INDICE

PRÓLOGO

En los últimos 30 años, el control digital de procesos industriales ha pasado de ser la excepción a ser la norma. Sin embargo, el dominio del control digital no se limita a los procesos industriales, sino que se ha extendido a otros muchos ámbitos, como por ejemplo el doméstico, debido en gran parte a los avances en el diseño de hardware y la reducción de los costes.

Siguiendo esta línea, la adopción de computadores para el control de procesos ha incrementado el rango de actividades que estos pueden llevar a cabo, no sólo encargándose del control digital directo (DDC), sino dotando al operador o usuario del sistema de una visión global del estado del mismo. De hecho, muchos de los modernos sistemas de control por computador, dan a este un papel supervisor y no lo emplean para el DDC.

En sus inicios el control por computador era centralizado, fundamentalmente por motivos económicos, de modo que solía disponerse de un solo computador para el control de toda una planta. Si bien, la continua reducción de costes en el diseño y

fabricación de hardware llevó al desarrollo del microprocesador que ha hecho posible la existencia de sistemas multi-procesador.

Entre ellos se encuentran los **sistemas de control distribuido**, que se caracterizan por estar integrados por diversos procesadores que ejecutan tareas de naturaleza similar en paralelo, y que intercambian información a través de una determinada red de comunicaciones serie. El empleo de este tipo de sistemas se está extendiendo cada vez más, debido principalmente a las ventajas que estos nos ofrecen:

- La capacidad del sistema se ve notablemente incrementada por el reparto de las tareas entre varios procesadores.

- El sistema es mucho más flexible y escalable, ya que si aparece una nueva tarea a ser realizada por un nuevo procesador, basta con añadir un nodo a la red de comunicación.

- Un fallo en una de las unidades no afecta a las demás. Es posible implementar mecanismos de protección que permitan que otro procesador tome el control de la tarea de manera automática cuando se produzca el fallo.

- Es mucho más fácil hacer cambios (tanto hardware como software) o reparaciones en el sistema al tratarse de módulos independientes.

- Al estar unidos por una red de comunicaciones serie, los procesadores pueden estar dispersos por un área extensa, ya que no es necesario que las señales sean cableadas directamente al computador.

Todas estas ventajas, unidas al abaratamiento y aumento de la fiabilidad de los microprocesadores, han convertido a los sistemas distribuidos en la elección óptima para muchas aplicaciones.

La potencia y flexibilidad que estos sistemas ofrecen han suscitado gran interés en el autor y el tutor del presente proyecto y han motivado la elaboración del mismo.

1. OBJETIVOS DEL PROYECTO

Una vez expuestos los motivos que han llevado a la elaboración del presente proyecto en base al interés suscitado por los sistemas de control distribuido, es posible ya, entrar en detalle acerca de los objetivos perseguidos por el mismo.

Como antecesores de este proyecto podrían citarse todos aquellos que, basados en la tarjeta de evaluación del Microcontrolador 68HC11 de Motorola usada en el Área de Tecnología Electrónica del Departamento de Tecnología Electrónica de la E.S.I. de la Universidad de Sevilla, se propusieron para dar solución, diseñando aplicaciones específicas, a problemas concretos.

Este proyecto no persigue los mismos objetivos que sus antecesores sino que tiene como fundamento integrar todos ellos de modo que diversas aplicaciones independientes puedan ser supervisadas y controladas por un único usuario, al tiempo que permita la interacción entre las propias aplicaciones.

Por tanto, lo que se pretende es ensamblar en un sólo sistema un número indeterminado de tarjetas, cada una con su aplicación local independiente a la que el usuario puede acceder para obtener información o dar consignas de manera que estas no se vean afectadas por tales actuaciones. De este modo habríamos diseñado un sistema de control multipunto, ya que el usuario podría supervisar y controlar diferentes aplicaciones distribuidas geográficamente entorno a un determinado área desde un único puesto fijo.

El objetivo fundamental del proyecto es por consiguiente, establecer comunicaciones entre el usuario, por medio de un PC, y todos los nodos del sistema y por extensión entre los propios nodos, al tiempo que definir un lenguaje común para que puedan entender la información intercambiada. Este objetivo ha de lograrse usando los recursos que nos ofrecen los equipos que compondrán el sistema, es decir, comunicación serie asíncrona en enlaces punto a punto (RS232), lo que implica una dificultad añadida, ya que habrá por lo tanto que ingeniar algún método que nos permita obtener un enlace multipunto a partir de dichos enlaces punto a punto.

2. DESCRIPCIÓN DEL SISTEMA

Nuestro sistema de control multipunto consiste en un sistema distribuido integrado por un número indeterminado de nodos de los que uno de ellos actúa como estación de operación, con funciones de mando y supervisión y que además incluye una pantalla de visualización de datos y presenta una Interfase Hombre-Máquina sencilla e intuitiva.

