* (C_MEDIO) FIN_EFICH.*indication*
	Σe4: Repetir envío	Ae5	Espera 1er ACK	Se4: NBF, C_R, n DATA.*request* (bloques) n≤4 , DATA.*request* (SVB), activa T_{EC}
	Σe5: C_R vencido	Ae6	Inicio	Se5: C_R, desactiva T_{EC} , ABORT_TA.*indication*
	Σe6: ABORT_TA.*request*	Ae7	Inicio	Se6: C_R , desactiva T_{EC}
	Otra entrada	Ae8	Espera 1er ACK	Se7: C_R, activa T_{EC}
Espera resto ACK	Σe2: ACK	Ae3	Espera resto ACK	Se1: NBF, C_R, n DATA.*request* (bloques) n≤4 , DATA.*request* (SVB), activa T_{EC}
	Σe3: Ultimo ACK	Ae4	Inicio	Se3: C_R,desactiva T_{EC} , DATA.*request* (C_MEDIO) FIN_EFICH.*indication*
	Ee4: Repetir envío	Ae9	Espera resto ACK	Se4: NBF, C_R, n DATA.*request* (bloques) n≤4 , DATA.*request* (SVB), activa T_{EC}
	Σe5: C_R vencido	Ae6	Inicio	Se5: C_R , desactiva T_{EC} ABORT_TA.*indication*
	Σe6: ABORT_TA.*request*	Ae7	Inicio	Se6: C_R , desactiva T_{EC}
	Otra entrada	Ae10	Espera resto ACK	Se7: C_R, activa T_{EC}

TABLA 4.7

ESPECIFICACIÓN DE LAS ACCIONES. Las acciones incluyen el PGS proporcionando las salidas del protocolo.

Variables que manipulan los procesos:
variables del protocolo:
(C_R , *Contador de reintentos*)

(NBF, Número de bloques pendientes de enviar)

variables internas inicializadas en cada proceso:

(SVB, Secuencia de verificación de bloque)
(NBE, Número de bloque enviado)

Ae1. $C_R = 0$

1. NBF = 0
   SVB = 0
   NBE = 0
   Determinar NBF. *Con parámetro LON_MEM*
   Conformar APDU de 1er bloque (CCA, LON_MEM, LON_FICH, DAT_FI)
   SVB = [SVB + $\sum x_i$ del campo DA de la APDU] módulo 256
   Enviar DATA.*request con APDU* al nivel Sesión-LL

2. NBE = NBE+1

   Si. NBE = NBF
   Enviar DATA.*request* con SVB al nivel Sesión-LL
   Activar T_{EC}
   Nuevo estado = Espera 1er ACK

   Si. NBE < NBF
   Si NBE = 4
   Enviar DATA.*request* con SVB al nivel Sesión-LL
   Activar T_{EC}
   Nuevo estado = Espera 1er ACK

   SI NBE < 4
   Conformar APDU de próximo bloque (CCA, DA)
   SVB = [SVB + $\sum x_i$ del c. DA de la APDU] mod. 256
   Enviar DATA.*request con APDU* al nivel Sesión-LL
   Ir a 2.

Ae2. Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:
Fecha, Hora, Nivel, Estado, Entrada
Nuevo estado = Inicio

Ae3. Desactivar T_{EC}
$C_R = 0$
NBF = NBF- 4

1. SVB = 0
   NBE = 0

2. Conformar APDU de próximo bloque (CCA, DAT)
   SVB = [SVB + $\sum x_i$ del campo DA de la APDU] módulo 256
   Enviar DATA.*request con APDU* al nivel Sesión-LL
   NBE = NBE+1

   Si. NBE = NBF
   Enviar DATA.*request* con SVB al nivel S-LL
   Activar T_{EC}

Nuevo estado = Espera resto ACK

Si. NBE < NBF

Si NBE = 4

Enviar DATA.*request* con SVB al nivel S-LL
Activar T_{EC}
Nuevo estado = Espera resto ACK

SI NBE < 4

Conformar APDU de próximo bloque (CCA, DA)
SVB = [SVB + $\sum x_i$ del c. DA de la APDU] mod. 256
Enviar DATA.*request con APDU* al nivel Sesión-LL
Ir a 2.

Ae4. Desactivar T_{EC}
$C_R = 0$
Enviar DATA.*request con* C_MEDIO al nivel Sesión-LL
Enviar FIN_EFICH.*indication* a la aplicación de usuario
Nuevo estado = Inicio

Ae5. Desactivar T_{EC}
$C_R = C_R + 1$

El resto de operaciones igual que A1. desde 1.

Ae6. Desactivar T_{EC}
$C_R = 0$
Enviar ABORT_TA.*indication* a la aplicación de usuario. *Se acompañarán parámetros indicando el* ***motivo de la interrupción*** *(Vencimiento de C_R) y* ***causa*** *(eventos que provocaron el incremento de C_R)*
Nuevo estado = Inicio

Ae7. Desactivar T_{EC}
$C_R = 0$
Nuevo estado = Inicio

Ae8. Desactivar T_{EC}
$C_R = C_R + 1$
Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:
Fecha, Hora, Nivel, Estado, Entrada
Activar T_{EC}
Nuevo estado = Espera 1er ACK

Ae9. Desactivar T_{EC}
$C_R = C_R + 1$
NBF = NBF

El resto de operaciones igual que A3. desde 1.

Ae10. Desactivar T_{EC}
$C_R = C_R + 1$

Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:
Fecha, Hora, Nivel, Estado, Entrada

Activar T_{EC}
Nuevo estado = Espera resto ACK

La figura 4.19 muestra el diagrama de estados de la MM que modela el núcleo del protocolo de la aplicación específica *transferencia de archivos* para la entidad receptora. La tabla 4.8 recoge los estados, entradas a la MM, acciones del protocolo, nuevos estados y salidas del protocolo.

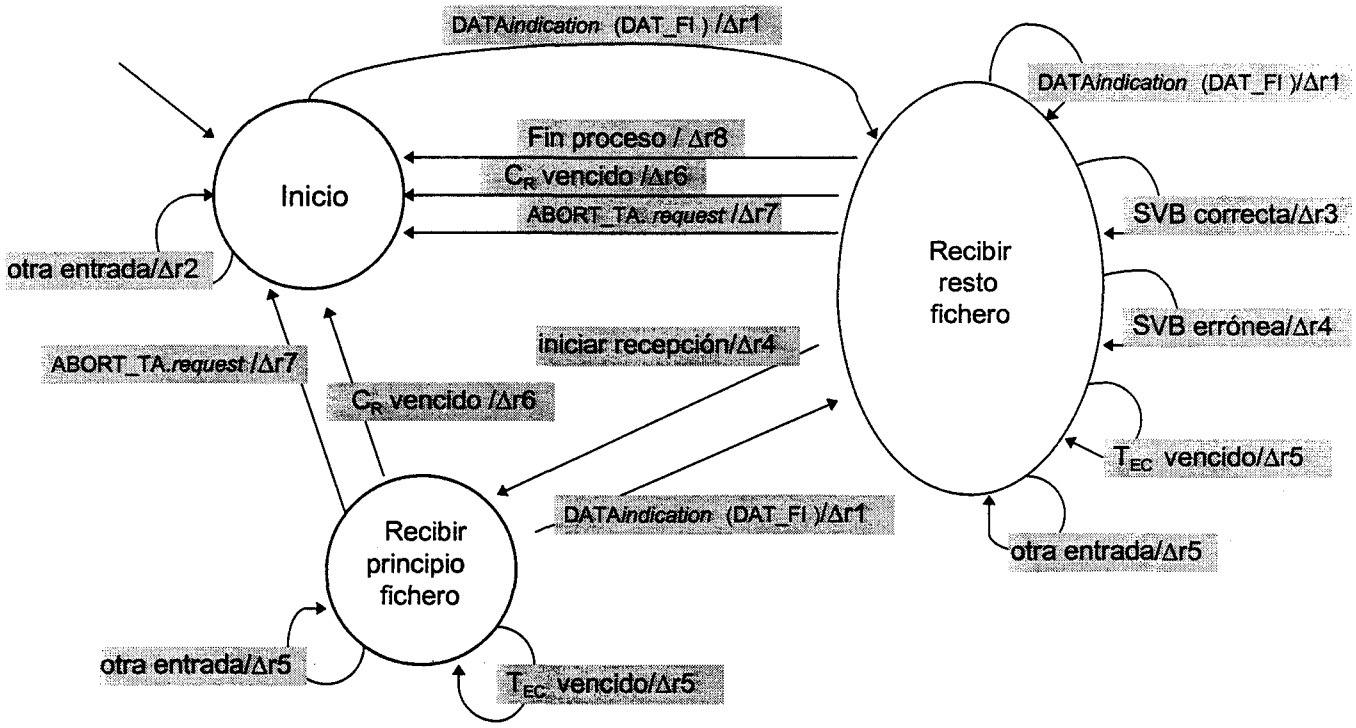

Figura 4.19 Diagrama de estados de la MEF de la aplicación "transferencia de archivos".
Entidad receptora

PROTOCOLO *TRANSFERENCIA DE ARCHIVOS* Entidad receptora				
Estado	**Entrada**	**Acción**	**Nuevo estado**	**Salida del protocolo**
Inicio	Σr0: E_FICH.*request*	La entidad actúa como emisora (*véase tabla correspondiente*)		
	Σr1: DATA.*indication* (*DAT_FI*)	Ar1	Recibir resto fichero	Sr1: C_SVB C_R activa T_{EC}
	Otra entrada	Ar2	Inicio	Sr2: ∅
Recibir resto fichero	Σr2: DATA.*indication* (*DAT_FI*)	Ar3	Recibir resto fichero	Sr1 C_SVB C_R activa T_{EC}
	Σr3: SVB correcta	Ar4	Recibir resto fichero	Sr3: C_SVB , C_R DATA.*request* (ACK_FI) activa T_{EC}
	Σr4: SVB errónea	Ar5	Recibir resto fichero	Sr4: C_SVB, C_R DATA.*request* (NACK_FI) activa T_{EC}
	Σr5: iniciar recepción	Ar6	Recibir principio fichero	Sr4: C_SVB, C_R DATA.*request* (NACK_FI) activa T_{EC}
	Σr6: T_{EC} Vencido	Ar7	Recibir resto fichero	Sr5: C_R activa T_{EC}
	Σr7: C_R vencido	Ar8	Inicio	Sr6: C_R desactiva T_{EC} ABORT_TA.*indication*
	Σr8: ABORT_TA.*request*	Ar9	Inicio	Sr7: C_R desactiva T_{EC}
	Σr9: Fin proceso	Ar10	Inicio	Sr8: C_R desactiva T_{EC} FIN_RFICH.*indication*
	Otra entrada	Ar11	Recibir resto fichero	Sr5: C_R activa T_{EC}
Recibir principio fichero	Σr2: DATA.*indication* (*DAT_FI*)	Ar12	Recibir resto fichero	Sr1 C_SVB C_R activa T_{EC}
	Σr6: T_{EC} Vencido	Ar13	Recibir principio fichero	Sr5: C_R activa T_{EC}
	Σr7: C_R vencido	Ar8	Inicio	Sr6: C_R desactiva T_{EC} ABORT_TA.*indication*
	Σr8: ABORT_TA.*request*	Ar9	Inicio	Sr7: C_R desactiva T_{EC}
	Otra entrada	Ar14	Recibir principio fichero	Sr5: C_R activa T_{EC}

TABLA 4.8

ESPECIFICACIÓN DE LAS ACCIONES. En este caso no es necesario el *PGS* porque ninguna variable del protocolo modifica la salida de la MM ($S = \Delta$).

Variables del protocolo que manipulan los procesos:
(C_R, *Contador de reintentos*)
(C_SVB, Secuencia de verificación de bloque calculada en el receptor)

Ar1. $C_R = 0$
C_SVB = 0

Guardar LON_MEM y LON_FCH *(para enviar como parámetros a la aplicación de usuario)*
Guardar datos de fichero en buffer temporal
C_SVB = [C_SVB + $\sum x_i$ del campo DA de la SDU recibida] módulo 256

Activar T_{EC}
Nuevo estado = Recibir resto fichero

Ar2. Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:
Fecha ,Hora, Nivel, Estado, Entrada

Nuevo estado = Inicio

Ar3. Desactivar T_{EC}
$C_R = 0$

Guardar datos de fichero en buffer temporal
C_SVB = [C_SVB + $\sum x_i$ del campo DA de la SDU recibida] módulo 256

Activar T_{EC}
Nuevo estado = Recibir resto fichero

Ar4. Desactivar T_{EC}
C_SVB = 0
$C_R = 0$

Enviar DATA.*request con ACK_FI* al nivel Sesión-LL

Activar T_{EC}
Nuevo estado = Recibir resto fichero

Ar5. Desactivar T_{EC}
C_SVB = 0
$C_R = C_R + 1$

Enviar DATA.*request con NACK_FI* al nivel Sesión-LL

Activar T_{EC}
Nuevo estado = Recibir resto fichero

Ar6. Desactivar T_{EC}
C_SVB = 0
$C_R = C_R + 1$

Enviar DATA.*request con NACK_FI* al nivel Sesión-LL

Activar T_{EC}
Nuevo estado = Recibir principio fichero

Ar7. $C_R = C_R + 1$

Activar T_{EC}
Nuevo estado = Recibir resto fichero

Ar8. Desactivar T_{EC}
$C_R = 0$

Enviar ABORT_TA.*indication* a la aplicación de usuario. *Se acompañarán parámetros indicando el* ***motivo de la interrupción*** *(Vencimiento de C_R) y* ***causa*** *(eventos que provocaron el incremento de C_R)*

Nuevo estado = Inicio

Ar9. Desactivar T_{EC}
$C_R = 0$

Nuevo estado = Inicio

Ar10 Desactivar T_{EC}
$C_R = 0$

Enviar FIN_RFICH.*indication* a la aplicación de usuario.
(Se acompañarán parámetros: Estación que envía la información, identificativo para localizarla, LON_MEM y LON_FICH

Nuevo estado = Inicio

Ar11. Desactivar T_{EC}
$C_R = C_R + 1$

Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:
Fecha ,Hora, Nivel, Estado ,Entrada

Activar T_{EC}
Nuevo estado = Recibir resto fichero

Ar12. $C_R = 0$

Guardar LON_MEM y LON_FICH (*para enviar como parámetros a la aplicación de usuario)*
Guardar datos de fichero en buffer temporal
C_SVB = [C_SVB + $\sum x_i$ del campo DA de la SDU recibida] módulo 256

Activar T_{EC}
Nuevo estado = Recibir resto fichero

Ar13. $C_R = C_R + 1$

Activar T_{EC}
Nuevo estado = Recibir principio fichero

Ar14. Desactivar T_{EC}
$C_R = C_R + 1$

Anotar en fichero de errores "*mensaje de error*" con el siguiente formato:
Fecha ,Hora, Nivel, Estado, Entrada

Activar T_{EC}
Nuevo estado = Recibir principio fichero

PROTOCOLO DE LA APLICACIÓN ESPECÍFICA *ENTRADA REMOTA DE TRABAJOS.* La MM que modela este protocolo se compone de un único estado, figura 4.20.

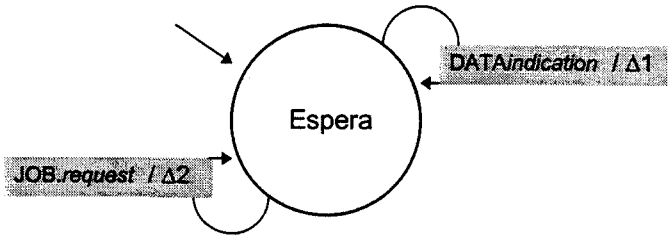

Figura 4.20 Diagrama de estados de la MEF de la aplicación específica *entrada remota de trabajos*

Los estados, entradas a la MM, acciones del protocolo, nuevos estados y las salidas del protocolo pueden verse en la tabla 4.9.

PROTOCOLO *ENTRADA REMOTA DE TRABAJOS*				
Estado	**Entrada**	**Acción**	**Nuevo estado**	**Salida del protocolo**
Espera	Σ1: DATA.*indication*	A1	Espera	S1: JOB.*indication*
	Σ2: JOB.*request*	A2	Espera	S2: Data.*request*

Tabla 4.9

Especificación de las acciones:

A1. Determinar tipo de trabajo con campo CCA *[Campo de control de la aplicación*]

Salida = Enviar JOB.*indication* a la aplicación de usuario

[parámetros: estación origen y tipo de trabajo;

SDU = campo DA recibido (datos de la aplicación)]

Nuevo estado = Espera

A2. Conformar APDU [*unidad de datos del protocolo para la aplicación*]

[CCA con parámetro recibido (Tipo de trabajo), campo DA con la SDU recibida]

Salida = Enviar DATA.*request* al nivel Sesión-LL

[parámetro: estación destino, SDU = APDU conformada]

Nuevo estado = Espera

4.5.3 OTRAS CONSIDERACIONES SOBRE LOS PROTOCOLOS

La especificación formal de los protocolos mediante el método descrito en el apartado anterior, permite utilizar técnicas conocidas para *verificar* su correcto funcionamiento. La *verificación* del protocolo de nivel *N* se reduce básicamente a demostrar que tiene las propiedades necesarias para que su comportamiento sea el deseado, es decir, que ofrezca el servicio *N* tal como es esperado. Para ello se supone que el nivel *N-1* funciona correctamente [Beltrào_89].

La elección de una técnica de verificación particular, está condicionada por el método de especificación formal del protocolo. En nuestro caso puede utilizarse una técnica mixta que emplearía *análisis de alcance*[11] para determinar las propiedades generales del protocolo (propiedades de control), a partir de las especificaciones de la MEF definida para cada protocolo o su núcleo. La verificación de las propiedades

---

[11] *Análisis de alcance:* Determinación de todos los estados que pueden ser alcanzados en una MEF, a partir de un estado inicial, en respuesta a todas las posibles entradas que perturban a la MEF en cada estado.

específicas puede llevarse a cabo con la *prueba de aserciones*[12] del(de los) programa(s), que completan la definición del protocolo.

Así, la verificación de cada protocolo puede realizarse en dos partes distintas y complementarias. Esta separación permite que las propiedades de la MEF se verifiquen de forma independiente, suministrando resultados parciales, pero muy útiles, porque los errores graves del protocolo pueden detectarse y corregirse rápidamente. La verificación del contexto del protocolo puede hacerse a continuación por la prueba de aserciones, que puede simplificarse porque la parte de control del protocolo ya ha sido investigada.

Para abordar la cuestión de implementar los protocolos definidos, deben tenerse en cuenta las facilidades de los sistemas en los cuales va a ejecutarse la implementación. En principio la implementación de un protocolo puede realizarse en *hardware* o *software*. Es corriente que en redes locales y en redes de larga distancia, la implementación de los protocolos de los niveles inferiores de la arquitectura se realice en *hardware*, controlado por algún tipo de microprocesador. En los sistemas informáticos utilizados en la confección de esta red, únicamente el puerto serie está controlado por *hardware*. La implementación, excepto en algunos aspectos del nivel físico, deberá realizarse en *software*.

La forma de presentar la especificación formal de los protocolos en esta memoria permite identificar cada nivel definido con un proceso. Así, podría estructurarse el *software* de cada nivel como un proceso independiente, conectando sus salidas a la entrada de los procesos relativos a su nivel superior e inferior, según corresponda [Cox_86]. Las interfases entre niveles definirían los métodos de comunicación entre procesos.

---

[12] *Prueba por aserciones:* Introducir aserciones (condiciones) en un programa, que se devuelven verdaderas cuando la ejecución del programa alcanza ese lugar.

Utilizando las tablas de estados, entradas y acciones, podría construirse un compilador que generara de forma automática parte del código en un determinado lenguaje de programación. Este código se completaría con la escritura, en el mismo lenguaje, de las acciones especificadas en pseudocódigo. Finalmente, escribiendo el *software* relativo al PGE (si existe), podría obtenerse una implementación ejecutable de cada protocolo con relativa comodidad. Debe tenerse en cuenta que gran parte del trabajo se ha realizado ya en la especificación y que, prácticamente, resta traducir el pseudocódigo a un lenguaje de programación determinado.

El tipo de implementación descrita implica que, en los sistemas donde deben ejecutarse los protocolos, deben existir varios procesos abiertos simultáneamente; como mínimo uno por nivel más el (los) proceso (s) de la aplicación de usuario. Para esto necesitamos plataformas, es decir, equipos con un sistema operativo que admitan multiproceso.

Aunque en el ordenador personal, utilizado en el centro de proceso de la red, se pueden instalar sistemas operativos como Windows NT o Linux, que admiten multiproceso, esta opción no es recomendable, porque el usuario tiene poco control sobre la prioridad con la cual será atendido cada proceso por el sistema operativo.

Por otra parte, debe tenerse en cuenta que las transiciones entre los estados de los protocolos definidos son siempre secuenciales. Para que en la MEF del nivel *N* se produzca una transición es necesario que se haya producido, previamente, una transición en la MEF del nivel *N+1*, o del de nivel *N-1* o venza un temporizador interno. En un mismo sistema no se ejecutarán dos acciones de forma simultánea, por lo tanto, no tiene mucho sentido que el procesador dedique tiempo a atender procesos que están en espera de una entrada, que solo podrá producirse cuando finalice una operación que se está ejecutando en otro de los procesos en un momento dado.

El único lugar donde pueden producirse dos sucesos simultáneos es en los puertos serie de los sistemas, pero, en este caso, el *hardware* que controla estos dispositivos permite atender a la emisión y recepción de *bytes* del medio físico a la vez

(con esta facilidad debe contarse a la hora de implementar el nivel físico). En estas circunstancias es aconsejable recurrir a la programación secuencial para obtener un mejor rendimiento del procesador de cada sistema. De esta forma se optimizará la utilización de recursos del sistema dedicados a la comunicación, haciendo uso de los mismos únicamente cuando sus servicios son requeridos.

Así, en cada sistema informático de la red se definirá un proceso principal, que contendrá la aplicación de usuario y que estará en espera de entradas: a través del interfase de usuario, de la propia aplicación o recepción de información por el puerto serie. En función de las entradas ajecutará acciones que irán llamando a distintos módulos (estos forman parte de la aplicación principal), que proporcionarán los servicios requeridos de los distintos niveles de la arquitectura.

En los apéndices I y II de la memoria aparece un ejemplo de implementación de los procesos principales que se ejecutan en el ordenador personal de la estación central (programado en Visual Basic) y en los sistemas informáticos remotos de las estaciones de campo (programados en Ensamblador). En estos programas está incluido el *software* relativo a la red de ordenadores, aunque no de una forma estructurada. Por ejemplo, existen funciones como "*quitar transparencia y determinar SVB*" que presta servicios adjudicados a niveles distintos en la arquitectura. También distintas acciones como *conformar tramas de enlace para transmitir al nodo de la subred*, aparecen repetidas en distintas funciones o procedimientos de la aplicación. A pesar de estos, y otros, aspectos que podrían modificarse para mejorar la presentación, se ha preferido dejar estos apéndices, tal cual aparecen, por varias razones.

En primer lugar, este *software* es una muestra de como se ha ido elaborando el trabajo que ha permitido formular la arquitectura de la red. Así, a partir de una necesidades de la aplicación de usuario se han ido construyendo módulos, donde se incluía el tratamiento de la información necesario para esta pudiera viajar a través de la red de comunicaciones con garantías. Esto explica que una misma acción aparezca en más de un módulo de la aplicación. Pero es esta forma de proceder, la que ha permitido ***abstraer*** que servicios de red eran necesarios y que estructura de red era la más

adecuada para que estos servicios se prestaran de una forma eficaz. Este *software* es, por tanto, una parte de la base experimental que ha servido para diseñar la arquitectura de red que aparece en la memoria.

En segundo lugar, es un *software* que está instalado y funcionando en trabajo real desde marzo de 1996. Esto es una forma de corroborar los planteamientos acerca de la red de ordenadores que se efectúan en la memoria y, a la vez, una manera de verificar experimentalmente que los protocolos definidos cumplen con las especificaciones realizadas en su diseño.

Por último, los programas que aparecen en estos apéndices, son una forma posible de implementar el *software* de red. Evidentemente esta no sería la implementación más adecuada si partimos de una arquitectura ya confeccionada, con sus protocolos especificados conforme aparecen en esta memoria. Pero no deja de ser una implementación válida, ya que cualquier decisión sobre la implementación de un protocolo es correcta si esta no afecta a su funcionamiento, es decir, no le impide prestar el servicio para el cual fue diseñado conjuntamente con otra entidad, la cual, puede estar a su vez implementada de forma diferente.

El diseño de la red de ordenadores no ha partido de un desarrollo teórico, comprobándose a continuación su operatividad de forma experimental. Su concreción final es el resultado de aplicar una metodología de diseño, apoyada en las pruebas realizadas con los elementos que conforman la red y en las conclusiones obtenidas a partir de estas experiencias. Desde este punto de vista, el *software* contenido en los apéndices I y II es una importante referencia que ayuda a entender el planteamiento final de la arquitectura propuesta para la red de ordenadores.

Ahora bien, una vez formalizada esta arquitectura y especificados los protocolos de los niveles que la conforman, se puede abordar desde otra perspectiva la implementación del *software* relativo a la red. Para llevar a cabo esta implementación utilizando programación secuencial, se propone construir módulos de *software* independientes, conformados a partir de las distintas funciones que pueden identificarse

en los niveles definidos. Veamos un ejemplo implementando en el ordenador personal de la estación central, un módulo que proporciona el establecimiento de un enlace lógico con otra estación. Este módulo formaría parte del servicio CONNECT del nivel de Sesión-LL. La implementación se realiza en Visual Basic (plataforma de programación utilizada en este sistema).


```
Function ConectarEstacion (nombreest As String) As Integer

  'Averiguar canal libre ==> definida 'canalTX'
  ' Fijar servicio ==> será siempre conexion="C"

  NReint = 0
REINTENTAR:
  If NReint > 3 Then ConectarEstacion = 0;  Exit;  End If
  SPDU = "C " + nombreest
  EnviarNCA "C", canalTX, SPDU                ' DATA.request al nivel NCA

RELEERSERIE:
  Tiempo = 5
  RtoRX = RecibirNCA(Tiempo, CC, canalRX, SPDU)
  If RtoRX = 1 Then
    If canalRX = canalTX Then
      If CC = "C" Then
        If Mid$(SPDU, 1, 1) = "C" Then
          ConectarEstacion = 1; Exit
          ' Anotar tabla estaciones conectadas y sus canales asociados
        Else
          NReint = NReint + 1
          GoTo REINTENTAR
        End If
      Else
        ' seria D o cualquier otra
        NReint = NReint + 1
        GoTo REINTENTAR
      End If
    Else
      'seria un REQUEST de otra estación ó tramaEnl = ""
      GoTo RELEERSERIE
    End If
  Else   'trama vacía
    NReint = NReint + 1
    GoTo REINTENTAR
  End If

End Function
```

La implementación de esta tarea se realizaría en Visual Basic por medio de una función, a la que se pasa como parámetro el identificativo de la estación con la que se

desea establecer el enlace. La función devolvería el resultado de la variable *ConectarEstación*. Para simplificar solo se especifican dos valores posibles: "1" (*enlace establecido*) y "0" (*enlace no establecido*), pero puede ampliarse con facilidad el número de valores para que el usuario conozca, en caso de no establecerse el enlace, el motivo que lo provoca.

Esta función se encarga únicamente del establecimiento de enlace lógico y sería llamada por otra función, de orden superior, encargada de establecer una sesión con la misma estación, ambas formarían parte del nivel Sesión_LL.

Para que este nivel pueda prestar el servicio CONNECT necesita un servicio DATA de su nivel inferior (nivel NCA). Este servicio se solicita por medio de la primitiva DATA.*request*, que en el *software* especificado es una llamada al procedimiento *EnviarNCA* (se trata de un procedimiento porque *EnviarNCA* ejecuta una acción sin devolver el resultado de variables). Una vez solicitado este servicio la función *ConectarEstación* queda en espera de recibir una contestación (*RecibirNCA*).

El servicio DATA (en emisión) del nivel NCA es prestado por el procedimiento *EnviarNCA* y la función *PonerT* (poner transparencia a la información). El procedimiento llama a la función para que ejecute su tarea y una vez obtiene la información transparente, conforma la unidad de datos de protocolo del nivel NCA (NPDU) y solicita un servicio DATA al nivel Físico (DATA.*request*) realizando una llamada al procedimiento *EscribirSerie*. Su especificación en Visual Basic, sería:

```
Sub EnviarNCA (CC As String, canal As String, SPDU As String)

' Conformar NPDU ==>
' cte. global Port=2
   SPDU2 = PonerT(SPDU)
   NPDU = CC + Port + canal + SPDU2
   EscribirSerie NPDU
End Sub
```

```
Function PonerT (cadena As String) As String

Dim i As Integer, longitud As Integer
Dim cadaux As String
   longitud = Len(cadena)
```

```
  For i = 1 To longitud
    Select Case Mid$(cadena, i, 1)
      Case Chr$(FEND)
        cadaux = cadaux + Chr$(FESC) + Chr$(TFEND)
      Case Chr$(FESC)
        cadaux = cadaux + Chr$(FESC) + Chr$(TFESC)
      Case Else
        cadaux = cadaux + Mid$(cadena, i, 1)
    End Select
  Next i
  PonerT = cadaux

End Function
```

Para confeccionar el procedimiento *EscribirSerie* y la función *LeerSerie*, se utilizan las facilidades que proporciona el Visual Basic en su librería MSCOMM.VBX, como manejador del puerto serie a bajo nivel. Su especificación debe ser la misma que aparece en el apéndice I.

Las funciones *RecibirNCA* y *QuitarT* prestarían el servicio DATA del nivel NCA en recepción. El nivel NCA recibe una primitiva DATA.*indication* del nivel Físico, cuando la función *RecibirNCA* obtiene una trama de enlace (TramaEnl) distinta de vacío, proporcionada por la función *LeerSerie* perteneciente al nivel Físico. La función *RecibirNCA* recurre a la función *QuitarT* para que elimine la transparencia de la información, determina las campos CC , canal y datos en la trama de enlace y envía una primitiva DATA.*indication* al nivel Sesión-LL (RecibirNCA = 1). La especificación de estas funciones en Visual Basic, sería:

```
Function RecibirNCA (Tiempo As Integer, CC As String, canal As String,
                     SPDU As String) As Integer

Dim t As Long
' Espera recibir contestacion en un tiempo Timeout
  tramaEnl = ""
  rec_trama = False
  t = Timer
  While (Timer - t <= Tiempo) And (rec_trama = False)
    ' Espera recibir una trama por el puerto serie (LeerSerie)
    DoEvents
  Wend

  If tramaEnl <> "" Then
    tramaEnl = QuitarT(tramaEnl)
    CC = Mid$(tramaEnl, 1, 1)
    canal = Mid$(tramaEnl, 3, 1)
```

```
    SPDU = Mid$(tramaEnl, 4 )
    RecibirNCA = 1
  Else
    RecibirNCA = 0
  End If
End Function
```

Function QuitarT (cadena As String) As String

```
Dim i As Integer, longitud As Integer
Dim cadaux As String, car As String
' Recibe una cadena y elimina la transparencia

  longitud = Len(cadena)
  i = 1
  While i <= longitud
    car = Mid$(cadena, i, 1)
    Select Case car
      Case Chr$(FESC)    'FESC habrá pocos caracteres en cada trama
        i = i + 1
        Select Case Mid$(cadena, i, 1)
          Case Chr$(TFEND)
            cadaux = cadaux + Chr$(FEND)
          Case Chr$(TFESC)
            cadaux = cadaux + Chr$(FESC)
        End Select
      Case Else
        cadaux = cadaux + car   'usando car evito Mid$ de nuevo
    End Select
    i = i + 1
  Wend
  QuitarT = cadaux

End Function
```

De esta forma la función *ConectarEstación* desarrolla el trabajo para el que ha sido diseñada (establecer el enlace lógico con otra estación o informar acerca de porqué este enlace no puede ser establecido), operando conforme a la estructura lógica definida en el diseño de la red.

Los módulos que se han definido para prestar el servicio DATA en el nivel NCA, serán utilizados siempre por el nivel de Sesión-LL, independientemente del servicio que este nivel preste a su nivel superior (CONNECT, DATA o DISCONNECT), pues en todos ellos es necesario este servicio del nivel NCA. Para solicitar este servicio (DATA.*request*), basta incorporar una llamada a *EnviarNCA* y esperar *RecibirNCA*

(DATA.*indication*) en los módulos correspondientes, como se ha hecho en la función *ConectarEstación*.

La confección de módulos de *software* sencillos como los descritos, permite implementar el *software* de la red con relativa comodidad. Además, así se puede independizar el *software* de la red y de la aplicación de usuario, de forma que el primero sería utilizable para cualquier aplicación que se diseñe. Para solicitar la aplicación de usuario los servicios de la red bastará que se incluyan en los diferentes módulos que lo integran, llamadas a los módulos de red que tienen comunicación con la aplicación de usuario (llamadas a módulos como: EstablecerSesión, EnviarFichero, EnviarTrabajo, etc., dependiendo del servicio que se solicite de la red). La jerarquización de los módulos del *software* de red será acorde con la jerarquización de niveles definidos, por lo que los módulos de red que comunican con la aplicación de usuario se encargarían de enlazar con los módulos necesarios para prestar el servicio.

Esta forma de proceder, factible en el ordenador personal de la estación central, no siempre es posible en todos los sistemas, pues debe contarse con las características de los equipos donde el *software* debe ejecutarse. Así en los SIR de las estaciones de campo tenemos algunas limitaciones que impiden llevar a la práctica esta filosofía de implementación en su totalidad.

Por ejemplo, los microprocesadores utilizados en la confección de los SIR, en la aplicación práctica implementada, no disponen de una memoria interna donde almacenar cadenas de *bytes*. Trabajar con la memoria externa para realizar determinadas operaciones restaría eficacia al rendimiento del microcontrolador que se encarga de actuar con la red, pues debe competir con otro microcontrolador para ganar el acceso a esta memoria. Por otra parte, el control del puerto serie se realiza por el mismo microcontrolador que actúa con la red y ha de gestionar directamente el envío de *bytes* al medio físico, uno a uno.

En estas circunstancias es aconsejable que cada vez que se lea un *byte* de la memoria externa para enviarlo a otro sistema a través de la red, se realicen sobre el

mismo las operaciones que pueden modificar este *byte* (poner transparencia), todos los cálculos en los que este *byte* intervenga (determinar SVB y determinar el número de *bytes* en la trama de enlace) y finalmente se envíe al *byte* . Esta forma de operar, además de proporcionar un mayor rendimiento del sistema, permite ahorrar líneas de código; detalle importante, pues en estos mismos microprocesadores existe una limitación de 4 k*bytes* para el programa ejecutable, que debe contener: el *software* de red, el *software* del sistema operativo del SIR y el *software* de la aplicación de usuario.

Las características específicas de la plataforma donde debe ejecutarse el *software*, impiden llevar a cabo una implementación tan estructurada y acorde con la arquitectura definida como en el caso anterior. No obstante la forma de proceder que se sugiere no afecta al funcionamiento de los protocolos por lo que una implementación de estas características sigue siendo válida.

En el *software* relativo al microcontrolador 1 que aparece en el apéndice II de la memoria se presenta una programación modular del *software* de red (nivel de Sesión-LL y niveles inferiores), intentando que los módulos definidos (funciones en lenguaje ensamblador) correspondan con operaciones relativas a niveles específicos. En algunos casos, como se ha explicado, es imprescindible mezclar tareas correspondientes a distintos niveles en un mismo módulo. Dadas las características de los SIR utilizados, que serán analizados en capítulo siguiente, entendemos que esta sería la mejor forma de implementar el *software* de red en estos dispositivos.

CAPITULO 5

EL SISTEMA INFORMÁTICO REMOTO

- ***Características generales del sistema***

- ***Adaptación del sistema informático remoto al control de estaciones de campo con sistemas de adquisición de datos autónomos***

5.1 CARACTERÍSTICAS GENERALES DEL SISTEMA.

Para confeccionar la red de ordenadores necesitamos disponer, en cada estación de campo, de un sistema informático capaz de realizar un control local de los elementos que integran la estación y de interconectarse con el resto de sistemas que conforman la red. Dado el trabajo especializado que debe efectuar este sistema, conviene definir que características técnicas debe reunir para realizar adecuadamente estas funciones.

Lógicamente debe ser un sistema desarrollado en base a la complejidad del control local que se le asigne. Esta complejidad normalmente viene determinada por la necesidad de tener que ejecutar varias tareas de forma simultánea. Así, además de realizar operaciones matemáticas con los distintos datos que maneja, estos sistemas pueden tener misiones como: la adquisición de datos, el almacenamiento de datos en algún dispositivo, elaborar información depurada para enviar a otros sistemas, controlar el estado de diversos elementos locales, controlar distintas variables, actuar sobre motores, válvulas, etc., y en este caso, atender también las comunicaciones en la red.

En estas circunstancias, la potencia del sistema no debe estar orientada a la velocidad de proceso en la ejecución de un trabajo específico, si no a la diversidad de tareas que deben atenderse a la vez. Por otra parte, las funciones que deben ejecutarse para realizar un control local pueden ser muy distintas en cada situación. Es conveniente que el sistema tenga una arquitectura flexible, en cuanto a su capacidad de expansión, con posibilidad de añadir o quitar elementos para adecuar su capacidad a unas necesidades determinadas.

Considerando todos estos aspectos, no es aconsejable delegar la ejecución de todas estas funciones en un solo procesador, aún siendo este muy potente. En este último caso, aunque los trabajos pueden desarrollarse de forma satisfactoria el sistema no sería flexible y se encarecería notablemente el producto. Una solución más acorde con la problemática planteada consistiría en la utilización de un sistema basado en una arquitectura paralela.

Como el sistema debe ejecutar tareas variadas y con un elevado grado de independencia entre sí, el tipo de paralelismo que se requiere es el que puede proporcionar una arquitectura *multiprocesador* o *multicomputador*. Estas arquitecturas tienen como base la utilización de varias unidades centrales de proceso (UCP), que pueden ejecutar cada una su propio programa y que interacionan entre sí de forma síncrona o asíncrona. Los sistemas con más de una UCP se pueden clasificar en dos tipos [Ibarra_95]:

1. *Multiprocesadores*. Las distintas UCP cooperan en la ejecución de los programas compartiendo información y procesos. La comunicación entre las UCP se realiza a través de una memoria principal compartida o mediante el paso de mensajes en un sistema donde la memoria está distribuida. El sistema está bajo el control de un único sistema operativo.
2. *Multicomputadores*. Sistemas formados por varias UCP independientes interconectadas mediante una red de comunicación. Cada UCP dispone de su memoria local (en principio no existen direcciones globales de memoria). Cada UCP está gobernada por su propio sistema operativo y no hay una interacción directa entre las UCP cuando se ejecutan los programas.

En base a las características que presentan los sistemas *multiprocesador* y *multicomputador* se observa que estos últimos resultan mas adecuados a las necesidades planteadas, porque garantizan una mayor independencia entre las tareas que se ejecutan de forma simultánea y tienen un alto grado de flexibilidad, en cuanto a añadir y suprimir elementos del sistema.

Se descarta el uso de arquitecturas paralelas *segmentadas*[1] o *matriciales*[2], porque su objetivo es aumentar la ganancia de velocidad de una sola tarea, ejecutando en paralelo distintas operaciones del proceso [Hawang_85].

Tanto la facilidad de expansión como la agilidad en el intercambio de información entre las UCP que componen un sistema *multicomputador*, depende en gran medida de su red de interconexión. Establecer el tipo de red más conveniente para estos sistemas requiere un estudio en profundidad de las posibles topologías con sus interfases asociados, de la gestión del acceso al medio por las UCP del sistema y de la gestión del enlace lógico para garantizar la eficiencia y fiabilidad de la comunicación. Aún así, existen múltiples factores que pueden condicionar la elección de uno u otro procedimiento para ajustarse a situaciones específicas. No obstante, y sin descartar otras posibles soluciones, en este apartado se sugieren algunas ideas para diseñar un sistema que proporcione resultados satisfactorios.

Como elementos procesadores del dispositivo se pueden utilizar integrados microprocesadores o microcontroladores estándar, con los circuitos adicionales necesarios, como circuitos de entradas/salida, memorias, relojes, etc. Estos componentes son fáciles de adquirir, de bajo coste y hay disponible mucha información sobre los mismos. De esta forma, el diseño *hardware* del sistema se convierte en una cuestión de electrónica digital, fácil de resolver.

El sistema debe incorporar todas las UCP en un ámbito local y reducido, por lo que las conexiones entre las UCP deben ser cableadas. Para lograr accesos rápidos,

---

[1] *Arquitectura paralela segmentada*: Técnica que descompone las funciones que realiza un circuito en etapas especializadas, de forma similar a como se hace en una cadena de montaje de un proceso industrial En esa cadena, con respecto a una pieza determinada, se trabaja de forma secuencial, pero si se consideran las distintas piezas que se están elaborando en la cadena se puede apreciar que se realizan operaciones paralelas.

[2] *Arquitectura paralela matricial*: Sistemas que agrupan gran número de unidades de proceso que operan paralelamente y de forma síncrona bajo la supervisión de una sola unidad de control. Este tipo de sistemas está diseñado para ejecutar operaciones sobre grandes conjuntos de datos, como vectores o matrices.

transmisiones ágiles e interfases sencillos, estas conexiones deben ser líneas que soporten señales digitales conectadas directamente a los puertos de los procesadores. Así, se evita tener que emplear modulaciones y la codificación de la información en el medio es tan simple como poner los niveles lógicos correspondientes a los *bits* transmitidos.

En la idea de que las interfases entre los procesadores y la red de interconexión sean sencillas es recomendable utilizar un medio de multidifusión, como un bus, al que estén conectadas todas las UCP. Un medio secuencial, con enlaces entre parejas de UCP, necesitaría interfases más complejos y presentaría mayores dificultades de expansión.

Para poder utilizar como UCP el mayor número de dispositivos estándar no debe condicionarse que los procesadores deban tener un puerto serie implementado. El interfase con la red de interconexión se realizará por los puertos de entrada/salida paralela de los procesadores.

El número de líneas del bus vendrá condicionado por las necesidades de capacidad de transmisión y por el método de acceso al medio seleccionado. En un principio debe de tratarse que el bus esté compuesto por pocas líneas, disponiéndose así de mayor capacidad en los puertos E/S de los procesadores para dedicarlos a sus aplicaciones específicas. Pero si se desea una transmisión muy eficaz es conveniente emplear más de una línea para transmitir información, en lugar de realizar esta tarea mediante una transmisión serie pura. Teniendo en cuenta que los microcontroladores comerciales disponen de instrucciones para trabajar con mitades de palabras de 8 *bits*; se puede diseñar, por ejemplo, un bus formado por cuatro líneas de datos para transmitir 4 *bits* cada vez.

Además del bus de datos pueden ser necesarias otras líneas para gestionar el acceso al medio y, si se utilizan transmisiones síncronas, alguna línea para sincronizar la información. Inicialmente podría pensarse en utilizar transmisiones asíncronas de datos o en codificaciones de la información que incluyan la señal de

reloj. Pero esta opción no es aconsejable ya que resulta mucho más operativo la utilización de una línea adicional para enviar la señal de reloj. Así, puede sincronizarse la información enviada entre los procesadores, sin necesidad de añadir *hardware* para gestionar transmisiones asíncronas o codificaciones complejas que complicarían el interfase de los procesadores con la red.

En cuanto a la gestión del acceso al medio de comunicación caben múltiples alternativas. Algunas casas comerciales han desarrollado procedimientos que permiten interconectar distintos elementos (microcontroladores, memorias, displays, circuitos de entrada/salida, etc.), para poder confeccionar sistemas más complejos.

En [Martínez_91] se presenta uno de estos procedimientos basado en un bus con dos líneas. Por una de ellas se transmite la señal de reloj. La otra línea soporta la transmisión de datos síncronos y la gestión de la reserva del medio para poder transmitir. Los distintos elementos que pueden conectarse a este bus se dividen en maestros y esclavos. Los primeros deben establecer una contienda para poder conseguir el medio de transmisión, pudiendo producirse colisiones durante este proceso. Una vez conseguido el medio, el elemento que lo obtiene transmite o solicita información a otros elementos conectados a la red direccionándolos adecuadamente. Los elementos esclavos no pueden competir por el medio y únicamente pueden enviar información al mismo cuando así se lo solicite el elemento maestro que lo ha capturado. Los dispositivos electrónicos que tienen una implementación *hardware* de este procedimiento, disponen de un conjunto de registros especiales que permiten conocer los estados de la comunicación.

Un estudio más detallado de posibles alternativas en el diseño de una red de interconexión para microcontroladores se aborda en [Candelas_96]. En el mismo trabajo se propone una red de interconexión para los procesadores, donde el acceso para enviar información al medio se gestiona, como en el caso anterior, mediante un proceso de reserva implícita del bus de datos. En este caso la reserva es más ágil por utilizarse dos líneas adicionales en el bus (línea de aviso y línea de reserva), independientes de la(s) línea(s) de datos. La resolución de las colisiones que pueden

producirse en el proceso de reserva se realiza de una forma bastante eficaz, gracias a la gestión coordinada de las líneas de aviso y reserva. De esta forma, la mayoría del tiempo que un procesador invierte en la gestión de la red lo utiliza para transmitir o recibir información y para ciertas temporizaciones necesarias en el proceso de reserva. Si estas temporizaciones se controlan, como se propone en el trabajo, mediante un reloj *hardware* y el uso de interrupciones, el tiempo de proceso que requiere el subnivel de acceso al medio se limita casi al empleado por las funciones de envío y recepción de información.

El subnivel de enlace lógico, en ambos procedimientos, debe garantizar la transmisión fiable (sin errores de nivel físico) de la información entre las distintas UCP. En ambos casos puede definirse un mismo método, ya que este subnivel debe ser independiente de la tecnología del medio y del procedimiento de acceso al mismo.

La red de interconexión diseñada en [Candelas_96] se ajusta mucho mejor a los requerimientos definidos para los sistemas informáticos remotos que deben conformar esta red. Aunque utiliza 2 líneas más de los puertos de E/S de los procesadores, que el procedimiento descrito en [Martínez_91], la transmisión de la información es mucho más eficiente y permite la utilización de cualquier tipo de procesador estándar para confeccionar las UCP del sistema. En cualquier caso, la elección de uno de estos dos procedimientos, u otros que pudieran diseñarse, estará condicionada por las necesidades de comunicación que se definan para los distintos procesos que se atienden por el sistema multicomputador, en función de la aplicación específica a la cual se destine.

El diseño del sistema informático remoto en base a una arquitectura multicomputador y con una red de interconexión eficiente entre los procesadores, permite disponer de un sistema de propósito general, cuyos procesadores pueden abordar la ejecución de distintas tareas de forma independiente y con una comunicación ágil entre las mismas. La flexibilidad del sistema propuesto posibilita dimensionar el número de procesadores que lo integran, adecuándolo a necesidades

puntuales con suma facilidad. La capacidad de estos procesadores puede, también, ajustarse a las tareas específicas que deben desarrollar (adquisición de datos, cálculos complejos, almacenamiento de información, control del estado de determinadas variables, actuación sobre elementos diversos, atender a comunicaciones externas, etc.)

Al margen de la utilidad de un sistema de estas características en diversos campos, sería aplicable al control de procesos por computador y estaría especialmente indicado como un elemento que puede proporcionar altas prestaciones en sistemas de control distribuido.

5.2 ADAPTACIÓN DEL SISTEMA INFORMÁTICO REMOTO AL CONTROL DE ESTACIONES DE CAMPO CON SISTEMAS DE ADQUISICIÓN DE DATOS AUTÓNOMOS.

Un caso particular a la situación planteada en el apartado anterior, se presenta cuando en las estaciones de campo de un sistema de control distribuido se dispone de distintos tipos de dispositivos de adquisición de datos (estaciones de medida autónomas EMA), capaces de discriminar la información útil y almacenarla en forma de ficheros de datos.

Las estaciones de medida a las que se hace referencia están destinadas a la elaboración de información relativa a eventos físicos variados, presentando en común el hecho de disponer de un puerto serie (normas RS232C) y un protocolo de enlace conocido. Mediante la conexión de un ordenador personal a este puerto serie puede vaciarse la información de la estación para procesarla con posterioridad. La misma conexión permite acceder a la estación para modificar su configuración en función de los requisitos que determine el usuario. Sistemas de estas características utilizados en el registro de: datos medioambientales, datos sobre eventos sísmicos, datos hidrográficos, etc., están hoy en día comercializados, siendo bastante usual el procedimiento utilizado para descargar la información contenida en los mismos.

Disponiendo de estos sistemas en las estaciones de campo el trabajo de los sistemas informáticos remotos (SIR) se reduce sensiblemente. La adquisición, la elaboración de ficheros de datos y el almacenamiento de la información se realiza por la EMA. La actuación del SIR sobre ésta se limitará a una supervisión para comprobar su correcto funcionamiento y para solicitar los ficheros de datos, cuando estos sean creados, a fin de ponerlos a disposición del centro de registro para su evaluación.

El SIR también servirá de puente para permitir a un operador, situado en el centro de registro, acceder a la EMA a través de la red de ordenadores para su parametrización. Una función adicional que puede asumir este sistema es hacerse cargo de la supervisión de determinados elementos de la estación, por ejemplo, la comprobación de los niveles de tensión de la alimentación eléctrica (en muchas ocasiones esta alimentación provienen de baterías solares) generando avisos al centro de registro cuando se sobrepasen ciertos umbrales. Y, para dar mayores prestaciones a la red de ordenadores, el SIR podría actuar sobre reles para efectuar "*reinicializaciones hardware*" sobre elementos activos de la estación de campo (EMA, emisora de radio-TNC y el propio SIR), activados también desde el centro de registro a voluntad del operador de la red.

Para realizar estas tareas puede diseñarse un dispositivo multicomputador como el descrito en el apartado anterior, pero, dadas las funciones más limitadas que deben abordarse, bastaría con la utilización de dos microcontroladores para su confección. Teniendo en cuenta además que, tanto el nodo de la red de comunicaciones como la EMA disponen de un acceso bajo normas RS232C, los procesadores que se utilicen en su construcción, deberían disponer de un puerto serie con un buffer controlado por *hardware* para facilitar la conexión del SIR a estos sistemas.

Cada microcontrolador dispondría de su propio sistema operativo, atendiendo a la red de ordenadores (μC1) y a la EMA (μC2) de forma independiente. La comunicación entre los procesadores puede realizarse a través de una memoria compartida, que pueden utilizar también para realizar operaciones de tipo temporal. De esta forma no es necesario que los microcontroladores utilizados dispongan de una memoria interna amplia, lo cual encarecería el producto.

Es importante que la memoria externa utilizada en la confección del SIR, disponga de funciones de reloj y de una batería para actuar en caso de fallo de la alimentación externa (memoria no volátil, SRAM). Esto permitiría mantener actualizados los datos de fecha y hora del sistema para que las anotaciones en el

fichero de errores del SIR sean concordantes con la situación que las produjo y para mantener los valores de ciertos parámetros de trabajo, residentes en memoria, en caso de pérdida de la alimentación eléctrica externa. Las memorias que reúnen estas características suelen disponer también de una lógica de control que monitoriza de forma continua la tensión de alimentación (V_{cc}), para variaciones fuera de un rango de tolerancia. Cuando ocurre esta condición, se habilita incondicionalmente una protección contra escritura con el fin de prevenir un posible deterioro de los datos debido al estado indeterminado y transitorio de las señales en el *bus* en el instante que el hecho se produce.

En el mapa de la memoria de uso compartido debe establecerse un *área de comandos* para el intercambio de instrucciones entre ambos μC. Para el dimensionamiento de este área debe preverse que los comandos pueden tener parámetros asociados. También se definirá un *área de diagnóstico* para la anotación de errores. El resto del espacio disponible en la memoria se destinará al *área de datos*.

Como las conexiones del sistema con el exterior son escasas: conexión a la EMA y al nodo de la red de comunicaciones, que se realiza a través de los puertos serie de cada microcontrolador; y actuación sobre relés y circuito para supervisar la tensión de alimentación externa, que se puede realizar a través de puertos de E/S de los microcontroladores; el resto de estos puertos puede utilizarse para confeccionar el bus de datos y de direcciones para acceso a la memoria externa. Así puede diseñarse un bus de datos con 8 líneas de forma que puede transmitirse un *byte* a cada señal de reloj haciendo muy efectiva la transmisión interna.

La existencia únicamente de dos procesadores permite diseñar un sencillo algoritmo de contienda para el acceso de estos a la memoria compartida. Como soporte físico de este algoritmo puede utilizarse una sola línea de control. Otra línea de control debe habilitarse para indicar el modo de funcionamiento de la memoria (lectura o escritura).

Para que el acceso compartido de los microcontroladores a la memoria común sea posible, es necesario la utilización de dispositivos *triestado*[3] para una correcta implementación del bus [Mandado_87]. Los puertos de los microcontroladores y de la memoria deben cumplir esta condición para que operen correctamente.

Además de las líneas de direcciones, datos y control, se propone la utilización de dos líneas anexas, cuya finalidad es la de proporcionar las señales de interrupción necesarias para el tratamiento de sucesos entre ambos procesadores. Cuando un microcontrolador desea avisar al otro del envío de un comando, activará esta interrupción después de depositar el comando en una dirección específica de la memoria. Esto agilizará el intercambio de mensajes entre los procesadores.

Otro elemento importante en la confección del SIR es la integración de un subsistema que evite que el primero quede bloqueado por cualquier causa no prevista, tanto por el *firmware*[4] como por el *hardware* (subsistema de *Reset y Watch-dog*). El subsistema debe diseñarse de forma que si existe una eventual perdida de control, se produzca una reinicialización. Esta configuración permitirá, así mismo, la generación de la señal de *reinicialización* necesaria en el instante de la conexión o activar esta señal a voluntad mediante una instrucción externa.

La frecuencia del reloj de referencia para la ejecución de los procesos internos se eligirá de 11'059 MHz. Esta frecuencia permite, mediante las oportunas divisiones, obtener velocidades de transmisión estándar (2400, 4800, 9600, 19200, etc.). Al mismo tiempo proporciona, junto con la arquitectura propuesta para el SIR, unas buenas características en cuanto a velocidad de proceso.

---

[3] *Dispositivos triestado:* Los posibles estados de estos elementos son: +V, -V ó alta impedancia (carga pasiva). Permiten enviar un estado lógico o quedar inhibidos para que otros dispositivos, conectados al mismo bus, puedan ocupar el canal.

[4] *Firmware:* Microcódigo asociado a un procesador que se ubica en una memoria de solo lectura (puede residir en el propio procesador o en un dispositivo externo).

El SIR deberá incorporar además otros dispositivos como:

- Circuito integrado MAX232. Destinado a la conversión de niveles RS232C a los niveles TTL utilizados por los procesadores.
- Regulador de tensión. Destinado a estabilizar la alimentación eléctrica del SIR.
- Otros componentes electrónicos adicionales (inversores, resistencias, condensadores, etc.). Necesarios para la perfecta coordinación del sistema.

En la figura 5.1 puede verse un diagrama de bloques del diseño propuesto para este SIR específico.

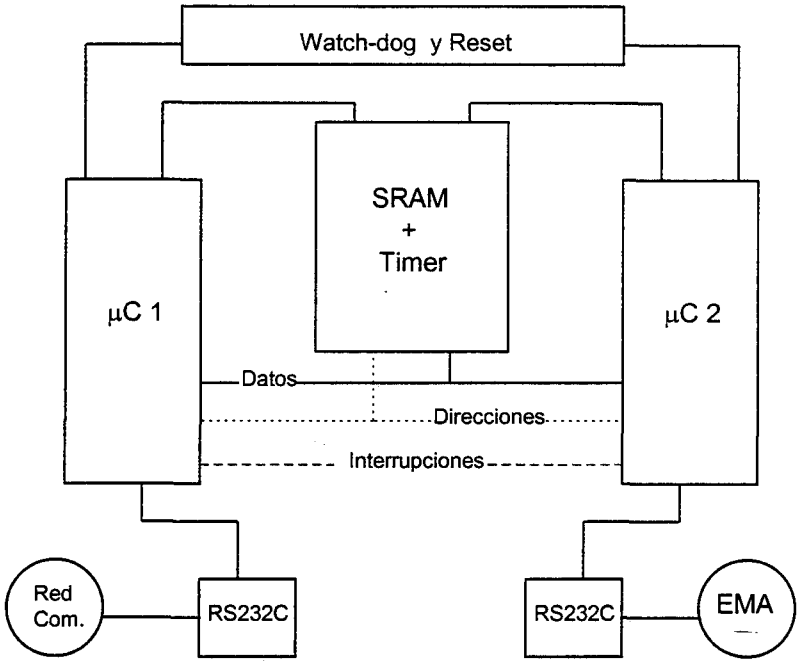

Figura 5.1 Diagrama de bloques del SIR

En cuanto al *firmware* necesario para que este *hardware* pueda realizar las funciones requeridas puede dividirse en dos grupos. El primer grupo lo constituirían un conjunto de funciones diseñadas para realizar las operaciones internas del SIR y proporcionar una salida hacia el exterior. Conformaría lo que puede denominarse el *sistema operativo básico* del SIR . Dentro de estas funciones pueden significarse:

- Funciones de inicialización del sistema y control del Watch-dog.
- Funciones de acceso y liberación de la memoria compartida
- Funciones de lectura y escritura sobre la memoria.
- Funciones para el procesado de interrupciones.
- Funciones de envío y recepción de datos por el puerto serie.
- Funciones lógicas de propósito general. Destinadas a facilitar la ejecución de ciertas tareas rutinarias como: sumas, restas, comparaciones, generación de retardos, conversiones de código, etc.
- Funciones de diagnóstico. Para controlar un fichero de errores alojado en un área de memoria y cuya recuperación, en el centro de diagnostico, permitirá una evaluación del estado del sistema.

Ambos microcontroladores dispondrían de un *sistema operativo* (SO) de estas características con algunas diferencias mínimas entre sí. Existirían diferencias en el algoritmo de contienda para el acceso a memoria, en cuanto a los tiempos de espera entre dos intentos de acceso consecutivo, para poder solventar las situaciones de colisión que pudieran producirse. El área de memoria para anotación errores de diagnóstico será diferente para cada μC. Y también serán diferentes las funciones lógicas de propósito general, incorporándose en cada uno de ellos solo aquellas que vayan a utilizarse en función de los requerimientos de la aplicación de control.

La otra parte del *firmware* estará compuesta por las funciones necesarias para implementar los protocolos definidos para la red de ordenadores (solo en el μC1) y las funciones requeridas para confeccionar la aplicación específica a la que se desea destinar el SIR (distribuidas entre ambos μC). Estas funciones se sustentarán en el sistema operativo especificado para cumplir sus objetivos.

La programación debe hacerse en forma modular para facilitar una sencilla adaptación a distintas aplicaciones y realizar actuaciones sobre el SO básico, sin

afectar al funcionamiento del sistema. Con esto se obtiene un código más compacto y mayor facilidad a la hora de localizar posibles errores.

La implementación práctica de un dispositivo que reúne las especificaciones descritas en este apartado, ha sido desarrollada en el Departamento de Ingeniería de Sistemas y Comunicaciones de la Universidad de Alicante, siendo objeto de una patente de invención [Rosa_96].

CAPITULO 6

Utilización de la red de ordenadores como soporte para realizar un control distribuido sobre estaciones de registro sísmico

- ***Características específicas del proceso***
- ***Aplicación de control. Modos de operación***
- ***Ventajas que aporta este procedimiento***

6.1 CARACTERÍSTICAS ESPECÍFICAS DEL PROCESO

Con objeto de comprobar experimentalmente el funcionamiento de la red de ordenadores, la operatividad del sistema informático remoto y ofrecer, a la vez, una solución práctica a los requerimientos planteados por los gestores de la red de microsismicidad de la provincia de Alicante, se confeccionó un procedimiento que permitiera controlar, desde el centro de proceso ubicado en el Campus de la Universidad de Alicante, un conjunto de estaciones de medida autónoma destinadas a registrar eventos sísmicos. Las estacines de medida se encuentran distribuidas en puntos dispersos de la geografía provincial.

La idea básica del procedimiento propuesto es llevar a cabo un control distribuido de las estaciones de medida autónomas (EMA), encargándose los sistemas informáticos remotos (SIR), ubicados en cada estación de campo, de realizar un control local sobre la EMA. La red de ordenadores hace posible que desde el centro de registro, se pueda ejercer un control de orden superior sobre las estaciones de medida, realizando sobre estas todas las operaciones que el usuario podría efectuar con una conexión directa desplazándose a su lugar de ubicación, incluida la posibilidad de efectuar una reinicialización (*reset*) *hardware*.

Las estaciones de medida utilizadas son sistemas autónomos capaces de discriminar y almacenar eventos sísmicos, generando ficheros de datos a partir de las señales analógicas suministradas por un transductor. Disponen de una conexión a un sistema GPS (*Global Position System*), que le permite confeccionar estos ficheros con tiempos de referencia comunes para todas las EMA de la red. La conexión a una EMA para extraer los ficheros de datos, parametrizarla y realizar otras funciones que

tiene programadas, se lleva a cabo mediante una conexión RS232C, utilizando un protocolo de enlace estándar (modo terminal).

El elemento que controla directamente cada EMA es el SIR. Este está conectado a la EMA por uno de sus puertos serie, emulando su protocolo de enlace e incorporando el *software* necesario para realizar con la EMA todas aquellas funciones que son posibles con la misma (conexión, petición de estado, solicitud de ficheros de datos, desconexión, parametrización, etc.). La ejecución de alguna de estas funciones se solicita directamente por el SIR de forma automática y otras se solicitan, desde el centro de registro, por el operador de la red.

Los SIR situados en las estaciones de campo junto con el ordenador personal (PC) del centro de registro conforman los sistemas informáticos de la red de ordenadores, de acuerdo con las especificaciones descritas en la memoria. En ellos se ejecutan los procesos que deben comunicar entre sí para efectuar el control distribuido de las EMA.

El PC es el elemento que proporciona el interfase de usuario para operar con la red, establece sesiones con las estaciones de campo para el intercambio de mensajes, bien por iniciativa propia o a petición de alguna de ellas, y gestiona el tráfico de datos en la red de ordenadores controlando su correcto funcionamiento. La subred de comunicaciones que hace posible la interconexión de los sistemas informáticos es la red digital de radio descrita en el capítulo 3.

El procedimiento propuesto contempla que la recopilación de ficheros de datos en el centro de registro, se lleve a cabo de forma automática por los sistemas que conforman la red. La actuación específica sobre una determinada estación de campo, para realizar una función distinta, requiere de la actuación de un operador que utilizará los servicios proporcionados por la red de ordenadores para este fin.

El *software* diseñado para cumplir este objetivo se encuentra especificado en loa apéndices I y II de esta memoria.

Apéndice I : *Software* residente en el PC de la estación central

Apéndice II : *Software* residente en los SIR de las estaciones de campo

a) *Software* del μC1. Controla la comunicación con la red y dialoga con μC2 a través de SRAM.

b) *Software* del μC2. Controla la estación de medida y dialoga con μC1 a través de SRAM.

Estos programas contienen la aplicación de usuario y el *software* de red necesarios para que el control pueda realizarse de forma distribuida. Para su elaboración han sido de gran utilidad las recomendaciones encontradas en: [Angulo_89], [Ceballos_94], [IAR_89] e [Intel_83].

En este ejemplo práctico existen dos diferencias en la red de ordenadores, respecto al diseño final de la misma que figura en la memoria. En primer lugar, las aplicaciones específicas de la red *transferencia de archivos* y *entrada remota de trabajos*, forman parte de la aplicación de usuario. Como se ha mencionado en el capítulo 4 la confección de este *software* es anterior a la formalización definitiva de la arquitectura de la red de ordenadores. La previsión inicial era que los servicios de la red finalizaran en el nivel de Sesión-LL, que proporcionaría a la aplicación de usuario los servicios: CONNECT, DATA y DISCONNECT. Así está elaborado el *software* de red de este caso particular.

Una vez confeccionada la aplicación de usuario y habiendo desarrollado en la misma un procedimiento para transferir archivos y otro para ejecutar tareas en sistemas distintos a aquel que lo solicita, se pensó incluir estos servicios como aplicaciones específicas de la red para ponerlos a disposición del usuario de la misma (aplicación de control). De esta forma si se desea confeccionar una aplicación de usuario diferente, para controlar un proceso distinto, pueden utilizarse estos servicios sin necesidad de volver a diseñarlos.

Dado el lugar que el nivel de aplicación de red ocupa en la estructura de la arquitectura (nivel superior de la jerarquía), la utilización de estos servicios es

opcional. Consideramos que, para la mayoría de situaciones en la que el uso de esta red resulte satisfactorio, también será útil disponer de los servicios de aplicación específica propuestos, de ahí que estos se incorporen al diseño definitivo de la red. No obstante, si en algún caso particular resulta más efectivo otro mecanismo distinto de intercambio de información entre las aplicaciones de usuario, puede prescindirse de las aplicaciones específicas en red y utilizar el resto de su estructura como plataforma para interconectar las aplicaciones de usuario entre sí.

La otra diferencia en este ejemplo práctico está en la forma de realizar el establecimiento de sesiones. Cuando un evento sísmico tiene lugar, prácticamente todas las estaciones de la red detectarán el suceso (dependerá de su intensidad y de la sensibilidad de los sistemas de medida). Ante esta circunstancia es aconsejable que, cuando una estación de campo solicite el establecimiento de una sesión para comunicar alguna información, la estación central establezca una sesión con todas las estaciones de la red en previsión de que esa comunicación contenga información de un movimiento sísmico. Si no se procede de esta forma, la recuperación de un fichero puede ser interrumpida por colisiones con solicitudes de establecimiento de sesiones de otras estaciones.

Por este motivo se diseña en la estación central una función para conectar *todas* las estaciones de campo de la red. Para que esta resulte más eficaz en su labor, la estación central debe ignorar las solicitudes de establecimiento de sesión de las estaciones remotas actuando siempre como entidad emisora. Para lograr esto, basta configurar el nodo de la subred de comunicaciones al que está conectado el PC del centro de registro, de forma que éste no admita ninguna conexión de enlace que no haya solicitado él mismo. Así, se penalizan los casos en los que una estación remota desea comunicar datos de un aviso, pero se consigue mayor eficiencia en la recuperación de ficheros de datos, aspecto de mayor interés para la gestión de la red de microsismicidad.

Cuando la aplicación de usuario, residente en el sistema de la estación central, solicita a la red el establecimiento de una sesión con una estación de campo, todo se

desarrolla conforme a la especificación descrita en la memoria para el nivel de Sesión-LL. Ahora bien, si es una estación de campo la que solicita una sesión con la estación central, cuando lance la trama conteniendo la solicitud del enlace previo, obtendrá siempre como respuesta una trama de sistema (S) del nodo de la subred, informándole que el destino está ocupado. Siguiendo el diagrama de estados de la MEF del nivel Sesión-LL, definido para el sistema emisor, puede observarse que este quedaría en espera de una conexión que no llegaría a producirse.

Esta situación se solventa fácilmente si conseguimos que los SIR actúen siempre como entidades receptoras a la hora de establecer una sesión con la estación central. Así, el SIR emitirá una trama solicitando una conexión del enlace lógico (C,Connect) y en lugar de pasar al estado *esperando conexión*, continuará en el estado *inicio*. El nodo al que está conectado la estación central no acepta esta conexión de enlace, pero el sistema es avisado mediante una trama (R,Connect) y es éste el que solicita la sesión con la estación remota que originó la indicación. Como precaución el SIR que origina la solicitud activa un tiempo de espera cada vez que lanza una trama (C,Connect) y vuelve a enviarla si el tiempo vence, por si su trama colisionó en el medio.

Entendemos que el caso práctico propuesto es una situación particular, por lo que en la especificación del nivel de Sesión-LL realizado en la memoria, se ha optado por definir el caso más general en el que las sesiones pueden ser establecidas a solicitud de cualquier sistema conectado en red.

6.2 APLICACIÓN DE CONTROL. MODOS DE OPERACIÓN

Para llevar a cabo el trabajo propuesto se diseña una aplicación de control, distribuida en los sistemas que componen la red de ordenadores, en la que se definen tres *Modos de Operación* en función de los distintos trabajos que se efectúan:

- Modo Normal
- Modo Transparente
- Modo Parametrización

MODO NORMAL. Este modo de operación está relacionado con la transferencia de ficheros de datos de las estaciones de campo al centro de registro. También se transmiten los avisos de anomalías producidas en las estaciones de campo a la estación central. El sistema de la estación central es el que proporciona el interfase de usuario, de forma que estas anomalías pueden comunicarse al operador de la red a través del mismo.

Todas las estaciones arrancarán en este modo de operación. La estación central establecerá una sesión con las estaciones de campo cada T_s (T_s se determinará en función de las características específicas del proceso a controlar, en este caso práctico se ha fijado en una hora, siendo su valor programable), con el único fin de actualizar su LISTA_DE_ESTACIONES_ACTIVAS, y tener informado al operador de la red de las variaciones que se produzcan.

Por su parte el SIR de las estaciones de campo, al arrancar en este modo de operación, establece un enlace con su EMA y la interroga regularmente por si dispusiera de algún fichero de datos para ser transmitido.

El medio permanece sin tráfico mientras no se produzca un evento (fichero de datos o aviso por anomalías) en alguna estación de campo o venza T_s . Producido un evento, la estación que lo registra emite una trama (C,Connect) a la estación central indicando a esta última que desea se establezca una sesión para comunicarle un suceso. La estación central establece una sesión con las estaciones de campo de la red e interroga secuencialmente a estas, comenzando por la que originó la indicación. A partir de este momento, la coordinación de la comunicación está centralizada en el PC del centro de registro evitándose las colisiones en la red.

La aplicación de control, para recuperar los ficheros que contienen datos relativos a eventos sísmicos, está diseñada de forma que la estación del centro de registro solicita información de un determinado SIR hasta que vacía el contenido de su memoria (el contenido de la memoria del SIR puede ser un fichero completo o parte del mismo). Este hecho es indicado a la aplicación de la estación central mediante un comando específico. Recibida la notificación, la estación central continuará solicitando ficheros de datos a otro SIR mientras que el anterior recupera de su EMA la continuación del fichero que acaba de transmitir (si esta transmisión era el contenido parcial de un fichero).

Cuando la aplicación de la estación central acaba el ciclo de solicitudes a todos los SIR y vuelve al primero, este ya ha recuperado de la EMA la siguiente porción de fichero e inicia la transmisión de forma inmediata. El mecanismo permite que la recepción de datos significativos en el centro de registro sea continuada, y los tiempos que el SIR emplea en recuperar fragmentos de fichero, son aprovechados por la estación central para obtener porciones de fichero pertenecientes a otros SIR. Cuando los ficheros de cada EMA de la red son enviados de forma completa y la estación del centro de registro recibe un mensaje de "*no datos*" de todos los SIR, la estación central finaliza el proceso de sondeo y libera las sesiones, permaneciendo de nuevo el medio en silencio.

Las únicas colisiones posibles pueden producirse en el establecimiento de las sesiones. En este paso del proceso la red de ordenadores proporciona los mecanismos

para resolver estas situaciones. Según se ha especificado en el servicio CONNECT del nivel de Sesión-LL la entidad que actúa como emisora de la solicitud de sesión (en este caso ubicada en el sistema central), es la responsable de arbitrar estos mecanismos.

Dentro de este modo se define un conjunto de comandos (sería el equivalente al CCA [campo de control de la aplicación] definido para las aplicaciones específicas en red), que permitirá realizar a la aplicación de usuario las funciones atribuidas al modo Normal. En la tabla 6.1 se especifican los comandos utilizados.

C. enviado por E. central	C. enviado por E. de campo	FUNCIÓN
SONDEO		Solicitud para que se envíe información (datos aviso a datos de fichero)
	NO_DAT	No hay datos
	WAIT	Espera
	DAT_AV	Datos de aviso
	DAT_FI	Datos de fichero
	SVB	Secuencia de verificación de bloque de datos fichero
ACK_AV		Datos de aviso recibidos
ACK_FI NACK_FI		Datos correctos \ sólo se producen después de Datos incorrectos / comprobar SVB
	C_MEDIO	Devolución del control del medio a la E. Central
TNACK_P	TNACK_R	Campo CCA recibido es erróneo o formato de la APDU mal conformado

Tabla 6.1

El modo de operación normal sólo se abandona a petición del usuario de la red para realizar actuaciones específicas con las estaciones de campo.

MODO TRANSPARENTE. Este modo de operación está relacionado con la entrada de trabajos en las estaciones de campo desde el centro de registro. Se accede al mismo a petición del usuario de la red y una vez que se determina la estación sobre la que va a actuar, la estación central establece una sesión con la misma y le envía un comando de puesta en modo transparente. Aceptado éste por la estación de campo, el operador estará en disposición de solicitarle que realice una serie de funciones y le devuelva los resultados.

Las funciones están programadas en el SIR de forma que a la estación central le basta enviar un comando para que estas se ejecuten. Dentro de este modo de operación se implementan todas las funciones necesarias para realizar un control efectivo de las estaciones de campo.

Finalizada la actuación específica con una estación se regresará al modo normal de operación.

MODO PARAMETRIZACIÓN. Este modo de operación también está relacionado con la entrada de trabajos en las estaciones de campo, pero estos trabajos hacen referencia a la parametrización de los elementos activos de los sistemas remotos: SIR y EMA.

A diferencia del modo anterior, cada comando de actuación irá acompañado de un fichero de datos suministrado por la estación central, donde se especificará a quién van dirigidos los parámetros (SIR ó EMA) e información relativa a que parámetros se han de modificar y con qué contenido.

La forma de entrar y abandonar el modo parametrización es igual que en el caso anterior.

Para que la aplicación pueda realizar las funciones asignadas a los modos Transparente y Parametrización, se define un conjunto de comandos. Estos comandos y su significado pueden encontrarse en los Apéndices I y II de la memoria.

La tabla 6.2 muestra los comandos definidos para realizar los cambios de *modo de operación* en la aplicación de usuario.

C. enviado por E. central	C. enviado por E. de campo	FUNCIÓN
MODO_N		Solicitud para que la estación de campo se ponga en modo normal de operación
	R_MDN	E. de campo en modo normal
MODO_T		Solicitud para que la estación de campo se ponga en modo transparente
	R_MDT	E. De campo en modo transparente
MODO_P		Solicitud para que la estación de campo se ponga en modo parametrización
	R_MDP	E. De campo en modo parametrización

Tabla 6.2

El diseño de los procesos principales que se ejecutan en cada SIR y en la estación central incluye un *procedimiento de diagnóstico* que refleja, en un *fichero de errores*, las situaciones de anómalas que pueden producirse en el funcionamiento de la red. El análisis del fichero de errores permite conocer al detalle todo lo que acontece en la red, facilitando su mantenimiento. La implementación *software* de un procedimiento de diagnóstico *inicial*, como paso previo al desarrollo de las aplicaciones de control y al *software* de red, ha sido de vital importancia para progresar con cierta agilidad en la elaboración de los programas diseñados. Este procedimiento de diagnóstico inicial ha ido evolucionando paralelamente al desarrollo de las aplicaciones, hasta convertirse en el producto final que utiliza la red para evaluar su operatividad.

Seguidamente se especifica la aplicación de control mediante el diagrama de estados de una MEF para obtener una apreciación más clara de su funcionamiento. Para esta aplicación se definen dos MEF complementarias para los dos tipos de estaciones distintas que participan en la red. La MEF de la aplicación residente en la estación central consta de cuatro estados, figura 6.1

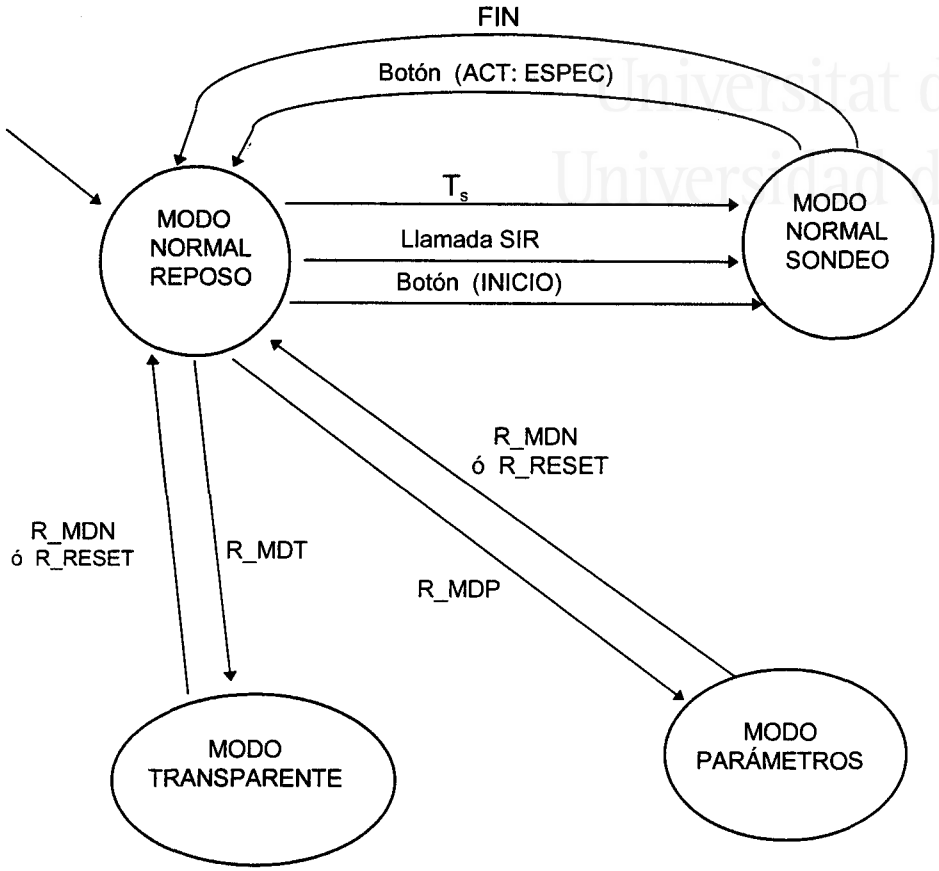

Figura 6.1 Diagrama de estados de la MEF de la Aplicación de usuario Estación Central

- **ESTADO** MODO NORMAL REPOSO. La MEF comienza en este estado, produciéndose un cambio del mismo cuando:

  1. Lo decida el operador de la red activando el botón "INICIO". Pasa al estado MODO NORMAL SONDEO.
  2. Recibe una llamada de una estación de campo para que la atienda. Pasa al estado MODO NORMAL SONDEO.
  3. Vence T_s. Pasa al estado MODO NORMAL SONDEO.
  4. Cuando a solicitud del operador de la red se le envía un comando MODO_T o MODO_P a una estación de campo y ésta devuelve la aceptación de este modo de operación con R_MDT o R_MDP. Pasará en ese caso al estado que define el correspondiente modo de operación.

- **ESTADO** MODO NORMAL SONDEO. Al entrar en este estado enviará una primitiva CONNECT.*request* a su capa de Sesión-LL para que se establezca una sesión con todas las estaciones de la red. Una vez establecidas las sesiones comenzará un ciclo de sondeo a las estaciones de campo para solicitarle los datos que pudieran tener almacenados. Finalizado el ciclo la MEF regresaría al estado de MODO NORMAL REPOSO. Si durante la ejecución del bucle de sondeo el operador de la red lo solicita (botón ACTUACIÓN ESPECÍFICA), la MEF abandonará este estado para regresar al MODO NORMAL REPOSO.

- **ESTADO** MODO TRANSPARENTE Ó MODO PARÁMETROS. Cuando la MEF entra en alguno de estos estados queda en espera, limitándose a gestionar las solicitudes del operador para que se ejecuten las funciones que pueden realizarse dentro de cada uno de estos modos. Estos estados se abandonan cuando se recibe una respuesta de la estación de campo específica con la que se está actuando, aceptando el regreso al modo Normal de funcionamiento con un comando (R_MDN). O bien, deben de abandonarse cuando se recibe una respuesta afirmativa a un comando P_RESET (comando que puede enviarse dentro de estos dos modos de operación para efectuar una reinicialización *hardware* en el SIR). Si se recibe un comando R_RESET implicará que el SIR arrancará de nuevo y lo hará en MODO NORMAL REPOSO, por lo que la MEF de la estación central debe regresar a este estado.

La MEF de la aplicación de control residente en las estaciones de campo consta de cuatro estados, figura 6.2.

- **ESTADO** MODO NORMAL REPOSO. La MEF comenzará en este estado abandonándolo únicamente si recibe un CONNECT.*indication* de su capa de Sesión-LL, en cuyo caso pasará al estado MODO NORMAL CONECTADO.

- **ESTADO** MODO NORMAL CONECTADO. Cuando entra en este estado la aplicación de usuario envía a su nivel de Sesión-LL un CONNECT.*response* y permanece en el mismo hasta recibir una primitiva DISCONNECT.*Indication*, pasando al estado MODO NORMAL REPOSO. Si estando en el estado MODO

NORMAL CONECTADO acepta un comando MODO_T o MODO_P, pasará al estado correspondiente.

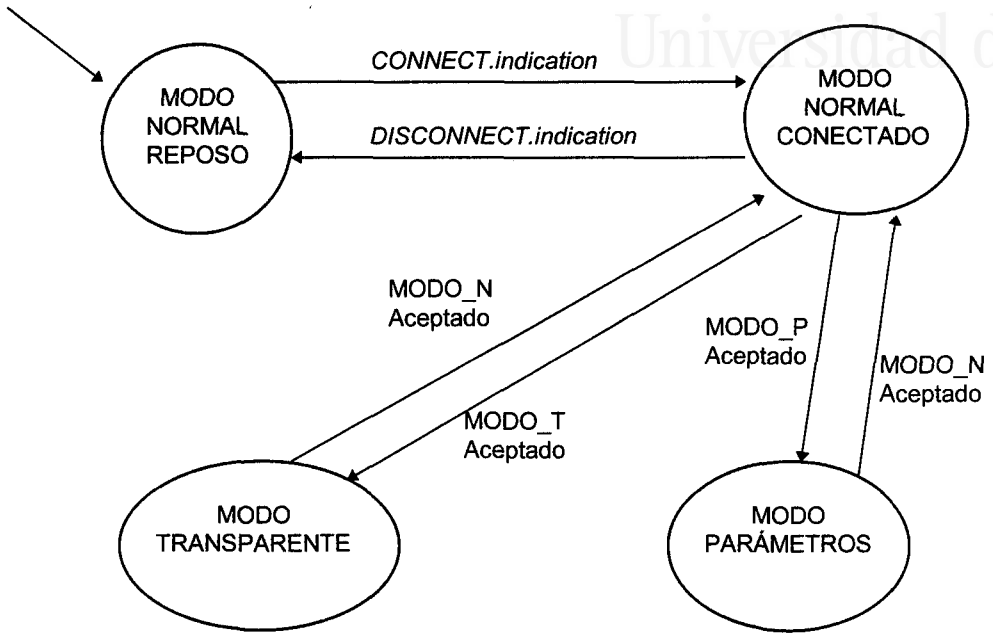

Figura 6.2 Diagrama de estados de la MEF de la Aplicación de usuario Estaciones remotas

- **ESTADO** MODO TRANSPARENTE Ó MODO PARÁMETROS. Al entrar en alguno de estos dos estados la aplicación de usuario envía la correspondiente respuesta de aceptación R_MDT o R_MDP. En estos estados se espera la llegada de los comandos definidos para estos modos de operación y ejecuta las funciones correspondientes. La MEF abandona estos estados cuando recibe un comando MODO_N y lo acepta. En ambos casos enviará la aceptación del mismo mediante el comando R_MDN y pasará al estado MODO NORMAL CONECTADO.

Dentro de cada uno de los estados especificados la parte de la aplicación de usuario residente en la estación central y la residente en los SIR de la estaciones de campo, se intercambiarán todos aquellos comandos que sean necesarios para que se ejecuten adecuadamente las funciones previstas en cada uno de los modos de operación.

Las figuras 6.3, 6.4, 6.5 y 6.6 muestran el aspecto que presenta el interfase gráfico de usuario, en los tres modos de operación definidos para la aplicación de control. En letra negrita aparecen los botones de las funciones que están habilitadas en cada uno de los modos. La función de superusuario (botón SUPER_U) puede ser activada por un gestor especialista de la red (necesita una palabra de paso). Esta opción permite labores de mantenimiento de la red (realizar el chequeo de la memoria de un SIR, cambiar de modo de operación al nodo de la subred de comunicaciones, desactivar manualmente un enlace lógico, etc.). Todas estas funciones son servicios adicionales de la aplicación de usuario, que amplían las prestaciones del procedimiento propuesto. En la figura 6.5 aparece habilitada esta última opción.

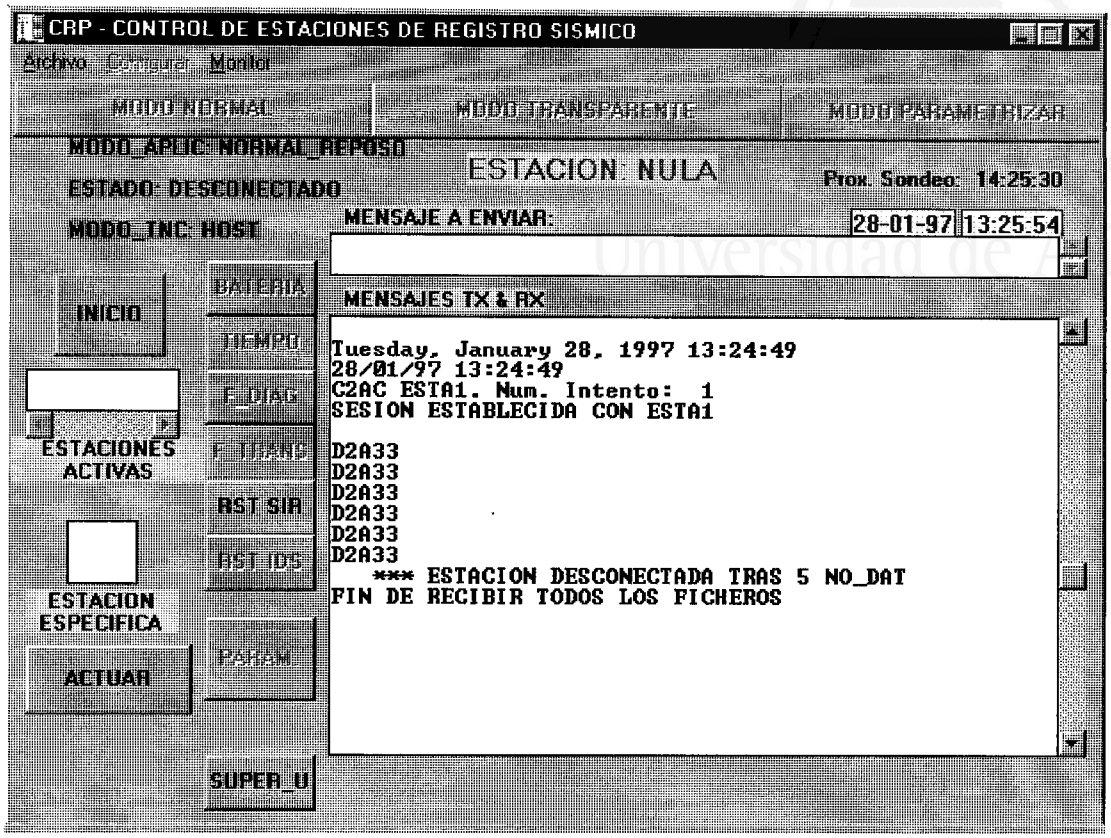

Figura 6.3 Aplicación de la estación central en modo normal

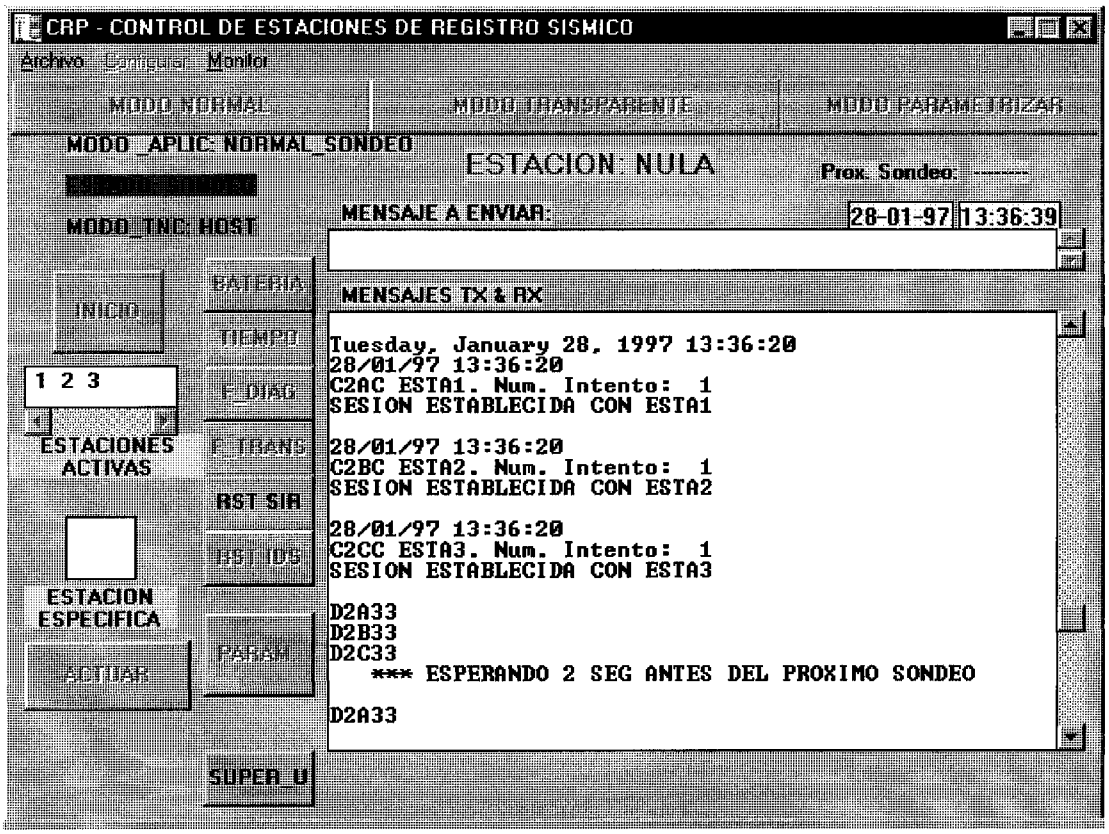

Figura 6.4 Aplicación de la estación central en modo normal (proceso de sondeo)

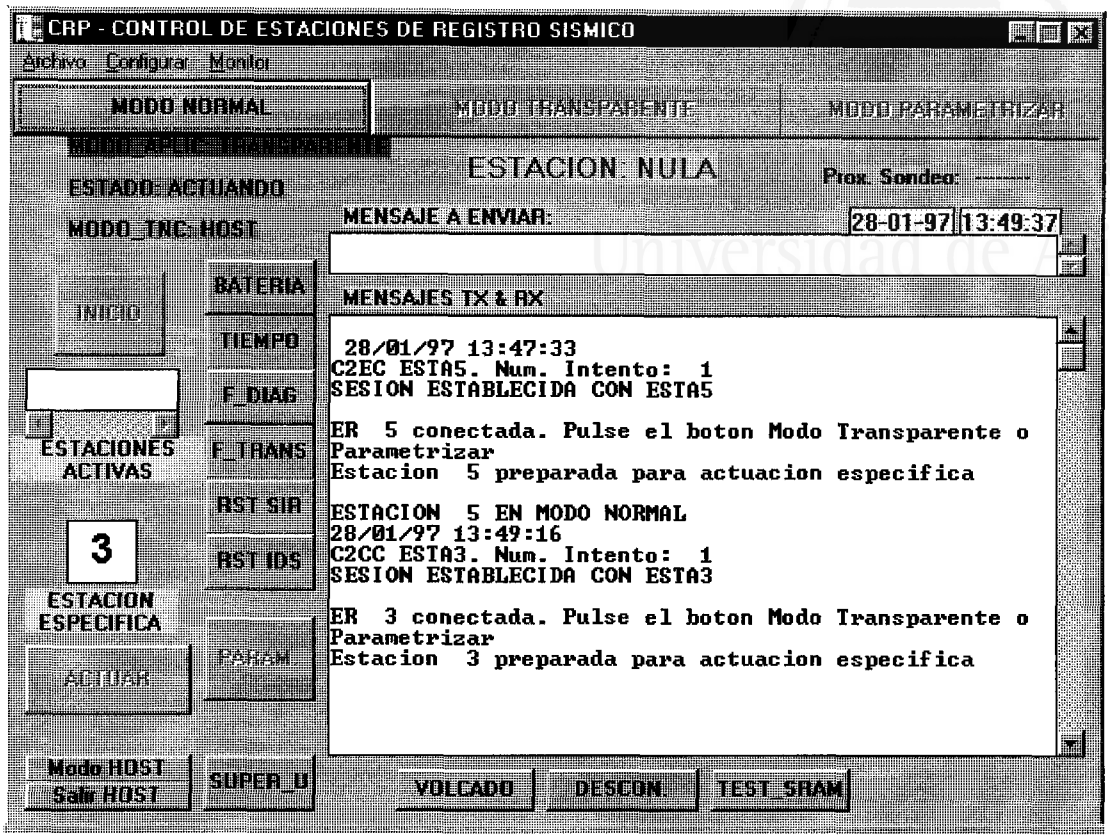

Figura 6.5 Aplicación de la estación central en modo transparente

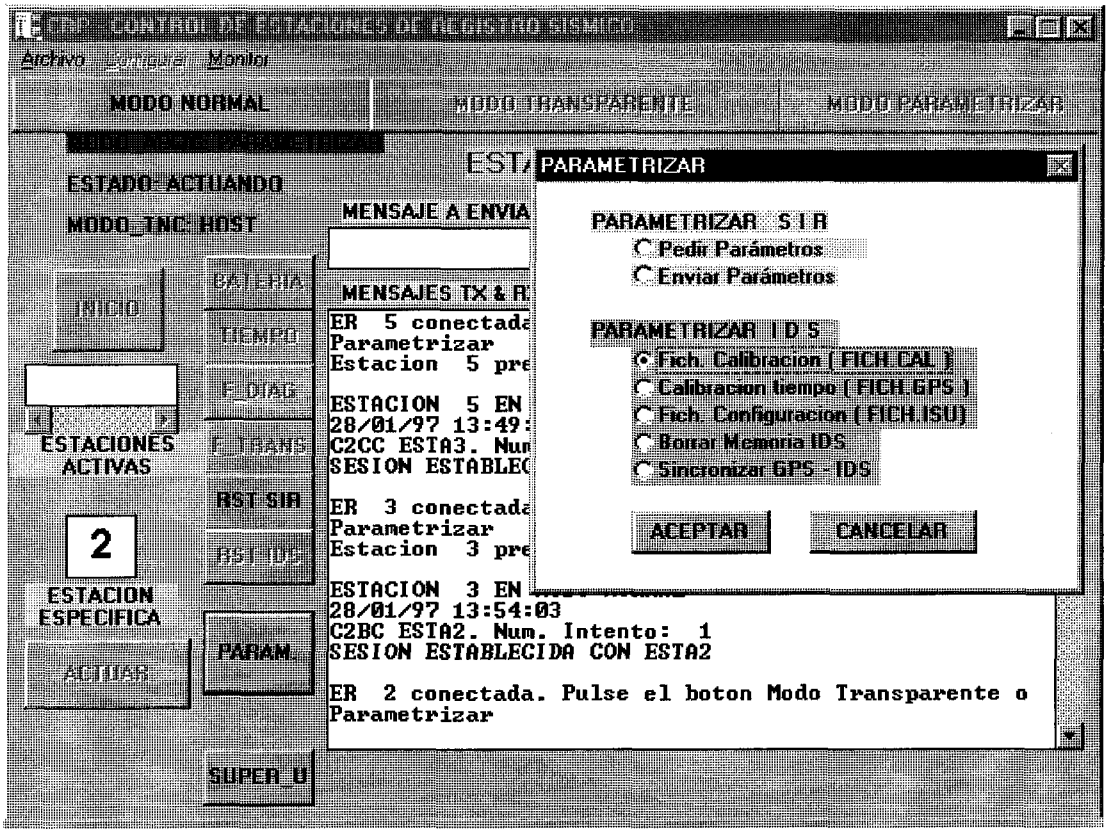

Figura 6.6 Aplicación de la estación central en modo parametrizar

6.3 VENTAJAS QUE APORTA ESTE PROCEDIMIENTO

La solución propuesta permite potenciar significativamente la capacidad de una red de microsismicidad, por varias razones:

La primera ventaja apreciable es la reducción de bandas de radiofrecuencia que se utilizan en los sistemas actuales. El procedimiento típico utilizado el una red de sismicidad se sirve de transmisiones de radio analógicas (12'5 kHz de ancho de banda) para transferir la información recogida por los sensores ubicados en las estaciones de campo al centro de registro. Esto supone la necesidad de que exista una frecuencia de radio distinta para cada enlace estación de campo-centro de registro. El procedimiento descrito en esta memoria se sirve de una sola banda de radiofrecuencia para interconectar todas las estaciones de la red. El número máximo de enlaces que pueden establecerse (26 según las características técnicas de los nodos de la subred utilizados), sobrepasa el dimensionamiento normal de una red de ámbito provincial.

Si fuera necesario ampliar el número de estaciones sería factible confeccionar dos subredes interconectadas entre sí, trabajando cada una de ellas en una banda de radiofrecuencia distinta. En cualquier caso la relación 26/1 es considerablemente ventajosa para este procedimiento, aunque el ancho de banda requerido (25 kHz, si se desea trabajar a 9600 bps) es mayor en el caso que nos ocupa.

La utilización de transmisiones digitales supone otra notable mejoría, ya que evita la pérdida de rango dinámico implícita en las transmisiones analógicas y garantiza la integridad de la información recibida en el centro de registro. Los posibles errores en la información recibida, debidos a interferencias en el espacio

radioeléctrico, son recuperados por la red de ordenadores proporcionando los datos tal como han sido capturados por la estación de medida.

Aunque la adquisición final de los datos no se produce en tiempo real la velocidad de transmisión utilizada es suficiente para que, en combinación con un análisis automatizado de los datos recibidos, se disponga de conclusiones en un tiempo óptimo para los requisitos de la red de microsismicidad.

La utilización de una red de ordenadores para el control de la red de microsismicidad permite automatizar la recuperación de datos en el centro de registro, manteniendo una vigilancia permanente sobre el estado operativo de la red. Además, facilita al usuario la posibilidad de actuar sobre los elementos ubicados en las estaciones de campo desde la consola situada en el centro de registro. Aspectos tan importantes como la supervisión de la alimentación eléctrica de las estaciones, la ejecución de un *reset hardware* en la EMA o el SIR, la parametrización de estos dispositivos y otras facilidades descritas en la memoria, son posibles gracias a las prestaciones de la red de ordenadores.

El análisis de los ficheros de diagnóstico, existentes en cada sistema y recuperables en el centro de registro a través de la red de ordenadores, proporciona un conocimiento minucioso de lo que acontece en la red de microsismicidad, simplificando su mantenimiento.

La existencia de un sistema informático en las estaciones de campo para realizar un control local permite llevar a cabo un tratamiento previo de los datos, por ejemplo, encriptándolos o comprimiéndolos, antes de ser enviados al medio. En el caso concreto de que la red de sismicidad sea una *red de vigilancia*, los datos necesarios para la obtención de una localización previa pueden ser elaborados por cada SIR y enviados al centro de registro inmediatamente. En un segundo paso se recuperarían los ficheros completos elaborados por las estaciones de medida para un análisis más detallado.

Otro aspecto de relevancia es la utilización de una red digital de radio como soporte de las comunicaciones. Una red de ordenadores similar a la descrita podría construirse, redefiniendo los niveles correspondientes, utilizando enlaces telefónicos. Esto solo sería posible si existe una infraestructura adecuada, pero aún así, el procedimiento propuesto es mucho más ágil en las transacciones de información entre los sistemas que integran la red. Debe tenerse en cuenta que para establecer un enlace lógico entre dos sistemas conectados a la red telefónica conmutada, es necesario realizar una llamada previa para conformar el circuito de datos.

Para que ambos procedimientos pudieran competir, se necesitaría disponer, en el segundo de los casos, de un enlace directo y en línea permanente entre cada estación de campo y la estación de registro (línea punto a punto). Esto sería desproporcionado puesto que normalmente los periodos de inactividad en el medio son mucho mayores que aquellos en los que se intercambia información. Por otra parte, el mantenimiento económico de este procedimiento sería insostenible. Cuando en la práctica se utiliza esta solución, las conexiones se realizan a través de la red telefónica conmutada con una única línea de entrada en el centro de registro. Si una estación de campo necesita enviar información, tiene que efectuar la llamada correspondiente y encontrar la línea libre para que la transmisión pueda llevarse a cabo.

La adaptación de este procedimiento a distintos tipos de estaciones de medida autónomas, que podemos encontrar en el mercado, es sencilla. Basta conocer el protocolo de enlace de las estaciones y las operaciones que pueden realizarse con las mismas. Implementadas estas nuevas funciones en los programas de control, contarían con la plataforma proporcionada por la red de ordenadores para ejecutarse desde cualquier punto de la red.

Diseñando sistemas informáticos más complejos para las estaciones de campo, en base a las consideraciones realizadas en el capítulo 5 de la memoria, se puede abordar la adquisición de datos desde estos dispositivos. De esta forma, se pueden integrar las estaciones de medida en el SIR de la estación de campo.

Como consecuencia podemos decir que, la versatilidad y las elevadas prestaciones que proporciona el procedimiento propuesto, unido al bajo coste de mantenimiento que requiere, hacen que el mismo aporte soluciones óptimas para mejorar la capacidad de una red microsismicidad, en concordancia con el objetivo inicialmente planteado.

Un trabajo relativo a este procedimiento, en el que planteamos la utilización de una red de ordenadores para el control de estaciones digitales de registro sísmico, así como su integración con los procedimientos de adquisición que se utilizan en la actualidad, fue presentado en una ponencia en la XXV *General Assembly of the European Seismological Commission* (Reykjavík, Iceland, septiembre de 1996). Un *Paper* conteniendo los aspectos fundamentales de este trabajo se encuentra publicado en *Seismology in Europe*, dentro del apartado *Data acquisition, theory and interpretation* [Jauregui_96].

CONCLUSIONES

CONCLUSIONES

Como resultado de este trabajo se han obtenido las siguientes conclusiones:

1. Confeccionar una red de ordenadores con sistemas informáticos dedicados a realizar un control distribuido de procesos, permite que estos puedan disponer de los servicios proporcionados por la red. Así, un conjunto de programas, datos y equipos son accesibles para cualquier sistema conectado a la red, que suministra los mecanismos para que el control se ejecute de forma eficiente.

2. La metodología utilizada para realizar el diseño de la red de ordenadores ha hecho posible que, todos los aspectos relevantes para elaborar una plataforma de interconexión ajustada a los requisitos específicos derivados de su utilidad final, hayan sido analizados con orden y rigurosidad. La estructura de la red, las características de la subred de comunicaciones, las aplicaciones en red y la arquitectura de red, que proporcionan las prestaciones más adecuadas para realizar un control de procesos remotos, son el resultado de un estudio sobre diferentes opciones que pueden plantearse. La alternativa elegida es la que mejor se ajusta a la peculiaridades de estos procesos, proporcionando cobertura a un amplio abanico de situaciones.

3. El análisis de las características del tráfico que debe cursar la subred de comunicaciones y de los condicionantes derivados de la ubicación dispersa de los sistemas conectados en red, pone de manifiesto las ventajas que ofrecen la redes de comunicaciones digitales de radio, frente a otras alternativas.

4. El diseño de un procedimiento mixto de acceso al canal compartido, proporciona un mecanismo que aprovecha al máximo las facilidades ofrecidas por los nodos de la subred de comunicaciones. Este procedimiento mantiene el medio en silencio si no existe información que intercambiar entre las estaciones, proporciona agilidad en cuanto a la comunicación de incidencias en la red de ordenadores y garantiza que, cuando existe información voluminosa que transmitir, la efectividad del canal no va a verse reducida por la existencia de colisiones en el medio.

5. La evaluación de prestaciones de la subred de comunicaciones, realizada en la memoria, suministra datos significativos acerca de su capacidad de trabajo así como información imprescindible para obtener su mayor productividad. El procedimiento seguido para la obtención de estos resultados, sirve de referencia para valorar la eficacia de otras redes de comunicación de similares características.

6. La estructura y funciones definidas para los niveles que conforman la arquitectura de la red de ordenadores, permiten disponer de los servicios necesarios para obtener la máxima productividad en el proceso de control y minimizar el número de procesos que se ejecutan para atender a la red de ordenadores, optimizando así, los recursos que los sistemas deben dedicar a la comunicación.

7. La especificación de los protocolos, que regulan el intercambio de información entre entidades pares ubicadas en sistemas diferentes, se debe realizar de forma exhaustiva (especificación informal y formal), con el fin de facilitar la comprensión de su funcionamiento, analizar en detalle su respuesta a los sucesos que les conciernen y proporcionar una descripción pormenorizada que acerque a su implementación.

8. Con la técnica desarrollada para especificar formalmente los protocolos se obtiene una descripción completa de cada protocolo, aunque este sea complejo. Al mismo tiempo esta técnica aporta una visión clara de su operatividad, ya que los aspectos

de control del protocolo son modelados mediante una máquina de estados finitos, que ilustra su funcionamiento mejor que un lenguaje de programación.

9. La definición de un proceso generador de entradas y otro de salidas, a la máquina de Mealy que modela el núcleo de un protocolo, contribuye a independizar los procesos en función de su actuación sobre las variables del protocolo. Así, el valor de estas variables es controlado por el proceso generador de entrada y el proceso generador de salidas para tomar sus decisiones, mientras que los procesos que tienen lugar en las transiciones de la máquina de Mealy solo se encargan de actualizar el valor de las variables.

10. La flexibilidad del sistema informático remoto propuesto en esta memoria posibilita dimensionar el número de procesadores que lo integran, adecuándolo a las necesidades puntuales del control local que debe efectuar. Una adaptación de este dispositivo, para confeccionar estaciones de campo que cuentan con sistemas de adquisición de datos autónomos, pone de manifiesto como se puede abordar la ejecución de distintas tareas de forma independiente y con una comunicación ágil entre las mismas.

11. Los planteamientos efectuados en esta memoria han sido comprobados experimentalmente, gracias a la elaboración de un procedimiento que utiliza la red de ordenadores diseñada, como soporte para interconectar una aplicación de control distribuida. Las ventajas que proporciona una solución de estas características, proyectada con la finalidad de mejorar las prestaciones de una red local de microsismicidad, quedan recogidas dentro del apartado 6.3.

12. El diseño final de la red de ordenadores es el resultado de un proceso interactivo entre una metodología de diseño y las conclusiones obtenidas a partir de pruebas experimentales realizadas con los elementos que conforman la red. Los programas que contienen la aplicación de usuario y el *software* de red (apéndices I y II), han seguido un camino paralelo, en su elaboración, al diseño de la red de ordenadores. La implementación del *software* de red que aparece en estos apéndices no es la

más idónea, si se parte de una especificación formal de los protocolos tal como se presenta en la memoria. Sin embargo, se ha preferido mantener, porque además de ser una implementación válida y operativa (está en funcionamiento desde marzo de 1996), ayuda a entender el planteamiento final de la arquitectura propuesta para la red de ordenadores.

No obstante, una vez formalizada la arquitectura y especificados los protocolos de los niveles que la conforman, se puede abordar desde otra perspectiva la implementación del *software* relativo a la red. Las indicaciones sobre como llevar a cabo esta implementación quedan recogidas en el apartado 4.5.3

BIBLIOGRAFÍA

BIBLIOGRAFÍA

[Abramson_85] N. Abramson, "Development of the ALOHANET", Trans. on Inform. Theory IEEE, vol. IT-31, pp.119-123, Marzo 1985

[Abramson_94] N. Abramson, "Multiple access in wireless digital networks", Proc. IEEE, vol 82, no. 9, pp. 1360-1370, Septiembre 1994

[Alabau_86] A. Alabau y J. Riera, *Teleinformática y redes de computadores* 2ª Ed., Ed Marcombo, Barcelona 1986

[Alonso_96] J. Alonso, *Protocolos de comunicaciones para sistemas abiertos,* Addison-Wesley Iberoamericana S.A., Wilmington, Delaware, E.U.A. 1996

[Angulo_89] J. Angulo, *Microprocesadores* 5ª Ed. , Ed. Paraninfo, Madrid 1989

[Aracil_87] R. Aracil y A. Jiménez, *Sistemas discretos de control. (Representación externa),* Cátedra de Automática E.T.S.I Industriales, U.P., Sección de Publicaciones, Madrid 1987

[ARRL_84] American Radio Relay League, *AX.25 amateur packet-radio link-layer protocol, version 2.0,* Newington, CT. 1984

[Beltrào_89] J. Beltrào, J. Sauvé, W. Ferreira y J. Marinho, *Redes locales de computadoras. Protocolos de alto nivel y evaluación de prestaciones,* Mc Graw-Hill / Interamericana, Madrid 1989

[Bertoni_94] H. Bertoni, W. Honcharenko, L. Rocha and H. Xia, "UHF propagation prediction of wireless personal communications", Proc. IEEE, vol 82, no. 9, pp. 1333-1358, Septiembre 1994

[Black_87] U. Black, *Computer networks: protocols, standarts and interfaces,* Prentice-Hall International, Englewood Cliffs 1987

[Bochmann_80] G. Bochmann (Von) y L. Sunshine, "Formal methods in communication protocol design", Trans. on Commun. IEEE, vol. COM-28, no. 8, pp. 642-731, Abril 1980

[Booch_83] G. Booch, *Software engineering with Ada*, Benjamin-Cumming, Menlo Park, California 1983

[Candelas_96] F. Candelas, *Propuesta de sistema multiprocesador para aplicaciones de control. Diseño de la red de interconexión,* Proyecto S. informáticos, E.P.S., Universidad de Alicante 1996

[Carse_94] P. Carse and G. Garrard, "A continental drift toward wireless data", Data Commun. International, vol. 23, no. 5, pp. 68-74, Marzo 1994

[Chu_74] W. Chu, "Optimal message block size for computers communications with error detection and retransmission strategies", Trans. on Commun. IEEE, pp. 1516-1525, Octubre 1974

[Ceballos_94] F. Ceballos, *Enciclopedia de Microsoft Visual Basic,* Ed. Ra-ma, Madrid 1994

[Cox_86] B. Cox, *Object oriented programming,* Addison-Wesley, Reading, Massachusetts 1986

[Danthine_80] A. Danthine, "Protocols representation with finite-state models", Trans. on Commun. IEEE, vol. COM-28, pp. 632-643, Abril 1980

[Dorf_86] R. Dorf, *Sistemas automáticos de control. Teoría y práctica,* Addison-Wesley Iberoamericana, S.A. Wilmington, Delaware, E.U.A. 1986

[ECMA_79] European Computer Manufacturers Association, ECMA-49/ *HDLC elements of procedure* 2ª Ed., ECMA, Ginebra 1979

[ECMA_81] European Computer Manufacturers Association, ECMA-71/ *HDLC selectes procedures,* ECMA, Ginebra 1981

[Everitt_94] D. Everitt, "Traffic enginnering of the radio interface for cellular mobile networks", Proc. IEEE, vol 82, no. 9, pp. 1371-1382, Septiembre 1994

[Fuhrmann_94] W. Fuhrmann and V. Brass, "Performance aspects of the GSM radio subsystem", Proc. IEEE, vol 82, no. 9, pp. 1449-1465, Septiembre 1994

[García_90] J. García Tomás, *Sistemas y redes teleinformáticas,* Ed. Ra-ma, Madrid 1990

[Halsall_92] F. Halsall, *Data communications, computer networks and open system* 3 rd Ed., Addison-Wesley, Reading, Massachusetts 1992

[Hawang_85] K. Hawang and F. Bringgs, *Computer architectures and parallel processing*, Mc Graw-Hill International, Singapore 1985

[Hopcroft_79] J. Hopcroft and J. Ullman, *Introduction to automata theory, languages and computation*, Addison-Wesley, Menlo Park, California 1979

[IAR_89] IAR Systems, *Micro series assembler, linker and librarian for microprocessor development* 5ª Ed., Uppsala, Sweden 1989

[Ibarra_95] F. Ibarra, *Introducción a las arquitecturas paralelas,* Secretariado de Publicaciones, Universidad de Alicante 1995

[Intel_83] Intel Corporation, *MCS_51 Macro Assembler user's guide*, Technical Publications, Santa Clara, California 1983

[ISO_ 81] International Organization for Standardization, "ISO/DP 7498 "ISO/TC97/SC16/537 Draft Proposal", Computer Commun. Review, vol. 11, no. 2, pp. 15-65, Abril 1981

[ISO_ 81a] International Organization for Standardization, "Guidelines for specification of services and protocols", ISO/TC97/SC16/N380 and N381, Ginebra 1981

[ISO_ 83] International Organization for Standardization, "Basic Reference Model for Open System Interconnection", ISO 7498, Ginebra 1983

[Jauregui_96] P. Jauregui, J. Rosa, C. Pastor, J. Giner and J. Delgado, "Integration of two seismic data acquisition systems. The University of Alicante local seismic network", Seismology in Europe ESC, XXV General Assembly, pp. 137-142, Reykjavík, Septiembre 1996

[Katz_94] R. Katz, "Adaptation and mobility in wireless information sistems", Personal Commun. IEEE, vol.1, no.1, pp. 6-17, 1994

[Kuo_82] B. Kuo, *Sistemas automáticos de control,* Cia Editorial Continental S.A. de C.V., México 1982

[Mandado_87] E. Mandado, *Sistemas electrónicos digitales*, Ed. Marcombo, Barcelona 1987

[Martínez_91] J. Martínez y M. Barrón, *Prácticas con microcontroladores de 8 bits. Aplicaciones industriales,* Mac Graw-Hill, Madrid 1991

[Merlin_76] P. Merlin and D. Faber, "Recoverability of communications protocols, implications of a theorical study", Trans. on Commun. IEEE, vol. COM-24, pp. 1036-1043, Septiembre 1976

[No&Angulo_87] J. No y J. Angulo, *Control de procesos industriales por computador*, Ed. Paraninfo, Madrid 1987

[Ogata_93] K. Ogata, *Ingeniería de control moderna,* Prentice-Hall Hispanoamericana, México 1993

[Pahlavan_94] K. Pahlavan y A. Levesque, "Wireless data communications". Proc. IEEE, vol. 82, no. 9 , pp. 1389-1430, Septiembre 1994

[POS_74] J. Postel, *A graph model analysis of computer communications protocols*, PhD of Th., UCLA, ENG7410, 1974

[Purser_89] M. Purser, *Comunicación de datos para programadores,* Addison-Wesley Iberoamericana, Wilmington, Delaware, E.U.A. 1989

[Rosa_92] J. Rosa, *Teleinformática*, Apuntes curso 92/93, Escuela Politécnica Superior, Universidad de Alicante 1992

[Rosa_96] J. Rosa, C. Pastor, L. Crespo, F. Candelas, F. Botella y M. Marhuenda, "Nueva tarjeta de comunicaciones", P9600701, Oficina Española de Patentes y Marcas, Solicitante: Universidad de Alicante, Marzo 1996

[Sánchez_93] M. Sánchez-Uran, L. Jiménez y A. Perales, "Control distribuido (control avanzado)", E.T.S.I. Industriales, U.P. de Madrid 1993

[Schwartz_94] M. Schwartz, *Redes de telecomunicaciones. Protocolos, modelado y análisis,* Addiso-Wesley Iberoamericana, Wilmington, Delaware, E.U.A. 1994

[Stallings_91] W. Stallings, *Data and computer communications* 3rd Ed., Macmillan Publishing Company, New York 1991

[Tanenbaum_91] A. Tanenbaum, *Redes de ordenadores* 2ª Ed, Prentice-Hall Hispanoamericana, México 1991

[TEN_78] A. Teng and M. Liu, *A formal approach to the design and implementation of network communications protocols*, Anais da Compsac, 1978

[Vidaller_78] L. Vidaller, J. Riera y J. Viñas, *Transmisión de datos,* E.T.S.I. Telecomunicación, Dpto. Publicaciones, Madrid 1978

ABREVIATURAS

A	Acción que se produce en un protocolo como respuesta a una entrada, en un estado específico
Ae	Acción que se produce en un protocolo cuando la entidad actua como emisora de la información
Ar	Acción que se produce en un protocolo cuando la entidad actua como receptora de la información
ABORT_TA	Servicio de la aplicación transferencia de archivos, utilizado para indicar o solicitar el fin del proceso
ACK_FI	Confirmación del receptor (información de fichero correcta)
ADC	*Analog to Digital Converter* (convertidor analógico-digital)
APDU	*Aplication Protocol Data Unit* (unidad de datos de protocolo de las entidades de aplicación)
ARB	Variable binaria utilizada para indicar si se ha recibido algún bloque con SVB correcta
ARRL	*American Radio Relay League* (sociedad americana de radio)
c	Número de *bits* de una trama de asentimiento
C	Condiciones de un protocolo (externas = *primitivas,* internas = *temporizadores*
C_R	Contador de reintentos
CC	Campo de control para el enlace lógico
CCA	Campo de comandos para el control de la aplicación
CCITT	Comité Consultivo Internacional Telegráfico y Telefónico
C_MEDIO	Comando de la aplicación TA, utilizado por el emisor de un fichero para especificar el final del envio correspondiente a una solicitud
CRC	*Ciclic Redundancy Check* (código de redundancia cíclica)
CSMA	*Carrier Sense Múltiple Accees* (acceso múltiple por detección de portadora)
C_SVB	Secuencia de verificación de bloque calculada en el receptor
CTS	*Clear to Send* (preparado para recibir)
DA	Campo de datos para la aplicación
DAC	D*igital to Analog Converter* (convertidor digital-analógico)
DAT_FI	Datos de fichero
DATA	Servicio de envío y recepción de datos proporcionado por un nivel
DISC	Comando para finalizar un enlace entre dos estaciones