El resto de los nodos ejercen sobre la planta el control digital directo (DDC), de modo que reciben señales de sensores y actuadores cableadas físicamente.. Estos nodos ejecutan una aplicación específica de control almacenada en memoria local no volátil, si bien también pueden ejecutar programas descargados desde la estación de operación o bien no ejecutar ninguna aplicación local dependiendo de las necesidades de la planta.

Todos estos nodos están conectados mediante algún medio de comunicación serie que les permite intercambiar información. De este modo, todas las variables de los nodos son accesibles desde la estación de operación tanto en lectura como en escritura. Ello incluye no solo las entradas y salidas de los nodos, sino también las variables internas y registros de control de estos, lo que permite al operador gobernar todos los procesos de la planta. Además los nodos pueden generar alarmas de forma automática hacia la estación de operación, lo que dota al operador de un conocimiento preciso sobre las posibles anomalías que se produzcan en la planta.

En el apartado correspondiente al manual de usuario se detallan todas las posibilidades que ofrece el sistema. En cualquier caso, el sistema cuanta con una amplia reserva que permite al usuario ampliar dichas posibilidades en función de las necesidades del sistema.

Este sistema está especialmente diseñado para aplicaciones en el ámbito doméstico como la domótica, en las que las restricciones temporales no son críticas y el número de señales de entrada y salida no es tan elevado como para requerir PLC’s para su procesado, que resultan excesivos para este tipo de aplicaciones, sobre todo económicamente.

3. ARQUITECTURA

En este apartado se describe la arquitectura del sistema y se discuten sus requerimientos a nivel de hardware. Es conveniente resaltar que el sistema se implementó físicamente atendiendo a los esquemas presentes en este apartado.

3.1 EQUIPOS

ALCANCE DEL SUMINISTRO

El alcance del suministro que se necesita para la instalación y puesta en marcha del sistema (de N nodos) es el siguiente:

- Un (1) PC con al menos un (1) puerto para comunicaciones serie libre.
- N nodos consistentes en tarjetas de evaluación del microcontrolador 68HC11 desarrollada por el Departamento de Tecnología Electrónica de la Escuela de Ingenieros de la Universidad de Sevilla.

Cables y conexionado:

- N-1 conectores para separación de Tx y Rx desarrollados específicamente para el sistema
- N-1 cables de 5 hilos terminados en conectores RS232 DB-9

Alimentación:

- 7,5 Vdc para alimentar a los N-1 nodos del Sistema
- 220 Vac, 50 Hz para alimentar al PC

DESCRIPCIÓN DE LAS TARJETAS-68HC11

Estas tarjetas están ampliamente descritas en toda la literatura que se encuentra a disposición del Departamento de Tecnología Electrónica, por lo que sólo se describirán

aquellos aspectos relativos a las comunicaciones.

El microcontrolador 68HC11 de Motorola dispone de un periférico para comunicaciones serie asíncronas (SCI – Asynchronous Serial Peripheral Interface), que será el que nos permita acceder a la red de datos del sistema.

Este periférico funciona con niveles de tensión propios del micro, esto es 0 - 5V, que no son adecuados para las comunicaciones a través del standard elegido (RS232), si bien las tarjetas están dotadas del integrado MAX232, que convierte estas señales a ? 12 V que son los niveles de tensión RS232. Este integrado dispone de dos puertos, si bien, sólo uno de ellos es usado, quedando desaprovechado el restante.

Por último, estas tarjetas disponen de un conector DB-9 para el conexionado físico.

Además del SCI el micro dispone de otros periféricos que quedan a disposición de la aplicación para usos de control, en concreto dispone de las siguientes entradas y salidas, que pueden ser señales digitales de propósito general según la direccionalidad indicada, o estar configuradas para llevar a cabo una función alternativa concreta :

PIN	DIRECCIONALIDAD	FUNCIÓN ALTERNATIVA
PA0-PA2	Entrada	Input Capture
PA3-PA6	Salida	Output Capture
PA7	Bidireccional	Pulse Acumulator
PD2	Bidireccional	SPI: MISO
PD3	Bidireccional	SPI: MOSI
PD4	Bidireccional	SPI: SCK
PD5	Bidireccional	SPI: SS
PE0-PE7	Entrada	Entradas Analógicas

3.2 TOPOLOGÍA

El sistema está formado por un número indeterminado de nodos entre 1 y N. Además de dichos nodos, siempre está presente el nodo implementado en el PC, al que llamaremos de ahora en adelante Nodo 0, que constituye la Estación de Operación.