DM	Respuesta que se envía cuando una estación recibe una trama distinta de SABM o UI mientras está en modo desconectado
E_FICH	Servicio de la aplicación transferencia de archivos, utilizado para solicitar el envío de un fichero
ECMA	*European Computer Manufacturers Association* (asociación Europea de fabricantes de ordenadores)
ED	Estación Destino
EIA	*Electronic Industries Association* (asociación de industrias electrónicas)
EMA	Estación de Medida Autónoma
ERT	Entrada Remota de Trabajos
E/R	Variable binaria utilizada para indicar si la entidad participante en la transferencia de un fichero, actúa como emisora o receptora
E/S	Entrada - Salida
ETCD	Equipo Terminal de Circuito de Datos
ETD	Equipo Terminal de Datos
FEND	Carácter guión para las tramas de enlace (C0 hexadecimal, 192 decimal)
FESC	Carácter de transparencia para las tramas de enlace(DB hex)
FIN_EFICH	Servicio de la aplicación transferencia de archivos, utilizado para indicar la finalización del proceso enviar fichero
FIN_RFICH	Servicio de la aplicación transferencia de archivos, utilizado para indicar la finalización del proceso recibir fichero
FM	Frecuencia Modulada
FRMR	Respuesta que se envía para comunicar que una trama no se puede procesar con éxito y que no es corregible enviando la trama errónea de nuevo
HDLC	*High-Level Data Link Control* (control de enlace de datos, alto nivel)
i	Intentos de conexión del enlace
ICI	*Interfase Control Information* (información para el control de la interfase)
IDU	*Interfase Data Unit* (unidad de datos de la interfase)
IEEE	*Institute of Electrical and Electronic Engineers* (instituto de ingenieros eléctricos y electrónicos)
IF	Identificador de Función
ISO	*International Standards Organization* (organismo internacional de estandarización)
JOB	Servicio de la aplicación específica entrada remota de tareas

k	Tasa de error medio de un circuito (nº *bits* erróneos/nº de *bits* transmitidos)
LAN	*Local Area Network* (red de área local)
LAPB	*Link Access Procedure Balanced* (procedimiento de enlace con acceso balanceado)
LL	*Logical Link* (enlace lógico)
LLC	*Logical Link Control* (control de enlace lógico)
LON_FICH	Longitud total de un fichero
LON_MEM	Longitud de información que se envía en una determinada solicitud
MAC	*Medium Access Control* (control de acceso al medio)
MEF	Máquina de Estados Finitos
Σ	Conjunto finito de posibles entradas de una MEF
Σe	Conjunto finito de posibles entradas de la MM que modela un protocolo o su núcleo, cuando la entidad actua como emisora
Σr	Conjunto finito de posibles entradas de la MM que modela un protocolo o su núcleo, cuando la entidad actua como receptora
Δ	Conjunto finito de posibles salidas de una MEF
Δe	Conjunto finito de posibles salidas de la MM que modela un protocolo o su núcleo, cuando la entidad actua como emisora
Δr	Conjunto finito de posibles salidas de la MM que modela un protocolo o su núcleo, cuando la entidad actua como receptora
δ	Función de transición de estado de una MEF
λ	Función de salida de una MEF
MEFA	Máquina de Estados Finitos Ampliada
MM	Máquina de Mealy
MODO_N	Comando de la aplicación. Solicitud para pasar a modo normal
MODO_P	Comando de la aplicación. Solicitud para pasar a modo parámetros
MODO_T	Comando de la aplicación. Solicitud para pasar a modo transparente
μC	Microcontrolador
n	Número de *bit* útiles en una trama de enlace
NACK_FI	Confirmación del receptor (información de fichero errónea)
NBE	Variable, número de bloque enviado
NBF	Variable, número de bloques de un fichero que faltan por enviar
NCA	*Network Communications Access* (acceso a la red de comunicaciones)
NPDU	*Network Protocol Data Unit* (unidad de datos de protocolo, nivel NCA)

N(R)	Número de secuencia recibido (para tramas de enlace)
N(S)	Número de secuencia enviado (para tramas de enlace)
N_t	Proporción de transmisiones mas retransmisiones que corresponden a cada trama
OC	Ordenador Central
OSI	*Open Systems Interconnection* (interconexión de sistemas abiertos)
p	Probabilidad de que una trama se vea afectada por errores de transmisión
P_i	Probabilidad de que una trama requiera *i* retransmisiones
PC	*Personal Computer* (ordenador personal)
PDU	*Protocol Data Unit* (unidad de datos del protocolo)
P/F	Bit poll/final (bit nº 4 del campo de control de una trama de enlace)
PGE	Proceso Generador de Entradas
PGS	Proceso Generador de Salidas
PID	*Protocol Identifier* (identificador de protocolo)
PSDU	*Physical Service Data Unit* (unidad de datos de servicio físico)
PTT	*Push-to-Talk* (activación de la portadora en una emisora de radio)
Q	Conjunto finito de estados de una MEF
Q_0	Estado inicial de una MEF
Q'	Conjunto finito de estados de un protocolo
Q'_0	Estado inicial de un protocolo
r	Número de *bit* redundantes en una trama de enlace
R	Rendimiento en una red de comunicaciones
R_FICH	Servicio de la aplicación transferencia de archivos, utilizado para indicar la recepción de un fichero
REJ	*Reject* (rechazo de trama)
RM-OSI	*Reference Model for Open Systems Interconnection* (modelo de referencia para la interconexión de sistemas abiertos)
R_MDN	Comando de la aplicación. Aceptado el modo normal de operación
R_MDP	Comando de la aplicación. Aceptado el modo parametrización
R_MDT	Comando de la aplicación. Aceptado el modo transparente
RNR	*Receive not Ready* (recepción imposible)
RR	*Receive Ready* (preparado para recibir)
RTS	*Request to Send* (preparado para emitir)
S	Conjunto finito de posibles salidas de un protocolo

Se	Salida de un protocolo cuando la entidad actua como emisora
Sr	Salida de un protocolo cuando la entidad actua como receptora
SABM	Comando para poner un enlace entre dos estaciones en modo asíncrono balanceado
SAP	*Service Access Point* (punto de acceso al servicio)
SDU	*Service Data Unit* (unidad de datos del servicio)
SIR	Sistema Informático Remoto
SO	Sistema Operativo
SVB	Secuencia de Verificación de Bloque
T_0	Tiempo de ocupación de un circuito para enviar una información
T_1	Tiempo de espera para la recepción de una confirmación en el nivel de enlace de la subred de comunicaciones
T_2	Retraso introducido por el receptor entre la recepción de una trama I y la respuesta resultante
T_3	Tiempo de espera utilizado cuando T_1 no esta en marcha, para mantener la integridad del enlace
T_a	Tiempo de asentimiento. Tiempo desde que el emisor envía el último *bit* de una trama, hasta que interpreta su confirmación o rechazo
T_{AC}	Tiempo para emitir una nueva trama de solicitud de enlace lógico
T_c	Tiempo de escritura de una trama de confirmación en el medio
T_C	Tiempo de espera para la conexión del enlace lógico
T_{EC}	Tiempo de espera de la aplicación TA para una contestación de la entidad par
T_p	Tiempo de activación de portadora antes de enviar información
T_r	Tiempo de respuesta. $[T_a - (T_p + T_c)]$
T_R	Tiempo de referencia para el cálculo de T_{AC}
T_t	Tiempo de escritura de una trama de información en el medio.
T_{resp}	Tiempo de espera para asentimientos
T_s	Tiempo de espera entre sondeos para la aplicación de control
T-slots	Tiempo de espera igual al retardo máximo de propagación en un canal. (Ida y vuelta entre los dos puntos más alejados)
TA	Transferencia de Archivos
TNC	*Terminal Node Controller* (nodo de la subred de comunicaciones)
Tramas *I*	Tramas de enlace que contienen información del nivel superior
Tramas *S*	Tramas de supervisión del enlace

Tramas *U*	Tramas de enlace no numeradas
UA	Respuesta que se envía como confirmación a la recepción y aceptación de un SABM o DISC
UCP	Unidad Central de Proceso
UI	*Unnumbered Information* (tramas de información no numeradas)
V	Variables de un protocolo no incluidas en la definición de los estados de la MEF que modela su núcleo
V_c	Velocidad de un canal de comunicaciones (*bit*/s)
V_{cc}	Tensión de alimentación de un circuito
V_e	Velocidad efectiva de un canal de comunicaciones (*bit* útiles/s)
WAN	*Wide Area Network* (red de área extendida)
WLAN	*Wireless Local Area Network* (red de área local inalámbrica)

APÉNDICE I

Software de la aplicación de control y la red de ordenadores, residente en la estación central

Estación central. "*Control de estaciones de registro sísmico*"

CRP.BAS

```
' En este modulo se definen todas las variables necesarias para
' la aplicación, sus estados y los comandos del sistema

Option Explicit


'Variables de control de estaciones
Global Const TotalEst = 5              'Máximo 26= Canales TNC de A..Z
Global NomEstac As String              'indicativo de la estación
Global EstActivasElegidas As Integer   'el usuario ha elegido EstActivas
Global EstEntrante As Integer          'num estación que solicita REQUEST
Global EstAct(TotalEst) As Integer     'lista de estaciones activas
Global LEU(TotalEst) As Integer        'lista de estaciones elegidas por el usuario
Global NumEstCon As Integer            'numero de estaciones conectadas
Global EstEspecifica As Integer        'Estación especifica
Global NEA As Integer                  'Numero de Estaciones Activas

'Variables internas de la aplicación
Global CRLF As String, CR As String
Global FichTX$, FichRX$                'fichero a enviar, fichero a recibir
Global SondeosEnProceso As Integer     'Booleano
Global OpcionTime As Integer           'opción de TIME: 1=Mirar, 2=Introducir
Global OpcionParam As Integer          'opción de MODO PARAM: 1=Tarjeta, 2=IDS
Global OpcionSRAM As Integer           'opción de TEST SRAM
Global MODOSUPERV As Integer           'Modo superusuario
Global passw As String                 'password para SUPERVISOR
Global cadParam As String              'cadena con parámetros de Modo Parametrizar

'directorios donde se ubican los distintos ficheros que utiliza la aplicación
Global dir_app$, dir_eve$, dir_err$, dir_dia$
Global dir_jul$, dir_ids$

'Variables para gestión de tramas
Global tramaEnl As String      ' trama para nivel enlace
Global tramaSer As String      ' trama para nivel serie
Global tramaRec As String      ' trama recibida en modo Host
Global THE As String           ' trama host ENVIADA
Global EstSer As Integer       ' estado del autómata recepción serie
Global rec_trama As Integer    ' trama recibida (T o F)

'Variables para control de visualización de mensajes por pantalla
Global MonitorRX As Integer    ' booleano para escribir por pantalla
                               ' las tramas recibidas en modo Normal
Global MonitorFIDS As Integer  ' booleano para escribir en el Formulario CRP o
                               ' en el formulario FIDS
Global MonTie As Integer       ' booleano para ver el tiempo empleado en cada recepción


'Caracteres de Transparencia Modo HOST
Global Const FEND = 192
Global Const FESC = 219
Global Const TFEND = 220
```

```
Global Const TFESC = 221

' Modos de la TNC
Global ModoTNC As Integer
Global Const Normal = 10
Global Const HOST = 11

' Posibles modos de la aplicación
Global ModoAplic As Integer
Global Const NORMALSONDEO = 90
Global Const NORMALREPOSO = 91
Global Const TRANSPARENTE = 92
Global Const PARAMETRIZAR = 93


' Posibles estados de la aplicación
Global EstadoAplic As Integer
Global Const CONECTADO = 100
Global Const DESCONECTADO = 101
Global Const ENVIANDO = 102
Global Const RECIBIENDO = 103
Global Const CONECTANDO = 104
Global Const SONDEANDO = 105
Global Const ACTUANDO = 106


'Comandos del Sistema
Global Const SONDEO = &H33
Global Const ERROR_F = &HEF
Global Const NO_DAT = &H66
Global Const WAIT = &H75
Global Const DAT_AV = &H68
Global Const DAT_FI = &H6A
Global Const SVB = &H6B
Global Const ACK_AV = &H83
Global Const ACK_FI = &HA3
Global Const NACK_FI = &HA5
Global Const C_MEDIO = &H99
Global Const TNACK_P = &H31
Global Const TNACK_R = &H13
Global Const SESION_R = &H15
Global Const DEL_FI = &H7A
Global Const REP_FI = &H7B

Global Const C_IDS = &H87
Global Const R_CIDS = &H78

Global Const MODO_T = &H73
Global Const R_MDT = &H37
Global Const DAT_TP = &H63
Global Const DAT_TR = &H36

Global Const MODO_N = &H54
Global Const R_MDN = &H45

Global Const BATT = &H53
Global Const R_BATT = &H35
```

```
Global Const FDIAG = &H5B
Global Const R_FDIAG = &HB5

Global Const P_RESET = &H9A
Global Const R_RESET = &HA9

Global Const MODO_P = &H4A
Global Const R_MDP = &HA4
Global Const DAT_PP = &H4B
Global Const DAT_PR = &HB4

Global Const RESET_ID = &H93
Global Const R_RSTID = &H39

Global Const P_TIME = &H56
Global Const R_TIME = &H65

Global Const TEST_E = &H18
Global Const TEST_L = &H1A
Global Const R_TEST = &H81


'Constantes de los eventos MSComm
Global Const MSCOMM_EV_SEND = 1
Global Const MSCOMM_EV_RECEIVE = 2
Global Const MSCOMM_EV_CTS = 3
Global Const MSCOMM_EV_DSR = 4
Global Const MSCOMM_EV_CD = 5
Global Const MSCOMM_EV_RING = 6
Global Const MSCOMM_EV_EOF = 7

'Constantes de los códigos de error de MSComm
Global Const MSCOMM_ER_BREAK = 1001
Global Const MSCOMM_ER_CTSTO = 1002
Global Const MSCOMM_ER_DSRTO = 1003
Global Const MSCOMM_ER_FRAME = 1004
Global Const MSCOMM_ER_OVERRUN = 1006
Global Const MSCOMM_ER_CDTO = 1007
Global Const MSCOMM_ER_RXOVER = 1008
Global Const MSCOMM_ER_RXPARITY = 1009
Global Const MSCOMM_ER_TXFULL = 1010
```

Sub AnotarError (tipoerror As String, funcion As String)

```
' Anota los errores producidos en la aplicación en el fichero ERRORES.LOG

  Open dir_err + "errores.log" For Append As #2
  Write #2, String(70, "-") + CRLF + Format$(Now, "dd-mm-yy hh:mm:ss")
  Write #2, "Funcion: " + funcion
  Write #2, "Error: " + tipoerror
  Select Case ModoAplic
    Case NORMALREPOSO
      Write #2, "Modo Aplic: NORMALREPOSO"
    Case NORMALSONDEO
      Write #2, "Modo Aplic: NORMAL SONDEO"
```

```
      Case TRANSPARENTE
         Write #2, "Modo Aplic: TRANSPARENTE"
      Case PARAMETRIZAR
         Write #2, "Modo Aplic: PARAMETRIZAR"
      Case Else
         Write #2, "Modo Aplic: INDETERMINADO"
   End Select

   Close #2
End Sub

Sub EscribirSerie (cad As String)
' Esta funcion solo se accede desde el formulario FIDS
' Escribe en el puerto serie


   If ModoTNC = HOST Then
      FCRP.Comm1.Output = Chr$(FEND) + cad + Chr$(FEND)
   Else
      FCRP.Comm1.Output = cad
   End If
End Sub

Sub LeerSerie ()
Dim nct As Integer  ' num car de trama
Dim ncr As Integer  ' num car recibidos en buffer serie
Dim BufferEnt$      ' bufer leído de serie
Dim ncb%
' Analiza los bytes recibidos por el puerto serie, delimita tramas de enlace
' por los FEND y las envía al nivel superior (tramaEnl)

   If Err Then
      ShowTexto FCRP.Text2, "[LeerSerie] Error: " + Error$ + CRLF
      AnotarError Error$, "LeerSerie"
   End If

   BufferEnt = FCRP.Comm1.Input     ' lee el contenido del bufer de serie
   ncr = Len(BufferEnt)
   If ncr = 0 Then
      Exit Sub
   End If

   ncb = 1
   While ncb <= ncr   ' analizo el bufer del serie
      Select Case EstSer
       Case 1
         If Mid$(BufferEnt, ncb, 1) = Chr$(FEND) Then
            EstSer = 2
         End If
         ncb = ncb + 1   ' sig car a analizar
       Case 2
         If Mid$(BufferEnt, ncb, 1) = Chr$(FEND) Then
            'ncb = ncb + 1
            EstSer = 3
         Else
            tramaSer = tramaSer + Mid$(BufferEnt, ncb, 1)
```

```
            ncb = ncb + 1
          End If
        Case 3
          rec_trama = True
          tramaEnl = tramaSer
          tramaSer = ""
          EstSer = 1
          ncb = ncb + 1
      End Select
    Wend
End Sub

Function QuitarT (cadena As String) As String
Dim i%, longitud As Integer
Dim cadaux$, car As String
' Recibe una cadena y elimina la transparencia

    longitud = Len(cadena)
    i = 1
    While i <= longitud
      car = Mid$(cadena, i, 1)
      Select Case car
        Case Chr$(FESC)    'FESC habrá pocos caracteres en cada trama
          i = i + 1
          Select Case Mid$(cadena, i, 1)
            Case Chr$(TFEND)
              cadaux = cadaux + Chr$(FEND)
            Case Chr$(TFESC)
              cadaux = cadaux + Chr$(FESC)
          End Select
        Case Else
          cadaux = cadaux + car   'usando car evito Mid$ de nuevo
      End Select
      i = i + 1
    Wend
    QuitarT = cadaux
End Function

Function RecibirHost (Timeout As Integer) As String
Dim t As Long
' Espera recibir contestación en un tiempo Timeout

    tramaEnl = ""
    rec_trama = False
    t = Timer
    While (Timer - t <= Timeout) And (rec_trama = False)
      'LeerSerie
      DoEvents
    Wend
    RecibirHost = tramaEnl
End Function

Sub ShowTexto (caja As TextBox, cad As String)
Dim Nd%, i%
' Extraído de un ejemplo de Visual Basic 3.0 para mostrar caracteres por
' una caja de texto
```

```
  ' Asegurarse de que el texto existente no sea demasiado largo
  Nd = Len(caja.Text)
  If Nd >= 16384 Then
    caja.Text = Mid$(caja.Text, 4097)
    Nd = Len(caja.Text)
  End If

  'Apuntar al final de la caja de texto
  caja.SelStart = Nd

  'Filtrar caracteres Retroceso
  Do
    i = InStr(cad, Chr$(8))
    If i Then
      If i = 1 Then
        caja.SelStart = Nd - 1
        caja.SelLength = 1
        cad = Mid$(cad, i + 1)
      Else
        cad = Left$(cad, i - 2) + Mid$(cad, i + 1)
      End If
    End If
  Loop While i

  'Eliminar caracteres LF
  Do
    i = InStr(cad, Chr$(10))
    If i Then
      cad = Left$(cad, i - 1) + Mid$(cad, i + 1)
    End If
  Loop While i

  'Asegurar que todos los caracteres CR tienen un LF
  i = 1
  Do
    i = InStr(i, cad, Chr$(13))
    If i Then
      cad = Left$(cad, i) + Chr$(10) + Mid$(cad, i + 1)
      i = i + 1
    End If
  Loop While i

  'Añadir los datos filtrados a la caja de texto
  caja.SelText = cad
End Sub
```

CRP.FRM

```
' Declaración de variables del formulario CRP.FRM

Option Explicit

'evitar cargar dos veces la aplicación
Declare Function GetModuleHandle Lib "Kernel" (ByVal lpModuleName As String) As Integer
Declare Function GetModuleUsage Lib "Kernel" (ByVal hModule As Integer) As Integer
```

```
'variables de control del numero de ficheros
Dim NumFDiag As Integer
Dim NumFichSondeo As Integer

'variables de control para la funcion AnalizarCCA
Dim numTNACK_P As Integer    'num de TNACK_P consecutivos enviados
Dim numTNACK_R As Integer    'num de TNACK_R consecutivos enviados

'Variable de control de los sondeos de estaciones remotas
Dim tsondeo As Integer       'tiempo entre sondeos
Dim Timer2Vencido As Integer 'booleano
Dim ProxSondeo As String

'Parámetros del puerto serie
Dim NUMPUERTO As Integer
Dim PARAMETROS As String
Dim VELOCTNC As Integer


Sub AbrirPuerto ()
' Configura el puerto serie

   If Comm1.PortOpen Then Comm1.PortOpen = False

   'Establecer el numero del puerto
   Comm1.CommPort = NUMPUERTO

   'Abrir el puerto de comunicaciones. Si el numero del
   ' puerto no es valido se genera el error 68
   On Error Resume Next
   FCRP.Comm1.RThreshold = 1
   FCRP.Comm1.Settings = PARAMETROS
   FCRP.Comm1.InputLen = 0
   FCRP.Comm1.PortOpen = True

   If Err Then
         MsgBox "Error al abrir el puerto COM" & NUMPUERTO, 16
         Exit Sub
   End If


End Sub


Sub ActualizarListaEstAct ()
Dim i%
' Actualiza la lista de estaciones activas

   Text3.Text = ""
   For i = 1 To TotalEst
         If EstAct(i) = 1 Then
            Text3.SelText = Str$(i) + " "
            Text3.SelStart = Len(FCRP.Text3.Text)
         End If
   Next i
End Sub
```

```
Sub ActualizarLModoTNC (modo As Integer)
' Actualiza el estado del TNC

   Select Case modo
         Case NORMAL
            LModoTNC.Caption = "MODO_TNC: NORMAL"
            LModoTNC.BackColor = QBColor(14)    'Ligth Yellow

         Case HOST
            LModoTNC.Caption = "MODO_TNC: HOST"
            LModoTNC.BackColor = QBColor(11)    'Ligth Cyan
   End Select
End Sub

Sub AddTexto (caja As TextBox, cad As String)
Dim Nd%
' Escribe en las cajas de texto, y vacía su contenido en el fichero
' VOLCADO.LOG cada 16Kb

   '--- Nos aseguramos de que el texto existente no es demasiado largo.
   Nd = Len(caja.Text)
   If Nd >= 16384 Then
         Open dir_err + "VOLCADO.LOG" For Append As #1
         Write #1, Format$(Date$, "dd-mm-yy"), Time$
         Write #1, Mid$(caja.Text, 1, 16300) + CRLF
         Close #1
         'le quito 16K a la caja.Text
      caja.Text = Mid$(caja.Text, 16301) + "VOLCADOS 16K" + CRLF
      Nd = Len(caja.Text)
   End If

   '--- Apuntar al final de caja
   caja.SelStart = Nd

   '--- Añadir cad a caja.Text
   caja.SelText = cad
End Sub

Function AnalizarCCA (cad As String)
' Es una funcion de la aplicación de control que verifica si el campo CCA
' esta duplicado (siempre que el campo DA este vacío)
' (Devuelve OK=1, Repetir=2)
Dim estacion As Integer

  estacion = Asc(Mid$(cad, 3, 1)) - 64

  Select Case Mid$(cad, 4, 1)

   'POSIBLES COMANDOS RECIBIDOS POR EP
   Case Chr$(ERROR_F), Chr$(NO_DAT), Chr$(WAIT), Chr$(TNACK_R), Chr$(R_MDT),
Chr$(R_MDN), Chr$(R_MDP), Chr$(R_RESET), Chr$(SESION_R), Chr$(R_TEST)
         If Mid$(cad, 4, 1) <> Mid$(cad, 5, 1) Then
            AnotarEnDiag "[AnalizarCCA] " + THE, cad
            numTNACK_P = numTNACK_P + 1
            If numTNACK_P > 3 Then
```

```
                AddTexto Text2, "ESTA" + LTrim$(Str$(estacion)) + " dada de baja por envío
de 3 TNACK_P consecutivos" + CRLF
                'If modoAplic = NORMAL_SONDEO Then DesconectarDe 1
                DesconectarDe estacion
                NEA = NEA - 1
                'en modo Transparente o Parametrizar no se desconecta
                numTNACK_P = 0
            Else
                EscribirSerie "D2" + Mid$(cad, 3, 1) + Chr$(TNACK_P) + Chr$(TNACK_P)
                AddTexto Text2, "ENVIADO TNACK_P" + CRLF
            End If
            AnalizarCCA = 2
        Else
            AnalizarCCA = 1
        End If
    'RESTO DE TRAMAS SON DATOS Y NO SE DUPLICA EL BYTE
    Case Else
        AnalizarCCA = 1   'devolver OK
  End Select
End Function

Sub AnotarEnDiag (tramaE As String, tramaR As String)
' Anota en el fichero DIAG.LOG la recepción errónea o no recepción de la
' respuesta a un trabajo solicitado

    Open dir_err + "diag.log" For Append As #2
    Write #2, String(70, "-") + CRLF + Format$(Now, "dd-mm-yy hh:mm:ss")
    Write #2, "trama enviada", tramaE
    Write #2, "trama recibida", tramaR
    Close #2
End Sub

Sub BActuar_Click ()
Dim RESPUESTA%
' Permite al usuario solicitar una sesión con una estacion especifica

BActuar.Enabled = False
SondeosEnProceso = True 'NO atender a mas REQUEST hasta acabar de ACTUAR !!


    DetModoAplic NORMALREPOSO   'establece próximo sondeo
    timer4.Enabled = False      'paro prox sondeo
    LTSondeo.Caption = "--------"

    FEstEsp.Show 1

    ConectarCon estEspecifica

    If NumEstCon = 0 Then
        DetEstadoAplic DESCONECTADO
        DetModoAplic NORMALREPOSO       'para que establezca el próximo sondeo
        AddTexto Text2, "Imposible conectar con ER Num. " + Str$(estEspecifica) + ". Pulse
ACTUAR de nuevo" + CRLF
    Else
        BMTransp.Enabled = True
        BMParam.Enabled = True
```

```
        AddTexto Text2, "ER Num. " + Str$(estEspecifica) + " conectada. Pulse Modo
Transparente o Modo Parametrizar" + CRLF
        Text4.Text = LTrim$(Str$(estEspecifica))
        DetEstadoAplic ACTUANDO
   End If
'BActuar.Enabled = True
SondeosEnProceso = False 'atender a los REQUEST
End Sub

Sub BBateria_Click ()
' Solicitud del estado de la alimentación eléctrica de la estacion
' de registro


  If estadoAplic = ACTUANDO Then
   EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(BATT) + Chr$(BATT)
   AddTexto Text2, "ENVIADO BATT. ESPERANDO RESPUESTA..." + CRLF
   FCRP.Enabled = False
   Do
        tramarec = RecibirHost(20)
        Select Case Mid$(tramarec, 4, 1)
          Case Chr$(R_BATT)
               'No tratarlo aquí y que se reciba a nivel de serie
               ShowTexto Text2, CRLF + "  *** R_BATT: " + CRLF + Mid$(tramarec, 5) +
CRLF
          Case Chr$(TNACK_R)

          Case Else
               AddTexto Text2, "NO RESPUESTA A BATT" + CRLF
               'ANOTAR EN DIAGNOSTICO
               AnotarEnDiag "D2A-BATT", tramarec
        End Select
   Loop Until Mid$(tramarec, 4, 1) = Chr$(R_BATT) Or tramarec = "" Or Mid$(tramarec, 8, 4)
= "DISC"
  Else
   AddTexto Text2, "ESTACION DESCONECTADA: NO PUEDO ENVIAR BATT" + CRLF
  End If
   FCRP.Enabled = True

End Sub

Sub BDisconnect_Click ()
Dim EstAplic%
' DESCONEXION DEL ENLACE

   If EstAplic = ACTUANDO Then
        Text4.Text = "" 'borro lista estac. especif. por si había
   End If

   If ModoTNC = NORMAL Then BModoHost_Click

   DetEstadoAplic DESCONECTADO

   DesconEstaciones

   NEA = 0
```

```
    SondeosEnProceso = False
    'tras desconectar a todas las estaciones ya puedo atender los REQUEST
    BActuar.Enabled = True
    Text1.SetFocus
End Sub

Sub BFDiag_Click ()

    EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(FDIAG) + Chr$(FDIAG)
    AddTexto Text2, "ENVIADO FDIAG. ESPERANDO RESPUESTA..." + CRLF

    FCRP.Enabled = False
    tramarec = RecibirHost(5)
    If Mid$(tramarec, 4, 1) <> Chr$(R_FDIAG) Then
        AddTexto Text2, "NO RESPUESTA A FDIAG" + CRLF
        'ANOTAR EN DIAGNOSTICO
        AnotarEnDiag "D2A-FDIAG", tramarec
    Else
        RecibirFicheroDiag
    End If
    FCRP.Enabled = True
    NumFDiag = (NumFDiag + 1) Mod 10
    Open dir_app + "NFD.CFG" For Output As #1
    Write #1, Str$(NumFDiag)
    Close #1
End Sub

Sub BFichTransp_Click ()
' Visualiza el formulario FIDS para solicitar un fichero bloque a bloque

    MonitorFIDS = True
    FIDS.Show 1
End Sub

Sub BInicio_Click ()

timer4.Enabled = False  'paro el timer de sondeo 1 hora
LTSondeo.Caption = "--------"
BInicio.Enabled = False 'impido que me pulsen otro sondeo hasta que no acabe con el actual
BActuar.Enabled = False

'limpio el buffer del puerto serie
CerrarPuerto
AbrirPuerto

SondeosEnProceso = True 'NO atiendo a mas REQUEST hasta acabar la ronda de
SONDEOS

    AddTexto Text2, Format(Now, "dddd, mmmm dd, yyyy hh:mm:ss") + CRLF

' conecto las estaciones empezando por 1
' podemos usar USERS 0 y así no permito que nadie me conecte
' la TNC me advertirá que alguien intenta conectarse con "CONNECT REQUEST: ESTARx"
' con MAXUSERS 5 me reservo 5 canales de salida para mi

    If EstEntrante = 0 Then EstEntrante = 1
```

```
   ConectarEstaciones EstEntrante  'conecto todas estaciones empezando por
                                   'EstEntrante que es la que ha lanzado el REQUEST
   If NumEstCon > 0 Then
         DetModoAplic NORMALSONDEO
         RecibirTodosFicheros EstEntrante
         DetModoAplic NORMALREPOSO
         DetEstadoAplic DESCONECTADO
   Else
         AddTexto Text2, "NO HAY ESTACIONES CONECTADAS. NO INICIO SONDEOS" +
CRLF
   End If

BInicio.Enabled = True
BActuar.Enabled = True
   EstEntrante = 0 'para iniciar la ronda de sondeos cada hora desde ESTA1

   SondeosEnProceso = False

   LFecha = Format(Now, "dd-mm-yy")  'fecha del sistema

   EstableceProxSondeo
   timer4.Enabled = True
End Sub

Sub BMNormal_Click ()

BMNormal.Enabled = False
   If estadoAplic = ACTUANDO Then
         'AddTexto Text2, "D2" + Chr$(64 + EstEspecifica) + Chr$(MODO_N) + CRLF
         EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(MODO_N) + Chr$(MODO_N)
'espero 10 seg a que ER me conteste
         tramarec = RecibirHost(10)
         If Mid$(tramarec, 4, 1) = Chr$(R_MDN) Then
            AddTexto Text2, CRLF + "ESTACION Num. " + Str$(estEspecifica) + " EN MODO
NORMAL" + CRLF
            Text4.Text = ""
            'EstableceProxSondeo
            'timer4.Enabled = True

            'timer4.Enabled = True
            'LTSondeo.Caption = ProxSondeo

            DesconEstaciones
            DetEstadoAplic DESCONECTADO
            DetModoAplic NORMALREPOSO   'establece ya prox. sondeo
            BActuar.Enabled = True
         Else
            AddTexto Text2, "NO RESPUESTA A MODO_N" + CRLF
         End If
   Else
         AddTexto Text2, "IMPOSIBLE ENTRAR EN MODO NORMAL: No Responde" +
CRLF
         AnotarError "IMPOSIBLE ENTRAR EN MODO NORMAL: No Responde",
"BMNormal_Click"
   End If
BMNormal.Enabled = True
```

```
End Sub

Sub BModoHost_Click ()
Dim tramarec As String

  If ModoTNC = NORMAL Then
        BModoHost.Enabled = False
        EscribirSerie "interface host" + Chr$(13)
        tempoms (1000)

        tramaEnl = ""
        EscribirSerie "reset" + Chr$(13)
        'ya debe de entrar en Modo Host y responde con una trama HOST!!
        tempoms (1000)
        ModoTNC = HOST
        ActualizarLModoTNC HOST
        AddTexto Text2, "" + CRLF
        BModoHost.Enabled = True
        Text1.SetFocus

        MonitorRX = False
 End If
End Sub


Sub BMParam_Click ()

  BMParam.Enabled = False
  If estadoAplic = ACTUANDO Then
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(MODO_P) + Chr$(MODO_P)
        tramarec = RecibirHost(5)
        Select Case Mid$(tramarec, 4, 1)
          Case Chr$(R_MDP)
                '**** MODO PARAMETRIZAR
                DetModoAplic PARAMETRIZAR
                'BInicio.Enabled = False
                BParam_Click
          Case Chr$(WAIT)
                BMParam.Enabled = True
                AddTexto Text2, "Estacion Num. " + Str$(estEspecifica) + " en WAIT. No
puede ponerse en MODO PARAMETRIZAR" + CRLF
          Case Else
                If estadoAplic = DESCONECTADO Then
                  AddTexto Text2, "ESTACION NO CONECTADA: Vuelva a pulsar
ACTUAR" + CRLF
                  DetModoAplic NORMALREPOSO
                  Text4.Text = ""
                  Exit Sub
                Else
                  BMParam.Enabled = True
                  AddTexto Text2, "NO RESPUESTA A MODO_P. Vuelva a pulsar MODO
PARAMETRIZAR" + CRLF
                End If
        End Select
  Else
        AddTexto Text2, "ESTACION NO CONECTADA: IMPOSIBLE ENTRAR EN MODO
PARAMETRIZAR" + CRLF
```

```
        DetModoAplic NORMALREPOSO
        Text4.Text = ""
    End If
End Sub

Sub BMTransp_Click ()

    If estadoAplic = ACTUANDO Then
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(MODO_T) + Chr$(MODO_T)
        tramarec = RecibirHost(5)
        BMTransp.Enabled = True
        Select Case Mid$(tramarec, 4, 1)
          Case Chr$(R_MDT)
                DetModoAplic TRANSPARENTE
                AddTexto Text2, "Estacion Num. " + Str$(estEspecifica) + " preparada para
actuación especifica" + CRLF
          Case Chr$(WAIT)
                BMTransp.Enabled = False
                AddTexto Text2, "Estacion Num. " + Str$(estEspecifica) + " en WAIT. Vuelva
a pulsar ACTUAR" + CRLF
                DesconectarDe estEspecifica
                DetModoAplic NORMALREPOSO
                Text4.Text = ""
          Case Else
                If estadoAplic = DESCONECTADO Then
                  AddTexto Text2, "ESTACION NO CONECTADA: Vuelva a pulsar
ACTUAR" + CRLF
                  DetModoAplic NORMALREPOSO
                  Text4.Text = ""
                  Exit Sub
                Else
                  BMTransp.Enabled = True
                  AddTexto Text2, "NO RESPUESTA A MODO_T. Vuelva a pulsar MODO
TRANSPARENTE" + CRLF
                End If
        End Select
    Else
        AddTexto Text2, "ESTACION NO CONECTADA: IMPOSIBLE ENTRAR EN MODO
TRANSPARENTE" + CRLF
        DetModoAplic NORMALREPOSO
        Text4.Text = ""
    End If
End Sub

Sub BParam_Click ()
Dim svbb%, RESPUESTA%, i%

    FParam.Show 1
    Select Case opcionParam
    Case 0
        'Se pulso cancelar
    Case 11  ' 1 = Parametrizar Tarjeta, Pedir Datos
    'Parametrizar Tarjeta
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_PP) + Chr$(&H1) +
Chr$(&H0)
        tramarec = RecibirHost(5)
        tramarec = QuitarTySVB(tramarec, svbb)
```

```
        Select Case Mid$(tramarec, 4, 1)
        Case Chr$(DAT_PR)
          AddTexto Text2, "   *** DATOS TARJETA: "
          For i = 6 To 15
                AddTexto Text2, Hex(Asc(Mid$(tramarec, i, 1))) + ", "
          Next i
          AddTexto Text2, " " + CRLF + CRLF

        Case Else
          AddTexto Text2, "   *** Respuesta NO DAT_PR" + CRLF + CRLF
        End Select
    Case 12 ' Param. Tarjeta, Introducir Datos
        FPARAMSIR.Show 1
        If cadParam <> "" Then  'si no se pulso cancelar
          EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_PP) + Chr$(&H1) +
Chr$(&H55) + cadParam
          tramarec = RecibirHost(5)
          tramarec = QuitarTySVB(tramarec, svbb)
          Select Case Mid$(tramarec, 4, 1)
          Case Chr$(DAT_PR)
                AddTexto Text2, "   *** DATOS TARJETA: "
                  For i = 6 To 15
                        AddTexto Text2, Hex(Asc(Mid$(tramarec, i, 1))) + ", "
                  Next i
                AddTexto Text2, " " + CRLF + CRLF
          Case Else
                AddTexto Text2, "   *** Respuesta NO DAT_PR" + CRLF + CRLF
          End Select
        End If
    Case 21 'Parametrizar IDS, Fichero Calibración
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_PP) + Chr$(&H5) +
Chr$(&H1)
        AddTexto Text2, CRLF + "ENVIADO SOLICITUD FICH. CAL. ESPERANDO
RESPUESTA DE ESTACION REMOTA..." + CRLF
        FCRP.Enabled = False
        tramarec = RecibirHost(120)
        Select Case Mid$(tramarec, 4, 1)
        Case Chr$(DAT_PR)
          ' Llega D2A<DAT_PR><05><TEXTO>
          AddTexto Text2, "   *** RESPUESTA FICH. CALIBRACION: " + Mid$(tramarec, 6)
+ CRLF + CRLF
        Case Else
          AddTexto Text2, "   *** Respuesta NO DAT_PR" + CRLF + CRLF
        End Select
        FCRP.Enabled = True
    Case 22 'Parametrizar IDS, Borrar Mem IDS
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_PP) + Chr$(&H5) +
Chr$(&H2)
        AddTexto Text2, CRLF + "ENVIADO BORRAR MEM. IDS. ESPERANDO
RESPUESTA DE ESTACION REMOTA..." + CRLF

        FCRP.Enabled = False
        tramarec = RecibirHost(120)
        Select Case Mid$(tramarec, 4, 1)
        Case Chr$(DAT_PR)
          ' Llega D2A<DAT_PR><05><TEXTO>
```

```
            AddTexto Text2, "   *** RESPUESTA BORRAR MEM. IDS: " + Mid$(tramarec, 6) +
CRLF + CRLF
        Case Else
            AddTexto Text2, "   *** Respuesta NO DAT_PR" + CRLF + CRLF
        End Select

        FCRP.Enabled = True

    Case 23 'Fichero Configuración IDS
        EnviarFicheroIDS estEspecifica

        FCRP.Enabled = False
        tramarec = RecibirHost(60)

        Select Case Mid$(tramarec, 4, 1)
        Case Chr$(R_CIDS)
            RecibirFicheroIDS tramarec
        Case Chr$(DAT_PR)
            AddTexto Text2, "   *** [BParam] DATOS PARAM: " + Mid$(tramarec, 5) + CRLF
        Case Else
            AddTexto Text2, "   *** NO RESPUESTA AL ENVIO DE FICHERO IDS" + CRLF +
CRLF
        End Select
        FCRP.Enabled = True
        DetEstadoAplic ACTUANDO

    Case 24 'Sincronizar GPS-IDS
        FCRP.Enabled = False
        AddTexto Text2, CRLF + "ENVIADO SOLICITUD SINCRONIZAR GPS-IDS.
ESPERANDO RESPUESTA DE ESTACION REMOTA..." + CRLF
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_PP) + Chr$(&H5) +
Chr$(&H4)

        tramarec = RecibirHost(5)

        Select Case Mid$(tramarec, 4, 1)
        Case Chr$(R_MDN)
            AddTexto Text2, "   *** RECIBIDA RESPUESTA A SINCRONIZAR GPS-IDS.
PASANDO A MODO NORMAL..." + CRLF + CRLF
        Case Else
            AddTexto Text2, "   *** NO Respuesta A SINCRONIZAR GPS-IDS. PASANDO A
MODO NORMAL..." + CRLF + CRLF
        End Select

            'Paso a Modo Normal
            DetModoAplic NORMALREPOSO
            AddTexto Text2, "Estacion Num. " + Str$(estEspecifica) + " en Modo Normal" +
CRLF
            DesconectarDe estEspecifica
            DetEstadoAplic DESCONECTADO

            Text4.Text = ""
            'BInicio.Enabled = True
            BActuar.Enabled = True

            EstableceProxSondeo
            timer4.Enabled = True
```

```
            'timer4.Enabled = True
            'LTSondeo.Caption = ProxSondeo

        FCRP.Enabled = True

    Case 25 'Parametrizar IDS, Calibración Tiempo
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_PP) + Chr$(&H5) +
Chr$(&H5)
        AddTexto Text2, CRLF + "ENVIADO SOLICITUD CREAR FICH. CAL. TIEMPO." +
CRLF + "ESPERANDO RESPUESTA DE ESTACION REMOTA..." + CRLF
        FCRP.Enabled = False
        tramarec = RecibirHost(120)
        Select Case Mid$(tramarec, 4, 1)
        Case Chr$(DAT_PR)
          ' Llega D2A<DAT_PR><05><TEXTO>
          AddTexto Text2, "   *** RESPUESTA FICHERO.GPS: " + Mid$(tramarec, 6) +
CRLF + CRLF
        Case Else
          AddTexto Text2, "   *** NO RESPUESTA A FICHERO.GPS !!" + CRLF + CRLF
        End Select
        FCRP.Enabled = True

    End Select

End Sub

Sub BResetIDS_Click ()
Dim resp%
    resp = MsgBox("Se va a enviar un RESET a la IDS de la ER" + Str$(estEspecifica) +
CRLF + "¿Desea continuar?", 49)
    '2=CANCEL
    If resp = 2 Then Exit Sub

    EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(RESET_ID) + Chr$(RESET_ID)
    AddTexto Text2, "ENVIADO RESET_ID. ESPERANDO RESPUESTA..." + CRLF
    FCRP.Enabled = False
    tramarec = RecibirHost(20)
    If Mid$(tramarec, 4, 1) <> Chr$(R_RSTID) Then
        AddTexto Text2, "NO RESPUESTA A RESET_ID" + CRLF
        'ANOTAR EN DIAGNOSTICO
        AnotarEnDiag "D2A-RESET_ID", tramarec
    Else
        'AddTexto Text2, "R_RSTID: " + Mid$(tramaRec, 5) + CRLF
    End If
    FCRP.Enabled = True
End Sub

Sub BResetTC_Click ()
Dim estRST%, resp%
'pedir antes el Nº de Estacion
' y enviar el RESET


    If estadoAplic = ACTUANDO Then
        resp = MsgBox("Se va a efectuar un RESET en la tarjeta de la ER" +
Str$(estEspecifica) + CRLF + "¿Desea continuar?", 49)
```

```
        '2=CANCEL
        If resp = 2 Then Exit Sub

        resp = MsgBox("Está seguro de efectuar un RESET en la tarjeta de ER" +
Str$(estEspecifica) + CRLF + "¿Confirmar?", 49)
        If resp = 2 Then Exit Sub

        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(P_RESET) + Chr$(P_RESET)
        AddTexto Text2, "ENVIADO P_RESET. ESPERANDO RESPUESTA..." + CRLF
        FCRP.Enabled = False

        tramarec = RecibirHost(5)

        If Mid$(tramarec, 4, 1) <> Chr$(R_RESET) Then
           AddTexto Text2, "NO RESPUESTA A P_RESET" + CRLF
           'ANOTAR EN DIAGNOSTICO
           AnotarEnDiag "D2A-P_RESET", tramarec
        Else
           AddTexto Text2, "   *** RECIBIDO CONFORME. R_RESET: " + Mid$(tramarec, 5)
+ CRLF
        End If
        FCRP.Enabled = True

        DesconectarDe estEspecifica
        DetModoAplic NORMALREPOSO
        DetEstadoAplic DESCONECTADO
   Else
        AddTexto Text2, CRLF + "!! DEBE ESTAR ACTUANDO CON UNA ESTACION
ESPECIFICA !!" + CRLF
   End If
End Sub

Sub BSalirHost_Click ()

   If ModoTNC = HOST Then     'Si ya estoy en modo NORMAL, no hago nada
        EscribirSerie "Q"
        ModoTNC = NORMAL
        ActualizarLModoTNC NORMAL
        Text1.SetFocus
        MonitorRX = True
   End If
End Sub

Sub BSuperUsu_Click ()

   If MODOSUPERV = False Then
        FSUPERV.Show 1

        If passw = "PEPE" Or passw = "FEDE" Then
           MODOSUPERV = True
           BModoHost.Visible = True
           BSalirHost.Visible = True
           BVolcado.Visible = True
           BDisconnect.Visible = True

           BTestSRAM.Visible = True
           BTestSRAM.Enabled = False
```

```
            MConfigurar.Enabled = True
            MMonitor.Enabled = True
        End If
    Else
        BModoHost.Visible = False
        BSalirHost.Visible = False
        BVolcado.Visible = False
        BDisconnect.Visible = False
        BTestSRAM.Visible = False
        MConfigurar.Enabled = False


        MODOSUPERV = False
    End If
End Sub
```

Sub BTestSRAM_Click ()

```
Dim svbb%, direc%, Byte1%, Byte2%, dato%
' Visualiza el formulario FTESTSRAM para trabajar directamente con la memoria
' del SIR. Se utiliza para realizar test remotos a la memoria del SIR.


    FTESTSRAM.Show 1

    Select Case OpcionSRAM
    Case 0
        'Se pulso cancelar
    Case 1  ' 1 = Parametrizar Tarjeta, Pedir Datos
        AddTexto Text2, "LEER. DIR: " + FTESTSRAM.Text1 + CRLF

        direc = Val(FTESTSRAM.Text1)
        Byte1 = direc \ 256     'cociente
        Byte2 = direc Mod 256   'resto

        AddTexto Text2, "LEER. DIR: " + FTESTSRAM.Text1 + " Byte1: " + Str$(Byte1) + "
Byte2: " + Str$(Byte2) + CRLF

        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(TEST_L) + Chr$(Byte2) +
Chr$(Byte1)

        tramarec = RecibirHost(6)
        tramarec = QuitarTySVB(tramarec, svbb)

        Select Case Mid$(tramarec, 4, 1)
        Case Chr$(R_TEST)
            AddTexto Text2, "   *** DATOS: " + Hex(Asc(Mid$(tramarec, 6, 1))) + CRLF
        Case Else
            AddTexto Text2, "   *** NO Respuesta A TEST" + CRLF + CRLF
        End Select

    Case 2
        AddTexto Text2, "ESCRIBIR. DIR: " + FTESTSRAM.Text1
        AddTexto Text2, "  DATO: " + FTESTSRAM.Text2 + CRLF

        direc = Val(FTESTSRAM.Text1)
        Byte1 = direc \ 256     'cociente
```

```
        Byte2 = direc Mod 256   'resto

        'Solo cojo los 2 primeros caracteres para asegurarnos
        dato = Val("&H" + Mid$(FTESTSRAM.Text2, 1, 2))

        AddTexto Text2, "   *** LEER. DIR: " + FTESTSRAM.Text1 + " Byte1: " + Str$(Byte1)
+ "  Byte2: " + Str$(Byte2)
        AddTexto Text2, "  DATO: " + Str$(dato) + CRLF


        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(TEST_E) + Chr$(Byte2) +
Chr$(Byte1) + Chr$(dato)

        tramarec = RecibirHost(6)
        tramarec = QuitarTySVB(tramarec, svbb)

        Select Case Mid$(tramarec, 4, 1)
        Case Chr$(R_TEST)
           AddTexto Text2, "   *** DATOS: " + Hex(Asc(Mid$(tramarec, 6, 1))) + CRLF
        Case Else
           AddTexto Text2, "   *** NO Respuesta A TEST" + CRLF + CRLF
        End Select



    End Select

End Sub

Sub BTime_Click ()
Dim horasist$, cadena$
Dim dd$, mm$, aa$, hh$, mi$, ss$
Dim a$, c$, m$, d$, cadrec$, h$, s$

On Error GoTo RE_BTIME

    FTime.Show 1

    Select Case opcionTime
    Case 1  'Mirar Hora ER
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(P_TIME) + Chr$(&H0)
        AddTexto Text2, " " + CRLF  'obligar un salto porque con Chr$(0) no lo hace
        tramarec = RecibirHost(5)
        If Mid$(tramarec, 4, 1) = Chr$(R_TIME) Then
           horasist = ConvertirEnHora(tramarec)
           AddTexto Text2, "   *** HORA DE EST. REMOTA: " + horasist + CRLF
        Else
           AddTexto Text2, "NO RESPUESTA A R_TIME" + CRLF
           'ANOTAR EN DIAGNOSTICO
           AnotarEnDiag "D2A-P_TIME", tramarec
        End If
    Case 2
    ' Introducir Hora Sistema a ER
        dd = Format$(Date, "dd")
        mm = Format$(Date, "mm")
        aa = Format$(Date, "yy")
        hh = Format$(Time, "hh")
```

```
        mi = Format$(Time, "nn")
        ss = Format$(Time, "ss")
        horasist = ConvBCD(dd) + ConvBCD(mm) + ConvBCD(aa) + ConvBCD(hh) +
ConvBCD(mi) + ConvBCD(ss)
        'convertir a binario cada dígito y enviar en BCD
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(P_TIME) + Chr$(&H55) +
horasist
        horasist = hh + ":" + mi + ":" + ss
        AddTexto Text1, "Hora Enviada: " + Format(Date, "dd-mm-yy ") + horasist + CRLF
        tramarec = RecibirHost(5)
        If Mid$(tramarec, 4, 1) = Chr$(R_TIME) Then
           horasist = ConvertirEnHora(tramarec)
           AddTexto Text2, "   *** R_TIME: " + horasist + CRLF
        Else
           AddTexto Text2, "NO RESPUESTA A R_TIME" + CRLF
           'ANOTAR EN DIAGNOSTICO
           AnotarEnDiag "D2A-R_TIME", tramarec
        End If
  Case 3  'Hora IDS
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) + "%DTB" + CR
        tramarec = RecibirHost(5)
        If Mid$(tramarec, 4, 1) = Chr$(DAT_TR) Then
           'cadena = D2A6%DTB<CR>aamdhmsxxx<CR>TTC>>
           cadena = Mid$(tramarec, 10)
           a = Hex$(Asc(Mid$(cadena, 1, 1)))
           c = Hex$(Asc(Mid$(cadena, 2, 1)))
           m = Hex$(Asc(Mid$(cadena, 3, 1)))
           d = Hex$(Asc(Mid$(cadena, 4, 1)))
           cadrec = Trim$(d) + "-" + Trim$(m) + "-" + Trim$(c) + Trim$(a)

           h = Hex$(Asc(Mid$(cadena, 5, 1)))
           m = Hex$(Asc(Mid$(cadena, 6, 1)))
           s = Hex$(Asc(Mid$(cadena, 7, 1)))
           cadrec = cadrec + " " + h + ":" + Trim$(m) + ":" + Trim$(s)

           AddTexto Text2, "   *** RECIBIDO HORA IDS: " + cadrec + CRLF
        Else
           AddTexto Text2, "NO RESPUESTA A HORA IDS" + CRLF
           'ANOTAR EN DIAGNOSTICO
           AnotarEnDiag "D2A-R_IDS", tramarec
        End If
  Case 4  'Hora GPS
        EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) + "%DTGB" + CR
        tramarec = RecibirHost(5)
        If Mid$(tramarec, 4, 1) = Chr$(DAT_TR) Then
           'cadena = D2A6%DTGB<CR>aamdhmsxxx<CR>TTC>>
           cadena = Mid$(tramarec, 11)
           a = Hex$(Asc(Mid$(cadena, 1, 1)))
           c = Hex$(Asc(Mid$(cadena, 2, 1)))
           m = Hex$(Asc(Mid$(cadena, 3, 1)))
           d = Hex$(Asc(Mid$(cadena, 4, 1)))
           cadrec = Trim$(d) + "-" + Trim$(m) + "-" + Trim$(c) + Trim$(a)


           h = Hex$(Asc(Mid$(cadena, 5, 1)))
           m = Hex$(Asc(Mid$(cadena, 6, 1)))
           s = Hex$(Asc(Mid$(cadena, 7, 1)))
```

```
            cadrec = cadrec + " " + h + ":" + Trim$(m) + ":" + Trim$(s)

            AddTexto Text2, "   *** RECIBIDO HORA GPS: " + cadrec + CRLF
         Else
            AddTexto Text2, "NO RESPUESTA A HORA GPS" + CRLF
            'ANOTAR EN DIAGNOSTICO
            AnotarEnDiag "D2A-R_GPS", tramarec
         End If

   End Select
   Exit Sub

RE_BTIME:
   If Err = 13 Then 'Type mismatch
         AddTexto Text2, "[BTime] Type mismatch" + CRLF
         AnotarError "[BTime] Type mismatch", "BTime_Click"
   End If
   AddTexto Text2, "Error[BTime]: " + Error$ + " - Ver ERRORES.LOG"
   AnotarError Error$, "BTime"

   Resume Next

End Sub

Sub BVolcado_Click ()

   Open dir_err + "VOLCADO.LOG" For Append As #1
   Write #1, Format$(Date$, "dd-mm-yy"), Time$
   Write #1, Text2.Text + CRLF
   Close

   Text2.Text = ""

End Sub

Sub CerrarPuerto ()
   If Comm1.PortOpen Then
         Comm1.PortOpen = False
   End If
End Sub

Static Sub Comm1_OnComm ()
Dim EVMsg As String
Dim ERMsg As String
Dim basura$
Dim vr%, cont As Integer
Dim antrama%
Dim estacion As Integer
Dim ik%
' Funcion interna del puerto serie "Comm1" de Visual Basic, que gestiona
' la recepción de caracteres y controla los errores del puerto serie.


On Error GoTo RE_COMM1

 'Saltar de acuerdo con la propiedad CommEvent
 Select Case Comm1.CommEvent
```

```
'Mensajes relativos a sucesos

Case MSCOMM_EV_RECEIVE
  If ModoTNC = NORMAL Then
      RecibirCom
  Else      'MODO HOST
        rec_trama = False
        While rec_trama = False
              LeerSerie
        Wend


        If Mid$(tramaEnl, 1, 3) = "R20" Then    'R20*** CONNECT REQUEST: ESTAR1
              If ModoAplic = NORMALREPOSO Then
                 AddTexto Text2, "   *** RECIBIDA: " + tramaEnl + CRLF
                 EstEntrante = Val(Mid$(tramaEnl, 30, 1))

                 'solo acepto REQUEST en Normal Reposo
                 If Not SondeosEnProceso Then BInicio_Click
              End If
        End If

        If MonitorFIDS = False Then
              Select Case Mid$(tramaEnl, 4, 1)
                 Case Chr$(DAT_AV), Chr$(R_BATT), Chr$(DAT_PR), Chr$(DAT_FI),
Chr$(R_FDIAG), Chr$(R_TIME)
                        ' nada
                 Case Chr$(ERROR_F)
                        AddTexto Text2, "   *** RECIBIDO ERROR_F DE ESTACION " +
Str$(Asc(Mid$(tramaEnl, 3, 1)) - 64) + CRLF
                 Case Chr$(R_FDIAG)
                        ShowTexto Text2, "   *** THR: " + tramaEnl + CRLF
                 Case Chr$(R_TEST)
                        AddTexto Text2, "   *** THR: " + Mid$(tramarec, 1, 3) + "*"

                        For ik = 4 To Len(tramarec)
                           AddTexto Text2, Hex(Asc(Mid$(tramarec, ik, 1))) + "*"
                        Next ik
                        AddTexto Text2, " " + CRLF

                 Case Else
                        ' los comandos "C" y los "CONNECT" no los muestro ya
                        If Mid$(tramaEnl, 4, 8) = "*** CONN" Or (Mid$(tramaEnl, 1, 1) = "C"
And Len(tramaEnl) = 3) Then
                        'If Mid$(tramaEnl, 4, 8) <> "*** CONN" Then
                           'AddTexto FCRP.Text2, "THR: " + Mid$(tramaEnl, 1, 4) + CRLF
                        Else
                           If Mid$(tramaEnl, 4, 2) = "6%" Then 'DAT_TR de hora IDS y Hora
GPS
                                 'nada
                           Else
                                 If ModoAplic = TRANSPARENTE Then
                                    ShowTexto FCRP.Text2, "   *** THR: " + tramaEnl + CRLF
                                 Else
                                    ShowTexto FCRP.Text2, "   *** THR: " + tramaEnl + CRLF
                                 End If
                           End If
```

```
                End If
            End Select
        Else
            ShowTexto FIDS.Text1, "   *** THR[" + Str$(Len(tramaEnl)) + "]: " + tramaEnl
+ CRLF
            Exit Sub
        End If
        'parece ser que se pierde tramaEnl si se llama a AnalizarCCA
        If Mid$(tramaEnl, 1, 1) = "D" Then
            antrama = AnalizarCCA(tramaEnl)
            If antrama = 2 Then
              tramaEnl = ""   'tiro la trama
              Exit Sub
            End If
        End If


        Select Case Mid$(tramaEnl, 4, 1)
            Case Chr$(TNACK_R)
              numTNACK_R = numTNACK_R + 1
            Case Else
              numTNACK_R = 0
        End Select

        'If estadoAplic <> CONECTANDO And Mid$(tramaEnl, 4, 8) = "*** DISC" Then
        '   AddTexto Text2, "RECIBIDA PETICION DE DESCONEXION " + CRLF + CRLF
        '   DetEstadoAplic DESCONECTADO
        'End If


        Select Case Mid$(tramaEnl, 4, 1)
            Case Chr$(TNACK_R)
              estacion = Asc(Mid$(THE, 3, 1)) - 64
              AnotarEnDiag "[Comm1_TNACKR] " + THE, tramaEnl
              'numTNACK_R = numTNACK_R + 1
              If numTNACK_R > 3 Then
                    AddTexto Text2, "Estacion ESTA" + LTrim$(Str$(estacion)) + " dada
de baja por recibir 3 TNACK_R consecutivos" + CRLF
                    DesconectarDe estacion
                    'desconecto la estacion
                    numTNACK_R = 0
              Else
                    If Mid$(THE, 4, 1) <> Chr$(SONDEO) Then
                    ' los sondeos no se repiten y se deja la Aplicación que lo repita
                    'donde toque
                      EscribirSerie THE   'repito la ultima trama enviada
                    End If
              End If
              tramaEnl = ""
        End Select

    End If

  Case MSCOMM_EV_SEND

  Case MSCOMM_EV_CTS
   EVMsg = "Cambio detectado en CTS"
```

```
  Case MSCOMM_EV_DSR
   EVMsg = "Cambio detectado en DSR"
  Case MSCOMM_EV_CD
   'EVMsg = "Cambio detectado en DCD"
  Case MSCOMM_EV_RING
   EVMsg = "Evento RING"
  Case MSCOMM_EV_EOF
   'EVMsg = "Fin de fichero"

  'Mensajes de error
  Case MSCOMM_ER_BREAK   '1001
   'EVMsg = "Interrupción detectada"
   Comm1.PortOpen = False
   tempoms (500)
   Comm1.PortOpen = True

  Case MSCOMM_ER_CTSTO
   ERMsg = "Tiempo para CTS sobrepasado"
  Case MSCOMM_ER_DSRTO
   ERMsg = "Tiempo para DSR sobrepasado"
  Case MSCOMM_ER_FRAME   '1004
   'MsgBox "Error de transmisión (encuadre)"

   MsgBox "TNC apagada?" + CRLF + "ENCIENDA TNC Y PULSE ACEPTAR", 32
   If ModoTNC = HOST Then
       cont = 1
       Do
          basura = Comm1.Input
          If InStr(basura, "S00") = 0 Then
               MsgBox "TNC apagada?" + CRLF + "ENCIENDA TNC Y PULSE
ACEPTAR", 32
               cont = cont + 1
          End If
          If cont = 5 Then
               MsgBox "ERROR FATAL: " + CRLF + "TNC NO RESPONDIO A 5
INTENTOS: AVERIADA?", 16
               Close
               Comm1.PortOpen = False
               End
          End If
       Loop Until InStr(basura, "S00") <> 0
   Else
       basura = Comm1.Input
   End If

  Case MSCOMM_ER_OVERRUN
   'ERMsg = "Error de sobrescritura"
  Case MSCOMM_ER_CDTO
   ERMsg = "Tiempo para DCD sobrepasado"
  Case MSCOMM_ER_RXOVER
   'ERMsg = "Buffer de recepción lleno"
  Case MSCOMM_ER_RXPARITY
   EVMsg = "Error de paridad"
  Case MSCOMM_ER_TXFULL
   ERMsg = "Buffer de transmisión lleno"
  Case Else
   ERMsg = "Error o suceso desconocido"
```

```
  End Select

  If Len(EVMsg) Then
    Open dir_err + "com1ev.log" For Append As #10
    Write #10, CRLF + Format$(Date$, "dd-mm-yy"), Time$, "Trama Rec: " + tramarec
    Write #10, "Error producido: " + EVMsg
    Write #10, "Estado puerto: " + CRLF
    Write #10, "PortOpen: " + Str$(Comm1.PortOpen)
    Write #10, "DSRHolding: " + Str$(Comm1.DSRHolding)
    Write #10, "CDHolding: " + Str$(Comm1.CDHolding)
    Close #10

    'Visualizar el mensaje EVMsg
    AddTexto Text2, EVMsg + " - Ver COM1.EV.LOG" + CRLF
    AnotarError EVMsg, "Comm1_OnComm"

    EVMsg = ""
  ElseIf Len(ERMsg) Then
    'Visualizar el mensaje de error
    Beep
    vr = MsgBox(ERMsg, 1, "Pulse Cancelar para salir, Aceptar para ignorar." + CRLF + "Ver
COM1EV.LOG")
    Open dir_err + "com1ev.log" For Append As #10
    Write #10, CRLF + Format$(Date$, "dd-mm-yy"), Time$, "Trama Rec: " + tramarec
    Write #10, "Error producido: " + ERMsg
    Write #10, "Estado puerto: " + CRLF
    Write #10, "PortOpen: " + Str$(Comm1.PortOpen)
    Write #10, "DSRHolding: " + Str$(Comm1.DSRHolding)
    Write #10, "CDHolding: " + Str$(Comm1.CDHolding)
    Close #10

    ERMsg = ""
    'Si se pulsó Cancelar
    If vr = 2 Then
      Comm1.PortOpen = False 'Cerrar el puerto y salir
      End
    End If
  End If

RE_COMM1:
    If Err = 76 Then    '76=Path not found
          AddTexto Text2, "   *** PETICION CD A DIR NO VALIDO" + CRLF
    End If
    'AddTexto Text2, "Error[Comm1_OnComm]: " + Error$ + " - Ver ERRORES.LOG"
    'AnotarError Error$, "Comm1_OnComm"

    Resume Next


End Sub


Sub ConectarCon (NEst As Integer)
Dim NumIntentos%, EstAConectar$, conected%, cadenv$, cadrec$
' Solicitud del establecimiento de una sesión con una estacion especifica
' CONNECT.request de la aplicación de usuario al nivel de SesionLL
```

```
    NumIntentos = 1
    EstAConectar = LTrim$(Str$(NEst))
    conected = False
    AddTexto Text2, Now + CRLF

  RepetirCONN:
    DetEstadoAplic CONECTANDO

    cadenv = "C2" + Chr$(64 + NEst) + "C ESTA" + EstAConectar

    AddTexto Text2, cadenv + ". NumIntento: " + Str$(NumIntentos) + CRLF

    EscribirSerie cadenv


    Do
    ' el TNC responde a un comando con "C00" y hay que tomarlo e ignorar
          tramarec = RecibirHost(5)
          cadrec = Mid$(tramarec, 4, 8)

          If Mid$(tramarec, 4, 1) = Chr$(SESION_R) Then
                AddTexto Text2, "SESION ESTABLECIDA CON ESTA" + EstAConectar +
CRLF + CRLF
                EstAct(NEst) = 1
                NumEstCon = NumEstCon + 1
                conected = True
                Exit Sub
          End If
          Select Case cadrec
            Case "**** DISC", "retry co"
                EstAct(NEst) = 0
            Case "**** CONN"
                AddTexto Text2, "ENLACE ESTABLECIDO CON ESTA" + EstAConectar +
CRLF
                EstAct(NEst) = -1
                ' conectada a nivel AX.25, pero NO sesion establecida!!
          End Select
    Loop Until cadrec = "**** DISC" Or tramarec = ""

    If (conected = False) And (NumIntentos < 3) Then
          NumIntentos = NumIntentos + 1
          If EstAct(NEst) = -1 Then DesconectarDe NEst
          GoTo RepetirCONN
    End If

    If conected = False Then
          AddTexto Text2, "SESION CON ESTA" + EstAConectar + " SIN EXITO" + CRLF
          If EstAct(NEst) = -1 Then DesconectarDe NEst
    End If

    Text1.SetFocus


  End Sub
```

```
Sub ConectarEstaciones (EstIni As Integer)
' Establece sesiones con todas las estaciones activas
' EstIni= num estacion por la que se empieza a conectar
Dim i%

    If ModoTNC = NORMAL Then    'poner TNC en Modo HOST
        BModoHost_Click
    End If
    NumEstCon = 0

    FCRP.Text3.Text = ""

    For i = 1 To TotalEst
        If EstIni > TotalEst Then EstIni = 1

        If LEU(i) = 1 Then          'si Lista_Estaciones_Usuario = 1
            ConectarCon EstIni
        End If
        EstIni = EstIni + 1
    Next i

    ActualizarListaEstAct

    If NumEstCon = 0 Then
        DetEstadoAplic DESCONECTADO
    Else
        DetEstadoAplic CONECTADO
    End If
End Sub

Function ConvBCD (N As String) As String
Dim cad$, dig$, i%, num%

    cad = ""
    For i = 1 To 2
        dig = Mid$(N, i, 1)
        Select Case dig
            Case "0"
                cad = cad + "0000"
            Case "1"
                cad = cad + "0001"
            Case "2"
                cad = cad + "0010"
            Case "3"
                cad = cad + "0011"
            Case "4"
                cad = cad + "0100"
            Case "5"
                cad = cad + "0101"
            Case "6"
                cad = cad + "0110"
            Case "7"
                cad = cad + "0111"
            Case "8"
                cad = cad + "1000"
            Case "9"
```

```
                cad = cad + "1001"
            End Select
        Next i
        For i = 1 To 8
            If Mid$(cad, 9 - i, 1) = "1" Then num = num + pot2(i - 1)
        Next i
        ConvBCD = Chr$(num)


    End Function

    Function ConvBCDenNum (BCD As Integer) As String
    Dim BCDAlto%, BCDBajo%, Rto%

        BCDAlto = BCD And 240
        BCDBajo = BCD And 15
        Rto = 0
        Do
            If BCDAlto >= 128 Then
                Rto = Rto + 8
                BCDAlto = BCDAlto - 128
            Else
                If BCDAlto >= 64 Then
                    Rto = Rto + 4
                    BCDAlto = BCDAlto - 64
                Else
                    If BCDAlto >= 32 Then
                        Rto = Rto + 2
                        BCDAlto = BCDAlto - 32
                    Else
                        If BCDAlto >= 16 Then
                            Rto = Rto + 1
                            BCDAlto = BCDAlto - 16
                        End If
                    End If
                End If
            End If
        Loop Until BCDAlto = 0
        ConvBCDenNum = LTrim$(Str$(Rto)) + LTrim$(Str$(BCDBajo))
    End Function

    Function ConvertirEnHora (trama As String) As String
    Dim d$, m$, a$, h$, mi$, s$

        trama = Mid$(trama, 5)   'le quito D2A<R_TIME>
        d = ConvBCDenNum(Asc(Mid$(trama, 1, 1)))
        m = ConvBCDenNum(Asc(Mid$(trama, 2, 1)))
        a = ConvBCDenNum(Asc(Mid$(trama, 3, 1)))
        h = ConvBCDenNum(Asc(Mid$(trama, 4, 1)))
        mi = ConvBCDenNum(Asc(Mid$(trama, 5, 1)))
        s = ConvBCDenNum(Asc(Mid$(trama, 6, 1)))
        ConvertirEnHora = d + "-" + m + "-" + a + " " + h + ":" + mi + ":" + s
    End Function
```

```
Sub DesconectarDe (NEst As Integer)
Dim comandrec$
' Solicita la desconexión de la sesion con una estacion especifica
' DISCONNECT.request (estacion X)

   AddTexto Text2, "Desconectando de ESTA" + LTrim$(Str$(NEst)) + CRLF
   EstAct(NEst) = 0   'la quito de la lista de estaciones activas
   EscribirSerie "C2" + Chr$(64 + NEst) + "D" 'envio DISC por canal adecuado

   Do
   ' la TNC responde a un comando con "C00" y hay que tomarlo e ignorar
         tramarec = RecibirHost(5)
         comandrec = Mid$(tramarec, 4, 6)
   Loop Until comandrec = "Can't" Or comandrec = "*** DI" Or tramarec = ""

   Text4.Text = "" 'borrar Estacion Especifica
   If ModoAplic = TRANSPARENTE Or ModoAplic = PARAMETRIZAR Then
         DetModoAplic NORMALREPOSO
   End If
End Sub

Sub DesconEstaciones ()
Dim i%
' Solicita la desconexión de todas las sesiones establecidas


   If NumEstCon > 0 Then
   'si hay alguna estacion conectada se inician las desconexiones
         For i = 1 To TotalEst
            If EstAct(i) <> 0 Then DesconectarDe i
         Next i
   End If

   ActualizarListaEstAct
   DetEstadoAplic DESCONECTADO
   DetModoAplic NORMALREPOSO

   NumEstCon = 0
   Text1.SetFocus
End Sub

Sub DeterminarModoTNC ()
' Funcion de la aplicación de usuario para acceder al estado del nodo de la
' subred de comunicaciones

   ModoTNC = HOST
   tramaEnl = ""
   EscribirSerie "C2AV"
   tempoms (500)
   If Mid$(tramaEnl, 1, 5) = "C00DE" Then
         ' estaba en HOST
         ModoTNC = HOST
         ActualizarLModoTNC HOST
   Else
         ' tramaEnl será ""
         ModoTNC = NORMAL
         ' le envio un CR para que trague el C2AV
```

```
        EscribirSerie Chr$(13)
        ' esperar 1 seg a que me devuelva el "eh?"
        tempoms (500)
        ActualizarLModoTNC NORMAL
    End If
    Text2.Text = ""    ' quito posible basura recibida
End Sub

Sub DetEstadoAplic (estado As Integer)
' Actualiza la etiqueta (Estado de la Aplicación) del interface gráfico
' de usuario

   estadoAplic = estado
   Select Case estadoAplic
        Case ACTUANDO
           LEstado.Caption = "ESTADO: ACTUANDO"
           BResetTC.Enabled = True
        Case DESCONECTADO
           LEstado.Caption = "ESTADO: DESCONECTADO"
           LEstado.BackColor = QBColor(11)   'ligth cyan
        Case CONECTADO
           LEstado.Caption = "ESTADO: CONECTADO"
           LEstado.BackColor = QBColor(10)   'ligth green
        Case ENVIANDO
           LEstado.Caption = "ESTADO: ENVIANDO"
        Case RECIBIENDO
           LEstado.Caption = "ESTADO: RECIBIENDO"
           LEstado.BackColor = QBColor(10)   'ligth green
        Case SONDEANDO
           LEstado.Caption = "ESTADO: SONDEANDO"
           LEstado.BackColor = QBColor(12)   'ligth red
        Case CONECTANDO
           LEstado.Caption = "ESTADO: CONECTANDO"
        Case Else
           LEstado.Caption = "ESTADO: INDETERMINADO!!"
   End Select
End Sub

Sub DetModoAplic (estado As Integer)
' Actualiza la etiqueta (Modo de la Aplicación) y habilita/inhibe los botones
' correspondientes del interface gráfico de usuario


   ModoAplic = estado
   Select Case ModoAplic
        Case NORMALREPOSO
           LModoAplic.Caption = "M_APLIC: NORMAL_REPOSO"
           LModoAplic.BackColor = QBColor(11)  'Ligth Cyan
           LModoAplic.ForeColor = QBColor(0)  'Black

           BMNormal.Enabled = False
           BMTransp.Enabled = False
           BMParam.Enabled = False

           BBateria.Enabled = False
           BFDiag.Enabled = False
           BResetIDS.Enabled = False
```

```
        BResetTC.Enabled = False
        BTime.Enabled = False
        BFichTransp.Enabled = False
        BParam.Enabled = False

        BInicio.Enabled = True
        BActuar.Enabled = True

        BTestSRAM.Enabled = False

        EstableceProxSondeo
        timer4.Enabled = True
        'timer4.Enabled = True
        'LTSondeo.Caption = ProxSondeo

    Case NORMALSONDEO
        LModoAplic.Caption = "M_APLIC: NORMAL_SONDEO"
        LModoAplic.BackColor = QBColor(10)  'Ligth Green
        LModoAplic.ForeColor = QBColor(0)  'Black

        BMNormal.Enabled = False
        'BMTransp.Enabled = False
        BMParam.Enabled = False



        timer4.Enabled = False
        LTSondeo.Caption = "--------"

    Case TRANSPARENTE
        LModoAplic.Caption = "M_APLIC: TRANSPARENTE"
        LModoAplic.BackColor = QBColor(13)  'Ligth Magenta
        LModoAplic.ForeColor = QBColor(0)  'Black

        BMNormal.Enabled = True
        BMTransp.Enabled = False
        BMParam.Enabled = False

        BBateria.Enabled = True
        BFDiag.Enabled = True
        BResetIDS.Enabled = True
        BTime.Enabled = True
        BFichTransp.Enabled = True

        If MODOSUPERV = True Then BTestSRAM.Enabled = True

        timer4.Enabled = False
        LTSondeo.Caption = "--------"

        BInicio.Enabled = False
        BActuar.Enabled = False

    Case PARAMETRIZAR
        LModoAplic.Caption = "M_APLIC: PARAMETRIZAR"
        LModoAplic.BackColor = QBColor(12)  'Ligth Red
        LModoAplic.ForeColor = QBColor(0)  'Black
```

```
            BMNormal.Enabled = True
            BMTransp.Enabled = False
            BMParam.Enabled = False

            BParam.Enabled = True
            BTestSRAM.Enabled = False

            timer4.Enabled = False
            LTSondeo.Caption = "-------"
            BInicio.Enabled = False
            BActuar.Enabled = False

        Case Else
            LModoAplic.Caption = "M_APLIC: INDETERMINADO"
            LModoAplic.BackColor = QBColor(4)
            LModoAplic.ForeColor = QBColor(0)  'Black
    End Select
End Sub

Sub EnviarFicheroIDS (numest As Integer)
' Procedimiento para enviar el fichero de configuración de la estacion de
' registro al SIR.
' Dado que el fichero es de 470 bytes, se envía completo sin SVB. Al
' finalizar se envía una SVB duplicada, el SIR envía este fichero a la
' estacion de registro y se lo pide de nuevo para remitirlo a la estacion
' central. Por comparación la estacion central determina si el fichero que
' envio coincide con el que actualmente opera en la estacion de registro.


Dim LonFich, NCE As Long
Dim car As String
Static NCFE As Long    'num car del fich enviados
Static ncfl As Long    'ultimo car fich leídos
Dim nct As Integer  'num car trama
Static nte%, ntre%, ntpe As Integer
Dim trama$, cadcrc$
Static crc%, NTCONF%, conexion As Integer
Static t1&, t2 As Long
Dim tiempo As Integer
Dim sigue As Integer
Dim Byte1%, Byte2 As Integer
Dim NCANAL As String
Dim fichero$

On Error GoTo RE_EnviarIDS


        NCANAL = LTrim$(Str$(numest))
        fichtx = dir_ids + "fisu_e" + NCANAL + "\isumod" + NCANAL + ".isu"


        LonFich = FileLen(fichtx)
        If LonFich > 65535 Then
            AddTexto Text2, "SOLICITUD DE ENVIO DE FICHERO MAYOR DE 64KB" +
CRLF
            AddTexto Text2, "FIN ENVIAR FICHERO" + CRLF
```

```
    GoTo SalirEnvIDS
  End If

  Open fichtx For Binary As #1
  fichero = Input$(LonFich, #1)
  Close #1


  DetEstadoAplic ENVIANDO
  nte = 0     ' num tramas enviadas
  ntre = 0    ' num tramas rechazadas
  ntpe = 0    ' num tramas parciales enviadas (entre 2 crc's)
  ncfl = 0    ' num car fich leidos
  NCFE = 0    ' num car fich enviados
  crc = 0
  NTCONF = 4  ' cada cuantas tramas envio crc
  NCE = 0     'num total de car enviados
  conexion = True

  t1 = Timer
  timer1.Enabled = False  ' deshabilito reloj para mayor velocidad


  While (ncfl < LonFich) And conexion   'Not EOF(1)
    nte = nte + 1
    ntpe = ntpe + 1
    nct = 0
    sigue = True

    trama = "D2" + Chr$(64 + numest) + Chr$(C_IDS)
    nct = 0


    While (nct < 256) And (ncfl < LonFich) And sigue
    'Not EOF(1)                'EOF en files Binary se pone a true cuando
                                  ' el ultimo Get efectuado no tenga éxito
        ncfl = ncfl + 1
        car = Mid$(fichero, ncfl, 1)

        Select Case car
        Case Chr$(FEND)
          If nct = 255 Then   'SI YA TENGO 255 EN LA TRAMA ...
                sigue = False
                ncfl = ncfl - 1
          Else
                trama = trama + Chr$(FESC) + Chr$(TFEND)
                nct = nct + 1   'la transparencia FEND la cuenta la TNC
          End If
        Case Chr$(FESC)
          If nct = 255 Then
                sigue = False
                ncfl = ncfl - 1
          Else
                trama = trama + Chr$(FESC) + Chr$(TFESC)
                nct = nct + 1   'la transparencia FEND la cuenta la TNC
          End If
        Case Else
```

```
                trama = trama & car
                nct = nct + 1        ' así las tramas son siempre de 255 car.
            End Select
        Wend

        EscribirSerie trama      ' trama ya lleva la transparencia

    Wend


    EscribirSerie "D2" + Chr$(64 + numest) + Chr$(SVB) + Chr$(SVB)


    AddTexto Text2, CRLF + "FICHERO " + fichtx + " ENVIADO" + CRLF
    DetEstadoAplic CONECTADO


SalirEnvIDS:
  Close
  timer1.Enabled = True
  Exit Sub

RE_EnviarIDS:
  If Err = 53 Then '53=file not found
        EscribirSerie "D2" + Chr$(64 + numest) + Chr$(NO_DAT) + "FICHERO NO EXISTE"
        DetEstadoAplic CONECTADO
        AddTexto Text2, "SOLICITUD DE ENVIO DE FICHERO NO EXISTENTE" + CRLF
        AddTexto Text2, "FIN ENVIAR FICHERO" + CRLF
        Resume SalirEnvIDS
  End If
  Open dir_err + "env-ids.err" For Append As #2
  Write #2, String(70, "-") + CRLF + Format$(Now, "dd-mm-yy hh:mm:ss")
  Write #2, Now, "nte: " + Str$(nte)
  Write #2, "trama enviada", trama
  Write #2, "trama recibida", tramarec
  Write #2, "Error producido: " + Error$
  Write #2, "NOMFICH: " + fichtx
  Close #2
  AddTexto Text2, "Error: " + Error$ + " - Ver ENV-IDS.ERR"
  AnotarError Error$ + " - Ver ENV-IDS.ERR", "EnviarFicheroIDS"

  Resume SalirEnvIDS
End Sub

Sub EscribirSerie (cad As String)
Dim i%
' Escribe en el puerto serie poniendo los guiones de principio y final
' de trama


  THE = cad   'trama host enviada
  If ModoTNC = HOST Then
        Comm1.Output = Chr$(FEND) + cad + Chr$(FEND)


        If Mid$(cad, 4, 1) <> "C" Then
        'las tramas de CONNECT las saca ConectarCon
```

```
        'si la trama son Datos_Fichero no sacar los 256 bytes por pantalla
          Select Case Mid$(cad, 4, 1)
          Case Chr$(DAT_FI), Chr$(C_IDS)
                cad = Mid$(cad, 1, 4)
                AddTexto Text2, cad + CRLF
          Case Chr$(REP_FI), Chr$(DEL_FI), Chr$(NACK_FI), Chr$(DAT_TP)
                AddTexto Text2, cad + CRLF

          Case Chr$(TEST_E), Chr$(TEST_L)
                AddTexto Text2, Mid$(cad, 1, 3) + "*"

                For i = 4 To Len(cad)
                   AddTexto Text2, Hex(Asc(Mid$(cad, i, 1))) + "*"
                Next i
                AddTexto Text2, " " + CRLF
          End Select
        End If
  Else   'MODO NORMAL
        Comm1.Output = cad
  End If

End Sub

Sub EstableceProxSondeo ()
Dim hora%, minu%, sec%, numsec
' Permite automatizar el tiempo del próximo sondeo a voluntad del usuario.


  ProxSondeo = Now
  hora = Hour(ProxSondeo)
  minu = Minute(ProxSondeo)
  sec = Second(ProxSondeo)
  numsec = TimeSerial(hora, minu, sec + tsondeo)
  ProxSondeo = Format$(numsec, "hh:mm:ss")

  LTSondeo.Caption = ProxSondeo

End Sub

Sub Form_Load ()
Dim i%, NFD$, nfs$, cadena$, NumCad%
Dim hModule As Integer, NumCargas As Integer
Dim msg$
On Error GoTo RE_FLoad
' Controla el arranque inicial de la aplicación y visualiza el interface
' gráfico de usuario.


 'evitar cargar dos veces CRP.EXE
 hModule = GetModuleHandle("crp.exe")
 NumCargas = GetModuleUsage(hModule)
 If NumCargas > 1 Then
   MsgBox "CRP.EXE ya está en ejecución", 16
   End
 End If
```

```
BModoHost.Visible = False
BSalirHost.Visible = False
BVolcado.Visible = False
BDisconnect.Visible = False
BTestSRAM.Visible = False
MConfigurar.Enabled = False
MODOSUPERV = False

BInicio.Enabled = False
BActuar.Enabled = False


'Establecer como directorio de trabajo el directorio
'donde se ejecuta la aplicación
ChDir App.Path

CRLF = Chr$(13) & Chr$(10)
CR = Chr$(13)
LFecha = Format(Now, "dd-mm-yy")  'fecha del sistema

'*** DIRECTORIOS
dir_eve = App.Path + "\eventos\"
dir_jul = App.Path + "\eventos\julio\"
dir_err = App.Path + "\errores\"
dir_dia = App.Path + "\diag\"
dir_app = App.Path + "\"
dir_ids = App.Path + "\fichids\"

'Establecer NumFDiag y NumFichSondeo
Open dir_app + "NFD.CFG" For Input As #1
Input #1, NFD
Close #1
NumFDiag = Val(NFD)

Open dir_app + "NFS.CFG" For Input As #1
Input #1, nfs
Close #1
NumFichSondeo = Val(nfs)


FCRP.Show  'hay que mostrar el Form antes de enfocar cualquier control

EstActivasElegidas = False
FEstAct.Show 1 'introducir estaciones activas

If EstActivasElegidas = True Then
      'ACTUALIZAR FICHERO
      Open dir_app + "EAUSER.CFG" For Output As #1
      'ultimas estaciones seleccionadas por el usuario
      For i = 0 To TotalEst - 1
         Write #1, LEU(i + 1)
      Next i
      Close #1
Else
      ' LEER DE FICHERO
      Open dir_app + "EAUSER.CFG" For Input As #1
```

```
        'ultimas estaciones seleccionadas por el usuario
          For i = 0 To TotalEst - 1
                Input #1, cadena
                LEU(i + 1) = Val(Trim$(cadena))
          Next i
        Close #1
    End If

    'determinar los parámetros de configuración del COM
    'leo del fichero soo al arrancar la aplicación CRP
    Open dir_app + "COM.cfg" For Input As #1
    NumCad = 0  'Numero de cadenas leídas
    While Not EOF(1)
        Line Input #1, cadena
        If Mid$(cadena, 1, 1) <> "#" Then
          NumCad = NumCad + 1
          Select Case NumCad
          Case 1
                NUMPUERTO = Val(cadena)
          Case 2
                PARAMETROS = cadena
          Case 3
                VELOCTNC = Val(cadena)
          End Select
        End If
    Wend
    Close #1
    AddTexto Text2, CRLF + "LEIDOS PARAMETROS DE CONFIGURACIÓN PUERTO
SERIE" + CRLF

    Select Case VELOCTNC
    Case 1200
        MVelTNC1200.Checked = True
    Case 2400
        MVelTNC2400.Checked = True
    Case 9600
        MVelTNC9600.Checked = True
    Case 19200
        MVelTNC19200.Checked = True
    Case Else
        msg = "Atención: COM.CFG tiene una velocidad TNC no valida" + CRLF
        msg = msg + "Adoptada VELOCTNC=9600 para cálculos de tiempos" + CRLF
        msg = msg + "DEBERÍA CORREGIR ESE FICHERO PARA REALIZAR BIEN LOS
CÁLCULOS"
        MsgBox msg, 16
        VELOCTNC = 9600
        MVelTNC9600.Checked = True
    End Select



    AbrirPuerto


    'INICIALIZAR VARIABLES LEER PUERTO SERIE
    EstSer = 1
```

```
    tramaEnl = ""
    tramaSer = ""
    rec_trama = False
    numTNACK_P = 0
    numTNACK_R = 0

    'monitor de tiempos desactivado
    MonTie = True
    MMonTiempo.Checked = True

    MonitorFIDS = False 'al inicio las tramas se muestran por FCRP


    'Dialogo inicial con la TNC
    DeterminarModoTNC
    NombreEstacion
    If ModoTNC = NORMAL Then    'pongo TNC en Modo HOST
        BModoHost_Click
    End If

    DetEstadoAplic DESCONECTADO
    DetModoAplic NORMALREPOSO

    Text1.Text = ""
    Text2.Text = ""

    EstEntrante = 0 'ninguna estacion pidió un REQUEST

FCRP.Enabled = True
FCRP.Text1.SetFocus


    'Arrancar timer de prox. sondeo
    tsondeo = 3600      'tiempo entre sondeos = 60 min.
    EstableceProxSondeo

    BInicio.Enabled = True
    BActuar.Enabled = True

    timer4.Enabled = True

    'no poner ninguna línea de código aquí: no se ejecutará !!!
    Exit Sub

RE_FLoad:
    If Err = 68 Then 'Device unavalaible
        MsgBox "CRP no pudo abrir el puerto serie" + CRLF + CRLF + "FIN DE LA
APLICACION", 16, "CRP"
        End
    End If
    Resume Next
End Sub

Sub MCentral_Click ()
Dim cadena As String
' Configura el TNC para operar como Estacion Central
```

```
    If estadoAplic = CONECTADO Then BDisconnect_Click

SondeosEnProceso = True 'NO atiendo a mas REQUEST hasta acabar CONFIG

    If ModoTNC = HOST Then
        BSalirHost_Click 'salir de modo Host
        tempoms (500)  'esperar 0,5 seg para no perder envio 1° comando
    End If
MonitorRX = True
    Open dir_app + "hostc.cfg" For Input As #1
    While Not EOF(1)
        Line Input #1, cadena
        If Mid$(cadena, 1, 1) <> "#" Then
            Comm1.Output = cadena + Chr$(13)
            AddTexto Text2, cadena + CRLF
            tempoms (500)
        End If
    Wend
    Close #1
    AddTexto Text2, CRLF + "    *** TNC CONFIGURADA COMO CENTRAL" + CRLF
    ModoTNC = HOST
    ActualizarLModoTNC HOST
    NombreEstacion
    DetEstadoAplic DESCONECTADO 'tras un RESET se pierden todas las conexiones
    DetModoAplic NORMALREPOSO
MonitorRX = False
SondeosEnProceso = False    'ATIENDO A LOS REQUEST
End Sub

Sub MEstAct_Click ()
' Permite al usuario cambiar la lista de estaciones activas

    FEstAct.Show 1
    AddTexto Text2, "Pulse INICIO para actualizar la Lista Estaciones Activas" + CRLF
End Sub

Sub MFecha_Click ()
Dim fecha$, resp%
' Permite al usuario cambiar la fecha del sistema central


    fecha = InputBox("Introducir la nueva fecha" + CRLF + "Formato: mm-dd-aa")
    If fecha <> "" Then
        resp = MsgBox("Se va actualizar la fecha del sistema a " + fecha, 36)
        If resp = 6 Then '6=YES
            Date$ = fecha
            LFecha = Format(Now, "dd-mm-yy")
        End If
    End If
End Sub

Sub MHora_Click ()
Dim hora$, resp%
' Permite al usuario cambiar la hora del sistema central

    hora = InputBox("Introducir la nueva hora" + CRLF + "Formato: hh:mm:ss")
```

```
    If hora <> "" Then
        resp = MsgBox("Se va actualizar la hora del sistema a " + hora, 36)
        If resp = 6 Then Time$ = hora
    End If
End Sub

Sub MMonTiempo_Click ()
' Activa/Desactiva variable interna para mostrar por pantalla el tiempo
' de recepción de los ficheros

    If MonTie = True Then
        MonTie = False
        MMonTiempo.Checked = False
    Else
        MonTie = True
        MMonTiempo.Checked = True
    End If

End Sub

Sub MRemota_Click ()
Dim cadena As String
' Configura el TNC para operar como Estacion Remota

    If estadoAplic = CONECTADO Then BDisconnect_Click
SondeosEnProceso = True 'NO atiendo a mas REQUEST hasta acabar CONFIG

    If ModoTNC = HOST Then
        BSalirHost_Click  'salir de modo Host por si estaba
        tempoms (500)   'esperar 100 mseg para no perder 1º comando
    End If
MonitorRX = True
    Open dir_app + "hostr.cfg" For Input As #1
    While Not EOF(1)
        Line Input #1, cadena
        If Mid$(cadena, 1, 1) <> "#" Then
            Comm1.Output = cadena + Chr$(13)
            AddTexto Text2, cadena + CRLF
            tempoms (500)
        End If
    Wend
    Close #1
    AddTexto Text2, CRLF + "    *** TNC CONFIGURADA COMO REMOTA" + CRLF
    ModoTNC = HOST
    ActualizarLModoTNC HOST
    NombreEstacion
    DetEstadoAplic DESCONECTADO 'tras un RESET se pierden todas las conexiones
    DetModoAplic NORMALREPOSO
MonitorRX = False
SondeosEnProceso = False    'ATIENDO A LOS REQUEST
End Sub

Sub MSalir_Click ()

    If estadoAplic <> DESCONECTADO Then DesconEstaciones

    CerrarPuerto
```

```
    BVolcado_Click
    Close              'cierro todos los ficheros abiertos

    End
    'FIN DE LA APLICACION
End Sub

Sub MTSondeo_Click ()
Dim tsondeoant%, resp$
' Permite al usuario cambiar el intervalo en el que la aplicación solicita
' un sondeo de todas las estaciones activas de forma automática.


    timer4.Enabled = False
    tsondeoant = tsondeo
    resp = InputBox("Introducir el tiempo entre sondeos (en segundos)", "TIEMPO DE
SONDEO", Str$(tsondeo))
    If resp = "" Or Val(resp) = 0 Then
          tsondeo = tsondeoant
    Else
          tsondeo = Val(resp)
    End If

    EstableceProxSondeo
    timer4.Enabled = True
End Sub

Sub MVelTNC1200_Click ()
    Select Case VELOCTNC
    Case 1200
          MVelTNC1200.Checked = False
    Case 2400
          MVelTNC2400.Checked = False
    Case 9600
          MVelTNC9600.Checked = False
    Case 19200
          MVelTNC19200.Checked = False
    End Select

    VELOCTNC = 1200
    MVelTNC1200.Checked = True

End Sub

Sub MVelTNC19200_Click ()
    Select Case VELOCTNC
    Case 1200
          MVelTNC1200.Checked = False
    Case 2400
          MVelTNC2400.Checked = False
    Case 9600
          MVelTNC9600.Checked = False
    Case 19200
          MVelTNC19200.Checked = False
    End Select

    VELOCTNC = 19200
```

```
    MVelTNC19200.Checked = True

End Sub

Sub MVelTNC2400_Click ()
    Select Case VELOCTNC
    Case 1200
        MVelTNC1200.Checked = False
    Case 2400
        MVelTNC2400.Checked = False
    Case 9600
        MVelTNC9600.Checked = False
    Case 19200
        MVelTNC19200.Checked = False
    End Select

    VELOCTNC = 2400
    MVelTNC2400.Checked = True

End Sub

Sub MVelTNC9600_Click ()
    Select Case VELOCTNC
    Case 1200
        MVelTNC1200.Checked = False
    Case 2400
        MVelTNC2400.Checked = False
    Case 9600
        MVelTNC9600.Checked = False
    Case 19200
        MVelTNC19200.Checked = False
    End Select

    VELOCTNC = 9600
    MVelTNC9600.Checked = True

End Sub

Sub MVerConfig_Click ()
Dim cadena As String, cambiamodo As Integer
' Solicita al TNC el valor de todos sus parámetros y lo almacena en el
' fichero DE-CONF.RES

    If estadoAplic = CONECTADO Then BDisconnect_Click
SondeosEnProceso = True 'NO atiendo a mas REQUEST hasta acabar CONFIG

    cambiamodo = False
    If ModoTNC = NORMAL Then
        BModoHost_Click
        cambiamodo = True
    End If
    Open dir_app + "de-CONF.cfg" For Input As #1
    Open dir_app + "de-CONF.res" For Output As #2
    Write #2, "CONFIGURACION DE LA TNC  -  " + Format$(Now, "dd-mm-yy  hh:mm:ss") +
CRLF + CRLF
    While Not EOF(1)
        Input #1, cadena
```

```
        If Mid$(cadena, 1, 1) <> "#" Then
            EscribirSerie "C2A" + cadena
            'AddTexto Text2, cadena + CRLF
            tempoms (500)
            Write #2, cadena, tramaEnl
        End If
    Wend
    Close #1, #2
    AddTexto Text2, "   *** VER FICHERO '.\FILES\DE-CONF.RES' " + CRLF
    Text1.Text = ""
    Text1.SetFocus
    If cambiamodo Then
        BSalirHost_Click
    End If
SondeosEnProceso = False    'ATIENDO A LOS REQUEST
End Sub

Sub NombreEstacion ()
Dim lonindic%, poscr As Integer
' Determina el nombre del nodo de la subred de comunicación al cual esta
' conectado este sistema

    NomEstac = ""
    tramaEnl = ""

    If ModoTNC = HOST Then
        EscribirSerie "C2AMY"
        tempoms (500)
        If tramaEnl <> "" Then
            NomEstac = Mid$(tramaEnl, 11, Len(tramaEnl) - 11) ' me quedo solo con los
indicativos
            lonindic = InStr(NomEstac, "/") - 2
            NomEstac = Mid$(NomEstac, 1, lonindic)
            LEstacion.Caption = "ESTACION: " + NomEstac
        Else
            LEstacion.Caption = "ESTACION: NULL"
        End If
    Else
        EscribirSerie "MY" + Chr$(13)
        tempoms (500)
        If tramaEnl <> "" Then
            NomEstac = Mid$(tramaEnl, 12) ' me quedo solo con los indicativos
            lonindic = InStr(NomEstac, "/") - 2
            'basta con coger el nombre del port1
            NomEstac = Mid$(NomEstac, 1, lonindic)
            LEstacion.Caption = "ESTACION: " + NomEstac
        Else
            LEstacion.Caption = "ESTACION: NULL"
        End If
    End If
End Sub

Function pot2 (e As Integer) As Integer
Dim result%, i%
' Funcion de calculo interno; devuelve la potencia de 2
    If e = 0 Then
        pot2 = 1
```

```
    Else
        result = 1
        For i = 1 To e
            result = result * 2
        Next i
        pot2 = result
    End If
End Function
```

Function QuitarTySVB (cadena As String, crc As Integer) As String
' Quita la transparencia a una cadena y devuelve la SVB al mismo tiempo

```
Dim i%, longitud As Integer
Dim cadaux$, car As String

    cadaux = ""
    longitud = Len(cadena)
    i = 1
    While i <= longitud
        car = Mid$(cadena, i, 1)
        Select Case car
            Case Chr$(FESC)
                i = i + 1
                Select Case Mid$(cadena, i, 1)
                    Case Chr$(TFEND)
                        cadaux = cadaux + Chr$(FEND)
                        crc = 255 And (crc + FEND)
                    Case Chr$(TFESC)
                        cadaux = cadaux + Chr$(FESC)
                        crc = 255 And (crc + FESC)
                End Select
            Case Else
                cadaux = cadaux + car
                crc = 255 And (crc + Asc(car))
        End Select
        i = i + 1
    Wend
    QuitarTySVB = cadaux

End Function
```

Sub RecibirCom ()
' Lee el puerto serie con el TNC en Modo NORMAL
' Se utiliza para dialogar con el TNC

```
Dim BufferEnt$

    BufferEnt = ""
    While Comm1.InBufferCount > 0
        BufferEnt = BufferEnt + FCRP.Comm1.Input
    Wend
    tramaEnl = tramaEnl + BufferEnt

    If MonitorRX = True Then
        AddTexto Text2, BufferEnt
    End If
End Sub
```

```
Function RecibirFichero (NEst As Integer, LonFichero As Long, LonFichRec As Long,
NumEnvios As Integer) As Integer
' Sondea a una estacion remota (NEst enviado por RecibirTodosFicheros) para
' recuperar la parte de fichero contenida en la memoria del SIR (NEst)

Dim nomfich$
Static trama4$, fichero$
Static svbb As Integer
Dim crcrec%
Static tiempo  As Integer
Static ntpr%, ntrech%, ntacep As Integer
Static t1&, t2 As Long
Static disc As Integer
Static NFallosRec As Integer
Dim nfs$, NCANAL$, i%
Dim LonMem As Long
Dim ntr As Integer
Dim Byte8%, Byte7%, Byte6%, Byte5%
Dim bps As Single, product As Single
Dim veloc%, Params$
Static MEDIO_RECIBIRFICH  As Integer

On Error GoTo RutErr


'-----------------
' PROCESO DE SONDEO
'-----------------------------------------------------------------
   DetEstadoAplic SONDEANDO

   nfs = Format(Format(NumFichSondeo, "000"), "@@@")
   NCANAL = Chr$(64 + NEst)
   nomfich = dir_eve + "EV-" + NCANAL + "." + nfs


   If NumEnvios = 0 Then       'en el primer grupo de tramas
         tiempo = 0
         ' Creo el fichero por si no existe
         Open nomfich For Append As #1
         Close
         ' Borro el fichero para quitar la basura si existía
         Kill nomfich
         'dejo el fichero creado
         Open nomfich For Append As #1
         Close
   End If


SONDEO1:
   ntr = 0        ' num tramas recibidas
   ntpr = 0       ' num tramas parciales recibidas (entre 2 crc's)
   ntrech = 0     ' num tramas rechazadas
   ntacep = 0     ' num tramas aceptadas
   trama4 = ""
   fichero = ""
```

```
   svbb = 0        ' crc que voy calculando
   disc = False
   NFallosRec = 0


   EscribirSerie "D2" + Chr$(64 + NEst) + Chr$(SONDEO) + Chr$(SONDEO)
   AddTexto Text2, "D2" + Chr$(64 + NEst) + "33" + CRLF

   tramarec = RecibirHost(5)

   If Mid$(tramarec, 4, 1) = Chr$(WAIT) Or Mid$(tramarec, 4, 1) = Chr$(R_MDN) Then
         GoTo REC_NO_DAT
   Else
         If tramarec = "" Then
            GoTo REC_NO_DAT
         End If
   End If

'-------------------------
'INICIO DE RECIBIR FICHERO
'-------------------------

   If Mid$(tramarec, 4, 1) = Chr$(NO_DAT) Then
REC_NO_DAT:
         If NumEnvios <> 0 Then
            RecibirFichero = 8  'estoy a medias de un fichero
         Else
            RecibirFichero = 4
         End If

         GoTo SalirRec
   Else
         If Mid$(tramarec, 4, 1) = Chr$(DAT_AV) Then
            AddTexto Text2, "AVISO DE ESTACION Num. 1" + CRLF
            AddTexto Text2, CRLF + "    *** DATOS DE AVISO: " + Mid$(tramarec, 5) + CRLF
            EscribirSerie "D2" + Chr$(64 + NEst) + Chr$(ACK_AV) + Chr$(ACK_AV)
            tramarec = RecibirHost(5)

            RecibirFichero = 5
         Else

         MEDIO_RECIBIRFICH = False
         DetEstadoAplic RECIBIENDO

'-----------------------------------------
'INICIO BUCLE RECEPCIÓN
'-----------------------------------------
         While MEDIO_RECIBIRFICH = False
            If Mid$(tramarec, 8, 4) = "DISC" Then
                  DetEstadoAplic DESCONECTADO
                  t2 = Timer
                  MEDIO_RECIBIRFICH = True
                  disc = True

                  Close    'cierro todos los ficheros
                  tiempo = t2 - t1
                  AddTexto Text2, "   *** [RecFich] recibida PETICION de desconexión" + crlf
```

```
                If MonTie Then AddTexto Text2, "Tiempo: " + Str$(tiempo) + " seg" + "
Recibidas: " + Str$(ntr) + CRLF
            Else
                Select Case Mid$(tramarec, 4, 1)
                Case Chr$(ERROR_F)
                  t2 = Timer
                  AddTexto Text2, "error en tarjeta. imposible acceso a s_ram" + crlf
                  AddTexto Text2, "   *** DATOS DE TRAMA E_F:"
                  For i = 5 To Len(tramarec)
                        AddTexto Text2, Hex(Asc(Mid$(tramarec, i, 1))) + "*"
                  Next i
                  AddTexto Text2, " " + CRLF


                  ' salir del bucle y no guardar el trozo de fichero recibido
                  MEDIO_RECIBIRFICH = True
                  Close
                  tiempo = t2 - t1
                  If MonTie Then AddTexto Text2, "Tiempo: " + Str$(tiempo) + " seg" + "
Recibidas: " + Str$(ntr) + CRLF
                  RecibirFichero = 6

                  Open nomfich For Append As #1
                  Close
                  Kill nomfich
                  AddTexto Text2, "   *** FICHERO " + nomfich + " BORRADO" + CRLF
                  DesconectarDe NEst      'desconecto la estacion

                Case Chr$(DAT_FI)
                  NFallosRec = 0
                  ntr = ntr + 1
                  ntpr = ntpr + 1
                  tramarec = QuitarTySVB(Mid$(tramarec, 5), svbb)
                              ' calcula SVB con LM y LF incluidos
                              ' devuelve trama sin "D2Aj"
                  If NumEnvios = 0 Then   'si es el primer bloque de tramas

                        Byte8 = Asc(Mid$(tramarec, 4, 1))
                        Byte7 = Asc(Mid$(tramarec, 3, 1))
                        LonFichero = Byte8 * 256 + Byte7

                        AddTexto Text2, "   *** RECIBIDO: LF= " + Str$(LonFichero) + "    "
                  End If
                  If ntr = 1 Then
                        t1 = Timer      'control tiempos solo fichero
                        NumEnvios = NumEnvios + 1
                        Byte6 = Asc(Mid$(tramarec, 2, 1))
                        Byte5 = Asc(Mid$(tramarec, 1, 1))
                        LonMem = Byte6 * 256 + Byte5
                        LonFichRec = LonFichRec + LonMem
                        AddTexto Text2, "   *** LM= " + Str$(LonMem) + CRLF

                        tramarec = Mid$(tramarec, 3)
                  End If

                  trama4 = trama4 + tramarec
```

```
            Case Chr$(SVB)
              NFallosRec = 0
              tramarec = QuitarT(tramarec)

              crcrec = Asc(Mid$(tramarec, 5, 1))

              If svbb <> crcrec Then
                    EscribirSerie "D2" + Chr$(64 + NEst) + Chr$(NACK_FI) +
Chr$(NACK_FI)
                    ntrech = ntrech + ntpr
                    If ntr <= 4 Then
                    ' para que vuelva a calcular y sacar lm - If por pantalla
                      NumEnvios = NumEnvios - 1
                      ntr = 0
                    End If
              Else
                    EscribirSerie "D2" + Chr$(64 + NEst) + Chr$(ACK_FI) +
Chr$(ACK_FI)
                    fichero = fichero + trama4
                    ntacep = ntacep + ntpr
              End If
              trama4 = ""
              svbb = 0
              ntpr = 0


            Case Chr$(C_MEDIO)

              NFallosRec = 0
              Open nomfich For Binary As #1
              Seek #1, LOF(1) + 1   'añadir al final del fichero
              Put #1, , fichero
              Close

              ' SI NO RECIBIDO TODO EL FICHERO
              If LonFichRec < LonFichero Then
                    RecibirFichero = 2  'RECIBIDO FICHERO A MEDIAS

                    t2 = Timer        'CONTROL TIEMPOS SOLO FICHERO
                    tiempo = tiempo + (t2 - t1)

              Else
                    t2 = Timer
                    tiempo = tiempo + (t2 - t1)

                    RecibirFichero = 1      'RECIBIDO FICHERO COMPLETO
                    If LonFichRec > LonFichero Then
                      AddTexto Text2, "RECHAZANDO FICHERO..." + CRLF
                      EscribirSerie "D2" + Chr$(64 + NEst) + Chr$(REP_FI) +
Chr$(REP_FI)
                      tramarec = RecibirHost(5)
                      If Mid$(tramarec, 4, 1) <> Chr$(C_MEDIO) Then
                            AddTexto Text2, "   *** RECIBIDA TRAMA NO C_MEDIO" +
CRLF
                      End If
                      AddTexto Text2, "   *** FICHERO " + nomfich + " RECHAZADO" +
CRLF + CRLF
```

```
                        AnotarError "Fichero: " + nomfich + "RECHAZADO",
"RecibirFichero"

                        Open nomfich For Append As #1
                        Close
                        Kill nomfich
                      Else
                        EscribirSerie "D2" + Chr$(64 + NEst) + Chr$(DEL_FI) +
Chr$(DEL_FI)
                        tramarec = RecibirHost(5)
                        If Mid$(tramarec, 4, 1) <> Chr$(C_MEDIO) Then
                             AddTexto Text2, "   *** RECIBIDA TRAMA NO C_MEDIO" +
CRLF
                        End If
                        AddTexto Text2, "   *** FICHERO " + nomfich + " ACEPTADO" +
CRLF + CRLF


                        If tiempo = 0 Then tiempo = 1
                        bps = LonFichero * 8 / tiempo
                        product = bps / VELOCTNC

                        If MonTie Then
                             AddTexto Text2, "VelocTNC:" + Str$(VELOCTNC) + " -
Tiempo:" + Str$(tiempo) + " seg" + " - Bytes:" + Str$(LonFichero) + CRLF
                             AddTexto Text2, "NTR:" + Str$(ntr) + " - NTACEP:" +
Str$(ntacep) + " - NTRECH:" + Str$(ntrech) + CRLF
                             AddTexto Text2, "BPS: " + Format$(bps, "0.00") + " -
Productividad: " + Format$(product, "0.00%") + CRLF
                        End If

                        Open dir_err + "tiempos.log" For Append As #2
                        Write #2, String$(70, "-")
                        Write #2, Format$(Now, "dd-mm-yy  hh:mm:ss")
                        Write #2, "FicheroRX: " + nomfich + "  Long: " + Str$(LonFichero)
+ " bytes"
                        Write #2, "VelocTNC: " + Str$(VELOCTNC) + "  Tiempo=" +
Str$(tiempo)
                        Write #2, "NTR=" + Str$(ntr) + "  NTACEP=" + Str$(ntacep) + "
NTRECH=" + Str$(ntrech)
                        Write #2, "BPS=" + Format$(bps, "0.00") + "  Productividad=" +
Format$(product, "0.00%")
                        Close #2



                      End If
                  End If

                  MEDIO_RECIBIRFICH = True

                Case Else
TRAMA_INESPERADA:     'recibida otra trama o ninguna tras 5 seg.
                  AnotarEnDiag "[RecFich] " + THE, tramarec'Trama Host Enviada -
TramaRec
                  NFallosRec = NFallosRec + 1
                  If NFallosRec > 3 Then
```

```
                        AddTexto Text2, "ESTACION " + Str$(NEst) + " DESCONECTADA
DURANTE RECEPCIÓN DE FICHERO" + CRLF
                        DesconectarDe NEst
                        NEA = NEA - 1
                        AddTexto Text2, CRLF + "    *** FICHERO " + nomfich + "
BORRADO" + CRLF
                        Open nomfich For Append As #1
                        Close
                        Kill nomfich
                        RecibirFichero = 1
                        GoTo SalirRec
                    Else
                        EscribirSerie "D2" + Chr$(64 + NEst) + Chr$(NACK_FI) +
Chr$(NACK_FI)
                    End If
                End Select
            End If

'RECEPCIÓN DE SIGUIENTE TRAMA
            If MEDIO_RECIBIRFICH = False Then
                tramarec = RecibirHost(5)
            End If

        Wend
'----------------------------
'FIN BUCLE RECEPCIÓN FICHERO
'----------------------------
        If disc = True Then
            AddTexto Text2, "    *** FICHERO " + nomfich + " BORRADO" + CRLF

            Open nomfich For Append As #1
            Close
            Kill nomfich
            RecibirFichero = 7      'RECIBIDA SOLICITUD DE DESCONEXION
        End If

    End If
    End If

SalirRec:
    Close       ' cierro todos los ficheros por si he dejado alguno abierto

    Exit Function


RutErr:
    Open dir_err + "recibir.err" For Append As #2
    Write #2, String(70, "-") + CRLF + Format$(Now, "dd-mm-yy hh:mm:ss")
    Write #2, "trama recibida", tramarec
    Write #2, "Error producido: " + Error$
    Write #2, "NOM_FICH: " + nomfich
    Close #2
    AddTexto Text2, "Error: " + Error$ + " - Ver RECIBIR.ERR"
    AnotarError Error$ + " - Ver RECIBIR.ERR", "RecibirFichero"
    Resume Next

End Function
```

```
Sub RecibirFicheroDiag ()
 'RECIBE UN FICHERO DE DIAGNOSTICO
Dim nomfich$
Static fichero$
Static ntr As Integer
Static disc As Integer
Dim NFD$, NCANAL$, fecha$
Dim i%, j%
Static MEDIO_RECIBIRFICHfd  As Integer

On Error GoTo RE_RecibirFD

   fichero = ""
   ntr = 0
   disc = False

         timer1.Enabled = False
         MEDIO_RECIBIRFICHfd = False
         DetEstadoAplic RECIBIENDO

         While MEDIO_RECIBIRFICHfd = False
            If Mid$(tramarec, 8, 4) = "DISC" Then
                  DetEstadoAplic DESCONECTADO
                  MEDIO_RECIBIRFICHfd = True
                  disc = True
                  Close     'cierro todos los ficheros
                  AddTexto Text2, "   *** [RecibirFD] recibida petición de desconexión" + crlf
            Else

                  Select Case Mid$(tramarec, 4, 1)
                  Case Chr$(R_FDIAG)
                     ntr = ntr + 1
                     If ntr = 1 Then
                           tramarec = Mid$(tramarec, 5)
                     Else    'quito cabeceras de datos host a las tramas
                           tramarec = Mid$(tramarec, 5)
                     End If
                     tramarec = QuitarT(tramarec)
                     fichero = fichero + tramarec

                  Case Chr$(SVB)
                     EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(ACK_FI) +
Chr$(ACK_FI)

                  Case Chr$(C_MEDIO)
                     fecha = Format$(Now, "dd-mm-yy hh:mm:ss") + CRLF
                     NFD = Format(Format(NumFDiag, "000"), "@@@")
                     NCANAL = Mid$(tramarec, 3, 1)
                     nomfich = dir_dia + "IDSDIAG" + NCANAL + "." + NFD
                     ' Creo el fichero por si no existe
                     Open nomfich For Binary As #1
                     Close
                     ' Borro el fichero
                     Kill nomfich

                     ' Escribo el nuevo contenido
```

```
                Open nomfich For Binary As #1
                Put #1, , fecha
                Put #1, , fichero
                Close

                MEDIO_RECIBIRFICHfd = True
            '   AddTexto Text2, "RECIBIDO C_MEDIO " + CRLF
            End Select
        End If

        If MEDIO_RECIBIRFICHfd = False Then tramarec = RecibirHost(5)

    Wend
    If disc = False Then
        DetEstadoAplic ACTUANDO
        AddTexto Text2, "FICHERO " + nomfich + " RECIBIDO " + CRLF
    End If


  fichero = Mid$(fichero, 3)  'los dos primeros son LM y ya vienen datos DIAG

  AddTexto Text2, "   *** LONFICH= " + Str$(Len(fichero)) + "  Recibido Msg " + CRLF

                For i = 0 To (Int(Len(fichero) / 5)) - 1
                    AddTexto Text2, "   *** " + Str$(i) + ": "
                    For j = 1 To 4
                        AddTexto Text2, ConvBCDenNum(Asc(Mid$(fichero, i * 5 + j, 1)))
+ "-"

                    Next j
                    'el 5º car ya no es BCD
                    AddTexto Text2, Str$(Asc(Mid$(fichero, i * 5 + 5, 1)))
                    AddTexto Text2, " " + CRLF
                Next i
SalirSonFD:
  Close      ' cierro todos los ficheros por si he dejado alguno abierto
  timer1.Enabled = True
  Exit Sub


RE_RecibirFD:
  Open dir_err + "recibifd.err" For Append As #2
  Write #2, String(70, "-") + CRLF + Format$(Now, "dd-mm-yy hh:mm:ss")
  Write #2, "trama recibida", tramarec
  Write #2, "Error producido: " + Error$
  Write #2, "Nom_Fich: " + nomfich
  Close #2
  AddTexto Text2, "Error: " + Error$ + " - Ver RECIBIFD.ERR"
  AnotarError Error$ + " - Ver RECIBIFD.ERR", "RecibirFicheroDiag"
  Resume Next
End Sub
```

Sub RecibirFicheroIDS (TRAMA1 As String)

```
'recibe el fichero isurec.isu de ids
'compara isurec.isu con isumodxxx.isu y avisa si son distintos o no
Static fichero$, disc%
```

```
Static MEDIO_RECIBIRFICHfd  As Integer

On Error GoTo RE_RecibirIDS

   fichero = ""
   disc = False
   fichero = QuitarT(Mid$(TRAMA1, 7))  'OJO! LLEGA LONMEM EN LA 1º TRAMA


         timer1.Enabled = False
         MEDIO_RECIBIRFICHfd = False
         DetEstadoAplic RECIBIENDO

         tramarec = RecibirHost(15)

         While MEDIO_RECIBIRFICHfd = False
            If Mid$(tramarec, 8, 4) = "DISC" Then
                  DetEstadoAplic DESCONECTADO
                  MEDIO_RECIBIRFICHfd = True
                  disc = True
                  Close     'cierro todos los ficheros
                  AddTexto Text2, "   *** [RecibirFD] recibida petición de desconexión" + crlf
            Else

                  Select Case Mid$(tramarec, 4, 1)
                  Case Chr$(R_CIDS)
                     tramarec = Mid$(tramarec, 5)
                     tramarec = QuitarT(tramarec)
                     fichero = fichero + tramarec
                  Case Chr$(DAT_PR)
                     AddTexto Text2, "   *** [RecFichIDS] DATOS PARAM: " + Mid$(tramarec,
5) + CRLF

                     fichrx = dir_ids + "ISUREC.ISU"
                     Open fichrx For Binary As #1
                     Close
                     ' Borro el fichero
                     Kill fichrx
                     GoTo SalirRECIDS
                  Case Chr$(SVB)
                     EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(ACK_FI) +
Chr$(ACK_FI)
                  Case Chr$(C_MEDIO)
                     MEDIO_RECIBIRFICHfd = True

                     ' Creo el fichero por si no existe
                     fichrx = dir_ids + "ISUREC.ISU"
                     Open fichrx For Binary As #1
                     Close
                     ' Borro el fichero
                     Kill fichrx

                     ' Escribo el nuevo contenido
                     Open fichrx For Binary As #1
                     Put #1, , fichero
                     Close
                  Case Else
```

```
                    AddTexto Text2, "  *** [RecFichIDS] TRAMA NO ESPERADA: " +
Mid$(tramarec, 5) + CRLF
                End Select
            End If

            If MEDIO_RECIBIRFICHfd = False Then tramarec = RecibirHost(15)

        Wend
        If disc = False Then
            DetEstadoAplic ACTUANDO
            AddTexto Text2, "FICHERO ISUREC.ISU RECIBIDO" + CRLF
        End If


'************ COMPARAR FICHEROS ********************************
Dim ncl&, lon1&, lon2 As Long
Dim car1 As String, car2 As String
Dim distintos%, asc1%, asc2 As Integer
Dim vr%

    Open fichtx For Binary As #1
    Open fichrx For Binary As #2
    lon1 = LOF(1)
    lon2 = LOF(2)
    ncl = 0
    distintos = False
    AddTexto Text2, CRLF + "COMPARANDO FICHEROS ENVIADO-RECIBIDO" + CRLF
    car1 = Input$(1, #1)
    car2 = Input$(1, #2)
    ncl = 1
    Do
        vr = DoEvents()
        If car1 <> car2 Then
            distintos = True
            AddTexto Text2, "Pos: " + Str$(ncl)
            AddTexto Text2, "  car1(tx): " + Str$(Asc(car1)) + "  car2(rx): " + Str$(Asc(car2)) +
CRLF
        End If
        car1 = Input$(1, #1)
        car2 = Input$(1, #2)
        ncl = ncl + 1
    Loop While (ncl <= lon1) And (ncl <= lon2)
    If distintos = False Then AddTexto Text2, "LOS FICHEROS SON IDÉNTICOS" + CRLF
    Close


SalirRECIDS:
    Close      ' cierro todos los ficheros por si he dejado alguno abierto
    timer1.Enabled = True
    Exit Sub


RE_RecibirIDS:
    Open dir_err + "recibids.err" For Append As #2
    Write #2, String(70, "-") + CRLF + Format$(Now, "dd-mm-yy hh:mm:ss")', "nte: " +
Str$(nte)
    Write #2, "trama recibida", tramarec
```

```
    Write #2, "Error producido: " + Error$
    Write #2, "NOM_FICH: " + fichrx
    Close #2
    AddTexto Text2, "Error: " + Error$ + " - Ver RECIBids.ERR"
    AnotarError Error$ + " - Ver RECIBids.ERR", "RecibirFicheroIDS"

    Resume Next
End Sub

Sub RecibirTodosFicheros (NEst As Integer)
' Coordina la recepción de ficheros de datos de todas las estaciones,
' empezando por la primera que solicita una sesion
' Utiliza la funcion RecibirFichero para recibir cada trozo de fichero de
' cada una de las estaciones.

Dim i%, seguir%, NENV%, todasND%
Static Rto(TotalEst) As Integer        'array de resultados para cada estacion
Static NumEnvios(TotalEst) As Integer  'array del num envíos de cada estacion
Static LF(TotalEst), LFR(TotalEst) As Integer
Dim LonFichRec As Long
Dim LonFich As Long
Static NumNoDat As Integer
Static NumComproNDAT As Integer
Dim llegoDISC As Integer
Dim NumEstSondeadas As Integer
Dim nfs%, NCANAL As String, nomfich$


Static NumEstsEnSondeo As Integer


    ' iniciamos con NEA = NumEstCon que devolvió ConectarEstaciones
    NEA = NumEstCon

    NumNoDat = 0
    NumComproNDAT = 0
    NumEstsEnSondeo = 0
    NumEstSondeadas = 0

    For i = 1 To TotalEst
        NumEnvios(i) = 0
        LF(i) = 0
        LFR(i) = 0
    Next i

    seguir = True

INICIO_RONDA_SONDEOS:
    While (seguir And NEA > 0)

        NumEstsEnSondeo = NumEstsEnSondeo + 1

        If NEst > TotalEst Then NEst = 1

        If EstAct(NEst) = 1 Or EstAct(NEst) = 2 Then
            i = NEst
```

```
NENV = NumEnvios(i)
LonFich = LF(i)
LonFichRec = LFR(i)

Rto(i) = RecibirFichero(i, LonFich, LonFichRec, NENV)

Select Case Rto(i)
'Fichero Completo
Case 1
      EstAct(i) = 1
      NumEnvios(i) = 0 'Preparar para nueva ronda y nuevo fichero
      LF(i) = 0
      LFR(i) = 0

      ' preparar el NFS para siguiente ronda de sondeos
      ' a todas las estaciones
      NumFichSondeo = (NumFichSondeo + 1) Mod 100
      Open dir_app + "NFS.CFG" For Output As #1
      Write #1, Str$(NumFichSondeo)
      Close #1

      NumNoDat = 0       'INICIO CONTÉO DE NUM_NO_DATOS


'Fichero a medias
Case 2
      NumEnvios(i) = NENV
      'Guardar el num envíos recibidos por esa estacion
      LF(i) = LonFich
      LFR(i) = LonFichRec
      EstAct(i) = 1

      NumNoDat = 0       'INICIO CONTÉO DE NUM_NO_DATOS

'NO contesta a 3 sondeos
Case 3
      ' ********CASO ELIMINADO********

' NO DATOS
Case 4
      EstAct(i) = 2

      If NEA = 1 Then     'si hay una estacion activa
         NumNoDat = NumNoDat + 1
         'si hay 1 NEA entonces aumentar NumNoDat
         If NumNoDat > 5 Then
               EstAct(i) = 0   'dar de baja de EstAct
               AddTexto Text2, "   *** ESTACION DESCONECTADA TRAS 5
NO_DAT" + CRLF
               DesconectarDe i
               NEA = NEA - 1
         Else
               tempoms (2000)
         End If
      End If
```

```
        'DATOS AVISO
        Case 5
            EstAct(i) = 1

        'ERROR FICHERO
        Case 6
            EstAct(i) = 0  'dar de baja de EstAct
            NEA = NEA - 1

        ' llego un DISC a medias de la tramas
        Case 7
            EstAct(i) = 0   'dar de baja de EstAct
            NEA = NEA - 1
            llegoDISC = True

        ' llego un NO_DAT a medias UN FICHERO
        Case 8
            EstAct(i) = 2

            If NEA = 1 Then NumNoDat = NumNoDat + 1    'num tramas NO_DAT a
medias de un fich

            If NEA = 1 Then
              If NumNoDat > 6 Then
                  EstAct(i) = 0   'dar de baja de EstAct
                  AddTexto Text2, "   *** ESTACION DESCONECTADA POR
RECIBIR MAS DE 6 NO_DAT A MEDIAS DE UN FICHERO" + CRLF
                  DesconectarDe i
                  NEA = NEA - 1

                  nfs = Format(Format(NumFichSondeo, "000"), "@@@")
                  NCANAL = Chr$(64 + i)
                  nomfich = App.Path + "\FILES\EV-" + NCANAL + "." + nfs
                  AddTexto Text2, "   *** FICHERO " + nomfich + " BORRADO" +
CRLF
                  Kill nomfich
              Else
                  tempoms (2000)
              End If
            End If

        End Select

    End If 'If EstAct(NEst) = 1 Or EstAct(NEst) = 2 Then


    NEst = NEst + 1 'Num de siguiente estacion
    If NumEstsEnSondeo < TotalEst Then
      GoTo INICIO_RONDA_SONDEOS
    Else   'cuando haya pasado por todas las estaciones
      NumEstsEnSondeo = 0
    End If


If NEA > 1 Then

    For i = 1 To TotalEst
```

```
            'si todas las activas respondieron NO_DATOS==> Fin del SONDEO
            'seguir = (EstAct(i) = 2 Or EstAct(i) = 0)
            If EstAct(i) = 1 Then
                seguir = True
                NumComproNDAT = 0
                'num de veces que compruebo NO-DAT a medias de fich

                Exit For
            Else
                seguir = False
            End If
        Next i

        If Not seguir And NumComproNDAT < 3 Then
            NumComproNDAT = NumComproNDAT + 1
            seguir = True
        End If

        If seguir And NumComproNDAT >= 1 Then
            AddTexto Text2, "   *** ESPERANDO 2 SEG ANTES DE SIG. RONDA
SONDEOS" + CRLF + CRLF
            tempoms (2000)
            GoTo INICIO_RONDA_SONDEOS
        End If

    Else
     GoTo INICIO_RONDA_SONDEOS
    End If

    Wend

SalirRecTodos:
    DesconEstaciones
    AddTexto Text2, "FIN RECIBIR TODOS FICHEROS" + CRLF + CRLF

    ActualizarListaEstAct

    timer1.Enabled = True

End Sub

Sub tempoms (N As Integer)
' Temporizador de N milisegundos
' Permite habilitar un temporizador de espera de la aplicación

    timer2.Interval = N
    Timer2Vencido = False
    timer2.Enabled = True
    While Timer2Vencido = False
        DoEvents
    Wend
    timer2.Enabled = False
End Sub

Sub Text1_Click ()
        Text1.Text = ""
End Sub
```

```
Sub Text1_KeyPress (KeyAscii As Integer)
Static buftext$
Dim comando$, posb%
' Envía por el puerto serie el texto que el usuario escriba en la caja Text1


' no permitir que usuario introduzca comandos de tnc: MYCALL

  If KeyAscii = 13 Then
      If ModoTNC = HOST Then
        If estadoAplic = ACTUANDO Then 'en estado CONECTADO envio datos
            Select Case ModoAplic
            Case TRANSPARENTE
              If Mid$(buftext, 1, 1) = "@" Then
                  EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) +
buftext + CR
              Else
                  EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) +
buftext
              End If
            Case PARAMETRIZAR
              'EscribirSerie "D2A" + Chr$(DAT_PP) + Chr$(&H55) + buftext

            Case Else
' ver donde enviar los datos
' el usuario que escriba también D2<CANAL> y evito sacar pop-up,etc.
              EscribirSerie "D2" + Mid$(THE, 3, 1) + buftext  'envio a la ultima trama
enviada
            End Select
        Else   ' en el resto de estados envio comandos a la TNC
            EscribirSerie "C2A" + buftext
        End If
      Else
            EscribirSerie buftext + Chr$(13)
      End If
      ' averiguar si llamo a MYCALL
      Select Case Mid$(UCase(buftext), 1, 3)
      Case "MYC", "MY "
        'si estoy cambiando el nombre de la estacion
        If Len(buftext) > 3 Then NombreEstacion
      End Select

      Text1.Text = ""
      buftext = ""
  Else
      If KeyAscii = 8 Then
        If Len(buftext) >= 1 Then
            buftext = Mid$(buftext, 1, Len(buftext) - 1)
        End If
      Else
        buftext = buftext + Chr$(KeyAscii)
      End If
  End If


End Sub
```

```
Sub Text2_DblClick ()
Dim RESPUESTA%

   RESPUESTA = MsgBox("Esta seguro de borrar el contenido de esta ventana" + CRLF,
49)
   If RESPUESTA = 1 Then Text2.Text = " "
End Sub

Sub Text3_DblClick ()

   FEstAct.Show 1
   AddTexto Text2, "Pulse INICIO para actualizar la Lista Estaciones Activas" + CRLF
End Sub

Sub Timer1_Timer ()
' Actualiza la etiqueta 'Hora' cada segundo

   LHora.Caption = Time$
End Sub

Sub Timer2_Timer ()
' Para uso interno con la funcion Tempoms

   Timer2Vencido = True
End Sub

Sub Timer4_Timer ()
Dim hora As String
' Al vencer este timer se realizan los Sondeos a todas las estaciones


On Error GoTo RE_Timer4

   hora$ = Time$
   If Hour(ProxSondeo) = 0 And Hour(hora) = 23 Then
   ' las 0 horas son criticas con Time$ = 23h.
   ' pues entrara en un bucle
         If (ProxSondeo = hora$) Then
            If Not SondeosEnProceso Then BInicio_Click
         End If
   Else   'el resto del día se hace sondeo con <=
         If (ProxSondeo <= hora$) Then
            If Not SondeosEnProceso Then BInicio_Click
         End If
   End If
Salir_T4:
   Exit Sub

RE_Timer4:
   If Err = 13 Then 'Tipe mismatch
         Beep
         AddTexto Text2, "!!! Error 13 en tick de Timer4 en" + Date$ + hora$ + CRLF
   End If
   Resume Salir_T4
End Sub
```

FESTACT.FRM

```
' Formulario para introducir la lista de estaciones con las que el usuario
' desea establecer sesiones

Sub BAceptar_Click ()
Dim i%

   For i = 0 To LBEstAct.ListCount - 1
         If LBEstAct.Selected(i) Then
            LEU(i + 1) = 1 'pongo a true el array LEU
         End If
   Next i

   EstActivasElegidas = True
   FEstAct.Hide
End Sub

Sub BCancelar_Click ()
   FEstAct.Hide
End Sub

Sub Form_Activate ()
Dim i%
   For i = 1 To TotalEst
         LEU(i) = 0  'inicializo a False la Lista Estaciones User
   Next i

   t = Timer
   Do While ((Timer - t) < 10) And EstActivasElegidas = False
         vr = DoEvents()
   Loop

   If EstActivasElegidas = False Then
         FEstAct.Hide    'se acabo el tiempo y el usuario no eligió nada
   End If


End Sub

Sub Form_Load ()
Dim i%


   For i = 1 To TotalEst
         LBEstAct.AddItem Str$(i)
         LEU(i) = 0  'inicializar a False la Lista Estaciones User
   Next i
   LBEstAct.Selected(0) = True



End Sub
```

FESTESP.FRM

```
' Formulario para introducir la estacion con la que el usuario
' desea realizar una acción especifica

Sub BAceptar_Click ()
Dim i%

   For i = 0 To LBEstAct.ListCount - 1
      If LBEstAct.Selected(i) Then
         EstEspecifica = i + 1   'seleccionar la estacion especifica
      End If
   Next i
   FEstEsp.Hide
End Sub

Sub Form_Load ()
Dim i%

   For i = 1 To TotalEst
      LBEstAct.AddItem Str$(i)
   Next i
   LBEstAct.Selected(0) = True
End Sub
```

FIDS.FRM

```
' Formulario para recibir ficheros  bloque a bloque

Option Explicit

Dim n_eventos As Integer
Dim contador As Integer

Sub AddTextoIDS (cad As String)
Dim Nd%

   'Asegurar que el texto existente no es demasiado largo.
   Nd = Len(Text1.Text)
   If Nd >= 16384 Then
     Text1.Text = Mid$(Text1.Text, 4097)
     Nd = Len(Text1.Text)
   End If

   'Apuntar al final de Text1
   Text1.SelStart = Nd

   'Añadir cad a Text1.Text
   Text1.SelText = cad
End Sub

Sub BSalir_Click ()

   FIDS.Hide
   Label1.Caption = "N. EV: ---"
   MonitorFIDS = False
End Sub
```

```
Function Bytes_datos (cadena As String, pos10 As Integer)
Dim a, b, c, d As Integer

   MsgBox Hex(Asc(Mid$(cadena, pos10 + 2, 1))) + "*" + Hex(Asc(Mid$(cadena, pos10 + 1,
1)))

   a = ((Asc(Mid$(cadena, pos10 + 2, 1)) And 240) / 16) * 4096
   b = ((Asc(Mid$(cadena, pos10 + 2, 1)) And 15)) * 256
   c = ((Asc(Mid$(cadena, pos10 + 1, 1)) And 240) / 16) * 16
   d = (Asc(Mid$(cadena, pos10 + 1, 1)) And 15)
   Bytes_datos = a + b + c + d

End Function

Function Calcular_long_total (cadena As String)
Dim a, b, c, d As Integer

   MsgBox Hex(Asc(Mid$(cadena, 5, 1))) + "*" + Hex(Asc(Mid$(cadena, 4, 1)))

   a = ((Asc(Mid$(cadena, 5, 1)) And 240) / 16) * 4096
   b = ((Asc(Mid$(cadena, 5, 1)) And 15)) * 256
   c = ((Asc(Mid$(cadena, 4, 1)) And 240) / 16) * 16
   d = (Asc(Mid$(cadena, 4, 1)) And 15)
   Calcular_long_total = a + b + c + d

End Function

Sub Command1_Click ()

   EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) + "@AT" + Chr$(13)

End Sub

Sub Command2_Click ()

   EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) + "@EH" + Chr$(13)

End Sub

Sub Command3_Click ()
'NUMERO DE EVENTOS
Dim cadena As String

   EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) + "@NE" + Chr$(13)

   tramaRec = RecibirHost(5)

   cadena = ""
   cadena = Mid$(tramaRec, 5)

   If cadena <> "" Then
      If Asc(Mid$(cadena, 1, 1)) = 10 Then
         n_eventos = Asc(Mid$(cadena, 2, 1))
      Else
         n_eventos = Asc(Mid$(cadena, 1, 1))
      End If
   Else
```

```
      n_eventos = 0
   End If

   Label1.Caption = "N. EV: " + Str$(n_eventos)
End Sub

Sub Command4_Click ()
Static ii As Integer
Dim respuesta As Integer


   If n_eventos <> 0 Then
      For ii = 1 To n_eventos
         Open dir_jul + "ev" + LTrim$(Str$(ii)) For Binary As #2
         respuesta = MsgBox("[Datos] Pidiendo Fichero" + CRLF + "CANCELAR para cortar el
proceso", 49)
         If respuesta = 2 Then Exit Sub

         Pedir_Fichero ii
         Close #2
      Next ii
   Else
      MsgBox "No hay datos en IDS"
   End If
End Sub

Sub Command5_Click ()
'LEER
Dim cadena As String
Dim BUFFER As String
Dim j As Integer

   cadena = ""
   BUFFER = ""

   Open dir_jul + "ne.ids" For Binary As #1
   cadena = Input$(LOF(1), #1)
   Close #1
   For j = 1 To Len(cadena)
      AddTextoIDS Hex(Asc(Mid$(cadena, j, 1))) + "*"
   Next j
   AddTextoIDS "" + CRLF

End Sub


Function Es_Correcta (cad As String) As Integer

   If Mid$(cad, 1, 1) = Chr$(1) Then
      Es_Correcta = True
   Else
      Es_Correcta = False
   End If
End Function
```

```
Sub Form_Load ()

   n_eventos = 0
   Label1.Caption = "N. EV: ---"
End Sub

Function FuncSVB (cadena As String, ByVal inicio As Integer, ByVal nbytesdatos As
Integer)
Dim j%, POS As Integer
Dim auxiliar As Long

   FuncSVB = 0
   auxiliar = 0
   POS = 0
   j = Len(cadena)
   For j = 1 To nbytesdatos
      auxiliar = 255 And (auxiliar + Asc(Mid$(cadena, inicio + j, 1)))
   Next j
      POS = (inicio + 1) + nbytesdatos
      MsgBox "El crc vale" + Hex(Asc(Mid$(cadena, POS, 1)))

   FuncSVB = auxiliar
End Function

Function MMID (ByVal cadena As String, ByVal inicio As Integer, ByVal longitud As
Integer) As String
Dim cadena_aux As String
Rem Dim cadimprimir As String

   cadena_aux = Mid$(cadena, inicio, longitud)
   MsgBox "los 3 últimos son (ultimo)" + Hex(Asc(Mid$(cadena_aux, longitud - 2, 1))) + "*" +
Hex(Asc(Mid$(cadena_aux, longitud - 1, 1))) + "*" + Hex(Asc(Mid$(cadena_aux, longitud, 1)))
   MMID = cadena_aux
End Function

Sub Pedir_Cabecera (j As Integer, longfichero As Long)
Dim cadena As String
Static cadrec$
Dim CADAUX As String
Dim nbytesfichero%, nbytesdatos%, crc As Integer
Static salir As Integer
Static ncad As Integer
Dim respuesta As Integer

   CADAUX = Str$(j - 1)
   CADAUX = Mid$(CADAUX, 2)
   'MsgBox CADAUX
   cadrec = ""
   ncad = 0
   salir = False
   cadena = "@SH" & CADAUX & Chr$(13)

   EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) + cadena

   Do
      tramaRec = RecibirHost(5)
      tramaRec = QuitarT(tramaRec)
```

```
      cadrec = cadrec + Mid$(tramaRec, 5)
      'ncad = ncad + 1
      'AddTextoIDS "cadrec[" + Str$(ncad) + "-" + Str$(Len(tramaRec)) + "]: " + CRLF
   Loop Until InStr(tramaRec, ">>") <> 0
   cadena = cadrec
   'AddTextoIDS "cadena[" + Str$(ncad) + "]: " + cadena + CRLF

   respuesta = MsgBox("[PC] Recibido :" + Str$(Len(cadena)) + CRLF + "CANCELAR para
cortar el proceso", 49)
   If respuesta = 2 Then Exit Sub


   'MsgBox "Recibido :" + Str$(Len(cadena)) + cadena


   If Es_Correcta(cadena) = False Then
      MsgBox "cadena no empieza por 01"
   End If

   Rem Kill "c:\julio\ne.ids"



   'FIDS.Text1.Text = cadena
   nbytesfichero = Calcular_long_total(cadena)
   longfichero = nbytesfichero
   FIDS.Label1.Caption = "Cabecera nbytesfichero " + Str$(nbytesfichero)
   nbytesdatos = Bytes_datos(cadena, 1)
   Rem longfichero = nbytesdatos
   FIDS.Label1.Caption = "Cabecera nbytesdatos " + Str$(nbytesdatos)
   crc = FuncSVB(cadena, 3, nbytesdatos)
   crc = crc And 255
   MsgBox "El CRC calculado vale : " + Hex(crc)
   cadena = MMID(cadena, 4, nbytesdatos)
   Put #2, , cadena
End Sub
```

Sub Pedir_Datos (j As Integer, longfichero As Long)

```
MsgBox "Entrando en pedir Datos"
Dim y%, cociente%, NBLOQUE%, RESTO%, crc%, nbytesdatos As Integer
Dim cadena As String
Static cadrec$
Dim respuesta As Integer


   respuesta = MsgBox("[PD] Longfichero vale" + Str$(longfichero) + CRLF + "CANCELAR
para cortar el proceso", 49)
   If respuesta = 2 Then Exit Sub

   y = longfichero - 512
   cociente = y / 1024
   MsgBox "Cociente vale :" + Str$(cociente)
   RESTO = y Mod 1024

   For NBLOQUE = 0 To cociente - 1
      cadena = "@SB" + LTrim$(Str$(j - 1)) + "," + LTrim$(Str$(NBLOQUE)) + Chr(13)
```

```
      MsgBox "*" + cadena + "*"
      EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) + cadena
      cadrec = ""
      Do
         tramaRec = RecibirHost(5)
         tramaRec = QuitarT(tramaRec)
         cadrec = cadrec + Mid$(tramaRec, 5)
      Loop Until InStr(tramaRec, ">>") <> 0
      cadena = cadrec

      respuesta = MsgBox("[PD] He recibido :" + Len(cadena) + "bytes" + CRLF +
"CANCELAR para cortar el proceso", 49)
      If respuesta = 2 Then Exit Sub

      If Es_Correcta(cadena) = False Then
         MsgBox "la cadena recibida no empieza por 01"
      End If

      nbytesdatos = Bytes_datos(cadena, 1)
      FIDS.Label1.Caption = "Datos nbytesdatos:" + Str$(nbytesdatos)
      crc = FuncSVB(cadena, 3, nbytesdatos)
      crc = crc And 255
      MsgBox "El CRC calculado vale : " + Hex(crc)

      cadena = MMID(cadena, 4, nbytesdatos)

      Put #2, , cadena
   Next NBLOQUE

   If RESTO <> 0 Then
      cadena = "@SB" + Mid$(Str$(j - 1), 2) + "," + Mid$(Str$(NBLOQUE), 2) + Chr(13)
      MsgBox "*" + cadena + "*"

      EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) + cadena

      cadrec = ""
      Do
         tramaRec = RecibirHost(5)
         tramaRec = QuitarT(tramaRec)
         cadrec = cadrec + Mid$(tramaRec, 5)
      Loop Until InStr(tramaRec, ">>") <> 0
      cadena = cadrec

      MsgBox "He recibido :" + Len(cadena) + "bytes y resto vale " + Str$(RESTO)

      If Es_Correcta(cadena) = False Then
         MsgBox "la cadena recibida no empieza por 01"
      End If

      nbytesdatos = Bytes_datos(cadena, 1)
      FIDS.Label1.Caption = "Resto nbytesdatos:" + Str$(nbytesdatos)
      crc = FuncSVB(cadena, 3, nbytesdatos)
      crc = crc And 255
      MsgBox "El CRC vale : " + Hex(crc)

      cadena = MMID(cadena, 4, nbytesdatos)
```

```
      Put #2, , cadena
    End If

    cadena = "%MAB" + Mid$(Str$(j - 1), 2) + Chr(13)
    MsgBox cadena
    EscribirSerie "D2" + Chr$(64 + estEspecifica) + Chr$(DAT_TP) + cadena
    tramaRec = RecibirHost(5)
    cadena = Mid$(tramaRec, 5)
End Sub

Sub Pedir_Fichero (j As Integer)
Dim longfichero As Long

    Pedir_Cabecera j, longfichero
    Pedir_Datos j, longfichero
End Sub

Sub Text1_DblClick ()
Dim respuesta%

    respuesta = MsgBox("Esta seguro de borrar el contenido de esta ventana" + CRLF, 49)
    If respuesta = 1 Then Text1.Text = " "

End Sub

Sub Timer1_Timer ()

   contador = contador + 1
   If contador = 2 Then
      Timer1.Enabled = False
   End If

End Sub
```

FPARAM.FRM

```
' Formulario para operar sobre el SIR y sobre la estacion de registro

Sub BAceptar_Click ()
    FParam.Hide
End Sub

Sub BCancelar_Click ()
    OpcionParam = 0
    FParam.Hide
End Sub

Sub Form_Activate ()
    OpcionParam = 11      '11=Parametrizar Tarjeta, Pedir Datos
    OpPedirDatos.Value = True
End Sub

Sub OpBorrarMemIDS_Click ()
    OpcionParam = 22      '2=PArametrizar IDS, Borrar Mem IDS
End Sub
```

```
Sub OpCalTime_Click ()
   OpcionParam = 25      '2=PArametrizar IDS, Calibración Tiempo
End Sub

Sub OpFichCal_Click ()
   OpcionParam = 21      '2=PArametrizar IDS, Fichero Calibracion
End Sub

Sub OpFichConfigIDS_Click ()
   OpcionParam = 23      '3=Configurar IDS
End Sub

Sub OpIntDatos_Click ()
   OpcionParam = 12      '12=Parametrizar Tarjeta, Introducir Datos
End Sub

Sub OpPedirDatos_Click ()
   OpcionParam = 11      '11=Parametrizar Tarjeta, Pedir Datos
End Sub

Sub SincGPS_Click ()
   OpcionParam = 24      '2=PArametrizar IDS, Sincronizar GPS
End Sub
```

FPARASIR.FRM

```
' Formulario para enviar parámetros al SIR

Sub BAceptar_Click ()
Dim i%
   cadParam = ""
   For i = 0 To LBNumEst.ListCount - 1
      If LBNumEst.Selected(i) Then
         cadParam = Chr$(i)
         Exit For
      End If
   Next i

   For i = 0 To LBVel.ListCount - 1
      If LBVel.Selected(i) Then
         cadParam = cadParam + Chr$(i)
         Exit For
      End If
   Next i

   For i = 1 To 8
      cadParam = cadParam + Chr$(0)
   Next i

   FPARAMSIR.Hide

End Sub

Sub BCancelar_Click ()
   cadParam = ""
   FPARAMSIR.Hide
End Sub
```

```
Sub Form_Load ()
Dim i%
   For i = 0 To 255
      LBNumEst.AddItem Str$(i)
   Next i
   LBNumEst.Selected(0) = True

   LBVel.AddItem "0 =  2400 bps"
   LBVel.AddItem "1 =  4800 bps"
   LBVel.AddItem "2 =  9600 bps"
   LBVel.AddItem "3 = 19200 bps"
   LBVel.Selected(0) = True

End Sub
```

FTESTSRAM.FRM

```
' Formulario para operar directamente con al memoria del SIR

Sub BAceptar_Click ()
   FTESTSRAM.Hide
End Sub

Sub BCancelar_Click ()
   OpcionSRAM = 0
    FTESTSRAM.Hide

End Sub

Sub OpEscribir_Click ()
   Label1.Visible = True
   Text1.Visible = True
   Label2.Visible = True
   Text2.Visible = True
   OpcionSRAM = 2
End Sub

Sub OpLeer_Click ()
   Label1.Visible = True
   Text1.Visible = True
   Label2.Visible = False
   Text2.Visible = False
   OpcionSRAM = 1
End Sub
```

FTIME.FRM

```
' Formulario para introducir y solicitar fecha y hora del SIR,
' y para solicitar otros tiempos de la estacion de registro

Sub BAceptar_Click ()

   If OpMirar.Value Then OpcionTime = 1      '1=Mirar Hora
   If OpIntroducir.Value Then OpcionTime = 2      '2=Introducir Hora
   If OpHoraIDS.Value Then OpcionTime = 3      '3=Hora IDS
   If OpHoraGPS.Value Then OpcionTime = 4      '4=Hora GPS
```

```
   FTime.Hide
End Sub

Sub BCancelar_Click ()
   OpcionTime = 0
   FTime.Hide
End Sub
```

APÉNDICE II

Software de la aplicación de control y la red de ordenadores, residente en los sistemas informáticos remotos

Sistema informático remoto. "*Control de estaciones de registro sísmico*"

Código para el micro-controlador 1
(controla la comunicación con la red y dialoga con μC2 a través de SRAM)


```
; ---------------------------------------------------------------------
; NOTAS SOBRE LA COMUNICACION SERIE CON CONTROL POR INTERRUPCION
;
; Se utiliza del BANK1 de registros, R0 y R1.
; El tamaño de los buffers es de 10 bytes cada uno. Si se desea modificar
; se deben cambiar todas las líneas INITX+10 e INIRX+10 por el valor
; deseado. Además, añadir los oportunos mov INI..n,INI..n+1 en los
; módulos de acceso a los buffers.


;---------------------------------------------------------------------
; VARIABLES Y CONSTANTES DEL PROGRAMA
;---------------------------------------------------------------------


; VARIABLES Y REGISTROS. MAPA DE MEMORIA RAM INTERNA DEL μC
;---------------------------------------------------------------------
;  0 a  7: Banco de registros 0 de trabajo: R0 a R7
;  8 a 15: Banco de registros 1:
;          R0= Direc. ultimo byte en buffer de transmisión
;          R1= Direc. ultimo byte en buffer de recepción
;          R2 a R7: usados en bucles de retardo, espera y rutinas que no
;                   deben afectar a los registros del banco 0

TMP1      EQU   16   ; Datos temporales
TMP2      EQU   17   ;
VARSVB    EQU   18   ; Variable para cálculos de SVB
ARIT_L    EQU   19   ; Para cálculos aritméticos con rutinas RESTA y SUMA
ARIT_H    EQU   20
LON_L     EQU   21   ; Longitud de 16 bits de datos en memoria dada por
                     ; OPEN
LON_H     EQU   22
DAT_L     EQU   23   ; Variables para guardar datos de 16 bits
DAT_H     EQU   24
TMP       EQU   25   ; Variable Temporal, donde normalmente se guarda ACC
PTR_OPL   EQU   26   ; Parte baja del puntero OPEN
PTR_OPH   EQU   27   ; Parte alta del puntero OPEN
OP_TOPL   EQU   28   ; Parte baja fin fichero
OP_TOPH   EQU   29   ; Parte alta fin fichero
VARITL    EQU   30   ; Para guardar ARIT_L/H en LOG_ESC
VARITH    EQU   31
ULTCMD    EQU   32   ; Código de ultimo comando enviado a EP
TMPBUF    EQU   33   ; Usada para profundizar en buffers INITX y INIRX
CABEZA    EQU   34   ; Comando que debe enviar ENVI_DFI
TESTDAT   EQU   35   ; Ultimo dato de SRAM elido de TEST
C_CONTR   EQU   36   ; Valor del campo de control de una trama a enviar
                     ; (D,S)
C_CANAL   EQU   37   ; Numero de canal de una trama a enviar
C_CMD1    EQU   38   ; Valor para el campo de comando dentro de los datos
                      ; en una trama a enviar. 0 si no se emplea
C_CMD2    EQU   39   ; Valor para el campo de comando repetido dentro de
                     ; los datos en una trama a enviar. 0 si no se emplea

CR_CONTR  EQU   40   ; Campo de control de un trama recibida
CR_CANAL  EQU   41   ; Numero de canal de un trama recibida
```

```
CR_CMD     EQU   42   ; Campo de comando de una trama recibida (si es D)
                      ; 46: palabra de estado 2
                      ; 47: palabra de estado 1

INITX      EQU   49   ; Inicio del buffer de transmisión (INITX+10 bytes)
INIRX      EQU   62   ; Inicio del buffer de recepción (INIRX+10 bytes)

PILA       EQU   75   ; Inicio de pila (75...127)


;DEFINICIONES DE LOS BITS DE ESTADO
;-----------------------------------------------------------------------

; 46 Palabra de estado 2
;
; 7 6 5 4 3 2 1 0
; | | | | | | | | +- (BIT_4)   Bit indicador de resultado de rutinas
; | | | | | | | +--- (BIT_2)   Bit de propósito general
; | | | | | | +----- (BIT_3)   Bit de propósito general
; | | | | | +------- (BIT_OP)  Bit utilizado por rutinas OPEN
; | | | | +--------- (BIT_E)   Indicador de error al acceder a SRAM
; | | | +----------- (BIT_CM1) Indica si una trama recibida es D, C o S
; | | +------------- (BIT_CM2) Indica si una trama recibida es D, C o S
; | +--------------- libre

BIT_4      EQU   70h  ; Bit 46.0. Bit indicador de resultado de rutinas
BIT_2      EQU   71h  ; Bit 46.1. Bit de propósito general
BIT_3      EQU   72h  ; Bit 46.2. Bit de propósito general
BIT_OP     EQU   73h  ; Bit 46.3. Utilizado por OPEN,PUT y GET
BIT_E      EQU   74h  ; Bit 46.4. Indica error al acceder a memoria
BIT_CM1    EQU   75h  ; Bit 46.5. Indica tipo de trama recibida. Ver tabla
BIT_CM2    EQU   76h  ; Bit 46.6. Indica tipo de trama recibida. Ver tabla

;            BIT_CM1 BIT_CM2  Tipo de trama
;               0       0        Trama D
;               1       0        Trama C
;               0       1        Trama S

; 47 Palabra de estado 1
;
; 7 6 5 4 3 2 1 0
; | | | | | | | | +- (BIT_CIDS) 0=espera recibir primera trama C_IDS.
; | | | | | | | +--- (BIT_LOG)  Indica lectura de diagnóstico con OPEN-GET
; | | | | | | +----- (BIT_RRDY) 0=Sin datos recep.
; | | | | | +------- (BIT_ROV)  1=Overflow recep.
; | | | | +--------- (BIT_RTO)  0=Timeout recep.
; | | | +----------- (BIT_COM)  Mensaje pendiente µC Contrario 0=No 1=Si
; | | +------------- (BIT_TRDY) 0=Sin datos transm.
; | +--------------- (BIT_TOV)  1=Overflow transm.

BIT_CIDS   EQU   78h  ; Bit 47.0. A 0 se espera primera trama C_IDS
BIT_LOG    EQU   79h  ; Bit 47.1. Lectura de diagnostico con OPEN-GET
BIT_RRDY   EQU   7Ah  ; Bit 47.2. Datos en recepción
BIT_ROV    EQU   7Bh  ; Bit 47.3. Oferflow recepción
BIT_RTO    EQU   7Ch  ; Bit 47.4. Timeout recepción
BIT_COM    EQU   7Dh  ; Bit 47.5. Usado para indicar aviso del otro µC
BIT_TRDY   EQU   7Eh  ; Bit 47.6. Datos por transmitir
BIT_TOV    EQU   7Fh  ; Bit 47.7. Oferflow transmisión

; Otros bits utilizados:
BIT_CS1    EQU   0B7h ; Bit P3.7. Línea NOT RD conectada a NOT CS1
BIT_WE     EQU   0B6h ; Bit P3.6. Línea NOT WR conectada a NOT WE
BIT_INT    EQU   0B5h ; Bit P3.5. Disparo de INT0(Flanco descendente)
```

```
BIT_RST    EQU   0B4h ; Bit P3.4. En uC1 Reset de uC1-uC2.
BIT_0      EQU   0D5h ; Bit PSW.5 Propósito general
BIT_1      EQU   0D1h ; Bit PSW.1 Propósito general

; VALORES DE DEFINICIONES CONSTANTES
; ------------------------------------------------------------------
; Mapa de Memoria SRAM
; Area de comandos
;     A_CMD+0:  Código de comando
;     A_CMD+1:  Byte bajo de longitud de bloque de datos en RAM
;     A_CMD+2:  Byte alto de longitud de bloque de datos en RAM
;     ...       Libre
; Area de parámetros
;     A_PAR+0:  Numero de estación
;     A_PAR+1:  Código de velocidad de comunicación con TNC
;     ...       Libre
;     A_PAR+10: SVB del área de parámetros
; Area de datos: A_DATOS ... A_TOPDAT
; Area de mensajes de diagnostico:
;     Formato de un mensaje: día hora min seg código (5 bytes)
;     Hay dos zonas: para uC1 y para yC2. Caben 51 mensajes en cada zona:
;     ALOG1 ... ALOG1+255:          datos de uC1
;     ALOG2 ... ALOG2+255=A_TOPLOG: datos de uC2
; Area del reloj de tiempo real: A_TMCTRL...

A_CMD      EQU   0    ; Dirección de inicio del área de comandos
A_PAR      EQU   9    ; Dirección de inicio del área de parámetros
A_DATOS    EQU   100  ; Dirección de inicio de la zona de datos
A_TOPDAT   EQU   7499 ; Limite de la zona de datos (se incluye este byte)
A_PTRLOG1  EQU   7508 ; Posición del puntero del fichero LOG para uC1
A_PTRLOG2  EQU   7509 ; Posición del puntero del fichero LOG para uC2
A_LOG1     EQU   7510 ; Comienzo del fichero LOG o anotaciones de uC1
A_LOG2     EQU   7765 ; Comienzo del fichero LOG o anotaciones de uC2
A_TOPLOG   EQU   8020 ; Limite del área LOG (se incluye este byte)
A_TMCTRL   EQU   8184 ; Registro de control del reloj
A_TMSEGU   EQU   8185 ; Registro de segundos del reloj
A_TMMINU   EQU   8186 ; Registro de minutos del reloj
A_TMHORA   EQU   8187 ; Registro de horas del reloj
A_TMDIAS   EQU   8188 ; Registro de días de semana del reloj
A_TMDIAM   EQU   8189 ; Registro de días de mes del reloj
A_TMMES    EQU   8190 ; Registro de meses del reloj
A_TMANO    EQU   8191 ; Registro de anos del reloj

;CONSTANTES DE COMANDOS ENTRE LA RED, uC1 y uC2

SONDEO     EQU   33h  ; sondeo de EP a uc1
NO_DAT     EQU   66h  ; no hay datos. de uc1 a Ep
WAIT       EQU   75h  ; memoria ocupada por uc2. de uc1 a EP
DAT_AV     EQU   68h  ; datos de aviso. de uc1 a Ep y de uc2 a uc1
DAT_FI     EQU   6Ah  ; datos de fichero. de uc1 a Ep y de uc2 a uc1
SVB        EQU   6Bh  ; SVB de datos fichero. de uc1 a Ep
ACK_AV     EQU   83h  ; datos de aviso recibidos. de EP a uc1 y de uc1 a uc2
ACK_FI     EQU   0A3h ; datos de fichero recibido. De EP a uc1 y de uc1 a uc2
NACK_FI    EQU   0A5h ; datos fichero erróneos. de EP a uc1
C_MEDIO    EQU   99h  ; respuesta de E.remota a ACK_FI o ACK_AV. para
                      ; devolver el control del medio a EP.de uc1 a EP
TNACK_P    EQU   31h  ; trama errónea o mal conformada. de EP a uc1
TNACK_R    EQU   13h  ; trama errónea o mal conformada. de uc1 a EP
MODO_T     EQU   73h  ; petición de modo datos transparentes.
                      ; de EP a uc1 y de uc1 a uc2
R_MDT      EQU   37h  ; E.remota en modo transparente. De uc2 a uc1 y de uc1 a EP
DAT_TP     EQU   63h  ; datos en modo transparente.de EP a uc1 y de uc1 a uc2
DAT_TR     EQU   36h  ; respuesta con datos transp.de uc2 a uc1 y de uc1 a EP
MODO_N     EQU   54h  ; petición modo operación normal. De EP a uc1 y de uc1 a uc2
```

```
R_MDN      EQU   45h  ; E.remota en modo norma. de uc1 a EP
BATT       EQU   53h  ; petición estado batería. de EP a uc1 y de uc1 a uc2
R_BATT     EQU   35h  ; datos con estado batería. de uc2 a uc1 y de uc1 a EP
FDIAG      EQU   5Bh  ; petición fichero diagnostico. de EP a uc1
R_FDIAG    EQU   0B5h ; datos del fichero diagnostico. De uc1 a EP
P_RESET    EQU   9Ah  ; reset de E.remota. de EP a uc1
R_RESET    EQU   0A9h ; reset aceptado. de uc1 a EP
MODO_P     EQU   4Ah  ; parametrización de la estación de medida.
                      ; De EP a uc1 y de uc1 a uc2
R_MDP      EQU   0A4h ; E.remota dispuesta a aceptar parámetros.
                      ; De uc2 a uc1 y de uc1 a EP
DAT_PP     EQU   4Bh  ; datos para parametrización. De EP a uc1 y de uc1 a uc2
DAT_PR     EQU   0B4h ; respuesta de parametrización de la estación de
                      ; medida. De uc2 a uc1 y de uc1 a EP
RESET_ID   EQU   93h  ; comando de EP a uC1 y de uC1 a uC2 para reiniciar
                      ; por hardware la IDS
R_RSTID    EQU   39h  ; respuesta al comando anterior. De uC2 a uC1 y de uC1 a EP
P_TIME     EQU   56h  ; petición de configuración o lectura del reloj. De EP a uC1
R_TIME     EQU   65h  ; respuesta de uC1 a EP por el comando anterior
SESION_R   EQU   15h  ; la tarjeta indica a la EP sesión establecida
SESION_P   EQU   51h  ; la EP indica a la tarjeta sesión establecida
DEL_FI     EQU   7Ah  ; la EP indica a la tarjeta que puede borrar fichero
REP_FI     EQU   7Bh  ; la EP indica a la tarjeta que reenvíe un fichero
ERROR_F    EQU   0EFh ; Indicación de error fatal a EP
TEST_E     EQU   18h  ; Prueba de SRAM: escritura de dato en una dirección
TEST_L     EQU   1Ah  ; Prueba de SRAM: lectura de dato en una dirección
R_TEST     EQU   81h  ; Respuesta a prueba de SRAM: dato escrito o leído
C_IDS      EQU   87h  ; Configuración de IDS en modo parametrización
R_CIDS     EQU   78h  ; Respuesta a configuración de la IDS

; CODIGOS DE SUBCOMANDOS, (Dentro de P_TIME y DAT_PP)

TIMEPRG    EQU   55h  ; Valor de byte 2 para programación de timer
TIMELEC    EQU   00h  ; Valor de byte 2 para lectura de timer
DATPTAR    EQU   01h  ; Valor de byte 2 para parámetros de tarjeta
DATPIDS    EQU   05h  ; Valor de byte 2 para enviar parámetros a IDS
DATPPRG    EQU   55h  ; Valor de byte 3 para prog. parámetros de tarjeta
DATPLEC    EQU   00h  ; Valor de byte 3 para leer parámetros de tarjeta

; PARAMETROS DE LAS TRAMAS (Fijos en cada envío)

CANAL      EQU 'A'
PUERTO     EQU '2'

; CONSTANTES ESPECIALES DE LAS TRAMAS DE NIVEL HOST

FESC       EQU  0DBh
FEND       EQU  0C0h
TFESC      EQU  0DDh
TFEND      EQU  0DCh

; OTRAS CONSTANTES

INTENT     EQU    3   ; Intentos en envío de tramas DAT_FI o DAT_AV a EP
INTESC     EQU    3   ; Intentos de escritura de un comando para uC2

CR         EQU   0Dh  ; Caracteres ASCII de control
LF         EQU   0Ah
ESC        EQU   1Bh

;CUENTAS PARA TEMP.1 EN LA COMUNICACION SERIE

S96        EQU   0FDh ; 9600/19200 Bd, para cristal de 11.059Mhz (para
                      ; th1)
```

```
S24        EQU   0F4h ; 2400/4800 Bd, para cristal de 11.059Mhz (para th1)
;-----------------------------------------------------------------------
;                          CODIGO DEL PROGRAMA
;-----------------------------------------------------------------------

; INTERRUPCIONES
; ---------------------------------------------------------------------

; Reset
        ORG   0000h
        ljmp  INICIO

; Interrupción externa INT0 (aviso del otro µC)

        ORG   0003h
        setb  BIT_COM
        reti

; Interrupción de la entrada/salida serie

        ORG   0023h
        push  PSW
        push  ACC

        jnb   TI,RXINT        ; Si no está listo para transmitir salta

TXINT:  clr   TI
          ; se comprueba si en el buffer de transmisión hay algún byte.
          ; Si es así se envía


RXINT:  jnb   RI,OUTINT       ; Si no se ha recibido salta
        clr   RI
; lee el byte recibido y lo guarda en el buffer, si cabe.

OUTINT: pop   ACC             ; Recupera estado y selecciona banco
                              ; original
        pop   PSW
        reti


; INICIACION DEL SISTEMA
; ---------------------------------------------------------------------

INICIO:
        mov   SP,#PILA         ; Inicializa el puntero de pila

        call  PULSOINI         ; Inicia envío de pulsos al Wathdog

        mov   TCON,#00000001b  ; Inicializa TCON para INT0 dispare
                               ; por flanco descendente
        mov   TMOD,#00100001b  ; T0 en modo 1: temp. 16 bits.
                               ; T1 en modo 2: temp. 8 bits con
                               ;autorrecarga
        mov   A,#3             ; Para configurar 9600Bd
        call  CFGSERIE         ; Configura puerto serie
        setb  TR1              ; Arranca Temp. 1

        mov   47,#0             ;Inicializa a 0 bits de palabra de estado 1
                                ; BIT_CIDS  BIT_LOG  BIT_RRDY  BIT_ROV
                                ; BIT_RTO   BIT_COM  BIT_TRDY  BIT_TOV
```

```
        mov   9,#INIRX          ; Inicializa control buffer recepción y
        mov   8,#0              ; de transmisión: 9h=R1 BANK 1,8h=R0 BANK 1

        setb  IE.4              ; Habilita interrupciones serie
        setb  IE.0              ; Habilita INT0
        setb  IE.7              ; Habilita interrupciones en general

        call  LOG_INI           ; Inicia mensajes de diagnostico
        call  STRCLK            ; Pone a cero bit de STOP del reloj
        call  INICMD            ; Iniciar posición 0 de SRAM: comando

        clr   A
        call  LOG_ESC           ; Mensaje de iniciación completada


; ESTADO DE REPOSO, EN ESPERA DE CONEXION
; ---------------------------------------------------------------------

REP_1:  call  PULSO5            ; Reinicia watchdog
        jb    BIT_RRDY,REP_2    ; Espera recepción o aviso de uC2
        jb    BIT_COM,ATEND
        sjmp  REP_1

REP_2:  call  RECI_CAB          ; Recibe cabecera de la trama
        jb    BIT_0,REP_1       ; Si error trama mal conformada va al bucle
        jb    BIT_1,REP_1
        jb    BIT_CM1,REP_3     ; Si es trama C salta
        jnb   BIT_CM2,REP_3     ; Si no es trama S (es trama D) salta

        jb    BIT_3,REP_1       ; Si no es trama S valida vuelve al bucle
        jnb   BIT_4,REP_1       ; Si no es trama S Connect vuelve al bucle

        call  DLY20             ; Espera un poco
        mov   A,#SESION_R       ; Confirma aceptación de sesión
        call  ENVI_DAT
        jmp   MODON             ; Si es Connect pasa a estado de conexión

REP_3:  call  DESECHA           ; Lee resto de trama y vuelve al bucle
        sjmp  REP_1


; ATENDIENDO AVISO DE uC2 EN ESTADO DE REPOSO: SOLICITAR CONEXION
; ---------------------------------------------------------------------

ATEND:  clr   BIT_COM           ; Desactiva aviso de uC2
        call  LEECMD            ; Lee comando de uC2
        jz    REP_1
        mov   B,A               ; Guarda en B el comando leído

; Comprueba comando de uC2

ATEND_0:mov   A,#DAT_AV         ; Si es DAT_AV o DAT_FI va a ATEND_2
        xrl   A,B
        jz    ATEND_2
        mov   A,#DAT_FI
        xrl   A,B
        jz    ATEND_2
        mov   A,#R_MDN          ; Si es R_MDN salta a ATEND_1
        xrl   A,B
        jz    ATEND_1

        ; Tratar comando no esperado de uC2
```

```
        mov   A,#1              ; Anota mensaje en diagnostico
        call  LOG_ESC
        jmp   REP_1             ; Vuelve a espera

        ; Tratar comando R_MDN de uC2. Puede recibirse un R_MDN debido a
        ; que cuando uC1 desconecta envía MODO_N a uC2 y se pasa a REP_1

ATEND_1:  jmp   REP_1           ; Ignora y vuelve a estado de reposo

; Solicitar conexión para tratar comandos DAT_FI y DAT_AV de uC2

ATEND_2:  call  SOLI_CON        ; Solicitar conexión

; Va a estado de conexión en modo normal, que está a continuación


; ESTADO DE CONEXION EN MODO NORMAL
; ---------------------------------------------------------------------

MODON:  call  PULSO5            ; Reinicia watchdog
        jb    BIT_RRDY,MODON_2  ; Espera recepción o aviso de uC2
        jb    BIT_COM,MODON_5
        sjmp  MODON

; Recepción de trama de la EP en modo normal

MODON_2:call  RECI_CAB          ; Lee cabecera de trama
        jnb   BIT_1,MODON_3     ; Salta si no hay error trama mal
                                ; conformada

        mov   A,#0              ; Para que ENVINACK no anote en diagnostico
        call  ENVINACK          ; Envía TNACK_R si trama mal conformada
        sjmp  MODON             ; Pasa a espera en conexión

MODON_3:jnb   BIT_CM1,MODON_3A  ; Si no es trama C salta
        call  DESECHA           ; Desecha las tramas de comando
        sjmp  MODON

MODON_3A:jnb  BIT_CM2,EPMDN     ; Si es trama S sigue, y si es D salta
        jb    BIT_3,MODON       ; Si trama S no valida salta al bucle
        jb    BIT_4,MODON       ; Si trama S conect salta al bucle

; Se ha recibido disconnect: hay que avisar a uC2 y ponerlo en modo
; normal

        mov   A,#MODO_N         ; Indica MODO_N a uC2
        call  AUC2ESP
        jmp   REP_1             ; Pasa a estado de reposo

; Aviso de uC2 mediante BIT_COM

MODON_5:  clr   BIT_COM         ; Desactiva aviso de uC2

        call  LEECMD            ; Lee comando de uC2
        jz    MODON             ; Si comando erróneo espera otro
        jmp   UC2MDN            ; Va a tratar comando de uC2


; ATENDER COMANDO DE EP EN MODO NORMAL
; ---------------------------------------------------------------------

EPMDN:  call  RECI_DAT                  ; Recibe resto de comando
        jnb   BIT_4,EPMDN_1             ; Si no hay error sigue
```

```
        jmp   MODON                     ; Vuelve en caso de error

; Tratar comando P_RESET desde EP

EPMDN_1:cjne  A,#P_RESET,EPMDN_2        ; Si no se trata de P_RESET sigue

        mov   A,#R_RESET                ; Envía respuesta R_RESET
        call  ENVI_DAT
        call  RESETT                    ; Ejecuta reset, y aquí acaba
        jmp   MODON                     ; Vuelve en caso de error

; Tratar comando TNACK_P desde la EP

EPMDN_2:cjne  A,#TNACK_P,EPMDN_3        ; Si no se trata de TNACK_P sigue

        call  RNACK                     ; Tratar reenvío
        jmp   MODON                     ; Vuelve a bucle de espera

; Atender comandos MODO_T y MODO_P

EPMDN_3:cjne  A,#MODO_T,EPMDN_D         ; Si trama datos no MODO_T salta
        mov   B,#R_MDT                  ; Respuesta para EP a MODO_T
        sjmp  EPMDN_D1

EPMDN_D:cjne  A,#MODO_P,EPMDN_4         ; Si trama datos no MODO_P salta
        mov   B,#R_MDP                  ; Respuesta para EP a MODO_P

; Ver comando que hay en la SRAM

EPMDN_D1:call  LEECMD                   ; Lee comando puesto por uC2
        jz    EPMDN_Y                   ; Si comando erróneo lo ignora
        cjne  A,#WAIT,EPMDN_Y           ; Salta si uC2 no ha puesto WAIT
        call  ENVI_DAT                  ; Si uC2 ha puesto WAIT lo envía a EP
        sjmp  MODON                     ; Vuelve al bucle

; Enviar respuesta a EP, comando a uC2 y cambiar de modo

EPMDN_Y:mov   A,B                       ; Recupera respuesta para EP
        call  ENVI_DAT                  ; Envía respuesta a EP
        mov   A,B
        cjne  A,#R_MDP,EPMDN_Y1         ; Salta si no era MODO_P (era MODO_T)

        mov   A,#MODO_P                 ; Envía comando a uC2
        call  AUC2ESP
        jmp   MODOP                     ; Va a modo parametrización

EPMDN_Y1:mov   A,#MODO_T                ; Envía comando a uC2
        call  AUC2ESP
        jmp   CON_2                     ; Pasa a modo transparente

; Atender comando SONDEO

EPMDN_BX: jmp   EPMDN_B
EPMDN_4:  cjne  A,#SONDEO,EPMDN_BX      ; Si trama datos no SONDEO salta

; Ver comando que hay en la SRAM

EPMDN_5: call  VERCMD                   ; Lee comando puesto por uC2
        jb    BIT_4,EPMDN_Z             ; No pilla RAM
        jz    EPMDN_Z                   ; Si comando erróneo espera otro
        cjne  A,#WAIT,EPMDN_9           ; Si el comando no es WAIT salta
        call  ENVI_DAT                  ; Si el comando es WAIT lo envía a EP

EPMDN_Z:  jmp   MODON
```

```
; Ver si hay datos de fichero

EPMDN_9:cjne  A,#DAT_FI,EPMDN_7        ; Si no hay datos de fichero salta

        mov   CABEZA,#DAT_FI           ; Usa tramas de envío de fichero
        call  ENVI_DFI                 ; Envía datos de fichero a EP
        jb    BIT_4,EPMDN_8            ; Si hay error salta

        mov   A,#ACK_FI                ; Pasa confirmación a uC2
        call  AUC2ESP
        jmp   MODON                    ; Pasa a esperar aviso de uC2

; Ver si hay datos de aviso

EPMDN_7:cjne  A,#DAT_AV,EPMDN_C        ; Si no hay datos de aviso salta

        call  ENVI_DAV
        jb    BIT_4,EPMDN_8            ; Si hay error salta

        mov   A,#ACK_AV                ; Pasa confirmación a uC2
        call  AUC2ESP
        jmp   MODON                    ; Pasa a esperar aviso de uC2

; Ver tipo de error al enviar fichero o aviso

EPMDN_8:jb    BIT_0,EPMDN_A            ; Salta si recibido disconnect
        jb    BIT_1,EPMDN_5            ; Salta si sondeo recibido
        jmp   MODON                    ; Vuelve al bucle

; Tratar desconexión

EPMDN_A:mov   A,#MODO_N                ; Indica MODO_N a uC2
        call  AUC2ESP
        jmp   REP_1                    ; Pasa a estado de reposo

; No hay datos de fichero ni de aviso

EPMDN_C:mov   A,#NO_DAT
        call  ENVI_DAT
        jmp   MODON                    ; Vuelve al bucle


; Atender paso a modo normal estando en modo normal

EPMDN_B:cjne  A,#MODO_N,EPMDN_F
        mov   A,#R_MDN
        call  ENVI_DAT
        jmp   MODON

; Atender comandos DEL_FI y REP_FI de la EP

EPMDN_F:cjne  A,#DEL_FI,EPMDN_G        ; Si no es DEL_FI salta
        sjmp  EPMDN_G2

EPMDN_G:cjne  A,#REP_FI,EPMDN_E        ; Si no es REP_FI salta

EPMDN_G2:call  AUC2DIR                 ; Pasa comando a uC2
        mov   A,#C_MEDIO               ; Envía C_MEDIO a EP
        call  ENVI_DAT
        jmp   MODON                    ; Vuelve a bucle de modo normal

; Atender respuestas no esperadas
```

```
EPMDN_E:call  DESECHA                  ; Desecha resto de trama
        mov   A,#10                    ; Código de error para diagnostico
        call  ENVINACK                 ; Envía TNACK_R
        jmp   MODON                    ; Vuelve al bucle de espera


; ATENDER COMANDO DE uC2 EN MODO NORMAL
;-----------------------------------------------------------------------

; Comandos DAT_FI y DAT_AV se ignoran (se atienden en sondeo)

UC2MDN: cjne  A,#DAT_FI,UC2MDN_2       ; Si no es DAT_FI salta
        jmp   MODON                    ; Vuelve al bucle

UC2MDN_2:cjne  A,#DAT_AV,UC2MDN_3      ; Si no es DAT_AV salta
        jmp   MODON                    ; Vuelve al bucle

; Otros comandos de uC2 son erróneos

UC2MDN_3:mov   A,#2                    ; Anota mensaje en diagnostico
        call  LOG_ESC
        jmp   MODON                    ; Vuelve al bucle


; ESTADO DE CONEXION EN MODO TRANSPARENTE
; ---------------------------------------------------------------------

CON_2:  call  PULSO5            ; Reinicia watchdog
        jb    BIT_RRDY,CON_3    ; Espera recepción EP o aviso uC2
        jb    BIT_COM,CON_4
        sjmp  CON_2

; Recepción de una trama desde la EP

CON_3:  call  RECI_CAB          ; Lee cabecera de trama
        jnb   BIT_1,CON_3A      ; Salta si no hay error trama mal
                                ; conformada

        mov   A,#0              ; Para que ENVINACK no anote en diagnostico
        call  ENVINACK          ; Envía TNACK_R si trama mal conformada
        sjmp  CON_2             ; Pasa a espera en conexión

CON_3A: jnb   BIT_CM1,CON_3B    ; Si no es trama C salta
        call  DESECHA           ; Ignora tramas de comando
        sjmp  CON_2

CON_3B: jnb   BIT_CM2,RESPEP    ; Si es trama S sigue, si es trama D salta
        jb    BIT_3,CON_2       ; Si trama S no valida salta al bucle
        jb    BIT_4,CON_2       ; Si trama S connect salta al bucle

        mov   A,#MODO_N         ; Envía comando de modo normal a uC2
        call  AUC2ESP
        jmp   REP_1             ; Pasa a estado de reposo si Disconnect

; Aviso de uC2 mediante BIT_COM

CON_4:  clr   BIT_COM           ; Desactiva aviso de uC2

        call  LEECMD            ; Lee comando de uC2
        jz    CON_2             ; Si comando erróneo espera otro
        jmp   RESPUC2           ; Tratar comando de uC2
```

```
; ATENDER AVISO DE uC2 EN MODO TRANSPARENTE
; ------------------------------------------------------------------------

; Tratar comando R_BATT

RESPUC2:cjne  A,#R_BATT,RESPUC_1       ; Si no es R_BATT sigue
        call  EDTP                     ; Tratar comando
        jmp   CON_2                    ; Vuelve al bucle

; Tratar comando R_RSTID

RESPUC_1:cjne  A,#R_RSTID,RESPUC_2     ; Si no es R_RSTID sigue
        call  EDTP                     ; Tratar comando
        jmp   CON_2                    ; Vuelve al bucle

; Tratar comando DT_TR

RESPUC_2:cjne  A,#DAT_TR,RESPUC_3      ; Si no es DAT_TR sigue
        call  EDTP                     ; Tratar comando
        jmp   CON_2                    ; Vuelve al bucle

; Tratar comando R_MDN

RESPUC_3:cjne  A,#R_MDN,RESPUC_4       ; Si no es R_MDN sigue
        call  ENVI_DAT                 ; Envía respuesta R_MDN a EP
        jmp   MODON                    ; Pasa a modo normal

; Otros comandos no esperados

RESPUC_4: mov   A,#3                   ; Anota mensaje en diagnostico
        call  LOG_ESC
        jmp   CON_2                    ; Vuelve al bucle


; ATENDER COMANDOS DE EP EN MODO TRANSPARENTE
; ------------------------------------------------------------------------

; Tratar comando DAT_TP desde EP

RESPEP: cjne  A,#DAT_TP,RESPEP_7       ; Si no se trata de DAT_TP sigue
        call  RDTP
        jmp   CON_2                    ; Vuelve a bucle de espera

; Tratar comando P_TIME desde la EP

RESPEP_7:cjne  A,#P_TIME,RESPEP_A      ; Si no se trata de P_TIME sigue
        call  TIME                     ; Tratar comando
        jmp   CON_2                    ; Vuelve a bucle de espera

; Comando de test de escritura en SRAM

RESPEP_A:cjne  A,#TEST_E,RESPEP_B      ; Si no se trata de TEST_E sigue
        call  MEMTS                    ; Tratar comando
        jmp   CON_2                    ; Vuelve a bucle de espera

; Comando de test de lectura de SRAM

RESPEP_B:cjne  A,#TEST_L,RESPEP_0      ; Si no se trata de TEST_L sigue
        call  MEMTS                    ; Tratar comando
        jmp   CON_2                    ; Vuelve a bucle de espera
```

```
; El resto de comandos se tratan como comandos cortos

RESPEP_0:call  RECI_DAT                 ; Recibe resto de comando
         jnb   BIT_4,RESPEP_1           ; Si no hay error sigue
         jmp   CON_2                    ; Vuelve en caso de error

; Tratar comando P_RESET desde EP

RESPEP_1:cjne  A,#P_RESET,RESPEP_2      ; Si no se trata de P_RESET sigue
         call  RESETT                   ; Ejecuta reset, y aquí acaba
         jmp   CON_2                    ; Vuelve en caso de error

; Tratar comando TNACK_P desde la EP

RESPEP_2:cjne  A,#TNACK_P,RESPEP_3      ; Si no se trata de TNACK_P sigue
         call  RNACK                    ; Tratar reenvío
         jmp   CON_2                    ; Vuelve a bucle de espera

; Tratar comando BATT desde la EP

RESPEP_3:cjne  A,#BATT,RESPEP_4         ; Si no se trata de BATT sigue
         call  AUC2DIR                  ; Tratar comando directo a uC2
         jmp   CON_2                    ; Vuelve a bucle de espera

; Tratar comando RESET_ID desde EP

RESPEP_4:cjne  A,#RESET_ID,RESPEP_5     ; Si no se trata de RESET_ID sigue
         call  AUC2DIR                  ; Tratar comando directo a uC2
         jmp   CON_2                    ; Vuelve a bucle de espera

; Tratar comando FDIAG desde EP

RESPEP_5:cjne  A,#FDIAG,RESPEP_6        ; Si no se trata de FDIAG sigue
         call  LOG_ENV                  ; Envía mensajes diagnostico a EP
         jb    BIT_4,RESPEP_X           ; Si se recibió disconnect salta
         jmp   CON_2                    ; Vuelve a bucle de espera

RESPEP_X:mov   A,#MODO_N                ; Envía comando modo normal a uC2
         call  AUC2DIR
         jmp   REP_1                    ; Pasa a estado de reposo

; Tratar comando NACK_FI desde la EP

RESPEP_6:cjne  A,#NACK_FI,RESPEP_8      ; Si no se trata de NACK_FI sigue
         call  RNACK                    ; Tratar reenvío
         jmp   CON_2                    ; Vuelve a bucle de espera

; Tratar comando MODO_N desde la EP

RESPEP_8:cjne  A,#MODO_N,RESPEP_9       ; Si no se trata de MODO_N sigue
         call  AUC2DIR                  ; Tratar comando directo a uC2
         jmp   CON_2                    ; Vuelve a bucle de espera

; Atender paso a modo transparente estando en modo transparente

RESPEP_9: cjne  A,#MODO_T,RESPEP_C
         mov   A,#R_MDT
         call  ENVI_DAT
         jmp   CON_2

; Tratar otros comandos no esperados en el modo actual

RESPEP_C:call  DESECHA                  ; desecha resto de trama
```

```
        mov   A,#11                  ; Código de error para diagnostico
        call  ENVINACK               ; Envía TNACK_R
        jmp   CON_2                  ; Vuelve al bucle


; ESTADO DE CONEXION EN MODO PARAMETRIZACION
; ------------------------------------------------------------------------

MODOP:  call  PULSO5            ; Reinicia watchdog
        jb    BIT_RRDY,MODOP_3 ; Espera recepción EP o aviso uC2
        jb    BIT_COM,MODOP_4
        sjmp  MODOP

; Recepción de una trama desde la EP

MODOP_3:call  RECI_CAB         ; Lee cabecera de trama
        jnb   BIT_1,MODOP_3A   ; Salta si no hay error de trama mal
                               ; conformada

        mov   A,#0             ; Para que ENVINACK no anote en diagnostico
        call  ENVINACK         ; Envía TNACK_R si trama mal conformada
        sjmp  MODOP            ; Vuelve al bucle

MODOP_3A:jnb  BIT_CM1,MODOP_3B ; Si no es trama C salta
        call  DESECHA          ; Desecha tramas C
        sjmp  MODOP

MODOP_3B: jnb  BIT_CM2,EPMDP ; Si trama S sigue, si trama D salta
        jb    BIT_3,MODOP      ; Si trama S no valida va al bucle
        jb    BIT_4,MODOP      ; Si trama S connect salta al bucle

        mov   A,#MODO_N        ; Envía comando de modo normal a uC2
        call  AUC2ESP
        jmp   REP_1            ; Pasa a estado de reposo si Disconnect

; Aviso de uC2 mediante BIT_COM

MODOP_4:clr   BIT_COM          ; Desactiva aviso de uC2

        call  LEECMD           ; Lee comando de uC2
        jmp   UC2MDP           ; Tratar comando de uC2


; ATENDER COMANDOS DE EP EN MODO PARAMETRIZACION
; ------------------------------------------------------------------------

; Tratar comando DAT_PP desde EP

EPMDP:  cjne  A,#DAT_PP,EPMDP_7       ; Si no se trata de DAT_TP sigue
        call  RDPR
        jmp   MODOP                   ; Vuelve a bucle de espera

EPMDP_7:cjne  A,#C_IDS,EPMDP_0        ; Si no se trata de C_IDS sigue
        call  RDCIDS
        jmp   MODOP                   ; Vuelve a bucle de espera

; El resto de comandos se tratan como comandos cortos (sin datos
; adicionales)

EPMDP_0:call  RECI_DAT                ; Recibe resto de comando
        jnb   BIT_4,EPMDP_1           ; Si no hay error sigue
        jmp   MODOP                   ; Vuelve en caso de error
```

```
; Tratar comando P_RESET desde EP

EPMDP_1:cjne  A,#P_RESET,EPMDP_2      ; Si no se trata de P_RESET sigue
        call  RESETT                  ; Ejecuta reset, y aquí acaba
        jmp   MODOP                   ; Vuelve en caso de error

; Tratar comando TNACK_P desde la EP

EPMDP_2:cjne  A,#TNACK_P,EPMDP_4      ; Si no se trata de TNACK_P sigue
        call  RNACK                   ; Tratar reenvío
        jmp   MODOP                   ; Vuelve a bucle de espera

; Tratar comando MODO_N desde la EP

EPMDP_4:cjne  A,#MODO_N,EPMDP_5       ; Si no se trata de MODO_N sigue
        call  AUC2DIR                 ; Tratar comando directo a uC2
        jmp   MODOP                   ; Vuelve a bucle de espera

; Atender paso a modo parametrización estando en modo parametrización

EPMDP_5:cjne  A,#MODO_P,EPMDP_6       ; Si no se trata de MODO_P sigue
        mov   A,#R_MDP                ; Envía respuesta a EP
        call  ENVI_DAT
        jmp   MODOP

; Se recibe comando SVB como final de tramas C_IDS

EPMDP_6:cjne  A,#SVB,EPMDP_8          ; Ver si comando SVB
        call  CLOSE                   ; Acaba de meter datos en SRAM
        mov   A,#C_IDS                ; Comando para uC2
        call  AUC2DIR
        clr   BIT_CIDS                ; Se esperara primera trama C_IDS
        jmp   MODOP

; Tratar otros comandos no esperados en el modo actual

EPMDP_8:call  DESECHA                 ; Desecha resto de trama recibida
        mov   A,#12                   ; Código de error para diagnostico
        call  ENVINACK                ; Envía TNACK_R
        jmp   MODOP                   ; Vuelve a bucle de espera


; ATENDER AVISOS DE uC2 EP EN MODO PARAMETRIZACION
; ------------------------------------------------------------------------

; Tratar comando R_MDN

UC2MDP: cjne  A,#R_MDN,UC2MDP_1       ; Si no es R_MDN salta
        call  ENVI_DAT                ; Envía respuesta R_MDN a EP
        jmp   MODON                   ; Pasa a modo normal

; Tratar comando DAT_PR

UC2MDP_1:cjne  A,#DAT_PR,UC2MDP_2     ; Si no es DAT_TP salta
        call  EDTP                    ; Tratar comando
        jmp   MODOP                   ; Vuelve al bucle

; Tratar comando R_CIDS

UC2MDP_2:cjne  A,#R_CIDS,UC2MDP_3     ; Si no es R_CIDS salta

        mov   CABEZA,A                ; Enviar respuesta a config.IDS
```

```
        call  ENVI_DFI
        jnb   BIT_0,UC2MDP_4          ; Si no disconect recibido salta

        mov   A,#MODO_N               ; Si recibe DISCONECT avisa a uC2
        call  AUC2DIR                 ; para que pase a modo normal
        jmp   REP_1                   ; Pasa a estado de reposo

; Otros comandos no esperados

UC2MDP_3:mov   A,#4                   ; Anota mensaje en diagnostico
        call  LOG_ESC

UC2MDP_4: jmp   MODOP                   ; Vuelve al bucle


;-----------------------------------------------------------------------
;                  FUNCIONES VARIAS DE LA APLICACIÓN
;-----------------------------------------------------------------------

; FUNCIONES RELATIVAS A LA RECEPCION DE DATOS
; ---------------------------------------------------------------------

;.......................................................................
; Función:  Trata comando DAT_PP de la EP. Recibe datos de
;           parametrización para la tarjeta o para la IDS.
; Entrada:
; Salida:
; Modif.R.: A, B, DPTR
; Modif.B:  BIT_E
; Usa:      RDTP, PIDE_RAM, DESECHA, D_RAM, ENVINACK, RECITR
;           E_RAM, LOG_ESC, EDPR
;.......................................................................

RDPR:   call  RECITR          ; Recibe dato transparente (subcomando)
        jb    BIT_0,RDPR_X2   ; Salta si timeout
        jb    BIT_1,RDPR_X1   ; Salta si dato no esperado
        jb    BIT_3,RDPR_X2   ; Si ha recibido FEND acaba

; Ver si el comando es de parámetros para IDS

        cjne  A,#DATPIDS,RDPR_2 ; Si no es para IDS salta

        mov   A,#DAT_PP         ; Comando para uC2
        call  RDTP              ; Recibe datos de parametrización desde EP
        ret                     ; Acaba

; Ver si el comando es de parámetros para tarjeta

RDPR_2: cjne  A,#DATPTAR,RDPR_X3 ; Si no son param. para tarjeta salta

        call  RECITR          ; Recibe dato transparente (subcomando)
        jb    BIT_0,RDPR_X2   ; Salta si timeout
        jb    BIT_1,RDPR_X1   ; Salta si dato no esperado
        jb    BIT_3,RDPR_X2   ; Si ha recibido FEND acaba

; Ver si es comando para escribir datos en la tarjeta

        cjne  A,#DATPPRG,RDPR_5 ; Si no es escr. de par. de tarjeta salta

; Intenta pillar memoria

        call  PIDE_RAM        ; Intenta pillar memoria
```

```
        jnb   BIT_4,RDPR_3     ; Si pilla memoria sigue
        call  DESECHA          ; Desechar lo recibido
        ret

; Byte no esperado, no se encontró FEND final o timeout entre bytes

RDPR_X1:  call  DESECHA          ; Desecha resto de trama recibida
RDPR_X2:  call  D_RAM            ; Libera memoria si estaba pillada
          mov   A,#50            ; Código de error para diagnostico
RDPR_X4:  call  ENVINACK         ; Envía TNACK_R
          ret

; SubComando no esperado

RDPR_X3:call  DESECHA          ; Desecha resto de trama recibida
        mov   A,#60            ; Código de error para diagnostico
        sjmp  RDPR_X4

; Leer parámetros para tarjeta y ponerlos en SRAM

RDPR_3: clr   BIT_E            ; Borra indicador de error
        mov   B,#10            ; Numero de datos a recibir
        mov   DPTR,#A_PAR      ; Puntero al área de parámetros

RDPR_4: call  RECITR           ; Recibe dato transparente
        jb    BIT_0,RDPR_X2    ; Salta si timeout
        jb    BIT_1,RDPR_X1    ; Salta si dato no esperado
        jb    BIT_3,RDPR_X2    ; Si FEND acaba el bucle
        call  E_RAM            ; Escribe dato

        inc   DPTR             ; Apunta a siguiente dirección
        djnz  B,RDPR_4         ; Si quedan datos por recibir sigue

        call  DESECHA          ; Espera FEND
        jb    BIT_4,RDPR_X2    ; Si timeout salta

; Ver si hubo error y enviar datos de param. de la tarjeta a la EP:

        call  D_RAM
        jnb   BIT_E,RDPR_4B    ; Salta si no hubo error
        mov   A,#23            ; Anota mensaje error
        call  LOG_ESC

RDPR_4B:call  EDPR             ; Enviar datos param. de tarjeta
        ret

; Si es comando para leer datos de la tarjeta

RDPR_5: cjne  A,#DATPLEC,RDPR_X3 ; Si no es lectura de par. de tarjeta
                                 ; salta

        call  DESECHA          ; Espera FEND
        jb    BIT_4,RDPR_X2    ; Si timeout salta

; Enviar datos de param. de la tarjeta a la EP

RDPR_6: call  EDPR             ; Enviar datos param. de tarjeta
        ret


;.....................................................................
; Función:  Recibir datos de configuración de IDS y pasarlos a la SRAM.
; Entrada:  BIT_CIDS = 0 si se espera primera trama C_IDS
```

```
; Salida:
; Modif.D:  A
; Modif.R:  BIT_CIDS
; Usa:      OPEN, RDTP
;.....................................................................

RDCIDS: jb    BIT_CIDS,RDCIDS_1 ; Si no es la primera trama C_IDS salta

        call  OPEN              ; Va a empezar a meter datos en SRAM
        setb  BIT_CIDS          ; Se esperan otras tramas C_IDS

RDCIDS_1: clr   A               ; Recibir contenido transparente de trama
        call  RDTP
        ret

;.....................................................................
; Función:  Tratar comando P_TIME procedente de la EP y determina si se
;           trata de una petición de la hora actual o de una programación
;           de una nueva hora. Según eso, envía la hora actual (llama a
;           TIME_R) o recibe los datos y actualiza el reloj.
; Entrada:
; Salida:
; Modif.R:  A, R0 a R6, DPTR
; Modif.B:  BIT_E
; Usa:      TIME_R, DESECHA, D_RAM, ENVINACK, PIDE_RAM, RECITR, ESCCLK,
;           E_RAM, NOPCLK, LOG_ESC
;.....................................................................

TIME:   call  RECITR            ; Recibe dato transparente (subcomando)
        jb    BIT_0,TIME_X2     ; Salta si timeout
        jb    BIT_1,TIME_X1     ; Salta si dato no esperado
        jb    BIT_3,TIME_X2     ; Si ha recibido FEND acaba

; Ver si el comando es de lectura de hora

        cjne  A,#TIMELEC,TIME_3   ; Si siguiente no es el esperado salta

        call  DESECHA                 ; Espera FEND
        jb    BIT_4,TIME_X2           ; Si timeout salta

        call  TIME_R                  ; Envía hora actual
        ret                           ; Acaba

; Byte no esperado, no se encontró FEND final o timeout entre bytes

TIME_X1:  call  DESECHA         ; Desecha resto de trama recibida
TIME_X2:  call  D_RAM           ; Libera memoria si estaba pillada
        mov   A,#51             ; Código de error para diagnostico
TIME_X4:  call  ENVINACK        ; Envía TNACK_R
        ret

; SubComando no esperado

TIME_X3:  call  DESECHA         ; Desecha resto de trama recibida
        mov   A,#61             ; Código de error para diagnostico
        sjmp  TIME_X4

; Si el comando es de programación, guarda datos en R1 a R6

TIME_3:   cjne  A,#TIMEPRG,TIME_X3    ; Si siguiente no es esperado salta

; Intenta pillar memoria

        call  PIDE_RAM          ; Intenta pillar memoria
```

```
        jnb   BIT_4,TIME_4     ; Si pilla memoria sigue
        call  DESECHA          ; Desecha lo recibido
        ret

; Guarda datos recibidos en R1 a R6

TIME_4:   mov   R0,#1          ; Apunta a registro 1 del banco 0

TIME_5: call  RECITR           ; Recibe dato transparente
        jb    BIT_0,TIME_X2    ; Salta si timeout
        jb    BIT_1,TIME_X1    ; Salta si dato no esperado
        jb    BIT_3,TIME_X2    ; Si FEND acaba el bucle
        mov   @R0,A            ; Guarda dato en registros R1 a R6
        inc   R0               ; Apunta a siguiente registro
        cjne  R0,#7,TIME_5     ; Si siguiente registro no es R7 repite

        call  DESECHA          ; Espera FEND
        jb    BIT_4,TIME_X2    ; Si timeout salta

; Borra indicador de error al acceder a SRAM

        clr   BIT_E

; Desactivar bit de test de frecuencia por si esta activado

        mov   DPTR,#A_TMDIAS   ; Registro de días de semana
        clr   ACC.6            ; Desactiva bit 6 de A (FT)
        call  E_RAM

; Activar bit de escritura del reloj sin modificar el valor de calibración

        call  ESCCLK           ; Activa escritura en reloj

; Escribir estado del reloj

        mov   DPTR,#A_TMDIAM   ; Recupera y escribe día de mes
        mov   A,R1
        call  E_RAM

        mov   DPTR,#A_TMMES    ; Recupera y escribe mes
        mov   A,R2
        call  E_RAM

        mov   DPTR,#A_TMANO    ; Recupera y escribe año
        mov   A,R3
        call  E_RAM

        mov   DPTR,#A_TMHORA   ; Recupera y escribe hora
        mov   A,R4
        call  E_RAM

        mov   DPTR,#A_TMMINU   ; Recupera y escribe minutos
        mov   A,R5
        call  E_RAM

        mov   DPTR,#A_TMSEGU   ; Recupera y escribe segundos
        mov   A,R6
        clr   ACC.7            ; Mantiene a 0 bit de STOP
        call  E_RAM

; Desactivar bit de escritura del reloj sin modificar el valor de calibración

        call  NOPCLK           ; Desactiva lectura y escritura
```

```
        call  D_RAM

; Anotar error de acceso a SRAM si lo hubo

        jnb   BIT_E,TIME_5A

        mov   A,#25
        call  LOG_ESC

; Enviar hora programada

TIME_5A:call  TIME_R          ; Enviar datos
        ret                   ; Acaba

;.....................................................................
; Función:  Tratar comandos DAT_TP desde la EP. Recibe datos
;           transparentes desde la EP y los pasa a la SRAM
; Entrada:  A = comando para uC2 (DAT_TR o DAT_TP). Si A es 0 no se hace
;           OPEN ni CLOSE ni se avisa a uC2. En otro caso se hace OPEN,
;           CLOSE y se avisa a uC2.
; Salida:
; Modif.R:  A, B
; Modif.B:
; Usa:      OPEN, RECITR, CLOSE, AUC2ESP, PUT, DESECHA, CLOSE, ENVINACK
;.....................................................................

RDTP:   mov   B,A                   ; Guarda comando para uC2 en B
        jz    RDTP_0                ; Si A es cero no hace OPEN
        call  OPEN                  ; Abre para escribir datos para uC2

RDTP_0: call  RECITR                ; Recibe dato transparente
        jb    BIT_0,RDTP_7          ; Salta si timeout
        jb    BIT_1,RDTP_6          ; Salta si dato no esperado
        jnb   BIT_3,RDTP_5          ; Si no FEND sigue en bucle

RDTP_1: mov   A,B                   ; Ver si hacer CLOSE
        jz    RDTP_2
        call  CLOSE                 ; Cierra datos en memoria
        mov   A,B                   ; Envía comando y aviso a uC2
        call  AUC2ESP
RDTP_2: ret                         ; Acaba

RDTP_5: call  PUT                   ; Escribe dato en memoria
        jb    BIT_OP,RDTP_1         ; Se ha sobrepasado limite de seg.
        sjmp  RDTP_0                ; Repite bucle

RDTP_6: call  DESECHA               ; Lee resto de trama
RDTP_7: mov   A,B                   ; Ver si hacer CLOSE
        jz    RDTP_3
        call  CLOSE
RDTP_3: mov   A,#52                 ; Código de error para diagnostico
        call  ENVINACK              ; Envía TNACK_R
        ret


;FUNCIONES DE LA APLICACION RELATIVAS AL ENVIO DE DATOS
; ---------------------------------------------------------------------

;.....................................................................
; Función:  Envía dato leído de SRAM y contenido en TESTDAT
; Entrada:  TESTDAT = Ultimo dato leído de SRAM al atender comando de
;                     test de memoria (TEST_E o TEST_L)
```

```
; Salida:
; Modif.R:  A, ULTCMD, C_CMD1, C_CMD2
; Usa:      TIME_R, EDPR, EDTP, MEMTS_R, ENVI_DAT
;.....................................................................

MEMTS_R:mov   A,#R_TEST        ; Envía cabecera de respuesta R_TEST
        mov   C_CMD1,A         ; Byte de comando duplicado
        mov   C_CMD2,A
        mov   ULTCMD,A         ; Guarda ultima trama enviada
        clr   BIT_4            ; Trama tipo D
        call  ENVI_CAB
        mov   A,TESTDAT        ; Envía dato
        call  ENVI_TR
        call  ENVI_FND         ; Envía FEND
        ret


;.....................................................................
; Función:  RNACK. Reenvía anterior comando como respuesta a TNACK_P
; Entrada:  ULTCMD: código del ultimo comando anteriormente enviado a EP
; Salida:
; Modif.R:  A
; Modif.B:
; Usa:      TIME_R,EDPR,EDTP,MEMTS_R,ENVI_DAT
;.....................................................................

RNACK:  mov   A,ULTCMD                 ; Si hay comando anterior salta
        jnz   RNACK_1
        ret

; Tratar  comandos que implica reenvío de datos

RNACK_1:cjne  A,#R_BATT,RNACK_2        ; Si no es R_BATT salta
        sjmp  RNACK_8                  ; Reenviar respuesta

RNACK_2:cjne  A,#R_RSTID,RNACK_3       ; Si no es R_RSTID salta
        sjmp  RNACK_8                  ; Reenviar respuesta

RNACK_3:cjne  A,#DAT_TR,RNACK_5        ; Si no es DAT_TR salta
        sjmp  RNACK_8                  ; Reenviar respuesta
        ret

RNACK_5:cjne  A,#R_TIME,RNACK_6        ; Si no es R_TIME salta
        call  TIME_R                   ; Reenviar respuesta
        ret

RNACK_6:cjne  A,#DATPTAR,RNACK_7       ; Si no DAT_PR con datos de tarjeta
        call  EDPR                     ; Reenviar respuesta
        ret

RNACK_7:cjne  A,#DAT_PR,RNACK_9        ; Si no DAT_PR con datos de IDS
RNACK_8:call  EDTP                     ; Reenviar respuesta
        ret

RNACK_9:cjne  A,#R_TEST,RNACK_A        ; Si no R_TEST con datos de IDS
        call  MEMTS_R                  ; Reenviar respuesta
        ret

; Tratar comandos directos como WAIT,TNACK_R...

RNACK_A:call  ENVI_DAT                 ; Envía comando que hay en A (ULTCMD)
        ret
```

```
;.....................................................................
; Función:  Anota error en diagnostico y envía una trama T_NACK
; Entrada:  A = código de mensaje a anotar en diagnostico (0 no anota)
; Salida:
; Modif.R:  A
; Modif.B:
; Usa:      ENVI_DAT,LOG_ESC
;.....................................................................

ENVINACK: jz    ENVINAC1        ; Con A=0 no anota en diagnostico
          call  LOG_ESC         ; Anota error en diagnostico
ENVINAC1: mov   A,#TNACK_R      ; Comando TNACK_R
          call  ENVI_DAT        ; Envía comando a EP
          ret

;.....................................................................
; Función:  Leer hora del reloj de la memoria SRAM y enviarla a EP
; Entrada:
; Salida:   ULTCMD: Ultimo comando enviado a EP
; Modif.D:  A, ULTCMD, DPTR, C_CMD1, C_CMD2
; Modif.B:  BIT_E
; Usa:      PIDE_RAM, ENVI_CAB, LEECLK, L_RAM, ENVI_TR, NOPCLK, D_RAM,
;           ENVI_FND, LOG_ESC
;.....................................................................

TIME_R: call  PIDE_RAM          ; Intenta pillar memoria
        jnb   BIT_4,TIME_R1     ; Si pilla memoria sigue
        ret

TIME_R1:mov   A,#R_TIME         ; Envía cabecera de respuesta R_TIME
        mov   C_CMD1,A
        mov   ULTCMD,A          ; Guarda ultima trama enviada
        mov   C_CMD2,#0         ; No hay subcomando
        call  ENVI_CAB          ; (BIT_4 = 0 y se envía trama tipo D)

; Borra indicador de error al acceder a SRAM

        clr   BIT_E

; Activar bit de lectura del reloj sin modificar el valor de calibración

        call  LEECLK            ; Activa lectura de reloj

; Leer estado del reloj

        mov   DPTR,#A_TMDIAM    ; Lee y envía día de mes
        call  L_RAM
        call  ENVI_TR

        mov   DPTR,#A_TMMES     ; Lee y envía mes
        call  L_RAM
        call  ENVI_TR

        mov   DPTR,#A_TMANO     ; Lee y envía ano
        call  L_RAM
        call  ENVI_TR

        mov   DPTR,#A_TMHORA    ; Lee y envía hora
        call  L_RAM
        call  ENVI_TR

        mov   DPTR,#A_TMMINU    ; Lee y envía minutos
        call  L_RAM
```

```
        call  ENVI_TR

        mov   DPTR,#A_TMSEGU  ; Lee segundos
        call  L_RAM
        clr   ACC.7           ; Pone a cero MSB (bit de STOP)
        call  ENVI_TR         ; Envía segundos

; Desactivar bit de lectura del reloj sin modificar el valor de calibración

        call  NOPCLK          ; Desactiva lectura y escritura

        call  D_RAM
        call  ENVI_FND        ; Envía FEND final

; Anotar error de acceso a SRAM si lo hubo

        jnb   BIT_E,TIME_R2

        mov   A,#26
        call  LOG_ESC

TIME_R2:  ret                 ; Acaba


;.....................................................................
; Función:  Trata comando DAT_PP de la EP. Envía datos de parametr.
;           de la tarjeta.
; Entrada:
; Salida:
; Modif.D:  A, B, DPTR, ULTCMD, C_CMD1, C_CMD2
; Modif.B:  BIT_E
; Usa:      PIDE_RAM, ENVI_CAB, L_RAM, ENVI_TR, LOG_ESC, ENVI_FND
;.....................................................................

EDPR:   call  PIDE_RAM        ; Intenta pillar memoria
        jnb   BIT_4,EDPR_1    ; Si pilla memoria sigue
        ret

; Enviar datos de param. de la tarjeta a la EP

EDPR_1: mov   C_CMD1,#DAT_PR  ; Envía cabecera de respuesta DAT_PR
        mov   A,#DATPTAR      ; Subcomando: parámetros tarjeta
        mov   C_CMD2,A
        mov   ULTCMD,A        ; Guarda ultima trama enviada diferente de
                              ; DAT_PR para distinguirla en reenvíos
        call  ENVI_CAB        ; (BIT_4 es 0, para trama tipo D)

        mov   B,#10           ; Numero de datos a enviar
        mov   DPTR,#A_PAR     ; Puntero al área de parámetros
        clr   BIT_E           ; Borra indicador de error

EDPR_2: call  L_RAM           ; Lee dato
        call  ENVI_TR         ; Envía dato
        inc   DPTR            ; Apunta a siguiente dirección
        djnz  B,EDPR_2        ; Si quedan datos por enviar sigue

        call  D_RAM
        jnb   BIT_E,EDPR_2B   ; Si no hubo error salta
        mov   A,#24           ; Anota error
        call  LOG_ESC

EDPR_2B:call  ENVI_FND        ; Envía FEND final
        ret
```

```
;.....................................................................
; Función:  Tratar respuesta del uC2 para la EP. Envía respuesta
;           con datos transparentes de los comandos R_BATT, R_RSTID, DAT_TP
;           No se usa CLOSE para no alterar la longitud de datos en memoria
; Entrada:  A = código del comando de respuesta
; Salida:
; Modif.R:  A, DPTR, B, ULTCMD, C_CMD_1, C_CMD2
; Modif.B:  BIT_4
; Usa:      ENVI_CAB, GET, RESTA, ENVI_TR, ENVI_FND
;.....................................................................

EDTP:    mov   ULTCMD,A          ; Guarda ultimo comando enviado
         mov   C_CMD1,A          ; Transmitir trama tipo A
         mov   C_CMD2,#0         ; En principio no hay subcomando
         cjne  A,#DAT_PR,EDTP_0  ; Si no es DAT_PR no envía subcomando
         mov   C_CMD2,#DATPIDS   ; Envía subcomando: datos param. de IDS

EDTP_0:  clr   BIT_4             ; Trama tipo D
         call  ENVI_CAB

         call  OPEN              ; Abre datos para leer
         mov   DPL,#1            ; Inicia contador datos enviados a 1
         mov   DPH,#0

EDTP_1:  call  GET               ; Coge dato
         jb    BIT_OP,EDTP_4     ; Si no hay mas datos acaba
         mov   B,A               ; Guarda dato

         inc   DPTR              ; Incrementa contador de datos enviados
         mov   ARIT_L,#0         ; Ver si el dato cabe en trama
         mov   ARIT_H,#1         ;    (como mucho caben 256 bytes)
         call  RESTA
         J.C.    EDTP_3            ; Si no cabe pasa a enviar nueva trama

         mov   A,B               ; Recupera dato
         call  ENVI_TR           ; Transmite dato transparente a EP
         sjmp  EDTP_1            ; Repite mientras haya datos

EDTP_3:  call  ENVI_FND          ; Envía FEND final

         mov   A,ULTCMD          ; Transmitir cabecera trama del tipo
                                 ; adecuado
         mov   C_CMD1,A
         mov   C_CMD2,#0         ; No hay subcomando
         clr   BIT_4             ; Trama tipo D
         call  ENVI_CAB

         mov   DPL,#1            ; Inicia contador datos enviados a 1
         mov   DPH,#0
         mov   A,B               ; Recupera ultimo dato
         inc   DPTR              ; Incrementa numero de bytes
         call  ENVI_TR           ; Envía último dato que no cabía antes
         sjmp  EDTP_1            ; Sigue con nueva trama

EDTP_4:  call  ENVI_FND          ; Envía FEND final
         ret                     ; Acaba

;.....................................................................
; Función:  Envía tramas con datos de aviso (DAT_AV).
;           No se usa CLOSE para no alterar la longitud de datos en memoria
; Entrada:
; Salida:   BIT_4=1 si error y no logrado el envío tras los intentos
```

```
;           BIT_0=1 si disconnect recibido
;           BIT_1=1 si sondeo recibido
; Modif.R:  R5, DPTR, A, B, ARIT_L, ARIT_H, RXCLR, OPEN, GET, ENVI_CAB,
;          ENVI_TR, RECI_CAB, RESTA, RECI_DAT, ENVI_DAT, ENVI_FND, DLY20,
;          LOG_ESC,ENVINACK, C_CMD1, C_CMD2
; Modif.B:  BIT_4, BIT_0, BIT_1
; Uso de registros:
; DPTR:          numero de datos enviados en una trama
; R5:            numero de intentos restantes
;.....................................................................

ENVI_DAV: mov   R5,#INTENT              ; Intentos de transmisión

ENVI_DA1: call  RXCLR                   ; Borrar buffer recepción

          call  OPEN                    ; Abrir fichero en memoria

ENVI_DA2: mov   C_CMD1,#DAT_AV          ; Envía cabecera de DAT_AV
          mov   C_CMD2,#0
          clr   BIT_4                   ; Trama tipo D
          call  ENVI_CAB

          mov   DPL,#1                 ; Inicia contador datos enviados a 1
          mov   DPH,#0

ENVI_DA3: call  GET                    ; Coge dato (si quedan)
          jb    BIT_OP,ENVI_DA5        ; Si no quedan salta
          mov   B,A                    ; Guarda dato

          inc   DPTR
          mov   ARIT_L,#0              ; Ver si el dato cabe en trama
          mov   ARIT_H,#1              ;   (como mucho caben 256 bytes)
          call  RESTA
          jc    ENVI_DA4               ; Si no cabe salta a enviar FEND

          mov   A,B                    ; Recupera dato
          call  ENVI_TR                ; Si cabe en trama lo envía
          sjmp  ENVI_DA3               ; Vuelve al bucle

ENVI_DA4: call  ENVI_FND               ; Envía FEND final

          call  DLY20

          mov   C_CMD1,#DAT_AV         ; Envía cabeceRa de DAT_AV
          mov   C_CMD2,#0
          clr   BIT_4                  ; Trama tipo D
          call  ENVI_CAB

          mov   DPL,#1                 ; Inicia contador datos enviados a 1
          mov   DPH,#0
          mov   A,B                    ; Recupera ultimo dato
          inc   DPTR                   ; Incrementa numero de bytes
          call  ENVI_TR                ; Envía último dato que no cabía antes
          sjmp  ENVI_DA3               ; Enviar otra trama

ENVI_DAX: sjmp  ENVI_DA1               ; Para un salto largo

ENVI_DA5: call  ENVI_FND               ; Envía FEND final

; Recibir respuesta de EP

ENVI_DA6: call  RECI_CAB               ; Espera respuesta
          jb    BIT_0,ENVI_DA10        ; Si error de timeout salta
          jb    BIT_1,ENVI_DA8         ; Si trama mal conformada salta
```

```
        jb     BIT_CM1,ENVI_DA6        ; Si es trama de comando ignora
        jb     BIT_CM2,ENVI_DA9        ; Si es trama de sistema salta

; Se recibe un ACK_AV

        cjne   A,#ACK_AV,ENVI_DA7      ; Si no ACK_AV sigue

        call   RECI_DAT                ; Recibe resto
        jb     BIT_4,ENVI_DA6          ; Salta si error

        mov    A,#C_MEDIO              ; Envía respuesta EP
        call   ENVI_DAT
        clr    BIT_4                   ; Final correcto
        ret

; Se recibe un TNACK_P

ENVI_DA7: cjne  A,#TNACK_P,ENVI_DA11    ; Si no TNACK_P sigue

        call   RECI_DAT                ; Recibe resto
        jb     BIT_4,ENVI_DA6          ; Salta si error

        mov    A,#70
        call   LOG_ESC                 ; Anota mensaje en diagnostico

        djnz   R5,ENVI_DAX             ; Otro intento si quedan

; Si ocurre timeout en la espera de respuesta

ENVI_DA10: setb  BIT_4                 ; Indica error
        clr    BIT_0                   ; No disconnect recibido
        clr    BIT_1                   ; No sondeo recibido
        ret

; Se recibe un SONDEO

ENVI_DA11: cjne  A,#SONDEO,ENVI_DA8    ; Si no SONDEO sigue

        call   RECI_DAT                ; Recibe resto
        jb     BIT_4,ENVI_DA6          ; Salta si error

        setb   BIT_4                   ; Indica error
        clr    BIT_0                   ; No disconnect recibido
        setb   BIT_1                   ; SONDEO recibido
        ret

; Se recibe comando no esperado

ENVI_DA8: mov   A,#62                  ; Código de error para diagnostico
        call   ENVINACK
        sjmp   ENVI_DA6                ; Espera otra respuesta

; Se recibe trama de sistema

ENVI_DA9: jb    BIT_3,ENVI_DAZ ; Si trama S no valida salta
        jb     BIT_4,ENVI_DAZ  ; Si no es trama S disconnect salta

        setb   BIT_4           ; Indica error
        setb   BIT_0           ; Indica disconect
        clr    BIT_1           ; No sondeo recibido
        ret

ENVI_DAZ: jmp   ENVI_DA6       ; Sigue en espera de confirmación
```

```
;.....................................................................
; Función:  Envía datos de ficheros en grupos de 4 tramas, seguidas
;        de una tramas con SVB, esperando confirmaciones. No se usa
;        CLOSE para no alterar la longitud de datos en memoria
; Entrada:CABEZA,comando para las tramas de datos (DAT_FI,R_CIDS,R-FDIAG)
; Salida:   BIT_4=1 si error y no logrado el envío tras los intentos.
;           En tal caso se devuelve mas información:
;             BIT_0=1 si disconnect recibido
;             BIT_1=1 si SONDEO recibido
; Modif.R: R0 a R7, DPTR, A, B, VAR_SVB, PTR_OPL, PTR_OPH, C_CMD1, C_CMD2
; Modif.B:  BIT_4, BIT_2, BIT_1, BIT_0
; Usa:      OPEN, RXCLR, ENV_CAB, ENVI_TR, GET, RESTA, ENVI_FND, DLY20,
;           RECI_CAB, RECI_DAT, ENVI_DAT, LOG_ESC, AUC2DIR, ENVINACK
;
; Uso de las variables y registros:
; R1,R0:        bytes restantes al inicio para bloque de 4 tramas actual
; R3,R2:        bytes restantes en el bloque actual
; DPTR:         numero de datos enviados en una trama
; R4:           numero de trama en un bloque de 4 tramas
; VARSVB        SVB calculada
; R5:           numero de intentos restantes
; R7,R6:        puntero a datos para inicio de bloque actual
; BIT_2:        indica primer grupo de 4 tramas por enviar
;.....................................................................

ENVI_DFI: mov   R5,#INTENT          ; Intentos de transmisión
        setb  BIT_2                 ; Primer grupo de 4 tramas por enviar
        call  OPEN                  ; Abrir fichero en memoria
        mov   R0,LON_L              ; Bytes por leer al inicio de bloque
        mov   R1,LON_H
        mov   R6,PTR_OPL            ; Inicia puntero para inicio de bloque
        mov   R7,PTR_OPH

ENVI_DF0: mov   R4,#0               ; Inicia numero de trama del cada 4
        mov   VARSVB,#0             ; Inicia SVB a 0
        mov   A,R0                  ; Inicia bytes restantes en bloque
        mov   R2,A
        mov   A,R1
        mov   R3,A
        mov   PTR_OPL,R6            ; Puntero de datos a inicio de bloque
        mov   PTR_OPH,R7

ENVI_DF1: call  RXCLR               ; Borrar buffer recepción

        mov   DPL,#1                ; Inicia contador datos enviados a 1
        mov   DPH,#0

        mov   A,CABEZA              ; Envía cabecera de DAT_FI, R_FDIAG
        mov   C_CMD1,A              ; o R_CIDS según contenido de CABEZA
        mov   C_CMD2,#0
        clr   BIT_4                 ; Trama tipo D
        call  ENVI_CAB

        jnb   BIT_2,ENVI_DF2        ; Si no primer bloque de tramas salta
        mov   A,R4                  ; Si no primera trama de bloque salta
        jnz   ENVI_DF2

        mov   A,LON_L               ; En primera trama envía longitud
        inc   DPTR                  ;   del fichero en RAM. Esta se
        call  ENVI_TR               ;   incluye en el calculo de SVB
        mov   A,LON_L
        add   A,VARSVB
```

```
        mov   VARSVB,A

        mov   A,LON_H
        inc   DPTR
        call  ENVI_TR
        mov   A,LON_H
        add   A,VARSVB
        mov   VARSVB,A

ENVI_DF2: mov   A,R2                ; Salta si 0 bytes restantes en R3,R2
        orl   A,R3
        jz    ENVI_DF4

        call  GET                   ; Coge dato
        jb    BIT_OP,ENVI_DFX       ; Si no quedan salta (No se debe dar)
        mov   B,A                   ; Guarda dato

        inc   DPTR
        mov   ARIT_L,#0             ; Ver si el dato cabe en trama
        mov   ARIT_H,#1             ;   (como mucho caben 256 bytes)
        call  RESTA
        jc    ENVI_DF3              ; Si no cabe salta a enviar FEND

        mov   A,VARSVB              ; Recupera SVB
        add   A,B                   ; Suma dato
        mov   VARSVB,A              ; Guarda SVB

        mov   A,B                   ; Recupera dato
        call  ENVI_TR               ; Si cabe en trama lo envía

        clr   CY                    ; Decrementa bytes restantes
        mov   A,R2
        subb  A,#1
        mov   R2,A
        mov   A,R3
        subb  A,#0
        mov   R3,A

        sjmp  ENVI_DF2                ; Vuelve al bucle

ENVI_DF3: clr   CY                    ; Decrementa puntero a datos para
        mov   A,PTR_OPL               ;   apuntar a dato no enviado
        subb  A,#1
        mov   PTR_OPL,A
        mov   A,PTR_OPH
        subb  A,#0
        mov   PTR_OPH,A

        inc   R4                      ; Incrementa contador de tramas
        mov   A,#4
        xrl   A,R4
        jz    ENVI_DF4                ; Si se han enviado 4 tramas salta

        call  ENVI_FND                ; Envía FEND final

        call  DLY20

        jmp   ENVI_DF1                ; Enviar otra trama

ENVI_DFX: clr   A
        mov   R2,A                    ; Para caso de sobrepasar TOP_DAT
        mov   R3,A                    ; considera todos datos enviados

ENVI_DF4: call  ENVI_FND              ; Envía FEND final
```

```
        mov   C_CMD1,#SVB              ; Enviar trama SVB
        mov   C_CMD2,#0
        clr   BIT_4                    ; Trama tipo D
        call  ENVI_CAB

        mov   A,VARSVB                 ; Enviar código de la SVB
        call  ENVI_TR
        call  ENVI_FND                 ; Envía FEND final

; Recibir respuesta de EP

ENVI_DF5: call  RECI_CAB               ; Espera respuesta
        jb    BIT_0,ENVI_DFB           ; Si error de timeout salta
        jb    BIT_1,ENVI_DFA           ; Si trama mal conformada salta
        jb    BIT_CM1,ENVI_DF5         ; Si es trama C ignora
        jb    BIT_CM2,ENVI_DFC         ; Si es trama S salta

; Se recibe un ACK_FI

        cjne  A,#ACK_FI,ENVI_DF7       ; Si no ACK_FI sigue

        call  RECI_DAT                 ; Recibe resto
        jb    BIT_4,ENVI_DF5           ; Salta si error

        mov   A,R2                   ; Salta si 0 bytes restantes en R3,R2
        orl   A,R3
        jz    ENVI_DF6

        mov   R6,PTR_OPL             ; Inicia puntero para inicio de bloque
        mov   R7,PTR_OPH
        mov   A,R2                   ; Inicia bytes restantes en bloque
        mov   R0,A
        mov   A,R3
        mov   R1,A
        clr   BIT_2                    ; Ya no es primer grupo de 4 tramas
        mov   R4,#0                    ; Inicia numero de trama del cada 4
        mov   VARSVB,#0                ; Inicia SVB a 0
        mov   R5,#INTENT               ; Intentos de transmisión

        jmp   ENVI_DF1                 ; Enviar otra trama

; Final correcto de la rutina

ENVI_DF6: mov   A,#C_MEDIO             ; Envía respuesta a ACK_AV
        call  ENVI_DAT
        clr   BIT_4                    ; Final correcto
        ret

; Se recibe un NACK_FI

ENVI_DF7: cjne  A,#NACK_FI,ENVI_DF8    ; Si no NACK_FI sigue

        call  RECI_DAT                 ; Recibe resto
        jb    BIT_4,ENVI_DF5           ; Salta si error

ENVI_DFD: djnz  R5,ENVI_DFZ            ; Si quedan intentos repite envío
        sjmp  ENVI_DFB                 ; Salta al final de la rutina

; Se recibe un TNACK_P

ENVI_DF8: cjne  A,#TNACK_P,ENVI_DF9    ; Si no TNACK_P sigue

        call  RECI_DAT                 ; Recibe resto
```

```
        jb    BIT_4,ENVI_DF5          ; Salta si error

        mov   A,#71
        call  LOG_ESC                 ; Anota mensaje en diagnostico
        sjmp  ENVI_DFD

; Se recibe un SONDEO

ENVI_DF9: cjne  A,#SONDEO,ENVI_DFR    ; Si no SONDEO sigue

        call  RECI_DAT                ; Recibe resto
        jb    BIT_4,ENVI_DF5          ; Salta si error

        setb  BIT_4
        clr   BIT_0                   ; No disconnect recibido
        setb  BIT_1                   ; SONDEO recibido
        ret

; Atender REP_FI de la EP: se envían a uC2

ENVI_DFR: cjne  A,#REP_FI,ENVI_DFA
        call  AUC2DIR                 ; Pasa REP_FI a uC2

        clr   BIT_4                   ; No hay errores
        ret

; Para saltos largos al inicio del bucle

ENVI_DFZ: jmp   ENVI_DF0

; Si ocurre timeout en la espera de respuesta

ENVI_DFB: setb  BIT_4                 ; Indica error
        clr   BIT_0                   ; No disconnect recibido
        clr   BIT_1                   ; No SONDEO recibido
        ret

; Se recibe comando no esperado o trama mal conformada

ENVI_DFA: mov   A,#63                 ; Código de error para diagnostico
        call  ENVINACK
        jmp   ENVI_DF0                ; Reenvía las ultimas 4 tramas

; Se recibe trama de sistema

ENVI_DFC: jb    BIT_3,ENVI_DFW        ; Si trama S no valida salta
        jb    BIT_4,ENVI_DFW          ; Si no es trama S disconnect salta

        setb  BIT_4                   ; Indica error y acaba
        setb  BIT_0                   ; Si disconnect acaba con BIT_4=1
        clr   BIT_1                   ; No sondeo recibido
        ret

ENVI_DFW: jmp   ENVI_DF5              ; Sigue en espera de confirmación
```

```
;-----------------------------------------------------------------------
;                    FUNCIONES DE RED
;-----------------------------------------------------------------------

;FUNCIONES DEL NIVEL DE SESION_ENLACE LOGICO
;-----------------------------------------------------------------------+

;.......................................................................
; Función:  Solicitar conexión a la estación principal. Para ello se
;           envía una trama C con datos "c centro", y se espera la
;           recepción de una trama S con "connect" procedente de la EP
; Entrada:
; Salida:
; Modif.R:  DPTR, A
; Modif.B:
; Usa:      ENVI_COM, RECI_CAB, TESPERA, DLY20, ENVI_DAT, DESECHA
;.......................................................................

CAD_CNC:  db    'c centro',0  ; Cadena de datos para solicitar conexión

SOLI_CON: mov   DPTR,#CAD_CNC ; Cadena para solicitar conexión
          call  ENVI_COM      ; Envía un connect

SOLI_3:   call  RECI_CAB      ; Espera recibir trama
          jb    BIT_0,SOLI_4X   ; Salta si timeout en respuesta
          jb    BIT_1,SOLI_4X   ; Si error trama mal conformada
          jb    BIT_CM1,SOLI_3A ; Si es trama C salta a desechar
          jnb   BIT_CM2,SOLI_3A ; Si no es trama S (es trama D) salta

; Ver si ha llegado disconnect de respuesta al envío de connect de aviso

          jb    BIT_3,SOLI_3    ; Salta si trama S no valida
          jb    BIT_4,SOLI_7    ; Salta sí trama S connect

SOLI_4X:  call  TESPERA         ; Espera 10seg o llegada de trama
          jnb   BIT_RRDY,SOLI_CON ;Si no ha recibido salta a enviar connect

; Recibir petición de conexión de la EP

SOLI_5:   call  RECI_CAB        ; Recibe respuesta
          jb    BIT_0,SOLI_4X   ; Salta si timeout en respuesta
          jb    BIT_1,SOLI_4X   ; Si error trama mal conformada
          jb    BIT_CM1,SOLI_5A ; Si es trama C
          jnb   BIT_CM2,SOLI_5A ; Si es no trama S (es trama D)

          jb    BIT_3,SOLI_4X   ; Si trama S no esperada vuelve al bucle
          jnb   BIT_4,SOLI_4X   ; Si trama S Disconect vuelve al bucle

; Recibido Connect

SOLI_7:   call  DLY20           ; Espera un poco
          mov   A,#SESION_R     ; Confirma aceptación de sesión a EP
          call  ENVI_DAT
          ret                   ; Acaba

; Desechar tramas de comando o de datos

SOLI_3A:  call  DESECHA         ; Si no es trama de sistema se ignora
          sjmp  SOLI_3

SOLI_5A:  call  DESECHA         ; Si no es trama de sistema se ignora
```

```
        sjmp  SOLI_5

;.....................................................................
; Función:  Recibe resto de una trama D que contiene un comando corto.
;           Ya ha recibido la cabecera y el primer byte de comando con
;           RECI_CAB.
; Entrada:  A = Comando que se espera (el del primer byte)
; Salida:   BIT_4 = 1 trama mal conformada (bytes no validos o timeout)
; Modif.R:  A (solo si hay error), TMP
; Modif.B:  BIT_4
; Usa:      RXB, DESECHA, ENVINACK
;.....................................................................
RECI_DAT: mov   TMP,A                  ; Guarda comando esperado
        call  RXB                      ; Comprueba siguiente byte
        jnb   BIT_RTO,RECI_CM2         ; Si timeout salta
        cjne  A,TMP,RECI_CM1           ; Si siguiente no es esperado salta

        call  RXB                      ; Espera FEND
        jnb   BIT_RTO,RECI_CM2         ; Si timeout salta
        cjne  A,#FEND,RECI_CM1         ; Si no FEND salta

        mov   A,TMP                    ; Recupera comando y acaba
        clr   BIT_4
        ret

; Segundo byte incorrecto, no se encontró FEND final o timeout entre bytes
RECI_CM1: call  DESECHA
RECI_CM2: mov   A,#53                  ; Código de error para diagnostico
        call  ENVINACK
        setb  BIT_4
        ret

;.....................................................................
; Función:  Analiza los campos control y canal de la SDU suministrada por
;           el nivel NCA. Si CC = D se analiza el primer byte de datos
;           del campo DA,. Si CC = S se determina si se trata de una
;           trama Connect o Disconnet.
; Entrada:
; Salida:   BIT_0 = 1 si timeout al esperar FEND inicial
;           BIT_1 = 1 si trama mal conformada (bytes no validos o timeout
;               entre los bytes de la trama)
;           BIT_CM1 BIT_CM2  Tipo de trama
;              0       0        Trama D
;              1       0        Trama C
;              0       1        Trama S
;           Si trama S: BIT_3 = 1 si trama no esperada, 0 si es valida
;                       BIT_4 = 1 si Connect, 0 si disconnect
;           Si trama D: A = Código comando (byte 0 datos)
; Modif.R:  A
; Modif.B:  BIT_0, BIT_1, BIT_CM1, BIT_CM2, BIT_CM3
; Usa:      RECI_CNCA, DESECHA, LOG_ESC
;.....................................................................

RECI_CAB: clr   BIT_CM1                ; De momento considera que la
        clr   BIT_CM2                  ;   trama es tipo D

        call  RECI_CNCA                ; Recibe cabecera de nivel inferior
        jb    BIT_0,RECI_CB1           ; Salta si timeout de trama
        jb    BIT_1,RECI_CB1           ; Salta si error entre bytes

; Recibida una trama C (trama FC00...)

        mov   A,CR_CONTR
        cjne  A,#'C',RECI_CB2          ; Salta si no se trata de trama C
```

```
        mov   A,CR_CANAL
        cjne  A,#'0',RECI_CB4          ; Las tramas C deben llevar canal 0
                                         ;  y salta si el canal no es 0
        setb  BIT_CM1                  ; Indica que se ha recibido trama C
        ret

; Error al recibir la cabecera de nivel inferior

RECI_CB1: ret                     ; Acaba con BIT_0 y BIT_1 ya actualizados

; Recibida una trama D

RECI_CB2: cjne  A,#'D',RECI_CB3        ; Salta si no se trata de trama D
        mov   A,CR_CANAL
        cjne  A,#CANAL,RECI_CB4        ; Salta si el canal no es valido

        mov   A,CR_CMD                 ; Recupera el primer byte de datos
        ret                            ;   (comando) y acaba

; Recibida una trama S

RECI_CB3: cjne  A,#'S',RECI_CB4        ; Salta si no se trata de trama S
        mov   A,CR_CANAL
        cjne  A,#CANAL,RECI_CB4        ; Salta si el canal no es valido

        call  RECI_CNC                 ; Comprueba tipo de trama S, actua-
                                  ;   lizando BIT_1, BIT_3 y BIT_4
                                  ;   según corresponda
        setb  BIT_CM2                  ; Indica que es trama S
        ret

; Error de trama errónea

RECI_CB4: call  DESECHA                ; Lee resto de trama
        mov   A,#55                    ; Mensaje error de trama
        call  LOG_ESC                  ; Anota mensaje en diagnostico
        setb  BIT_1                    ; Indica error de trama
        ret

;.....................................................................
; Función:  Comprueba si una trama de sistema recibida indica un comando
;           Connect, Disconect u otro.
; Entrada:
; Salida:   BIT_1 = 1 si error al recibir bytes de trama, 0 si no error
;           BIT_3 = 0 si Connenct o Disconnect, 1 otro comando
;           BIT_4 = 1 si Connect, 0 si Disconnect
; Modif.R:  A
; Modif.B:  BIT_1, BIT_3, BIT_4
; Usa:      RECITR, AMAYS, DESECHA, LOG_ESC, RXB
;.....................................................................

; Recibir primer byte

RECI_CNC: call  RECI_X          ; Recibe dato transparente

; Tratar respuesta a conexión: ...CON...

RECI_CN0: cjne  A,#'C',RECI_CN1 ; Si byte no es 'C' salta

        lcall RECI_X          ; Recibe dato transparente
        cjne  A,#'O',RECI_CN0 ; Si byte no es 'O' salta

        lcall RECI_X          ; Recibe dato transparente
        cjne  A,#'N',RECI_CN0 ; Si byte no es 'N' salta
```

```
        clr   BIT_1            ; Recepción correcta
        clr   BIT_3
        setb  BIT_4            ; Recibido connect
        sjmp  RECI_CN3         ; Salta a leer resto de trama

; Tratar respuesta a desconexión: ...DIS...

RECI_CN1: cjne  A,#'D',RECI_CN2 ; Si byte no es 'D' salta

        lcall RECI_X           ; Recibe dato transparente
        cjne  A,#'I',RECI_CN0 ; Si byte no es 'I' salta

        lcall RECI_X           ; Recibe dato transparente
        cjne  A,#'S',RECI_CN0 ; Si byte no es 'S' salta

        clr   BIT_0            ; Recepción correcta
        clr   BIT_3
        clr   BIT_4            ; Recibido disconnect
        sjmp  RECI_CN3         ; Salta a leer resto de trama

; Tratar respuesta por conexión radio no posible: ...RET...

RECI_CN2: cjne  A,#'R',RECI_CNC ; Si byte no es 'R' salta

        lcall RECI_X           ; Recibe dato transparente
        cjne  A,#'E',RECI_CN0 ; Si byte no es 'E' salta

        lcall RECI_X           ; Recibe dato transparente
        cjne  A,#'T',RECI_CN0 ; Si byte no es 'T' salta

        setb  BIT_0           ; Recepción correcta, pero trama no esperada
        setb  BIT_3

; Leer resto de trama

RECI_CN3: call  RXB            ; Lee otro dato
        jnb   BIT_RTO,RECI_CN5 ; Salta si hay error de timeout
        xrl   A,#FEND          ; Ver si dato es FEND
        jnz   RECI_CN3         ; Sigue en bucle si no es FEND
        ret                    ; Acaba

RECI_CN5: mov   A,#54          ; Trama sistema mal conformada
        call  LOG_ESC          ; Anota mensaje en diagnostico
        setb  BIT_0            ; Indica trama no valida y acaba
        ret

; Recibir datos transparentes y gestionar errores: si no hay errores
; retorna a donde se llamo y si hay errores activa BIT_0 y vuelve al
; punto en donde se llamo a la rutina que llamo a RECI_X

RECI_X: call  RECITR           ; Recibe dato transparente
        call  AMAYS            ; Pasa byte a mayúsculas para comparar
        jb    BIT_0,RECI_X8    ; Si hay timeout salta para acabar
        jb    BIT_1,RECI_X7    ; Si hay error en la transparencia salta
        jb    BIT_3,RECI_X8    ; Si se lee FEND no esperado aun salta
        ret

; Si hay error de datos o de timeout

RECI_X7:  call  DESECHA
RECI_X8:  mov   A,#54          ; Trama sistema mal conformada

RECI_X9:  call  LOG_ESC        ; Anota mensaje en diagnostico
```

```
        setb  BIT_1          ; Indica trama no valida y acaba
        dec   SP             ; Sacar de pila dirección previa
        dec   SP
        ret                  ; Regresar a rutina que llamo a RECI_CNC


;.....................................................................
; Función:  Envía trama de comando (C)
; Entrada:  DPTR apunta a la cadena a enviar con la información a enviar
;           en el campo de datos, acabada en 0
; Salida:   C_CMD1 = 0 (no enviar comando en campo de datos)
; Modif.R:  A, DPTR, C_CMD1
; Modif.B:  BIT_4
; Usa:      ENVI_CAB, ENVI_TR, ENVI_FND
;.....................................................................

ENVI_COM: mov   C_CMD1,#0       ; No hay código de comando
        setb  BIT_4
        call  ENVI_CAB          ; Envía cabecera trama comando (BIT_4=1)

ENVI_CO0: clr   A
        movc  A,@A+DPTR         ; Lee siguiente byte de texto
        jz    ENVI_CO1          ; Si dato es 0 acaba
        call  ENVI_TR           ; Envío del byte transparente
        inc   DPTR
        jmp   ENVI_CO0          ; Mientras quede mensaje repite

ENVI_CO1: call  ENVI_FND        ; Envía FEND final
        ret

;.....................................................................
; Función:  Envía una trama tipo D con un comando corto (comando repetido
;           dos veces en el campo de datos)
; Entrada:  A = código del comando (no puede ser código especial como un ; FEND)
; Salida:   ULTCMD = código del comando enviado
;           C_CMD1, C_CMD2 = comando a enviar
; Modif.R:  C_CMD1, C_CMD2, ULTCMD, A
; Modif.B:
; Usa:      ENVI_CAB, ENVI_FND
;.....................................................................
ENVI_DAT: mov   ULTCMD,A        ; Guarda ultimo comando enviado
        mov   C_CMD1,A          ; Envía cod. de comando dos veces
        mov   C_CMD2,A
        clr   BIT_4             ; Cabecera de trama normal
        call  ENVI_CAB          ; Envía trama
        call  ENVI_FND          ; ... y envía el FEND final
        ret

;.....................................................................
; Función:  ENVI_CAB. Envía un SDU al nivel NCA con parámetros CC,canal.
;           Equivale a un DATA.request del nivel Sesión-LL al NCA.
; Entrada:  BIT_4 = 1 para enviar trama C, o 0 para trama D
; Salida:   C_CONTR = valor de campo de control para nivel inferior
;           C_CANAL = valor de campo de canal para nivel inferior
; Modif.R:  C_CONTR, C_CANAL
; Modif.B:
; Usa:      ENVI_CNCA
;.....................................................................
ENVI_CAB: mov   C_CONTR,#'D'    ; Envía tipo 'D' en principio
        jnb   BIT_4,ENVI_CB1    ; Si hay que enviar D salta y no carga C
        mov   C_CONTR,#'C'
```

```
ENVI_CB1: mov   C_CANAL,#CANAL        ; Canal
          call  ENVI_CNCA            ; Envía cabecera a nivel inferior
          ret


; FUNCIONES DEL NIVEL DE ACCESO A LA SUBRED (NCA)
;-----------------------------------------------------------------------

;.......................................................................
; Función:  Recibe datos serie hasta encontrar un FEND que delimite
;           el final de la trama.
; Entrada:  A = ultimo byte leído de la trama antes de llamar a esta
;           rutina, para comprobar que el FEND final no ha llegado ya
; Salida:   BIT_4 = 1 si timeout al esperar recepción
; Modif.R:  A
; Modif.B:  BIT_4
; Usa:      RXB
;.......................................................................

DESECHA:  xrl   A,#FEND          ; Ver si ultimo dato es FEND
          jz    DESECH_0         ; Si es FEND acaba
          call  RXB              ; Si no es FEND lee otro
          jb    BIT_RTO,DESECHA  ; Repite bucle si no timeout
          setb  BIT_4            ; Si timeout acaba devolviendo 1
          ret
DESECH_0: clr   BIT_4            ; Acaba devolviendo 0
          ret

;.......................................................................
; Función:  Recibe un dato quitando la transparencia
; Entrada:
; Salida:   A = dato recibido.
;           BIT_0=1 si timeout
;           BIT_1=1 si dato erróneo
;           BIT_3=1 si se ha recibido un FEND final antes de
;                   transparencia
; Modif.R:  A
; Modif.B:  BIT_0,BIT_1,BIT_3
; Usa:      RXB
;.......................................................................

RECITR:   clr   BIT_0                 ; No hay errores de momento
          clr   BIT_1
          clr   BIT_3                 ; En principio no hay FEND final
          call  RXB                   ; Recibe dato
          jnb   BIT_RTO,RECITR_4      ; Salta si timeout
          cjne  A,#FEND,RECITR_0      ; Si no FEND final sigue
          setb  BIT_3                 ; Acaba indicando FEND final
          ret

RECITR_0: cjne  A,#FESC,RECITR_2      ; Si no es FESC final correcto
          call  RXB                   ; Si es FESC lee siguiente dato
          jnb   BIT_RTO,RECITR_4      ; Salta si timeout
          cjne  A,#TFESC,RECITR_1     ; Si siguiente no es TFESC sigue
          mov   A,#FESC               ; Si siguiente es TFESC dato es FESC
          ret                         ; Final correcto

RECITR_1: cjne  A,#TFEND,RECITR_3     ; Si siguiente no es TFEND sigue
          mov   A,#FEND               ; Si siguiente es TFEND dato es FEND
RECITR_2: ret

RECITR_3: setb  BIT_1                 ; Error por dato no esperado
```

```
        ret
RECITR_4: setb  BIT_0                   ; Error por timeout
        ret

;.....................................................................
; Función:  Recibe inicio y cabecera de una trama, reconociendo el FEND
;           inicial, los campos de control, puerto y canal. En tramas tipo
;           D se lee también el primer byte, que representa un comando.
;           Verifica que el puerto recibido es el esperado
; Entrada:
; Salida:   BIT_0 = 1 si timeout al esperar primer byte (FEND)
;           BIT_1 = 1 si trama mal conformada (puerto no esperado, bytes
;               no validos o timeout entre bytes)
;           CR_CONTR = el campo de control leído (C, D o S)
;           CR_CANAL = el campo de canal leído
;           CR_CMD = Código comando (byte 0 datos) si la trama es tipo D
; Modif.R:  A, B, CR_CANAL, CR_CONTR
; Modif.B:  BIT_0, BIT_1
; Usa:      RXB, DESECHA, LOG_ESC
;.....................................................................

RECI_CNCA: clr   BIT_0                  ; Recepción correcta de momento
        clr   BIT_1

        mov   A,R0                      ; Guarda valor original de R0
        push  ACC

        mov   R0,#5                     ; Inicia contadores de ...
RECI_CA1: mov   B,#255                  ; ... intentos para espera de FEND

; Leer FEND inicial

RECI_CA0: call  RXB                     ; Leer byte 0 de trama HOST: FEND
        jnb   BIT_RTO,RECI_CA4          ; Salta si timeout
        cjne  A,#FEND,RECI_CA5          ; Salta si no es FEND

; Leer campo de control (C, D o S)

        call  RXB                       ; Leer byte 1 de trama HOST: control
        jnb   BIT_RTO,RECI_CA6          ; Salta si timeout
        mov   CR_CONTR,A                ; Guarda byte de control

; Leer y verificar puerto (para D y S es PUERTO, y para C es 0)

        cjne  A,#'C',RECI_CA8           ; Salta si la trama no es tipo C

        call  RXB                       ; Leer byte 2 de trama HOST: puerto
        jnb   BIT_RTO,RECI_CA6          ; Salta si timeout
        cjne  A,#'0',RECI_CA5           ; Si trama C, salta si puerto no '0'
        jmp   RECI_CA9

RECI_CA8: call  RXB                     ; Leer byte 2 de trama HOST: puerto
        jnb   BIT_RTO,RECI_CA6          ; Salta si timeout
        cjne  A,#PUERTO,RECI_CA5        ; Otro tipo, salta si no es PUERTO

; Leer campo de canal

RECI_CA9: call  RXB                     ; Leer byte 3 de trama HOST: canal
        jnb   BIT_RTO,RECI_CA6          ; Salta si timeout
        mov   CR_CANAL,A                ; Guarda byte de canal

; En tramas tipo D también lee el primer byte de datos (comando)

        mov   A,CR_CONTR
```

```
        cjne  A,#'D',RECI_CA3          ; Salta si la trama no es tipo D
        call  RXB                      ; Leer byte 0 de datos: comando
        jnb   BIT_RTO,RECI_CA6         ; Salta si timeout
        mov   CR_CMD,A                 ; Guarda primer byte (comando)

RECI_CA3: pop   ACC                    ; Recupera valor original de R0
        mov   R0,A
        ret                            ; Acaba

; Error timeout en la espera del primer FEND

RECI_CA4: djnz  B,RECI_CA0             ; Si quedan intentos vuelve a
        djnz  R0,RECI_CA1              ;   intentar leer un FEND

        mov   A,#90                    ; Mensaje de error de TO de trama
        call  LOG_ESC                  ; Anota mensaje en diagnostico

        setb  BIT_0                    ; Indica error TO de trama
        jmp   RECI_CA3

; Código no esperado o error timeout entre bytes de la trama HOST

RECI_CA5: call  DESECHA
RECI_CA6: mov   A,#55                  ; Mensaje error de trama
        call  LOG_ESC                  ; Anota mensaje en diagnostico

        setb  BIT_1                    ; Indica error de trama
        jmp   RECI_CA3

;.....................................................................
; Función:  Envío de un byte aplicando transparencia
; Entrada:  A = dato a enviar
; Salida:
; Modif.R:  A
; Modif.B:
; Usa:      TXB
;.....................................................................

ENVI_TR:  cjne  A,#FESC,ENVI_TR1       ; Si no es FESC salta
        call  TXB                      ; Si es FESC envía FESC y TFESC
        mov   A,#TFESC
        sjmp  ENVI_TR2
ENVI_TR1: cjne  A,#FEND,ENVI_TR2       ; Si no es FEND salta
        mov   A,#FESC                  ; Si es FEND envía FESC y TFEND
        call  TXB
        mov   A,#TFEND
ENVI_TR2: call  TXB                    ; Si no es FEND o FESC envía dato
        ret

;.....................................................................
; Función:  Envía la cabecera de una trama NCA: campos de control
;           puerto y canal. Además, envía los dos primeros bytes de datos
;           (para indicar comandos) si se especifican estos
; Entrada:  C_CONTR = campo de control
;           C_CANAL = numero de canal
;           C_CMD1 = valor para comando en el campo de datos
;               (0 si no se debe enviar)
;           C_CMD2 = valor repetido para comando en el campo de datos
;               (0 si no se debe enviar)
; Salida:
; Modif.R:  TMP
; Modif.B:
; Usa:      ENVI_FND,TXB
;.....................................................................
```

```
ENVI_CNCA: mov   TMP,A          ; Guarda A
        call  ENVI_FND          ; Envía FEND inicial

        mov   A,C_CONTR         ; Envía tipo trama (C, D, S)
        call  TXB

        mov   A,#PUERTO         ; Envía puerto (2)
        call  TXB

        mov   A,C_CANAL         ; Envía canal (A)
        call  TXB

        mov   A,C_CMD1          ; Si 1er byte de comando <> 0 lo envía
        jz    ENVI_CA1
        call  TXB

        mov   A,C_CMD2          ; Si 2° byte de comando <> 0 lo envía
        jz    ENVI_CA1
        call  TXB

ENVI_CA1: mov   A,TMP           ; Recupera A
        ret

;.....................................................................
; Función:  Envía el byte de FEND final de una trama, y reinicia watchdog
; Entrada:
; Salida:
; Modif.R:  A
; Modif.B:
; Usa:      TXB, PUSLO5
;.....................................................................
ENVI_FND: call  PULSO5          ; Reinicia watchdog
        mov   A,#FEND           ; Envía FEND al nivel físico
        call  TXB
        ret


; FUNCIONES DEL NIVEL FISICO
; ---------------------------------------------------------------------

;.....................................................................
; Función:  Coloca un byte en el buffer de transmisión, o directamente en
;           el registro de transmisión si el buffer esta vacío.
; Entrada:  A = dato a enviar
; Salida:
; Modif.R:  R0 banco 1
; Modif.B:  BIT_TRDY, BIT_TOV
; Usa:
;.....................................................................

TXB:    jb    BIT_TOV,$         ; Si el buffer de transmisión esta
                                  ; lleno espera a que deje de estarlo
        clr   IE.4              ; Desactiva interrupción del serie para
                              ;   evitar conflictos con los registros en
                              ;   caso de una interrupción del puerto serie

        push  PSW               ; Guarda estado y numero actual de banco
        mov   PSW,#00001000b    ; Selecciona el banco de registros 1

; Meter dato a enviar en el registro de transmisión si este esta vacío

        jb    BIT_TRDY,TXB2     ; Si el registro de transmisión no esta
                                  ; vacío salta
```

```
        mov   SBUF,A        ; Cuando el buffer esta vacío pasa dato a
                              ; enviar directamente al reg. de transmisión
        setb  BIT_TRDY      ; El registro de transmisión ya no esta vacío
        mov   R0,#0         ; Inicia índice al último dato del buffer de
                              ; trans. indicando que no hay datos en él.

TXB1:   pop   PSW         ; Restaura estado y vuelve a banco original
        setb  IE.4
        ret

; Meter dato en el buffer de transmisión

TXB2:   cjne  R0,#0,TXB4      ; Si el buffer ya tiene datos salta
        mov   R0,#INITX       ; Si no tiene datos inicia el índice al
                                ; ultimo dato
TXB3:   mov   @R0,A           ; Guarda dato en el buffer de transmisión
        sjmp  TXB1

TXB4:   cjne  R0,#(INITX+10),TXB5 ;Si buffer transmisión no lleno salta
        sjmp  TXB3

TXB5:   inc   R0                  ; Incrementa índice al ultimo dato
        cjne  R0,#(INITX+10),TXB3 ; Si buffer transmisión no lleno salta
        setb  BIT_TOV             ; Indica que se ha llenado buffer
        sjmp  TXB3


;.....................................................................
; Función:  Lectura del siguiente byte presente en el buffer de recepción
;           del puerto serie. Si el buffer esta vacío al realizar la
;           llamada espera a recibir durante un tiempo de timeout.
;           Durante la espera reinicia el watchdog
; Entrada:
; Salida:   ACC = dato leído (si ha se recibido algo)
;           BIT_RTO = 0 si acabo el timeout antes de recibir algo
; Modif.R:  R1 banco 1, R6 banco 1, R7 banco 1, ACC (si no acabo el
;           timeout)
; Modif.B:  BIT_RDY, BIT_OK
; Usa:      PULSO5
;.....................................................................

RXB:    push  PSW             ; Guarda estado y numero actual de banco
        push  ACC             ; Guarda acumulador
        mov   PSW,#00001000b  ; Selecciona el banco de registros 1

; Bucle de espera si no se ha recibido algo. Espera un tiempo a ver
; si llega algún dato

        mov   R6,#255         ; Inicia contador para la espera

RXB2:   call  PULSO5          ; Reinicia watchdog
        jb    BIT_RRDY,RXB1   ; Sale del bucle si recibe algo
        mov   R7,#16
RXB4:   djnz  R7,RXB4         ; Repite bucle para R6,R7
        djnz  R6,RXB2

; Fin cuando se ha acabado la espera sin recibir algo

        pop   ACC             ; Recupera acumulador
        pop   PSW             ; Activa banco de registros original
        clr   BIT_RTO
        ret                   ; Retorna indicando que paso el timeout
```

```
; Lectura de un byte recibido del buffer y actualización del buffer

RXB1:   pop   ACC              ; Saca de pila el valor de A guardado antes
        clr   IE.4             ; Desactiva interrupción del serie para
                          ;    evitar conflictos con los registros en
                          ;    caso de una interrupción del puerto serie

        mov   A,INIRX          ; Lee el siguiente byte

        push  ACC             ; Bucle para decalar 1 posición en el buffer
        mov   A,R0
        mov   R2,#10
        mov   R0,#(INIRX+1)
RXBOTRA:  mov   TMPBUF,@R0
        dec   R0
        mov   @R0,TMPBUF
        inc   R0
        inc   R0
        djnz  R2,RXBOTRA
        mov   R0,A
        pop   ACC

        dec   R1              ; Decrementa índice al ultimo dato
        setb  BIT_RTO         ; Activa bit de lectura correcta sin timeout

        cjne  R1,#INIRX,RXB3   ; Salta si aun quedan datos
        clr   BIT_RRDY         ; Borra bit de datos en buffer

RXB3:   pop   PSW             ; Activa banco de registros original
        setb  IE.4             ; Activa interrupción del serie
        ret


;-----------------------------------------------------------------------
;                    FUNCIONES DEL SO DEL SIR
;-----------------------------------------------------------------------

;.......................................................................
; PIDE_RAM: Intenta bloquear la memoria SRAM para este micro y si no lo
;           consigue envía comando WAIT a la EP
; Entrada:
; Salida:   BIT_4 = 1 si no se logra bloquear memoria
; Modif.R:  A
; Modif.B:  BIT_4
; Usa:      B_RAM
;.......................................................................
; VERCMD:   Lee comando procedente del otro micro. Parar ello
;           intenta bloquear la memoria y si no lo consigue envía WAIT
;           a la EP. Al final se libera la memoria.
; LEECMD:   También lee comando procedente del otro micro, pero
;           espera a bloquear la memoria. Al acabar libera la memoria.
; Entrada:
; Salida:   Para VERCMD, BIT_4 = 1 si no se pilla memoria
;           A = comando leído. A = 0 si error al leer comando
; Modif.R:  A, DPTR
; Modif.B:  BIT_E, BIT_4
; Usa:      PIDE__RAM, ESP_RAM, L_RAM, D_RAM, LOG_ESC
;.......................................................................
; AUC2ESP:  Pasa un comando al otro micro. La función espera bloquear la
;           memoria y al acabar la libera. También se avisa al
;           otro micro activando su BIT_COM.
```

```
; AUC2DIR:  Para comandos como MODO_T, MODO_P, MODO_N, BATT de EP.
;           Intenta pillar memoria y pasa comando al otro micro si lo
;           logra. Si no se bloquea la memoria envía WAIT a EP. Al acabar
;           se activa BIT_COM del otro micro y se libera la memoria.
; Entrada:  A = código del comando para el otro micro
; Salida:   Para AUC2DIR, BIT_4 = 1 si no se pilla memoria
; Modif.R:  A, DPTR
; Modif.B:  BIT_INT, B,BIT_E, BIT_4
; Usa:      PIDE_RAM, ESP_RAM, E_RAM, D_RAM, LOG_ESC
;.....................................................................
; INICMD:   Iniciar comando de la posición 0 de la SRAM
; Entrada:
; Salida:
; Modif.R:  A, DPTR
; Modif.B:  BIT_E
; Usa:      ESP_RAM, E_RAM, D_RAM, LOG_ESC
;.....................................................................
; LOG_INI:  Revisa índices para diagnostico. Los índices deben ser
;           valores múltiplos de 5, ya que cada mensaje tiene 5 bytes. Si
;           alguno de los 2 índices no es múltiplo de 5 se inicia a 0.
; Entrada:
; Salida:
; Modif.R: A,DPTR,B,
; Modif.B:
; Usa:      ESP_RAM, D_RAM, E_RAM, L_RAM
;.....................................................................
; LOG_ESC: Escribe anotación de error en el área de mensajes del micro 1
;          y reinicia watchdog. Si hay error grave al acceder a SRAM,
;          salta directamente a la función ERRORF
; Entrada: A = código del mensaje
; Salida:
; Modif.R: A
; Modif.B: BIT_E
; Usa:     PULSO5, B_RAM, D_RAM, L_RAM, E_RAM, LEECLK, NOPCLK, SUMA
;.....................................................................
; LOG_ENV:  Envía a la EP los mensajes de diagnostico
; Entrada:
; Salida: BIT_4 = 1 si se recibe disconnect al esperar respuesta de EP
; Modif.R: CABEZA
; Modif.B: BIT_LOG
; Usa:     ENVI_FI
;.....................................................................
; ERRORF:  Tratar errores críticos: queda en espera de P_RESET.
;        Un error critico se produce al no poder anotar en diagnostico.
; Entrada:
; Salida:
; Modif.R:  A
; Moduf.B:
; Usa:      D_RAM, RECI_CAB, ENVI_DAT, DLY20, RESETT, DESECHA
;.....................................................................
; OPEN:     Abre fichero en la SRAM, para datos o para diagnostico
; Entrada:  ACMD+1/ACMD+2 = longitud del fichero en memoria
;           BIT_LOG = 1 para acceder a área de diagnostico;
;                     0 para área datos
; Salida:   PTR_OPL/PTR_OPL = punt. datos
;           OP_TOP_L/OP_TOP_L = punt. ultimo byte
;           LON_L/LON_H = longitud del fichero
;           BIT_OP = 0
; Modif.D:  PTR_OPL, PTR_OPH, OP_TOPL, OP_TOPH, DAT_H, DAT_L, ARIT_L,
;           ARIT_H, A
; Modif.B:  BIT_OP, BIT_E
; Usa:      ESP_RAM, L_RAM, D_RAM, SUMA, RESTA, LOG_ESC
;.....................................................................
; PUT:      Escribe siguiente dato en área de datos
```

```
; Entrada:  A = dato a guardar
;           PTR_OPL/PTR_OPL = punt. de datos
; Salida:   PTR_OPL/PTR_OPL = punt. datos
;           BIT_OP = 1 si dirección A_TOPDAT sobrepasada
; Modif.R:  TMP, DAT_H, DAT_L, PTR_OPL, PTR_OPH, ARIT_L, ARIT_H, A
; Modif.B:  BIT_OP, BIT_E
; Usa:      ESP_RAM, L_RAM, E_RAM, D_RAM, RESTA, LOG_ESC
;.....................................................................
; GET:      Lee siguiente dato del área de datos o de diagnostico
; Entrada:  OP_TOPL/OP_TOPH = punt. ultimo dato
;           PTR_OPL/PTR_OPL = punt. de datos
;           BIT_LOG = 1 para acceder a área de diagnostico,
;                     0 para área datos
; Salida:   ACC = dato leído
;           PTR_OPL/PTR_OPL = punt. datos
;           BIT_OP = 1 si fin de datos alcanzado, 0 en otro caso
; Modif.R:  PTR_OPL, PTR_OPH, ARIT_L, ARIT_H, DAT_H, DAT_L, A
; Modif.B:  BIT_OP, BIT_E
; Usa:      ESP_RAM, L_RAM, D_RAM, RESTA, LOG_ESC
;.....................................................................
; CLOSE:    Actualiza área datos al acabar de escribir datos con PUT
; Entrada:  PTR_OPL/PTR_OPL = puntero al final de datos
; Salida:   Se actualiza A_CMD+1/A_CMD+2 con tamaño del fichero
;           BIT_OP = 0
; Modif.R:  ARIT_L, ARIT_H, DAT_H, DAT_L, A
; Modif.B:  BIT_E, BIT_OP
; Usa:      ESP_RAM, E_RAM, D_RAM, RESTA, LOG_ESC
;.....................................................................
; STRCLK:   Arranca reloj de la SRAM, si estaba parado
; Entrada:
; Salida:
; Modif.R:  A, DPTR, B
; Modif.B:  BIT_E
; Usa:      ESP_RAM, E_RAM, LEECLK, L_RAM, NOPCLK, ESCCLK, D_RAM, LOG_ESC
;.....................................................................
; CFGSERIE: Configura puerto serie (velocidad)
; Entrada:  A = código de velocidad deseada (Ver tabla)
; Salida:
; Modif.R:  A, SCON, timer 1,
; Modif.B:  bit SMOD de PCON (87h)
; Usa:
; Valores posibles de velocidad:
; A=1->     2400 bps: cuenta T1=S24 , SMOD=0
; A=2->     4800 bps: cuenta T1=S24 , SMOD=1
; A=3->     9600 bps: cuenta T1=S96 , SMOD=0
; A=4->    19200 bps: cuenta T1=S96 , SMOD=1
;.....................................................................
; RXCLR:    Borra el buffer de recepción para ignorar últimos bytes
;           recibidos
; Entrada:
; Salida:
; Modif.R:  R1 banco 1, índice del buffer de recepción
; Modif.B:  BIT_RRDY, BIT_ROV, BIT_RTO
; Usa:
;.....................................................................
;Operaciones sobre el reloj:
;           ESCCLK. Activa escritura en reloj
;           LEECLK. Activa lectura de reloj
;           NOPCLK. Desactiva operaciones anteriores
; Entrada:
; Salida:   BIT_E = 1 si error al leer o escribir en memoria SRAM
;           No se altera si no hay errores
; Modif.R:  A, DPTR
; Modif.B:  BIT_4
```

```
; Usa:        L_RAM, E_RAM
;.....................................................................
; RESTA:      Resta números de 16 bits
; Entrada:    ARIT_L/ARIT_H = Operando 1
;             DPTR = Operando 2
; Salida:     C = acarreo
;             ARITL/ARIT_H = resultado de la resta Operando 1 - Operando 2
; Modif.R:    ARITL/ARIT_H, A
; Modif.B:    C
; Usa:
;.....................................................................
; SUMA:       Suma números de 16 bits
; Entrada:    ARIT_L/ARIT_H = Operando 1
;             DPTR = Operando 2
; Salida:     ARITL/ARIT_H = resultado
;             C = acarreo
; Modif.R:    ARITL/ARIT_H, A
; Modif.B:    C
; Usa:
;.....................................................................
; DLY20:      Reinicia watchdog y produce un retardo de unos 20ms.
; Entrada:
; Salida:
; Modif.R:    R6 banco 1, R7 banco 1
; Modif.B:
; Usa:        PULSO5
;.....................................................................
; TESPERA:    Espera un tiempo de 10seg, aunque si se recibe algo se acaba
;             la espera. Reinicia el watchdog durante la espera
; Entrada:
; Salida:     BIT_RRDY=1 si recibe cualquier cosa.
; Modif.R:    B, R0, R1
; Modif.B:
; Usa:        PULSO5
;.....................................................................
; PULSO5:     Envía un pequeño pulso (5µs) al circuito externo de watchdog
;             para evitar reset por CPU no activa.
; Entrada:
; Salida:
; Modif.R:
; Modif.B:    BIT_RST
; Usa:
;.....................................................................
; PULSOINI:   Envía un pulso de 3 ms al circuito de reset para asegurar
;             la descarga del condesador que controla el tiempo del
;             watchdog
; Entrada:
; Salida:
; Modif.R:    R2, R1
; Modif.B:    BIT_RST
; Usa:
;.....................................................................
; RESETT:Hace un reset hardware de la tarjeta, y espera a que tenga efecto
; Entrada:
; Salida:
; Modif.D.:   BIT_RST
; Usa:
;.....................................................................
; ESP_RAM:    Espera a bloquear la memoria SRAM, reiniciando el watchdog
; Entrada:
; Salida:
; Modif.R:
; Modif.B:
```

```
; Usa:       B_RAM, PULSO5
;.....................................................................
; B_RAM:    Establece una contienda con el otro micro para intentar pillar
;           la memoria
; Entrada:
; Salida:   BIT_4=1 si no se ha logrado bloquear memoria,
;                 0 en el otro caso
; Modif.R:
; Modif.B:  BIT_CS1, BIT_4
; Usa:
;.....................................................................
; D_RAM:    Desbloquea la memoria que tiene pillada
; Entrada:
; Salida:
; Modif.R:
; Modif.B:  BIT_CS1
; Usa:
;.....................................................................
; E_RAM:    Escribe un dato de la memoria SRAM, que debe estar bloqueada
;           por este micro mediante la función B_RAM
; Entrada:  A = dato a escribir
;           DPTR = dirección donde escribir el dato
; Salida:   BIT_E=1 si error en escritura. No se altera si no hay error
; Modif.R:  P0, P1, P2, TMP
; Modif.B:  BIT_E, BIT_WE
; Usa:
;.....................................................................
; L_RAM:    Lee un dato de la memoria SRAM, que debe estar bloqueada por
;           este micro mediante la función B_RAM
; Entrada:  DPTR = dirección del dato a leer
; Salida:   A = dato leído
;           BIT_E = 1 si error en lectura. No modificado si no hay error
; Modif.R:  A, P0, P1, P2, TMP
; Modif.B:  BIT_E, BIT_WE
; Usa:
;.....................................................................
; MEMTS:    Pruebas de lectura y escritura en SRAM. Permite R/W sobre
;           SRAM desde la consola de la EP.
; Entrada:  A = comando a atender: TEST_E o TEST_L
; Salida:
; Modif.D:  A, R5, R0 a R5, DPTR, ULTCMD
; Modif.B:  BIT_3
; Usa:      ESP_RAM, E_RAM, L_RAM, D_RAM, DLY20, ENVI_CAB
;           RECITR, DESECHA, ENVINACK, MEMTS_R
;.....................................................................
; AMAYS:    Pasar un carácter a mayúsculas si es un carácter en ['a','z']
; Entrada:  A: carácter a convertir
; Salida:   A: carácter convertido
; Modif.R:  A (si es letra minúscula), TMP
; Modif.B:
; Usa:
;.....................................................................

END
```

Sistema informático remoto. "*Control de estaciones de registro sísmico*"

Código para el micro-controlador 2
(controla la estación de medida y dialoga con μC1 a través de SRAM)

Las funciones de acceso a memoria, acceso al puerto serie, diagnóstico, intercambio de información entre micros y las utilidades del SO que sean iguales en ambos micros, no se especificarán en este listado. Para confeccionar las funciones que permiten dialogar al μC2 con la estación de registro se ha tenido en cuenta el protocolo de enlace del sistema utilizado

```
; VARIABLES Y CONSTANTES DEL PROGRAMA. MAPA DE MEMORIA RAM INTERNA DEL μC
; ---------------------------------------------------------------------
;  0 a  7: Banco de registros 0 de trabajo: R0 a R7
;  8 a 15: Banco de registros 1:
;          R0= Direc. ultimo byte en buffer de transmisión
;          R1= Direc. ultimo byte en buffer de recepción
;          R2 a R7: usados en bucles de retardo, espera y rutinas que no
;                   deben afectar a los registros del banco 0

TMP1      EQU   16   ; Datos temporales
TMP2      EQU   17   ;
MAXTO     EQU   18   ; TO espera en RXB: 141->10s, 70->5s, 0->19s
ARIT_L    EQU   19   ; Para cálculos aritméticos con rutinas RESTA y SUMA
ARIT_H    EQU   20
FIN_L     EQU   21   ; Variables para guardar datos de 16 bits
FIN_H     EQU   22
DAT_L     EQU   23   ; Variables para guardar datos de 16 bits
DAT_H     EQU   24
TMP       EQU   25   ; Variable Temporal
NUMEVE    EQU   26   ; Eventos confirmados con DEL_FI o sig. evento a pedir
;libre    EQU   27
;libre    EQU   28
;libre    EQU   29
VARITL    EQU   30   ; Para guardar ARIT_L/H en LOG_ESC
VARITH    EQU   31
;libre    EQU   32
TMPBUF    EQU   33   ; Usada para profundizar en buffers INITX y INIRX

; 46: palabra de estado 2
; 47: palabra de estado 1

INITX     EQU   49   ; Inicio del buffer de transmisión (INITX+10 bytes)
INIRX     EQU   62   ; Inicio del buffer de recepción (INIRX+10 bytes)

PILA      EQU   75   ; Inicxio de pila (75...127)


; DEFINICIONES DE LOS BITS DE ESTADO
; ------------------------------------------------------------------
; 46 Palabra de estado 2
;
; 7 6 5 4 3 2 1 0
; | | | | | | | | +- (BIT_TMP) Status Temporal 0=Cierto 1=Falso
; | | | | | | | +--- (BIT_COM) Mensaje pendiente μC Contrario 0=No 1=Si
; | | | | | | +----- (BIT_E)   Indicador de error al acceder a SRAM
; | | | | | +------- libre
; | | | | +--------- libre
; | | | +----------- (BIT_AX)
; | | +------------- (BIT_2)
; | +--------------- (BIT_3)
```

```
BIT_TMP   EQU   70h  ; Bit 46.0. Propósito general.0=Cierto 1=Falso
BIT_COM   EQU   71h  ; Bit 46.1. Usado por INT0
BIT_E     EQU   72h  ; Bit 46.2. Indica error al acceder a memoria
BIT_AX    EQU   75h  ; Bit 46.5. Usado en la rutina ENV_ASC.
BIT_2     EQU   76h  ; Bit 46.6. De propósito general.
BIT_3     EQU   77h  ; Bit 46.7. De propósito general.

;  47  Palabra de estado 1
;
; 7 6 5 4 3 2 1 0
; | | | | | | | +- libre
; | | | | | | +--- tmp
; | | | | | +----- (BIT_RRDY) 0=Sin datos recep.
; | | | | +------- (BIT_ROV)  1=Overflow recep.
; | | | +--------- (BIT_RTO)  0=Timeout recep.
; | | +----------- libre
; | +------------- (BIT_TRDY) 0=Sin datos transm.
; +--------------- (BIT_TOV)  1=Overflow transm.
;

BIT_RRDY  EQU   7Ah  ; Bit 47.2. Datos en recepción
BIT_ROV   EQU   7Bh  ; Bit 47.3. Oferflow recepción
BIT_RTO   EQU   7Ch  ; Bit 47.4. Timeout recepción
BIT_TRDY  EQU   7Eh  ; Bit 47.6. Datos por transmitir
BIT_TOV   EQU   7Fh  ; Bit 47.7. Oferflow transmisión

; Otros bits utilizados

BIT_CS1   EQU   0B7h ; Bit P3.7. Línea NOT RD conectada a NOT CS1
BIT_WE    EQU   0B6h ; Bit P3.6. Línea NOT WR conectada a NOT WE
BIT_INT   EQU   0B5h ; Bit P3.5. Disparo de INT0(Flanco descendente)
BIT_RST   EQU   0B4h ; Bit P3.4. En uC1 Reset de uC1-uC2. En uC2 reset IDS

BIT_0     EQU   0D5h ; Bit PSW.5 Propósito general
BIT_1     EQU   0D1h ; Bit PSW.1 Propósito general


; VALORES DE DEFINICIONES CONSTANTES
; ----------------------------------------------------------------------

; Mapa de Memoria SRAM

; Area de comandos
;     A_CMD+0:  Código de comando
;     A_CMD+1:  Byte bajo de longitud de bloque de datos en RAM
;     A_CMD+2:  Byte alto de longitud de bloque de datos en RAM
;     ...       Libre
; Area de parámetros
;     A_PAR+0:  Numero de estación
;     A_PAR+1:  Código de velocidad de comunicación con TNC
;     ...       Libre
;     A_PAR+10: SVB del área de parámetros
; Area de datos: A_DATOS ... A_TOPDAT
; Area de mensajes de diagnostico:
;     Formato de un mensaje: día hora min seg código (5 bytes)
;     Hay dos zonas: para uC1 y para yC2. Caben 51 mensajes en cada zona:
;     ALOG1 ... ALOG1+255:          datos de uC1
;     ALOG2 ... ALOG2+255=A_TOPLOG: datos de uC2
; Area del reloj de tiempo real: A_TMCTRL...

A_CMD     EQU   0    ; Dirección de inicio del área de comandos
A_PAR     EQU   9    ; Dirección de inicio del área de parámetros
A_DATOS   EQU   100  ; Dirección de inicio de la zona de datos
```

```
A_TOPDAT  EQU   7499 ; Limite de la zona de datos (bloques de 1040Bytes)
A_PTRLOG1 EQU   7508 ; Posición del puntero del fichero LOG para uC1
A_PTRLOG2 EQU   7509 ; Posición del puntero del fichero LOG para uC2
A_LOG1    EQU   7510 ; Comienzo del fichero LOG o anotaciones de uC1
A_LOG2    EQU   7765 ; Comienzo del fichero LOG o anotaciones de uC2
A_TOPLOG  EQU   8020 ; Limite del área LOG (se incluye este byte)
A_TMCTRL  EQU   8184 ; Registro de control del reloj
A_TMSEGU  EQU   8185 ; Registro de segundos del reloj
A_TMMINU  EQU   8186 ; Registro de minutos del reloj
A_TMHORA  EQU   8187 ; Registro de horas del reloj
A_TMDIAS  EQU   8188 ; Registro de días de semana del reloj
A_TMDIAM  EQU   8189 ; Registro de días de mes del reloj
A_TMMES   EQU   8190 ; Registro de meses del reloj
A_TMANO   EQU   8191 ; Registro de anos del reloj

;Constantes de comandos entre EP, uC1 y uC2

SONDEO    EQU   33h  ; sondeo de EP a uc1
NO_DAT    EQU   66h  ; no hay datos. De uc1 a Ep
WAIT      EQU   75h  ; memoria ocupada por uc2. De uc1 a EP
DAT_AV    EQU   68h  ; datos de aviso. De uc1 a Ep y de uc2 a uc1
DAT_FI    EQU   6Ah  ; datos de fichero. De uc1 a Ep y de uc2 a uc1
SVB       EQU   6Bh  ; SVB de datos fichero. De uc1 a Ep
ACK_AV    EQU   83h  ; datos de aviso recibidos. De EP a uc1 y de uc1 a uc2
ACK_FI    EQU   0A3h ; datos de fichero recibido. De EP a uc1 y de uc1 a uc2
NACK_FI   EQU   0A5h ; datos fichero erróneos. De EP a uc1
C_MEDIO   EQU   99h  ; respuesta de E.remota a ACK_FI o ACK_AV. Para
                     ; devolver el control del medio a EP.de uc1 a EP
TNACK_P   EQU   31h  ; trama errónea o mal conformada. De EP a uc1
TNACK_R   EQU   13h  ; trama errónea o mal conformada. De uc1 a EP
MODO_T    EQU   73h  ; petición de modo datos transparentes.
                     ; De EP a uc1 y de uc1 a uc2
R_MDT     EQU   37h  ; E.remota en modo transparente. De uc2 a uc1 y de uc1 a EP
DAT_TP    EQU   63h  ; datos en modo transparente. De EP a uc1 y de uc1 a uc2
DAT_TR    EQU   36h  ; respuesta con datos transp. De uc2 a uc1 y de uc1 a EP
MODO_N    EQU   54h  ; petición modo operación normal. De EP a uc1 y de uc1 a uc2
R_MDN     EQU   45h  ; E.remota en modo norma. De uc1 a EP
BATT      EQU   53h  ; petición estado batería. De EP a uc1 y de uc1 a uc2
R_BATT    EQU   35h  ; datos con estado batería. De uc2 a uc1 y de uc1 a EP
FDIAG     EQU   5Bh  ; petición fichero diagnostico. De EP a uc1
R_FDIAG   EQU   0B5h ; datos del fichero diagnostico. De uc1 a EP
P_RESET   EQU   9Ah  ; reset de E.remota. De EP a uc1
R_RESET   EQU   0A9h ; reset aceptado. De uc1 a EP
MODO_P    EQU   4Ah  ; parametrización de la estación de medida.
                     ; De EP a uc1 y de uc1 a uc2
R_MDP     EQU   0A4h ; E.remota dispuesta a aceptar parámetros.
                     ; De uc2 a uc1 y de uc1 a EP
DAT_PP    EQU   4Bh  ; datos para parametrización. De EP a uc1 y de uc1 a uc2
DAT_PR    EQU   0B4h ; respuesta de parametrización de la estación de medida.
                     ; De uc2 a uc1 y de uc1 a EP
RESET_ID  EQU   93h  ; comando de EP a uC1 y de uC1 a uC2 para iniciar por
                     ; hardware la IDS
R_RSTID   EQU   39h  ; respuesta al comando anterior. De uc2 a uC1 y de uC1 a EP
P_TIME    EQU   56h  ; petición de configuración o lectura del reloj. De uC1
                     ; desde EP
R_TIME    EQU   65h  ; respuesta de uC1 a EP por el comando anterior
SESION_R  EQU   15h  ; la tarjeta indica a la EP sesión establecida
SESION_P  EQU   51h  ; la EP indica a la tarjeta sesión establecida
DEL_FI    EQU   7Ah  ; la EP indica a la tarjeta que puede borrar fichero
REP_FI    EQU   7Bh  ; la EP indica a la tarjeta que reenvíe un fichero
ERROR_F   EQU   0EFh ; Indicación de error fatal a EP
TEST_E    EQU   18h  ; Prueba de SRAM: escritura de dato en una dirección
TEST_L    EQU   1Ah  ; Prueba de SRAM: lectura de dato en una dirección
R_TEST    EQU   81h  ; Respuesta a prueba de SRAM: dato escrito o leído
```

```
C_IDS     EQU   87h  ; Configuración de IDS en modo parametrización
R_CIDS    EQU   78h  ; Respuesta a configuración de la IDS

; Códigos de subcomandos

TIMEPRG   EQU   55h  ; Valor de byte 2 para programación de timer
TIMELEC   EQU   00h  ; Valor de byte 2 para lectura de timer
DATPTAR   EQU   01h  ; Valor de byte 2 para parámetros de tarjeta
DATPIDS   EQU   05h  ; Valor de byte 2 para enviar parámetros a IDS
DATPPRG   EQU   55h  ; Valor de byte 3 para prog. parámetros de tarjeta
DATPLEC   EQU   00h  ; Valor de byte 3 para leer parámetros de tarjeta

; Códigos de operaciones de parametrización

PAR_CAL   EQU   01h  ; Comando de param. para calibrar IDS
PAR_BOR   EQU   02h  ; Comando de param. para borrar ficheros de IDS
PAR_GPS   EQU   05h  ; Comando de param. para calibrar GPS
PAR_SIN   EQU   04h  ; Comando de param. para sincronizar GPS-IDS

; Constantes especiales de las tramas de nivel HOST

FESC      EQU  0DBh
FEND      EQU  0C0h
TFESC     EQU  0DDh
TFEND     EQU  0DCh

; Otras constantes

INTENT    EQU    3   ; Intentos para petición de bloques, cabeceras, y
                     ; ficheros de configuración de IDS
INTESC    EQU    3   ; Intentos de escritura de un comando para uC1

CR        EQU   0Dh  ; Caracteres ASCII de control
LF        EQU   0Ah
ESC       EQU   1Bh

; Cuentas para Temp.1 en la comunicación serie

S96       EQU   0FDh ; 9600/19200 Bd, para cristal de 11.059Mhz (para th1)
S24       EQU   0F4h ; 2400/4800 Bd, para cristal de 11.059Mhz (para th1)


; ---------------------------------------------------------------------
;                          CODIGO DEL PROGRAMA
; ---------------------------------------------------------------------

; INTERRUPCIONES
; ---------------------------------------------------------------------
(Igual que micro-1); Reset, INT0 y Serie


; INICIACION DEL SISTEMA
; ---------------------------------------------------------------------

INICIO:   mov   SP,#PILA        ; Inicializa el puntero de pila

          mov   TCON,#00000001b ; Inicializa TCON para INT0 dispare
                                ; por flanco descendente
          mov   TMOD,#00100001b ; T0 en modo 1: temp. 16 bits.
                                ; T1 en modo 2: temp. 8 bits con autorecarga
          mov   A,#3            ; Para configurar 9600Bd
          call  CFGSERIE        ; Configura puerto serie
```

```
        setb  TR1              ; Arranca Temp. 1
        mov   47,#0            ; Inicializa palabra de estado 1

        mov   9,#INIRX         ; Inicializa control buffer recepción y
        mov   8,#0             ; de transmisión: 9h=R1 BANK 1, 8h=R0 BANK 1

        mov   MAXTO,#70        ; TO por defecto (5 seg)

        clr   TI               ; Antes de setb IE.4 , si no se dispararan
        clr   RI               ; Por defecto es así
        setb  IE.4             ; Habilita interrupciones serie
        setb  IE.0             ; Habilita INT0
        setb  IE.7             ; Habilita interrupciones en general

        clr   BIT_COM          ; Inicializa palabra de estado 2:
                               ; borra cualquier aviso esporádico
        mov   B,#4             ; Tiempo para que uC1 este preparado

ESP_UC1:  call  DLY            ; 4*250ms=1s
        djnz  B,ESP_UC1

        clr   A
        call  LOG_ESC          ; Mensaje de iniciación completada


; BUCLE DE ESPERA DE CONEXION DE LA IDS (estación de registro)
; ------------------------------------------------------------------------

ESPCON:    mov   R7,#10          ; 10 intentos de espera

ESPCON_1: mov   R6,#141          ; Esperar como mucho 10 seg por intento
        call  TXT                ; Enviar comando @AT a IDS
        db    '@AT',CR,ESC

ESPCON_2: mov   TL0,#0           ; Con T0=0 y R6=141 son 10.02seg
        mov   TH0,#0
        setb  TR0                ; Arranca temporizador

ESPCON_3: jb    BIT_RRDY,ESPCON_5 ; Si recibe algo de IDS sale de espera
        jnb   TF0,ESPCON_3
        clr   TR0                ; Para contador y borra fin cuenta
        clr   TF0
        djnz  R6,ESPCON_2        ; Repite R6 veces: espera 10seg

ESPCON_4: djnz  R7,ESPCON_1      ; Decrementa intentos y repite si quedan

; Fin de intentos de conexión.

        mov   A,#50
        call  LOG_ESC          ; Mensaje diag. de conexión no establecida

        mov   R2,#DAT_AV       ; Como datos de aviso
        call  ENVMSG           ; Envía aviso a uC1
        db    'IDS no resp. AT',ESC

        jmp   MODOTT           ; Entra en modo transparente total

; Ver respuesta de IDS y si hay conexión

ESPCON_5: clr   TR0              ; Para contador y borra fin cuenta
        clr   TF0

        call  ESPCMM           ; Espera respuesta de IDS acabada en C>>
        jb    BIT_TMP,ESPCON_4 ; Si error de timeout salta
```

```
; ESTADO DE IDS = CONECTADA. SIR EN MODO NORMAL
; ---------------------------------------------------------------------

IDSCON:    mov   NUMEVE,#0        ; Inicia contador de eventos a primer evento

IDSCON_0: call  TXT               ; Envía AT y espera respuesta
          db    '@AT',CR,ESC
          call  ESPCMM
          jb    BIT_TMP,ESPCON     ; Si NO hay respuesta sigue

          mov   R7,#3              ; 3 intentos para NE

IDSCON_7: mov   R6,#255           ; R6=15 seg
          call  TXT                ; Envía un @NE a IDS
          db    '@NE',CR,ESC

IDSCON_2: mov   TL0,#0            ; Inicia T0
          mov   TH0,#0
          setb  TR0                ; Arranca temporizador

IDSCON_3: jb    BIT_COM,AVISO       ; Espera algo de IDS o aviso de uC1
          jb    BIT_RRDY,IDSCON_5
          jnb   TF0,IDSCON_3
          clr   TR0                ; Para contador y borra fin cuenta
          clr   TF0
IDSCON_4: djnz  R6,IDSCON_2
          djnz  R7,IDSCON_7        ; Si acaba tiempo y quedan intentos
          sjmp  IDSCON_0           ; Si no quedan intentos

; Datos recibidos de IDS: se espera respuesta a NE o mensaje de desconexión

IDSCON_5: clr   TR0               ; Para contador y borra fin cuenta
          clr   TF0
          call  ESPIDS           ; Recibe respuesta
          jb    BIT_1,IDSCON_4   ; Si error de timeout sigue en espera de NE
          jb    BIT_0,IDSCON_6   ; Si es desconexión salta

          jz    IDSCON_4
          jmp   EVENT            ; Si no desconexión va tomar eventos

IDSCON_6: mov   R2,#DAT_AV       ; Como datos de aviso
          call  ENVMSG            ; Envía aviso a uC1
          db    'IDS desconectada',ESC
          jmp   ESPCON           ; Si desconexión va a espera de conexión

IDSCON_8: clr   BIT_COM          ; Esperar el aviso del disconect antes de
          call  TESPCOM           ; Ir al bucle de modo normal
          clr   BIT_COM
          sjmp  IDSCON_0


; ATENDER COMANDO DESDE uC1 EN MODO NORMAL
; ---------------------------------------------------------------------

AVISO:     clr   BIT_COM          ; Desactiva aviso de uC1
          clr   TR0              ; Para contador y borra fin cuenta
          clr   TF0

          call  LEECMD           ; Lee comando de SRAM
          jnz   AVISO_0
          jmp   IDSCON_0         ; Si error de lectura vuelve a bucle
```

```
; Tratar comando MODO_T desde uC1

AVISO_0:  cjne  A,#MODO_T,AVISO_1 ; Salta si no comando de paso a modo transp
          jmp   IDSCMT            ; Va a modo transparente

; Tratar comando MODO_P desde uC1

AVISO_1:  cjne  A,#MODO_P,AVISO_2 ; Salta si no comando de paso a modo param
          jmp   IDSCMP            ; Va a modo parametrización

; Tratar comando MODO_N desde uC1

AVISO_2:  cjne  A,#MODO_N,AVISO_3 ; Salta si no comando de paso a modo normal
          mov   R2,#R_MDN         ; Envía respuesta a uC1 (sin datos)
          call  CMDUC1
          jmp   IDSCON_8          ; Va a modo normal

; Otro comando de uC1

AVISO_3:  mov   A,#1            ; Anota diag. sobre comando erróneo
          call  LOG_ESC
          jmp   IDSCON_0        ; Vuelve a bucle de conexión en modo normal



; LEER EVENTOS DE LA IDS Y ENVIARLOS A uC1
; ---------------------------------------------------------------------------

EVENT_MM: jmp   EVENT_M       ; Para un salto largo a borrar eventos

EVENT:    mov   R6,NUMEVE     ; Ver numero de eventos enviados
          clr   C             ; Resta eventos enviados a eventos de IDS
          subb  A,R6
          jc    IDSCONX       ; Si eventos enviados>eventos IDS va a M. normal
          jz    EVENT_MM      ; Si eventos enviados=eventos IDS va a borrar

; Pide cabecera a IDS, leyendo datos y obteniendo numero de bytes restantes

EVENT_2:  call  SHIDS           ; Envía @SH y lee cabecera para evento de R6
          jnb   BIT_COM,EVENT_22 ; Si no hay BIT_COM sigue pidiendo bloques
          jmp   EVENT_C1        ; Si BIT_COM activo salta a EVENT_C1

EVENT_22: jb    BIT_TMP,EVENT_3 ; Si hay error en lectura de cabecera salta

          mov   A,R0            ; Ver si hay bloques por recibir
          orl   A,R1
          jnz   EVENT_4         ; Si hay bloques por recibir salta

          mov   DPTR,#(A_DATOS+512)
          mov   R2,#DAT_FI      ; Pone datos de fichero para uC1
          call  AUC1
          call  B_COM           ; Avisa a uC1
          jmp   EVENT_H

; Enviar datos de aviso a uC1 en caso de error en respuesta a SH

EVENT_3:  mov   R2,#DAT_AV      ; Como datos de aviso
          call  ENVMSG          ; Envía aviso a uC1
          db    'Error en SHx',ESC
IDSCONX:  jmp   IDSCON_0        ; Acaba
```

```
; Calcular numero de bloques a pedir según bytes que quedan por leer

EVENT_4:  mov   A,#0FCh          ; R1,R0 = numero bytes por leer
          anl   A,R1             ; R1,R0/1024 = (R1,R0 and FC00h) >> 10 =
          clr   CY               ;   = (R1 and FCh) >> 2 = numero de bloques
          rrc   A                ;   de 1024 bytes.
          clr   CY
          rrc   A
          mov   R4,A             ; R4= numero de bloques de 1K

          mov   A,#3             ; R1,R0= resto de R1,R0/1024 = R1,R0 and 3FFh
          anl   A,R1             ;   numero de bytes restantes en otro bloque
          mov   R1,A

          clr   BIT_3            ; Borra indicador de que hay otro bloque
          orl   A,R0             ; Ver si hay 0 bytes de resto en otro bloque
          jz    EVENT_5          ; Si no hay otro bloque salta
          setb  BIT_3            ; Si hay otro bloque menor a 1K activa bit

; Inicia variables del bucle de lectura de bloques

EVENT_5:  mov   DPTR,#(A_DATOS+512) ; Puntero a memoria. Ya hay 512 bytes
          mov   R5,#0                ; Inicia contador de bloques
          clr   BIT_2                ; Borra indicador de datos pasados a uC1

; Mientras hayan bloques se van leyendo bloques de IDS para uC1

EVENT_6:  mov   A,R4             ; Si no hay bloques de 1K salta
          jz    EVENT_7

          mov   ARIT_L,#0        ; DPTR + 1024 -> ARIT_H,ARIT_L
          mov   ARIT_H,#4
          call  SUMA

          mov   DAT_L,DPL        ; Guarda puntero de datos en DAT_H,DAT_L
          mov   DAT_H,DPH

          mov   DPTR,#(A_TOPDAT+2) ; ARIT_H,ARIT_L - (A_TOPDAT+1)
          call  RESTA              ; CY=0 si ARIT_H,ARIT_L > A_TOPDAT
          jnc   EVENT_B          ; Salta si no cabe el bloque en memoria

          dec   R4               ; Decrementa numero de bloques de 1K
          mov   DPL,DAT_L        ; Recupera puntero a datos
          mov   DPH,DAT_H

          sjmp  EVENT_9          ; Salta y sigue

EVENT_7:  jb    BIT_3,EVENT_8    ; Si hay bloque final salta
          jmp   EVENT_G          ; Si no hay bloque final menor a 1K salta

EVENT_8:  mov   ARIT_L,R0        ; DPTR + R1,R0 -> ARIT_H,ARIT_L
          mov   ARIT_H,R1
          call  SUMA

          mov   DAT_L,DPL        ; Guarda puntero de datos en DAT_H,DAT_L
          mov   DAT_H,DPH

          mov   DPTR,#(A_TOPDAT+2) ; ARIT_H,ARIT_L - (A_TOPDAT+1)
          call  RESTA              ; CY=0 si ARIT_H,ARIT_L > A_TOPDAT
          jnc   EVENT_B            ; Salta si no cabe el bloque en memoria

          clr   BIT_3              ; Ya se ha enviado el bloque final
          mov   DPL,DAT_L          ; Recupera puntero a datos
```

```
        mov   DPH,DAT_H

EVENT_9:  call  SBIDS              ; Envía petición de bloque a IDS y lee bloque
        jb    BIT_COM,EVENT_C1 ; Si BIT_COM activo salta a EVENT_C1
        jb    BIT_TMP,EVENT_A  ; Si hay error en lectura salta

        inc   R5               ; Incrementa contador de bloques
        setb  BIT_2            ; Activa indicador de datos pasados a uC1
        sjmp  EVENT_6          ; Ejecuta bucle para siguiente bloque

; Enviar datos de aviso a uC1 en caso de error en respuesta a SB

EVENT_A:  mov   R2,#DAT_AV       ; Como datos de aviso
        call  ENVMSG           ; Envía aviso a uC1
        db    'Error en SBxx',ESC
        jmp   IDSCON_0          ; Acaba

; Pasar datos a uC1, avisar a uC1 y esperar confirmación de uC1

EVENT_B:  mov   DPL,DAT_L        ; Recupera puntero a datos
        mov   DPH,DAT_H

        jb    BIT_COM,EVENT_C1 ; Si BIT_COM activo salta

        mov   R2,#DAT_FI       ; Pone datos de fichero para uC1
        call  AUC1
        call  B_COM            ; Avisa a uC1

EVENT_C:  call  ESPUC1           ; Si no recibió BIT_COM espera

EVENT_C1: clr   BIT_COM          ; Desactiva aviso de uC1

        call  LEECMD           ; Lee comando de SRAM
        jz    EVENT_F          ; Si error de lectura salta

        cjne  A,#ACK_FI,EVENT_D ; Salta si no comando de confirmación
        clr   BIT_2            ; Borra indicador de datos pasados a uC1
        mov   DPTR,#A_DATOS    ; Inicia puntero a datos en memoria
        jmp   EVENT_6          ; Vuelve al bucle de envío de bloques

; Comandos MODO_N, MODO_T, MODO_P o comando no esperado de uC1

EVENT_D:  cjne  A,#MODO_N,EVENT_D1 ; Salta si no comando MODO_N
        mov   R2,#R_MDN          ; Envía respuesta a uC1 (sin datos)
        call  CMDUC1
        jmp   IDSCON_0           ; Va a modo normal

EVENT_D1: cjne  A,#MODO_T,EVENT_D2 ; Salta si no comando MODO_T
        jmp   IDSCMT               ; Va a modo transparente

EVENT_D2: cjne  A,#MODO_P,EVENT_D3 ; Salta si no comando MODO_P
        jmp   IDSCMP               ; Va a modo parametrizar

EVENT_D3: cjne  A,#REP_FI,EVENT_D4 ; Salta si no comando REP_FI
        jmp   EVENT_2              ; Va a inicio de pedir eventos

EVENT_D4: mov   A,#2             ; Anotación sobre comando erróneo
        call  LOG_ESC
EVENT_F:  jmp   IDSCON_0         ; Acaba

; Acabar envío de evento anterior y pasar a siguiente evento si lo hay

EVENT_G:  mov   DAT_L,DPL        ; Guarda puntero de datos en DAT_H,DAT_L
        mov   DAT_H,DPH          ;   porque TXT lo modifica
```

```
        call  TXT                ; Comando MAB fichero a IDS
        db    '@MAB',ESC
        mov   A,R6
        call  ENVASC

EVENT_G1: mov   A,#CR
        call  TXB

        mov   DPL,DAT_L          ; Recupera puntero a datos
        mov   DPH,DAT_H

        call  ESPMM              ; Espera Respuesta de IDS
        jb    BIT_TMP,EVENT_G1   ; Si timeout envía CR otra vez

        jnb   BIT_2,EVENT_I      ; Si no se ha hecho traspaso a uC1 salta

        jb    BIT_COM,EVENT_H1   ; Si BIT_COM activo salta

        mov   R2,#DAT_FI         ; Pone datos de fichero para uC1
        call  AUC1
        call  B_COM              ; Avisa a uC1

EVENT_H:  call  ESPUC1           ; Si no recibió BIT_COM espera

EVENT_H1: clr   BIT_COM          ; Desactiva aviso de uC1

        call  LEECMD             ; Lee comando de SRAM
        jz    EVENT_F            ; Si error de lectura salta

        cjne  A,#ACK_FI,EVENT_DX ; Salta si no comando de confirmación

; Ya se ha recibido el ultimo ACK_FI

EVENT_I:  call  TESPCOM          ; Rutina de espera de 10seg
        jb    BIT_COM,EVENT_J    ; Se ha recibido aviso de uC1

        jmp   IDSCON_0           ; No se ha recibido aviso de uC1

EVENT_J:  clr   BIT_COM          ; Desactiva aviso de uC1

        call  LEECMD             ; Lee comando de uC1
        jz    EVENT_F            ; Salta si error al leer

        cjne  A,#REP_FI,EVENT_K  ; Si no es REP_FI salta
        jmp   EVENT_2            ; Vuelve a pedir ultimo evento

EVENT_K:  cjne  A,#DEL_FI,EVENT_DX ; SI no es DEL_FI salta
        inc   R6                 ; Inc. num. eventos enviados o sig. evento
        mov   NUMEVE,R6

        call  TXT
        db    '@NE',CR,ESC       ; Pregunta numero de eventos a IDS

        call  ESPIDS
        jb    BIT_1,EVENT_O      ; Si error de timeout salta
        jb    BIT_0,EVENT_P      ; Si es desconexión salta
                                 ; A=NUMERO DE EVENTOS DE IDS
        jz    EVENT_L            ; Si cero eventos acaba
        clr   C                  ; A-NUMEVE = ev en IDS - ev enviados a EP
        subb  A,NUMEVE           ; A-NUMEVE=0: NUMEVE=A Todos los eventos enviados
        jz    EVENT_M            ; Cy=0: A>NUMEVE Eventos por enviar
        jc    EVENT_O1           ; Cy=1: A<NUMEVE Todos los eventos enviados
```

```
        jmp   EVENT_2          ; Enviar resto de eventos

EVENT_M:  mov   R2,#WAIT        ; Pone WAIT para uC1 sin activar su BIT_COM
        mov   DPTR,#A_DATOS
        call  AUC1

        call  BORRAFI          ; Borrar fichero de IDS
        jb    BIT_0,EVENT_Q    ; Si hay error salta
        call  DLY              ; Esperar 1seg
        call  DLY
        call  DLY
        call  DLY

EVENT_L:  mov   R2,#ACK_FI      ; Quita WAIT para uC1 sin activar su BIT_COM
        mov   DPTR,#A_DATOS
        call  AUC1
        jmp   INICIO            ; Vuelve al bucle de envío de NE e inicia R6

EVENT_DX: jmp   EVENT_D

; Enviar datos de aviso a uC1 en caso de error de TO en respuesta a NE

EVENT_O:  mov   R2,#DAT_AV      ; Como datos de aviso
        call  ENVMSG            ; Envía aviso a uC1
        db    'TO en resp. NE',ESC

EVENT_O1: jmp   IDSCON_0        ; Acaba

; IDS ha desconectado

EVENT_P:  mov   R2,#DAT_AV      ; Como datos de aviso
        call  ENVMSG           ; Envía aviso a uC1
        db    'IDS desconectada',ESC
        jmp   ESPCON           ; Si desconexión va a espera de conexión

; Datos de aviso sobre error al borrar ficheros

EVENT_Q:  mov   R2,#DAT_AV      ; Como datos de aviso
        call  ENVMSG           ; Envía aviso a uC1
        db    'TO en resp. DEL_FI',ESC
        jmp   IDSCON_0         ; Si desconexión va a espera de conexión



; ESTADO DE IDS = CONECTADA. SIR EN MODO TRANSPARENTE
; ---------------------------------------------------------------------------

; Posibles tiempos para espera de respuesta de IDS:
; Para 30s -> R7=4 y R6=105
; Para 60s -> R7=8 y R6=105
; Para 90s -> R7=12 y R6=105

IDSCMT:   mov   R7,#8           ; R7=8 y R6=105 para 60 seg
IDSCMT_1: mov   R6,#105
IDSCMT_2: mov   TL0,#0          ; Inicia T0
        mov   TH0,#0
        setb  TR0               ; Arranca temporizador

IDSCMT_3: jb    BIT_RRDY,IDSCMT_5 ; Sale de espera si hay recepción
        jb    BIT_COM,ATEND       ; de IDS o aviso de uC1
        jnb   TF0,IDSCMT_3
        clr   TR0              ; Para contador y borra fin cuenta
        clr   TF0
```

```
IDSCMT_4: djnz  R6,IDSCMT_2      ; Repite R7, R6 veces
          djnz  R7,IDSCMT_1

          call  TXT              ; Si pasa tiempo envía un @AT a IDS para
          db    '@AT',CR,ESC     ; asegurar la conexión
          sjmp  IDSCMT           ; Y repite bucle

; Datos recibidos de IDS: se espera respuesta a NE o mensaje de desconexión

IDSCMT_5: clr   TR0              ; Para contador y borra fin cuenta
          clr   TF0
          call  ESPIDS           ; Recibe respuesta
          jb    BIT_1,IDSCMT_4   ; Si error de timeout sigue en espera de NE
          jnb   BIT_0,IDSCMT     ; Si no desconexión va a espera en conexión
          jmp   MODOTT           ; Si desconexión va a MTT


; ATENDER COMANDO DESDE uC1 EM MODO TRANSPARENTE
; ----------------------------------------------------------------------

ATEND:    clr   BIT_COM          ; Desactiva aviso de uC1
          clr   TR0              ; Para contador y borra fin cuenta
          clr   TF0

          call  LEECMD           ; Lee comando de SRAM
          jz    IDSCMT           ; Si error de lectura vuelve a bucle

; Tratar comando de RESET desde uC1

          cjne  A,#RESET_ID,ATEND_1 ; Salta si no comando de reset
          call  RIDS
          jmp   MODOTT               ; Entra en modo transparente total

; Tratar comando de datos transparentes desde uC1

ATEND_1:  cjne  A,#DAT_TP,ATEND_2 ; Salta si no comando DAT_TP
          call  MEMIDS           ; Datos de uC1 a IDS
          mov   R2,#DAT_TR       ; Comando de respuesta para uC1
          call  IDSMEM           ; Lee respuesta y envía a uC1
          jb    BIT_TMP,MODOTT_X ; Si respuesta es desconexión salta
          jmp   IDSCMT           ; Vuelve al bucle

; Tratar comando BATT desde uC1

ATEND_2:  cjne  A,#BATT,ATEND_3   ; Salta si no comando BATT
          call  TXT              ; Envía comando a IDS
          db    '@BATT',CR,ESC
          mov   R2,#R_BATT       ; Comando de respuesta para uC1
          call  IDSMEM           ; Lee respuesta y envía a uC1
          jb    BIT_TMP,MODOTT_X ; Si respuesta es desconexión salta
          jmp   IDSCMT           ; Vuelve al bucle

; Tratar comando MODO_N desde uC1

ATEND_3:  cjne  A,#MODO_N,ATEND_4 ; Salta si no comando MODO_N
          mov   R2,#R_MDN        ; Envía respuesta a uC1 (sin datos)
          call  CMDUC1
          jmp   IDSCON_8         ; Va a modo normal

; Otro comando de uC1

ATEND_4:  mov   A,#3             ; Anota diag. sobre comando erróneo
```

```
        call  LOG_ESC
        jmp   IDSCMT              ; Vuelve a bucle de conexión

; Salto largo al modo transparente total

MODOTT_X: jmp   MODOTT


; ESTADO DE IDS = CONECTADA. SIR EN MODO PARAMETRIZACION
; ---------------------------------------------------------------------------

; Posibles tiempos para espera de respuesta de IDS:
; Para 30s -> R7=4 y R6=105
; Para 60s -> R7=8 y R6=105
; Para 90s -> R7=12 y R6=105

IDSCMP:    mov   R7,#8            ; R7=8 y R6=105 para 60 seg
IDSCMP_1: mov   R6,#105
IDSCMP_2: mov   TL0,#0            ; Inicia T0
        mov   TH0,#0
        setb  TR0               ; Arranca temporizador

IDSCMP_3: jb    BIT_RRDY,IDSCMP_5 ; Sale de espera si hay recepción
        jb    BIT_COM,PARAM      ;   de IDS o aviso de uC1
        jnb   TF0,IDSCMP_3
        clr   TR0               ; Para contador y borra fin cuenta
        clr   TF0

IDSCMP_4: djnz  R6,IDSCMP_2      ; Repite R7,R6 veces
        djnz  R7,IDSCMP_1

        call  TXT               ; Si pasa tiempo envía un @NE a IDS para
        db    '@AT',CR,ESC      ; asegurar que esta conectada
        sjmp  IDSCMP            ; Y repite bucle

; Datos recibidos de IDS: se espera respuesta a NE o mensaje de desconexión

IDSCMP_5: clr   TR0              ; Para contador y borra fin cuenta
        clr   TF0
        call  ESPIDS            ; Recibe respuesta
        jb    BIT_1,IDSCMP_4    ; Si error de timeout sigue en espera de NE
        jnb   BIT_0,IDSCMP      ; Si no desconexión va a espera en conexión
        jmp   MODOTT            ; Si desconexión va a MTT


; ATENDER COMANDO DESDE uC1 EM MODO PARAMETRIZACION
; ---------------------------------------------------------------------------

PARAM:     clr   BIT_COM         ; Desactiva aviso de uC1
        clr   TR0               ; Para contador y borra fin cuenta
        clr   TF0

        call  LEECMD            ; Lee comando de SRAM
        jz    IDSCMP            ; Si error de lectura vuelve a bucle

; Tratar comando MODO_N desde uC1

        cjne  A,#MODO_N,PARAM_1 ; Salta si no comando MODO_N
        mov   R2,#R_MDN         ; Envía respuesta a uC1 (sin datos)
        call  CMDUC1
        jmp   IDSCON_8          ; Va a modo normal
```

```
; Tratar comando DAT_PP desde uC1

PARAM_1:  cjne  A,#DAT_PP,PARAM_2 ; Salta si no comando DAT_PP
          lcall PARIDS            ; Tratar comando
          jmp   IDSCMP            ; Vuelve a bucle de conexión

; Tratar comando C_IDS desde uC1

PARAM_2:  cjne  A,#C_IDS,PARAM_3  ; Salta si no comando C_IDS
          call  CONFIDS           ; Tratar comando
          jmp   IDSCMP            ; Vuelve a bucle de conexión

; Otro comando de uC1

PARAM_3:  mov   A,#4              ; Anota diag. sobre comando erróneo
          call  LOG_ESC
          jmp   IDSCMP            ; Vuelve a bucle de conexión



; MODO TRANSPARENTE TOTAL
; ---------------------------------------------------------------------------
; Tras un reset de la IDS, después de recibir un mensaje de desconexión o
; al no lograr conexión en el bucle de envío de AT. Solo se aceptan
; datos DAT_TP con comando AT, o comandos MODO_N (para desconexiones de
; uC1, que llevan a uC2 a reposo en modo normal) y MODO_T (que hace que
; se siga en MTT).

MODOTT:   mov   A,#60             ; Mensaje de diagnostico de entrada en MTT
          call  LOG_ESC

; Esperar aviso de uC1

MODOTT_0: jnb   BIT_COM,$         ; Espera datos de uC1
          clr   BIT_COM           ; Desactiva aviso de uC1

          call  LEECMD            ; Lee comando de uC1
          jz    MODOTT_0          ; Salta al bucle si error en lectura

; Ver si comando es MODO_N

          cjne  A,#MODO_N,MODOTT_1 ; Salta si no comando MODO_N
          mov   R2,#R_MDN          ; Envía respuesta a uC1 (sin datos)
          call  CMDUC1
          jmp   IDSCON_8           ; Va a modo normal

; Ver si comando en MODO_T

MODOTT_1: cjne  A,#MODO_T,MODOTT_2 ; Salta si no comando MODOTN
          mov   R2,#R_MDT          ; Envía respuesta a uC1 (sin datos)
          call  CMDUC1
          jmp   MODOTT_0           ; Sigue en MTT

; Ver si comando es DAT_TP

MODOTT_2: cjne  A,#DAT_TP,MODOTT_8 ; Si no DAT_TP salta

; Si comando es DAT_TP se comprueba si los datos son '@AT' o 'o/oAT'

MODOTT_3: call  B_RAM              ; Espera a bloquear SRAM
          jb    BIT_TMP,MODOTT_3
```

```
        clr   BIT_E              ; Borra indicador de error en SRAM
        mov   DPTR,#A_DATOS      ; Inicia puntero a datos en SRAM
        call  L_RAM              ; Lee dato de uC1
        jb    BIT_E,MODOTT_9     ; Si error salta
        cjne  A,#'@',MODOTT_4    ; Si no es '@' salta
        sjmp  MODOTT_5

MODOTT_4: cjne  A,#37,MODOTT_A   ; Si no es '%' salta

MODOTT_5: inc   DPTR             ; Apunta a segundo byte de datos
        call  L_RAM              ; Lee dato de uC1
        jb    BIT_E,MODOTT_9     ; Si error salta
        call  AMAYS
        cjne  A,#'A',MODOTT_A    ; Si no es 'A' salta

        inc   DPTR               ; Apunta a tercer byte de datos
        call  L_RAM              ; Lee dato de uC1
        jb    BIT_E,MODOTT_9     ; Si error salta
        call  AMAYS
        cjne  A,#'T',MODOTT_A    ; Si no es 'T' salta

; Enviar @AT a IDS y esperar respuesta acabada en C>>

        call  D_RAM              ; Libera memoria
        call  TXT                ; Enviar comando @AT a IDS
        db    '@AT',CR,ESC
        mov   R6,#141            ; Esperar como mucho 10seg la respuesta

MODOTT_6: mov   TL0,#0           ; Con T0=0 y R6=141 son 10.02seg
        mov   TH0,#0
        setb  TR0                ; Arranca temporizador

MODOTT_7: jb    BIT_RRDY,MODOTT_B ; Si se recibe algo de IDS sale de espera
        jnb   TF0,MODOTT_7
        clr   TR0                ; Para contador y borra fin de cuenta
        clr   TF0
        djnz  R6,MODOTT_6        ; Repite R6 veces: espera 10seg

        mov   A,#51              ; Mensaje diag. de conexión no establecida
        call  LOG_ESC
        sjmp  MODOTT_D           ; Salta a enviar respuesta de error a uC1

; Tratar error de comando no esperado de uC1

MODOTT_8: mov   A,#5             ; Anota diag. sobre comando erróneo
        call  LOG_ESC
        jmp   MODOTT_0           ; Vuelve a la espera de comando de uC1

; Tratar error en la lectura de datos de un comando DAT_TP de uC1

MODOTT_9: call  D_RAM            ; Libera memoria
        mov   A,#10              ; Anota error de lectura de datos
        call  LOG_ESC
        sjmp  MODOTT_D           ; Salta a enviar respuesta de error a uC1

; Tratar error de comando DAT_TP de uC1 que no contiene '@AT' o 'o/oAT'

MODOTT_A: call  D_RAM            ; Libera memoria
        sjmp  MODOTT_D           ; Salta a enviar respuesta de error a uC1

; Recibir respuesta de IDS a envío de '@AT'

MODOTT_B: clr   TR0              ; Para contador y borra fin de cuenta
```

```
        clr   TF0

        call  ESPCMM            ; Espera respuesta de IDS acabada en C>>
        jb    BIT_TMP,MODOTT_D  ; Salta si error de timeout

        mov   A,#61             ; Mensaje diag. sobre salida del MTT
        call  LOG_ESC
        mov   R2,#DAT_TR        ; Como datos transparentes
        call  ENVMSG            ; Envía respuesta a uC1: IDS conectada
        db    'IDS conectada',ESC

        jmp   IDSCMT            ; Va a modo transparente con IDS conectada

; Respuesta para uC1 si IDS no conectada por algún error

MODOTT_D: mov   R2,#DAT_TR      ; Como datos transparentes
        call  ENVMSG            ; Envía respuesta a uC1: IDS no conectada
        db    'IDS sigue desconectada',ESC

        jmp   MODOTT_0          ; Sigue en espera de AT


; ---------------------------------------------------------------------
;                    FUNCIONES DE LA APLICACIÓN
; ---------------------------------------------------------------------


; Estas funciones permiten actuar al SIR sobre la estación de registro. Así,
; pueden realizarse sobre la IDS acciones que habitualmente se llevan a cabo
; con una conexión directa de un OP al puerto serie de la estación. Para su
; confección debe ser conocido el protocolo de enlace con la estación de medida
; utilizada.


;.....................................................................
; Función: SHIDS. Pide cabecera a IDS (@SH ev) y lee datos de respuesta.
;          La función pilla y libera la memoria.
; Entrada: R6=Numero de evento
; Salida:  BIT_TMP = 1 si error en respuesta (TO, SVB, datos err)
;          R1,R0 = longitud del fichero - 512 = bytes por leer
; Modif.D: A, B, DPTR, DAT_L, DAT_H, R0, R1, R2, R3, BIT_E, ARIT_L, ARIT_H
;          BIT_TMP
; Usa:     TXT, RXB, ENVASC, B_RAM, E_RAM, D_RAM, ESPCMM, LOG_ESC
;.....................................................................

SHIDS:    mov   R3,#INTENT      ; Inicia contador de intentos

SHIDS_0:  mov   R2,#0           ; Inicia suma de SVB
        clr   BIT_E             ; Borra indicador error al escribir en SRAM

; Envía comando @SH ev a IDS

        call  TXT               ; Envía @SH
        db    '@SH',ESC
        mov   A,R6              ; Envía numero de evento
        call  ENVASC
        mov   A,#CR             ; Envía CR
        call  TXB

; Lee principio de respuesta: SOH,LSB-lonCab,MSB-lonCab

        call  RXB               ; Espera respuesta
        jnb   BIT_RTO,SHIDS_5X  ; Salta si error por timeout
```

```
        cjne  A,#1,SHIDS_7      ; Salta si primer byte recibido no es SOH

        call  RXB               ; Espera respuesta
        jnb   BIT_RTO,SHIDS_5X  ; Salta si error por timeout
        mov   DPL,A             ; Guarda LSB de longitud de cabecera

        call  RXB               ; Espera respuesta
        jnb   BIT_RTO,SHIDS_5X  ; Salta si error por timeout
        mov   DPH,A             ; Guarda MSB de longitud de cabecera

        mov   A,DPL            ; Salta si longitud no es 512 (0200h)
        jnz   SHIDS_7
        mov   A,#2
        xrl   A,DPH
        jnz   SHIDS_7

        mov   DPTR,#(A_DATOS+512)
        mov   DAT_L,DPL        ; #((A_DATOS+512)&255)
                               ; Valor de dirección final en DAT_H,DAT_L:
        mov   DAT_H,DPH        ; #((A_DATOS+512)/255)
                               ; dir final=A_DATOS+512

        mov   DPTR,#A_DATOS    ; Valor de dirección inicial
        sjmp  SHIDS_X          ; Pasa a leer longitud de fichero

; Tratar error de timeout en la lectura de un dato

SHIDS_5X: mov   A,#30          ; Mensaje diag. sobre timeout en la respuesta
        call  LOG_ESC
        jmp   SHIDS_8

; Tratar error en datos de la respuesta

SHIDS_7:  call  ESPCMM         ; Lee resto de respuesta: fin en C>>
        mov   A,#70            ; Mensaje diag. sobre error
        call  LOG_ESC
        jmp   SHIDS_8


; Lee longitud de fichero: bytes 2 y 3 de la cabecera

SHIDS_X:  call  RXB            ; Recibe dato de IDS
        jnb   BIT_RTO,SHIDS_5X ; Salta si error por timeout
        mov   ARIT_L,A         ; Guarda parte baja de la longitud de fichero
        add   A,R2             ; Suma dato a SVB
        mov   R2,A

SHIDS_A:  call  B_RAM          ; Espera a bloquear SRAM
        jb    BIT_TMP,SHIDS_A

        mov   A,ARIT_L         ; Recupera dato
        call  E_RAM            ; Escribe en SRAM el dato

        call  D_RAM            ; Libera memoria
        inc   DPTR             ; Incrementa y guarda puntero a SRAM

        call  RXB              ; Recibe dato de IDS
        jnb   BIT_RTO,SHIDS_5X ; Salta si error por timeout
        mov   ARIT_H,A         ; Guarda parte alta de la longitud de fichero
        add   A,R2             ; Suma dato a SVB
        mov   R2,A

SHIDS_C:  call  B_RAM          ; Espera a bloquear SRAM
        jb    BIT_TMP,SHIDS_C
```

```
        mov   A,ARIT_H          ; Recupera dato
        call  E_RAM             ; Escribe en SRAM el dato

        call  D_RAM             ; Libera memoria
        inc   DPTR              ; Incrementa y guarda puntero a SRAM

; Lee resto de datos de la cabecera

SHIDS_1:  call  RXB               ; Recibe dato de IDS
        jnb   BIT_RTO,SHIDS_5 ; Salta si error por timeout
        mov   B,A               ; Guarda dato
        add   A,R2              ; Suma dato a SVB
        mov   R2,A

SHIDS_2:  call  B_RAM             ; Espera a bloquear SRAM
        jb    BIT_TMP,SHIDS_2

        mov   A,B               ; Recupera dato
        call  E_RAM             ; Escribe en SRAM el dato

        call  D_RAM             ; Libera memoria
        inc   DPTR              ; Incrementa y guarda puntero a SRAM

        mov   A,DPH             ; Si DPTR no es direc. final repite bucle
        xrl   A,DAT_H
        jnz   SHIDS_1
        mov   A,DPL
        xrl   A,DAT_L
        jnz   SHIDS_1           ; Si DPTR = dirección final sigue

; Lee y comprueba SVB

        call  RXB               ; Recibe valor de SVB de IDS
        jnb   BIT_RTO,SHIDS_5 ; Salta si error por timeout
        xrl   A,R2              ; Si SVB calculada distinta de la leída salta
        jnz   SHIDS_6

; Calcula numero de bytes que quedan por leer del fichero (long-512)

        mov   DPTR,#512         ; ARIT_H,ARIT_L-512->R0,R1
        call  RESTA
        jnc   SHIDS_3           ; Si ARIT_H/L>=512 salta

        clr   A                 ; Si ARIT_H/L<512 ->R0=R1=0
        mov   ARIT_L,A
        mov   ARIT_H,A

SHIDS_3:  mov   R0,ARIT_L
        mov   R1,ARIT_H

; Lee resto de respuesta hasta C>>

        call  ESPCMM            ; Lee resto de respuesta: fin en C>>
        jb    BIT_TMP,SHIDS_8 ; Si error de timeout salta

; Final correcto

        jnb   BIT_E,SHIDS_4     ; Si no error al escribir datos salta
        mov   A,#11             ; Mensaje sobre error
        call  LOG_ESC

SHIDS_4:  clr   BIT_TMP           ; No hay error de timeout, SVB o dat err.
```

```
        ret                         ; Acaba

; Tratar error de timeout en la lectura de un dato

SHIDS_5:  mov   A,#30               ; Mensaje diag. sobre timeout en la respuesta
        call  LOG_ESC
        sjmp  SHIDS_8

; Tratar error en la verificación de SVB

SHIDS_6:  mov   A,#80               ; Mensaje diag. sobre error de SVB
        call  LOG_ESC
        call  ESPCMM               ; Lee resto de respuesta: fin en C>>

; Ver si queda otro intento

SHIDS_8:  djnz  R3,SHIDS_Z          ; Otro intento, si quedan

        jnb   BIT_E,SHIDS_9        ; Si no error al escribir datos salta
        mov   A,#11                ; Mensaje sobre error
        call  LOG_ESC

SHIDS_9:  setb  BIT_TMP             ; Hay error de timeout, SVB o dat err.
        ret

SHIDS_Z:  jmp   SHIDS_0


;.....................................................................
; Función: SBIDS. Pide un bloque a IDS (@SB ev,bl) y lee datos de respuesta.
;          La función pilla y libera la memoria.
; Entrada: DPTR= puntero a dirección inicial para datos en SRAM
;          R6=numero de evento, R5=numero de bloque
; Salida:  DPTR= puntero a dirección siguiente a la del ultimo dato escrito
;          BIT_TMP = 1 si error en respuesta (TO, SVB, datos err)
; Modif.D: A, B, DPTR, ARIT_L, ARIT_H, R0, R1, R2, R3, BIT_E, DAT_L, DAT_H
;          BIT_TMP
; Usa:     TXT, RXB, ENVASC, B_RAM, E_RAM, D_RAM, LOG_ESC,SUMA,ESPCMM
;.....................................................................

SBIDS:    mov   R3,#INTENT          ; Inicia contador de intentos
        mov   DAT_L,DPL            ; Guarda dirección inicial a memoria
        mov   DAT_H,DPH

SBIDS_0:  mov   R2,#0               ; Inicia suma de SVB
        clr   BIT_E                ; Borra indicador de error al escribir

; Envía comando @SB ev,bl a IDS

        call  TXT                  ; Envía @SH
        db    '@SB',ESC
        mov   A,R6                 ; Envía numero de evento
        call  ENVASC
        mov   A,#','               ; Envía una coma
        call  TXB
        mov   A,R5                 ; Envía numero de bloque
        call  ENVASC
        mov   A,#CR                ; Envía CR
        call  TXB

        mov   DPL,DAT_L            ; Recupera dirección inicial a memoria
        mov   DPH,DAT_H
```

```
; Lee principio de respuesta: SOH

        call  RXB               ; Espera respuesta
        jnb   BIT_RTO,SBIDS_5   ; Salta si error por timeout
        cjne  A,#1,SBIDS_7      ; Salta si primer byte recibido no es SOH

; Lee longitud de datos del bloque

        call  RXB               ; Espera respuesta
        jnb   BIT_RTO,SHIDS_5   ; Salta si error por timeout
        mov   ARIT_L,A          ; Guarda LSB de longitud de bloque

        call  RXB               ; Espera respuesta
        jnb   BIT_RTO,SHIDS_5   ; Salta si error por timeout
        mov   ARIT_H,A          ; Guarda MSB de longitud de bloque

        call  SUMA              ; ARIT_H,ARIT_L=long+dir_inicial=dir_final
                                ; (dir_inicial esta en DPTR)

; Leer resto de datos del bloque

SBIDS_1:  call  RXB             ; Recibe dato de IDS
        jnb   BIT_RTO,SBIDS_5   ; Salta si error por timeout
        mov   B,A               ; Guarda dato
        add   A,R2              ; Suma dato a SVB
        mov   R2,A

SBIDS_2:  call  B_RAM           ; Espera a bloquear SRAM
        jb    BIT_TMP,SBIDS_2

        mov   A,B               ; Recupera dato
        call  E_RAM             ; Escribe en SRAM el dato

        call  D_RAM             ; Libera memoria
        inc   DPTR              ; Incrementa y guarda puntero a SRAM

        mov   A,DPH             ; Si DPTR no es direc. final repite bucle
        xrl   A,ARIT_H
        jnz   SBIDS_1
        mov   A,DPL
        xrl   A,ARIT_L
        jnz   SBIDS_1           ; Si DPTR = dirección final sigue

; Lee y comprueba SVB

        call  RXB               ; Recibe valor de SVB de IDS
        jnb   BIT_RTO,SBIDS_5   ; Salta si error por timeout
        xrl   A,R2              ; Si SVB calculada distinta de la leída salta
        jnz   SBIDS_6

; Lee resto de respuesta hasta C>>

        call  ESPCMM            ; Lee resto de respuesta: fin en C>>
        jb    BIT_TMP,SBIDS_8   ; Si error de timeout salta

; Final correcto

        jnb   BIT_E,SBIDS_4     ; Si no error al escribir datos salta
        mov   A,#12             ; Mensaje sobre error
        call  LOG_ESC

SBIDS_4:  clr   BIT_TMP         ; No hay error de timeout, SVB o dat err.
        ret                     ; Acaba
```

```
; Tratar error de timeout en la lectura de un dato

SBIDS_5:  mov   A,#31          ; Mensaje diag. sobre timeout en la respuesta
          call  LOG_ESC
          sjmp  SBIDS_8

; Tratar error en datos de la respuesta

SBIDS_7:  call  ESPCMM         ; Lee resto de respuesta: fin en C>>
          mov   A,#71          ; Mensaje diag. sobre error de datos
          call  LOG_ESC
          jmp   SBIDS_8

; Tratar error en la verificación de SVB

SBIDS_6:  mov   A,#81          ; Mensaje diag. sobre error de SVB
          call  LOG_ESC

; Ver si queda otro intento

SBIDS_8:  djnz  R3,SBIDS_Z     ; Otro intento, si quedan

          jnb   BIT_E,SBIDS_9  ; Si no error al escribir datos salta
          mov   A,#12          ; Mensaje sobre error
          call  LOG_ESC

SBIDS_9:  setb  BIT_TMP        ; Hay error de timeout, SVB o dat err.
          ret

SBIDS_Z:  jmp   SBIDS_0


;.....................................................................
; Función: PARAM. Tratar comandos de parametrización de IDS
; Entrada:
; Salida:
; Modif.D: BIT_E,DPTR,A,R2,BIT_0,MAXTO
; Usa:     B_RAM,L_RAM,D_RAM,BORRAFI,PAR_CAL,ENVMSG,LOG_ESC,ENVPAR
;.....................................................................

; Leer código de comando de parametrización (BYTE 0 de datos)

PARIDS:   mov   DPTR,#A_DATOS

PARID_1:  call  B_RAM          ; Espera a bloquear SRAM
          jb    BIT_TMP,PARID_1

          clr   BIT_E          ; Borra indicador de error
          call  L_RAM          ; Lee byte de SRAM
          call  D_RAM          ; libera memoria
          jb    BIT_E,PARID_X  ; Si error salta

; Tratar comando de calibración de IDS

          cjne  A,#PAR_CAL,PARID_Y

          mov   MAXTO,#0       ; TO de unos 18seg
          clr   BIT_0          ; No es borrar fichero
          mov   DPTR,#DAT_CAL  ; Secuencia de calibración
          call  ENVPAR         ; Envía secuencia a IDS
          jb    BIT_0,PARID_W  ; Si hubo error en envío

          mov   R2,#DAT_PR     ; Envía respuesta a uC1
```

```
        call  ENVMSG
        db    'Fich.CAL creado',ESC
        ret

; Tratar comando de borrado de ficheros de IDS

PARID_Y:  cjne  A,#PAR_BOR,PARID_G

        call  BORRAFI          ; Envía a IDS parámetros de borrado
        jb    BIT_0,PARID_W    ; Si hubo error en envío

        mov   R2,#DAT_PR       ; Envía respuesta a uC1
        call  ENVMSG
        db    'Memoria IDS borrada',ESC
        ret

; Error al leer comando de uC1

PARID_X:  mov   A,#13            ; Mensaje de diagnostico
        call  LOG_ESC
        ret

; Error al enviar un parámetro a IDS (TO en respuesta de IDS)

PARID_W:  mov   R2,#DAT_PR       ; Envía respuesta a uC1
PARID_W1: call  ENVMSG
        db    'TO en resp. IDS',ESC
        call  TXT              ; Enviar START a IDS
        db    37,'START',CR,ESC
        ret

; Tratar comando de test de tiempo (Fichero calibración GPS)

PARID_G:  cjne  A,#PAR_GPS,PARID_H

        mov   MAXTO,#0         ; TO de unos 18seg
        clr   BIT_0            ; No es borrar fichero
        mov   DPTR,#DAT_GPS    ; Secuencia de calibración GPS
        call  ENVPAR           ; Envía secuencia a IDS
        jb    BIT_0,PARID_W    ; Si hubo error en envío

        mov   R2,#DAT_PR       ; Envía respuesta a uC1
        call  ENVMSG
        db    'Fich.GPS creado',ESC
        ret

; Tratar comando para sincronizar GPS-IDS

PARID_H:  cjne  A,#PAR_SIN,PARID_Z

        mov   R2,#R_MDN        ; Indica a uC1 que pase a modo normal
        call  CMDUC1

        clr   BIT_0            ; No es borrar fichero
        mov   DPTR,#DAT_SIN    ; Secuencia de sincronización GPS_IDS
        call  ENVPAR           ; Envía secuencia a IDS
        mov   R2,#DAT_AV       ; Respuesta para uC1 como aviso
        jb    BIT_0,PARID_W1   ; Si hubo error en envío

        call  TXT
        db    37,'TIMESET',CR,ESC
        call  ESPCUR           ; Espera respuesta ...CUR
        jb    BIT_TMP,PARID_W1
```

```
        call  IDSMEM          ; Resto de respuesta hasta C>> pasa a memoria

        call  TXT             ; Enviar START a IDS
        db    37,'START',CR,ESC
        ret

; Comando de parametrización no esperado

PARID_Z:  mov   A,#6            ; Anota diag. sobre comando erróneo
        call  LOG_ESC

        mov   R2,#DAT_PR      ; Envía respuesta a uC1
        call  ENVMSG
        db    'Comando param. no valido',ESC
        ret


;.....................................................................
; Función: CONFIDS. Enviar fichero de configuración a la IDS y esperar
;          respuesta, que se pasa a la memoria.
; Entrada:
; Salida:
; Modif.D: A,B,R1,R2,DPTR,ARIT_L,ARIT_H,FIN_L,FIN_H,BIT_E
; Usa:     B_RAM,E_RAM,L_RAM,D_RAM,TXB,RXB,ESPCMM,ENVPAR,TXT,ENVMSG,SUMA,AUC1
;.....................................................................

; Enviar secuencia TS,STOP,NE

CONFIDS:  mov   MAXTO,#141              ; TO de 10seg
        mov   DPTR,#DAT_CONF
        clr   BIT_0                   ; No se borra ficheros
        call  ENVPAR
        jnb   BIT_0,CONFID_0          ; Si no hay error de timeout salta
CONFI_X0: jmp   CONFID_F                ; En otro caso

; Espera a bloquear SRAM

CONFID_0: call  B_RAM
        jb    BIT_TMP,CONFID_0

; Enviar 37,ASU,CR

        call  TXT
        db    37,'ASU',CR,ESC

        call  DLY

; Enviar primeros 166 bytes de la SRAM con SVB

        mov   B,#0                    ; Inicia acumulador para SVB
        clr   BIT_E                   ; Borra indicador de error
        mov   DPTR,#(A_DATOS+166)     ; Guarda dirección final de datos
        mov   FIN_L,DPL               ; Guarda parte baja
        mov   FIN_H,DPH               ; Guarda parte alta
        mov   DPTR,#A_DATOS           ; Carga en DPTR direc inicio datos

        mov   A,#0A6h                 ; Envía A6 00
        call  TXB
        clr   A
        call  TXB

CONFID_3: call  L_RAM                   ; Leer byte de SRAM
        call  TXB                     ; Enviar byte a IDS
        add   A,B                     ; B=B+dato
```

```
        mov   B,A
        inc   DPTR                    ; Siguiente byte

        mov   A,DPH                   ; Si DPTR distinto de FIN_H,L sigue en
bucle
        xrl   A,FIN_H
        jnz   CONFID_3
        mov   A,DPL
        xrl   A,FIN_L
        jnz   CONFID_3

        jnb   BIT_E,CONFID_H          ; Si no error al leer longitud salta
        jmp   CONFID_Y                ; Si hay error

CONFID_H: call  D_RAM                   ; Libera memoria

; Enviar SVB y esperar respuesta de IDS

        mov   A,B
        call  TXB

        call  ESPCMM                  ; Espera C>>
        jb    BIT_TMP,CONFID_Z        ; Salta si error de timeout

; Espera a bloquear SRAM

CONFID_4: call  B_RAM
        jb    BIT_TMP,CONFID_4

; Enviar 37,AA,CR

        call  DLY

        call  TXT
        db    37,'AA',CR,ESC

        call  DLY

; Enviar resto de fichero de configuración

        mov   DPTR,#(A_CMD+1)         ; Apunta a byte bajo de longitud
        call  L_RAM                   ; Lee dato
        mov   ARIT_L,A                ; Guarda parte baja
        inc   DPTR                    ; Apunta a byte alto de longitud
        call  L_RAM                   ; Lee dato
        mov   ARIT_H,A                ; Guarda parte alta

        jb    BIT_E,CONFID_Y          ; Si hay error al leer datos salta

        mov   DPTR,#A_DATOS           ; Guarda dirección final de datos
        call  SUMA                    ; ARIT_L/H = dirección final de datos

        mov   DPTR,#(A_DATOS+166)     ; Carga en DPTR direc inicio datos

        mov   A,#30h                  ; Envía 30 01
        call  TXB
        mov   A,#1
        call  TXB

CONFID_6: call  L_RAM                   ; Leer byte de SRAM
        call  TXB                     ; Enviar byte a IDS
        inc   DPTR                    ; Siguiente byte

        mov   A,DPH                   ; Si DPTR distinto de ARIT_H,L
```

```
        xrl   A,ARIT_H                ;   sigue en bucle
        jnz   CONFID_6
        mov   A,DPL
        xrl   A,ARIT_L
        jnz   CONFID_6

        jb    BIT_E,CONFID_Y          ; Si hay error al leer datos

        call  D_RAM                   ; Libera memoria

; Enviar @ y esperar respuesta de IDS

        mov   A,#'@'
        call  TXB

        call  ESPCMM                  ; Espeta C>>
        jnb   BIT_TMP,CONFID_E        ; Salta si no hay error de timeout

; Error al enviar un parámetro a IDS (TO en respuesta de IDS)

CONFID_Z: mov   A,#32                   ; Mensaje diagnostico sobre timeout
        call  LOG_ESC
CONFID_F: mov   R2,#DAT_PR              ; Envía respuesta a uC1
        call  ENVMSG
        db    'TO en resp. CIDS',ESC
        ret

; Error al leer o escribir datos en memoria SRAM

CONFID_Y: call  D_RAM                   ; Libera SRAM
        mov   A,#14                   ; Anota el error
        call  LOG_ESC
        mov   R2,#DAT_PR              ; Envía respuesta a uC1
        call  ENVMSG
        db    'Err L/E SRAM',ESC
        ret

; Inicia contador de intentos de petición en caso de SVB errónea

CONFID_E: mov   R1,#INTENT

; Espera a bloquear SRAM

CONFID_G: call  B_RAM
        jb    BIT_TMP,CONFID_G

; Enviar 37,DSU,CR

        call  TXT
        db    37,'DSU',CR,ESC

; Esperar inicio de datos respuesta: 37,DSU,CR,LF,A6,00,166 bytes datos,SVB

CONFID_7: call  RXB                     ; Recibe dato de IDS
        jnb   BIT_RTO,CONFID_Z        ; Si timeout al recibir salta
        jnz   CONFID_7                ; Si no es 0 espera otro byte

; Recibir primeros 166 bytes de datos

        mov   B,#0                    ; Inicia acumulador para SVB
        clr   BIT_E                   ; Borra indicadores de error
        mov   DPTR,#A_DATOS           ; Inicia puntero de datos
```

```
CONFID_8: call  RXB                     ; Recibe byte de IDS
        jnb   BIT_RTO,CONFID_Z          ; Si timeout al recibir salta
        call  E_RAM                     ; Escribir byte de SRAM
        add   A,B                       ; B=B+dato
        mov   B,A
        inc   DPTR                      ; Siguiente byte

        mov   A,DPH                     ; Si DPTR distinto de FIN_H,L
        xrl   A,FIN_H                   ;  sigue en bucle
        jnz   CONFID_8
        mov   A,DPL
        xrl   A,FIN_L
        jnz   CONFID_8

        jb    BIT_E,CONFID_Y            ; Si hay error al escribir datos

        call  D_RAM                     ; Libera memoria

; Recibir SVB y comparar con la calculada

        call  RXB
        xrl   A,B                       ; Para ver si A=B
        jz    CONFID_9                  ; Salta si no hay error de SVB

; Error al comprobar SVB leída de IDS

        mov   A,#82                     ; Mensaje diag. sobre error de SVB
        call  LOG_ESC
        djnz  R1,CONFID_K               ; Si quedan intentos repite
        mov   R2,#DAT_PR                ; Envía respuesta a uC1
        call  ENVMSG
        db    'Error de SVB',ESC
        ret

CONFID_K: jmp   CONFID_4

; Espera a bloquear SRAM y espera C>>

CONFID_9: call  B_RAM
        jb    BIT_TMP,CONFID_9

        call  ESPCMM

; Enviar 37,DA,CR

        call  TXT
        db    37,'DA',CR,ESC

; Esperar inicio de datos respuesta: 37,DA,CR,LF,30,01, bytes datos

CONFID_A: call  RXB                     ; Recibe dato de IDS
        jb    BIT_RTO,CONFID_I          ; Si no timeout al recibir salta
CONFID_J: jmp   CONFID_Z                ; En caso de timeout

CONFID_I: cjne  A,#1,CONFID_A           ; Si no es 1 espera otro byte

; Recibir resto de bytes

        mov   DPTR,#(A_DATOS+166)       ; Carga en DPTR direc inicio datos

CONFID_B: call  RXB                     ; Recibe byte de IDS
        jnb   BIT_RTO,CONFID_J          ; Si timeout al recibir salta
        call  E_RAM                     ; Escribir byte de SRAM
        inc   DPTR                      ; Siguiente byte
```

```
        mov   A,DPH                  ; Si DPTR distinto de ARIT_H,L
        xrl   A,ARIT_H               ;   sigue en bucle
        jnz   CONFID_B
        mov   A,DPL
        xrl   A,ARIT_L
        jnz   CONFID_B

        call  ESPCMM                 ; Recibe resto de respuesta de IDS

CONFID_D: jnb   BIT_E,CONFID_L         ; Si no hay error al escribir datos
        jmp   CONFID_Y               ; En caso de error al escribir datos

CONFID_L: call  D_RAM                  ; Libera memoria

; Poner comando para uC1 y activar BIT_COM de uC1

        mov   R2,#R_CIDS             ; Comando para uc1
        call  AUC1
        call  B_COM

; Envía 37,START,CR a IDS y espera respuesta

        call  TXT
        db    37,'START',CR,ESC
        call  ESPCMM
        jb    BIT_TMP,CONFID_J
        ret                          ; Acaba rutina y vuelve


;.....................................................................
; Función: ENVPAR. Envía secuencia de parámetros a IDS y espera respuestas
;          Al acabar se restaura el máximo tiempo de espera a 5seg.
; Entrada: DPTR = dirección de secuencia con parámetros
;          BIT_0=1 para borrar ficheros y 0 en otro caso
; Salida:  BIT_0=1 si error de timeout en alguna respuesta de la IDS
; Modif.D: A,BIT_0,R2,DPTR,DAT_L,DAT_H,MAXTO
; Usa:     TXT,DLY,ESPCMM,ESPMM,LOG_ESC
;.....................................................................
; Formato de una secuencia de comandos:
; DIRECC: db    Numero_comandos,texto_comando1,CR,Texto_comando2,CR,..
; Puede no indicarse un comando y se envía solo el CR

DAT_CAL:  db    6,'TS',CR,'STOP',CR,'DSU',CR,'DMS',CR,'SCGA2',CR,'START',CR
DAT_BOR:  db    5,'TS',CR,'STOP',CR,'DELETEA',CR,'NEW',CR,'START',CR
DAT_GPS:  db    5,'TS',CR,'STOP',CR,'DMS',CR,'TT',CR,'START',CR
DAT_CONF: db    3,'TS',CR,'STOP',CR,'NE',CR
DAT_SIN:  db    2,'TS',CR,'STOP',CR

;.....................................................................

ENVPAR:   mov   DAT_L,DPL        ; Guarda dirección inicial a secuencia
        mov   DAT_H,DPH

ENVPAR1:  mov   DPL,DAT_L        ; Recupera dirección inicial a secuencia
        mov   DPH,DAT_H
        clr   A                 ; Carga en R2 el numero de comandos para IDS
        movc  A,@A+DPTR
        mov   R2,A

ENVPAR2:  mov   A,#37            ; Envía un por ciento
        call  TXB
ENVPAR7:  inc   DPTR             ; Apunta a siguiente carácter
        clr   A
```

```
        movc  A,@A+DPTR        ; Lee siguiente carácter
        call  TXB              ; Envía carácter
        xrl   A,#CR            ; Si no es CR vuelve al bucle
        jnz   ENVPAR7

        jnb   BIT_0,ENVPAR5    ; Si no esta borrando fichero salta o
        mov   A,#3             ; si no esta enviando DELETEA salta
        xrl   A,R2
        jnz   ENVPAR5

        call  ESPMM            ; Espera >>
        call  DLY              ; Espera un poco
        mov   A,#'Y'           ; Envía 'Y',CR
        call  TXB
        mov   A,#CR
        call  TXB

ENVPAR5:  call  ESPCMM          ; Espera C>>
ENVPAR6:  jb    BIT_TMP,ENVPAR3 ; Salta si error de timeout

        call  DLY              ; Espera un poco

        djnz  R2,ENVPAR2       ; Si quedan comandos para IDS vuelve al bucle

        clr   BIT_0            ; No hay error
        sjmp  ENVPAR4          ; Saltar al final

ENVPAR3:  mov   A,#33           ; Mensaje diagnostico sobre timeout
        call  LOG_ESC
        setb  BIT_0            ; Hay error de timeout

ENVPAR4:  mov   MAXTO,#70       ; Restaura TO en recepción por defecto (5s)
        ret


;.....................................................................
; Función: BORRAFI. Borrar todos los ficheros de eventos en la IDS
; Entrada:
; Salida:  BIT_0=1 si error de timeout en alguna respuesta de la IDS
; Modif.D: MAXTO,BIT_0,DPTR
; Usa:     ENVPAR
;.....................................................................

BORRAFI:  mov   MAXTO,#141      ; TO de unos 10seg
        setb  BIT_0            ; Indica que se va a borrar fichero
        mov   DPTR,#DAT_BOR    ; Secuencia de borrado
        call  ENVPAR           ; Envía secuencia a IDS
        ret


;.....................................................................
; Función: ESPCMM. Espera y lee fin de respuesta C>> dado por IDS
;          ESPMM.  Espera y lee fin de respuesta >> dado por IDS
;          ESPCUR. Espera y lee respuesta hasta ...CUR...
; Entrada:
; Salida:  BIT_TMP=1 si hay error de timeout y 0 en otro caso
; Modif.D: A, BIT_TMP
; Usa:     RXB, AMAYS, LOG_ESC
;.....................................................................

ESPCMM:   call  RXB              ; Recibe dato de IDS
        jnb   BIT_RTO,ESPCMM_1 ; Si timeout al recibir salta
        call  AMAYS            ; Pasa byte a mayúsculas para comparar
        cjne  A,#'C',ESPCMM    ; Si no es 'C' va al principio
```

```
        call  RXB               ; Recibe dato de IDS
        jnb   BIT_RTO,ESPCMM_1  ; Si timeout al recibir salta
        cjne  A,#'>',ESPCMM     ; Si no es '>' va al principio

        call  RXB               ; Recibe dato de IDS
        jnb   BIT_RTO,ESPCMM_1  ; Si timeout al recibir salta
        cjne  A,#'>',ESPCMM     ; Si no es '>' va al principio

        sjmp  ESPCMM_0

ESPMM:   call  RXB              ; Recibe dato de IDS
        jnb   BIT_RTO,ESPCMM_1  ; Si timeout al recibir salta
        cjne  A,#'>',ESPMM      ; Si no es '>' va al principio

        call  RXB               ; Recibe dato de IDS
        jnb   BIT_RTO,ESPCMM_1  ; Si timeout al recibir salta
        cjne  A,#'>',ESPMM      ; Si no es '>' va al principio

        sjmp  ESPCMM_0

ESPCUR:  call  RXB              ; Recibe dato de IDS
        jnb   BIT_RTO,ESPCMM_1  ; Si timeout al recibir salta
        cjne  A,#'C',ESPCUR     ; Si no es 'C' va al principio

        call  RXB               ; Recibe dato de IDS
        jnb   BIT_RTO,ESPCMM_1  ; Si timeout al recibir salta
        cjne  A,#'u',ESPCUR     ; Si no es 'u' va al principio

        call  RXB               ; Recibe dato de IDS
        jnb   BIT_RTO,ESPCMM_1  ; Si timeout al recibir salta
        cjne  A,#'r',ESPCUR     ; Si no es 'r' va al principio

ESPCMM_0: clr   BIT_TMP         ; No hay error de timeout
        ret

ESPCMM_1: mov   A,#34           ; Mensaje diagnostico sobre timeout
        call  LOG_ESC
        setb  BIT_TMP
        ret


;.....................................................................
; Función: ESPIDS. Espera respuesta de IDS. La respuesta se lee pero no
;          se escribe en memoria. Si la respuesta no es mensaje de
;          desconexión se considera respuesta a NE y se devuelve n. eventos
; Entrada:
; Salida:  BIT_0 a 1 si se ha recibido mensaje desconexión y 0 si normal
;          BIT_1 a 1 si error de timeout y 0 si no hay error
;          A= numero de eventos si hay respuesta a NE
; Modif.D: BIT_0, BIT_1, A,B
; Usa:     RXB, AMAYS, LOG_ESC
;.....................................................................

; Recibir primer byte

ESPIDS:  call  RXB              ; Recibe byte de IDS
        jnb   BIT_RTO,ESPIDS_2  ; Si timeout al recibir salta
        mov   B,A               ; Guarda dato como numero de eventos

; Recibir a partir de segundo byte

ESPIDS_X: call  RXB             ; Recibe byte de IDS
        jnb   BIT_RTO,ESPIDS_2  ; Si timeout al recibir salta
```

```
; Tratar respuesta normal a NE: ...C>>

ESPIDS_0: call  AMAYS            ; Pasa byte a mayúsculas para comparar
        cjne  A,#'C',ESPIDS_1 ; Si byte no es 'C' salta

        call  RXB              ; Recibe byte de IDS
        jnb   BIT_RTO,ESPIDS_2 ; Si timeout al recibir salta
        cjne  A,#'>',ESPIDS_0 ; Si byte no es '>' salta

        call  RXB              ; Recibe byte de IDS
        jnb   BIT_RTO,ESPIDS_2 ; Si timeout al recibir salta
        cjne  A,#'>',ESPIDS_0 ; Si byte no es '>' salta

        clr   BIT_0            ; Indica respuesta normal a NE recibida
        clr   BIT_1            ; No hay timeout
        mov   A,B              ; Recupera numero de eventos
        ret                    ; Acaba

; Tratar respuesta a desconexión: ...X1

ESPIDS_1: cjne  A,#'X',ESPIDS_X ; Si byte no es 'X' salta

        call  RXB              ; Recibe byte de IDS
        jnb   BIT_RTO,ESPIDS_2 ; Si timeout al recibir salta
        cjne  A,#'1',ESPIDS_0 ; Si byte no es '1' salta

        setb  BIT_0            ; Indica desconexión
        clr   BIT_1            ; No hay timeout
        clr   A                ; No hay eventos
        ret                    ; Acaba

; Tratar error de timeout

ESPIDS_2: mov   A,#35            ; Mensaje diag. sobre timeout en la respuesta
        call  LOG_ESC
        setb  BIT_1            ; Hay timeout
        clr   BIT_0
        clr   A                ; No hay eventos
        ret

;.....................................................................
; Función: IDSMEM. Lee respuesta de IDS y pasa datos transparentes a uC1
;          La función pilla y libera la memoria. Se esperan respuestas
;          acabadas en C>> (normales) o en X1 (de desconexión). Si hay error
;          al recibir la respuesta se pone ? al final de los datos en memoria
; Entrada: R2=comando para uC1
; Salida:  BIT_TMP a 1 si se ha recibido mensaje desconexión y no hubo TO
;          BIT_TMP a 0 si respuesta normal o error de TO
; Modif.D: DPTR, BIT_E, A
; Usa:     RXB, B_RAM, D_RAM, E_RAM, AMAYS, AUC1, B_COM, LOG_ESC
;.....................................................................

;*** No se comprueba si se supera el limite de memoria al enviar a uC1

IDSMEM:   mov   DPTR,#A_DATOS    ; Puntero a zona de datos
        clr   BIT_E            ; Indica que no hay error de momento

; Leer primer byte de final de respuesta de la IDS y escribirlo en memoria

IDSMEM_1: call  RXB              ; Recibe byte de IDS
        jnb   BIT_RTO,IDSMEM_R ; Salta si error de timeout

IDSMEM_2: call  B_RAM            ; Espera a bloquear SRAM
```

```
        jb    BIT_TMP,IDSMEM_2

        call  E_RAM            ; Escribe dato
        call  D_RAM            ; Libera SRAM
        inc   DPTR             ; Incrementa puntero de datos

; Recibir respuesta normal acabada en ...C>>

IDSMEM_4: call  AMAYS           ; Pasa byte a mayúsculas para comparar
        cjne  A,#'C',IDSMEM_A ; Si byte no es 'C' salta

        call  RXB              ; Recibe byte de IDS
        jnb   BIT_RTO,IDSMEM_R ; Salta si error de timeout

IDSMEM_5: call  B_RAM           ; Espera a bloquear SRAM
        jb    BIT_TMP,IDSMEM_5

        call  E_RAM            ; Escribe dato
        call  D_RAM            ; Libera SRAM
        inc   DPTR             ; Incrementa puntero de datos
        cjne  A,#'>',IDSMEM_4 ; Si byte no es '>' salta

        call  RXB              ; Recibe byte de IDS
        jnb   BIT_RTO,IDSMEM_R ; Salta si error de timeout

IDSMEM_7: call  B_RAM           ; Espera a bloquear SRAM
        jb    BIT_TMP,IDSMEM_7

        call  E_RAM            ; Escribe dato
        call  D_RAM            ; Libera SRAM
        inc   DPTR             ; Incrementa puntero de datos
        cjne  A,#'>',IDSMEM_4 ; Si byte no es '>' salta

        sjmp  IDSMEM_9

IDSMEM_R: sjmp  IDSMEM_O

; Recibir respuesta de desconexión acabada en ...X1

IDSMEM_A: cjne  A,#'X',IDSMEM_1 ; Si byte no es 'X' salta

        call  RXB              ; Recibe byte de IDS
        jnb   BIT_RTO,IDSMEM_O ; Salta si error de timeout

IDSMEM_B: call  B_RAM           ; Espera a bloquear SRAM
        jb    BIT_TMP,IDSMEM_B

        call  E_RAM            ; Escribe dato
        call  D_RAM            ; Libera SRAM
        inc   DPTR             ; Incrementa puntero de datos
        cjne  A,#'1',IDSMEM_4 ; Si byte no es '1' salta

; Tratar respuesta de desconexión

        call  AUC1             ; Actualiza comando (R2) y longitud para uC1

        jnb   BIT_E,IDSMEM_E   ; Si no error al escribir datos salta
        mov   A,#17            ; Mensaje sobre error
        call  LOG_ESC

IDSMEM_E: mov   A,#52           ; Mensaje para diagnostico sobre respuesta
        call  LOG_ESC

        call  B_COM            ; Avisa a uC1
```

```
        setb  BIT_TMP          ; Indica mensaje de desconexión recibido
        ret                    ; Acaba

; Tratar respuesta normal

IDSMEM_9: call  AUC1             ; Actualiza comando (R2) y longitud para uC1

        jnb   BIT_E,IDSMEM_0   ; Si no error al escribir datos salta
        mov   A,#17            ; Mensaje sobre error
        call  LOG_ESC

IDSMEM_0: call  B_COM            ; Avisa a uC1

        clr   BIT_TMP          ; Indica respuesta normal recibida
        ret                    ; Acaba

; Tratar error de timeout

IDSMEM_O: call  B_RAM            ; Espera a bloquear SRAM
        jb    BIT_TMP,IDSMEM_O

        mov   A,#'?'           ; Escribe una interrogación como ultimo dato
        call  E_RAM
        call  D_RAM            ; Libera SRAM
        inc   DPTR             ; Incrementa puntero de datos

        call  AUC1             ; Actualiza comando (R2) y longitud para uC1

        jnb   BIT_E,IDSMEM_Q   ; Si no error al escribir datos salta
        mov   A,#17            ; Mensaje sobre error
        call  LOG_ESC

IDSMEM_Q: mov   A,#36            ; Mensaje diag. sobre timeout en la respuesta
        call  LOG_ESC

        call  B_COM            ; Avisa a uC1

        clr   BIT_TMP          ; Indica no respuesta a desconexión recibida
        ret                    ; Acaba


;.....................................................................
; Función: MEMIDS. Lee datos transparentes desde uC1 y los envía a la IDS
;          La función pilla y libera la memoria. Si los datos no acaban en
;          CR, se envía un CR tras los datos a la IDS.
; Entrada:
; Salida:
; Modif.D: BIT_E, DPTR, ARIT_L, ARIT_H, A, B
; Usa:     TXB, B_RAM, D_RAM, L_RAM, SUMA, LOG_ESC
;.....................................................................

MEMIDS:   call  B_RAM            ; Espera a bloquear SRAM
        jb    BIT_TMP,MEMIDS

        clr   BIT_E            ; Borra indicadores de error

        mov   DPTR,#A_CMD+1    ; Apunta a low byte de longitud
        call  L_RAM            ; Lee dato
        mov   ARIT_L,A         ; Guarda parte baja

        inc   DPTR             ; Apunta a high byte de longitud
        call  L_RAM            ; Lee dato
        mov   ARIT_H,A         ; Guarda parte alta
```

```
        jnb   BIT_E,MEMID_6    ; Si no error al leer longitud salta
        call  D_RAM            ; Libera SRAM
        mov   A,#18            ; Anota el error
        call  LOG_ESC
        ret

MEMID_6:  mov   DPTR,#A_DATOS   ; Carga en DPTR direc inicio datos
        call  SUMA             ; AIRT_H,L = longitud+inicio = direc. final

        mov   B,#0             ; Inicia ultimo dato

MEMID_2:  mov   A,DPH           ; Si DPTR distinto de ARIT_H,L sigue en bucle
        xrl   A,ARIT_H
        jnz   MEMID_3
        mov   A,DPL
        xrl   A,ARIT_L
        jz    MEMID_5          ; Si DPTR=ARIT_H,L salta para acabar

MEMID_3:  call  L_RAM           ; Lee dado de uC1
        call  TXB              ; Envía dato a IDS
        mov   B,A              ; Guarda ultimo dato en B
        inc   DPTR             ; Incrementa puntero
        sjmp  MEMID_2          ; Va al bucle

MEMID_5:  call  D_RAM           ; Libera memoria

        jnb   BIT_E,MEMID_7    ; Si no error al leer datos salta
        mov   A,#19            ; Anota el error
        call  LOG_ESC

MEMID_7:  mov   A,#CR
        cjne  A,B,MEMID_8      ; Si ultimo carácter a IDS no es CR acaba
        ret

MEMID_8:  mov   A,#CR           ; Envía CR a IDS
        call  TXB
        ret

;.......................................................................
; Función: RIDS. Trata comando RESET_ID desde uC1. Hace RESET a la IDS y
;          espera la respuesta, que es copiada a la memoria. Si hay error al
;          recibir la respuesta se pone ? al final de los datos en memoria.
;          La función pilla y libera la memoria.
; Entrada:
; Salida:
; Modif.D: BIT_RST,BIT_E,B,DPTR,R2,A,
; Usa:     DLY,RXB,B_RAM,E_RAM,D_RAM,AMAYS,AUC1,LOG_ESC,B_COM
;.......................................................................

;*** No se comprueba si se supera el limite de memoria al enviar a uC1

; Activa reset IDS

RIDS:     clr   BIT_RST         ; Activa línea de reset hardware
        mov   B,#20            ; Ejecuta 20 veces
RIDS_Z:   call  DLY             ;   un retardo de 250ms.
        djnz  B,RIDS_Z         ; En total espera 5 seg
        setb  BIT_RST          ; Desactiva línea de reset

; Espera respuesta de IDS: ...one

        mov   DPTR,#A_DATOS    ; Inicia puntero a datos
        clr   BIT_E            ; Borra indicador de error
```

```
RIDS_0:   call  RXB              ; Recibe byte de IDS
          jnb   BIT_RTO,RIDS_9   ; Si error de timeout salta

RIDS_1:   call  B_RAM            ; Espera a bloquear SRAM
          jb    BIT_TMP,RIDS_1

RIDS_2:   call  E_RAM            ; Escribe byte
          call  D_RAM            ; Libera SRAM
          inc   DPTR             ; Incrementa puntero de datos
          call  AMAYS            ; Pasa a mayúsculas el dato
          cjne  A,#'O',RIDS_0    ; Si byte no es 'O' salta

          call  RXB
          jnb   BIT_RTO,RIDS_9   ; Si error de timeout salta

RIDS_4:   call  B_RAM            ; Espera a bloquear SRAM
          jb    BIT_TMP,RIDS_4

          call  E_RAM            ; Escribe byte
          call  D_RAM            ; Libera SRAM
          inc   DPTR             ; Incrementa puntero de datos
          call  AMAYS            ; Pasa a mayúsculas el dato
          cjne  A,#'N',RIDS_0    ; Si byte no es 'N' salta

          call  RXB
          jnb   BIT_RTO,RIDS_9   ; Si error de timeout salta

RIDS_6:   call  B_RAM            ; Espera a bloquear SRAM
          jb    BIT_TMP,RIDS_6

          call  E_RAM            ; Escribe byte
          call  D_RAM            ; Libera SRAM
          inc   DPTR             ; Incrementa puntero de datos
          call  AMAYS            ; Pasa a mayúsculas el dato
          cjne  A,#'E',RIDS_0    ; Si byte no es 'E' salta

          mov   R2,#R_RSTID      ; Respuesta para uC1
          call  AUC1             ; Actualiza comando y longitud para uC1

RIDS_B:   jnb   BIT_E,RIDS_8     ; Si no error al escribir datos salta
          mov   A,#20            ; Mensaje sobre error
          call  LOG_ESC

RIDS_8:   call  B_COM            ; Avisa a uC1
          ret                    ; Acaba

; Tratar error de timeout al recibir de IDS

RIDS_9:   call  B_RAM            ; Espera a bloquear SRAM
          jb    BIT_TMP,RIDS_9

          mov   A,#'?'           ; Escribe una interrogación como ultimo dato
          call  E_RAM
          call  D_RAM            ; Libera SRAM
          inc   DPTR             ; Incrementa puntero de datos

          mov   R2,#R_RSTID      ; Respuesta para uC1
          call  AUC1             ; Actualiza comando y longitud para uC1

          mov   A,#37            ; Mensaje diag. sobre timeout en la respuesta
          call  LOG_ESC

          sjmp  RIDS_B
```

```
;.....................................................................
; Función:  TXT. Transmisión de un texto introducido con DB
; Entrada:  Texto definido a continuación de la llamada con DB.
;           El texto acaba en ESC, que no se transmite.
; Salida:
; Modif.D.: DPTR, ACC
; Usa:      TXB
;.....................................................................

TXT:      pop   DPH              ; Recupera dirección del texto
        pop   DPL
        clr   A
        movc  A,@A+DPTR        ; Lee primer dato del texto

XSTR:     call  TXB              ; Transmite dato
        inc   DPTR             ; Lee siguiente dato
        clr   A
        movc  A,@A+DPTR
        cjne  A,#ESC,XSTR      ; Si el dato no es ESC sigue enviando
        mov   A,#1
        jmp   @A+DPTR          ; salta a dirección siguiente al texto


;---------------------------------------------------------------------
;                    FUNCIONES DEL SO DEL SIR
;---------------------------------------------------------------------

; Estas funciones son específicas para el µC2. Además este micro-controlador
; utiliza alguna de las funciones de SO definidas en el µC1.


;.....................................................................
; Función: CMDUC1. Manda comando a UC1 sin datos
; Entrada: R2=cmd para uC1
; Salida:
; Modif.D: DPTR
; Usa:     AUC1,B_COM
;.....................................................................
; Función: AUC1. Actualiza comando y longitud en SRAM y avisa a uC1
;          La función pilla la memoria y la libera.
; Entrada: DPTR: puntero al byte siguiente al ultimo escrito.
;          Cuando no se quiera pasar datos, DPTR debe ser A_DATOS.
;          R2=comando para uC1 (valor para la posición 0 de la SRAM)
; Salida:
; Modif.D: ARIT_L, ARIT_H, DPTR, A,B , BIT_1
; Usa:     RESTA, B_RAM, D_RAM, E_RAM, LOG_ESC
;.....................................................................
; Función: B_COM: Provoca un pulso en el BIT_COM del otro micro
; Entrada:
; Salida:
; Modif.D: BIT_INT
; Usa:
;.....................................................................
; Función:  DLY. Retardo de 250ms
; Entrada:
; Salida:
; Modif.D.: R6 Bank 1, R7 bank 1
; Usa:
;.....................................................................
; Función:  TESPCOM. Espera sobre 10 seg o aviso de otro micro.
; Entrada:
; Salida:   BIT_COM=1 si recibida conexión.
```

```
; Modif.D.: R5 a R7 bank 1
; Usa:
;.....................................................................
; Función:  ESPUC1. Espera BIT_COM de uC1
; Entrada:
; Salida:   BIT_COM=1 si recibida conexión.
; Modif.D.: PSW,R5,R6,R7
; Usa:      TXT,TESPCOM
;.....................................................................
; Función: LEECMD. Lee comando de uC1 en SRAM. Se pilla y libera la memoria
; Entrada:
; Salida:  A=comando elido, si lectura correcta. A=0 si hubo error
; Modif.D: A, DPTR, BIT_E
; Usa:     B_RAM, L_RAM, D_RAM, LOG_ESC
;.....................................................................
; Función: ENVMSG. envía datos a uC1.
; Entrada: Los datos se definen después de la rutina como: DB 'Texto',ESC
;          R2= comando respuesta para uC1 (DAT_TR o DAT_AV o DAT_PR)
; Salida:
; Modif.D: TMP1,TMP2,DAT_L,DAT_H,BIT_E,A,B,DPTR
; Usa:     B_RAM,E_RAM,D_RAM,AUC1,B_COM,LOG_ESC
;.....................................................................; ; ;
; Función:  ENVASC. Transmite un número como caracteres ASCII, decimales
; Entrada:  A=numero a enviar
; Salida:
; Modif.D.: ACC, B, BIT_AX
; Usa:      TXB
;.....................................................................

END
```