Tanto la Estación de Operación como los nodos (tarjetas de evaluación del Microcontrolador 68HC11 de Motorola) están provistos de un puerto serie asíncrono para comunicaciones RS232. En todos los casos se trata de conexiones punto a punto y comunicaciones full-dúplex.

En principio estos recursos no se adecuan a los requerimientos del sistema, de modo que habrá que explotarlos convenientemente para lograr nuestro objetivo, que no es otro que lograr emular el comportamiento de una conexión multipunto entre todos los nodos. A continuación se detallan las líneas con las que cuentan las tarjetas y el PC para establecer las comunicaciones:

TARJETAS:

LINEA	FUNCIÓN	PIN ASOCIADO	DIREC
TxD	Línea de transmisión de datos	2	Salida
RxD	Línea de recepción de datos	3	Entrada
GND	Tierra de referencia	5	-
RESET	Línea de reset de los micros	7	Entrada
XIRQ	Interrupción no enmascarable	4	Entrada

PC:

LINEA	FUNCIÓN	PIN ASOCIADO	DIREC
TxD	Línea de transmisión de datos	2	Salida
RxD	Línea de recepción de datos	3	Entrada
GND	Tierra de referencia	5	-
RTS	Línea de reset de los micros	4	Salida
XIRQ	Interrupción no enmascarable	Sin definir	Salida

La solución propuesta a partir de estos recursos es conectar los nodos usando una topología en anillo simple (unidireccional) de tal modo que cada nodo transmita al siguiente nodo en el anillo, y reciba del nodo inmediatamente anterior. En todos los casos las comunicaciones están gobernadas por el nodo transmisor que actúa como maestro del enlace, dejando al nodo receptor el papel de esclavo.

De este modo, el control de acceso al medio (MAC – Medium Access Control) no es necesario, ya que está totalmente gobernado por el nodo transmisor, que transmite cada vez que tiene algo que enviar sin preocuparse de avisar al receptor, el cual, se sobreentiende que siempre está listo para recibir ya que tan sólo recibe mensajes de un nodo transmisor, el anterior a él en el anillo.

Lo que realmente estamos haciendo es separar en cada nodo la comunicación full-dúplex en dos comunicaciones simplex, una en cada sentido en el anillo. De este modo se logra conectar físicamente todos los nodos del sistema. Sin embargo, al adoptar esta topología, se pierden prestaciones en los enlaces (petición de conexión o desconexión, asentimiento inmediato etc.), de modo que habrá que recuperar estas prestaciones vía software. La topología elegida se muestra en la figura Fig. 3.2.1 .

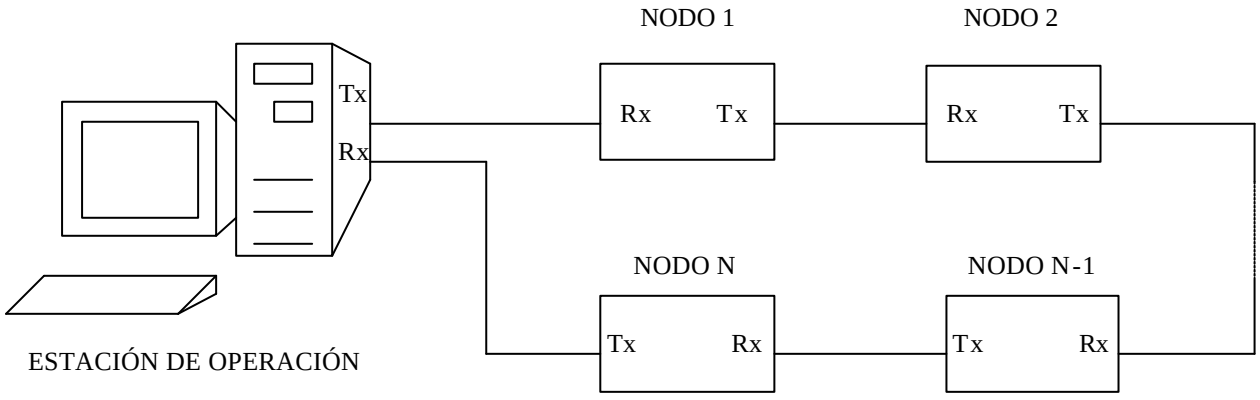


FIG 3.2.1

Por simplicidad y eficiencia del diseño, se ha decidido que físicamente el sistema se presente como una “tira” de tarjetas conectadas mediante un cable RS232,

del modo que se representa en la figura 3.2.2 . Ello obliga a que la conexión entre el PC y el primer nodo recorra toda la tira de modo que los datos atraviesen los conectores descritos más adelante sin ser "escuchados " por ninguno de los nodos intermedios.

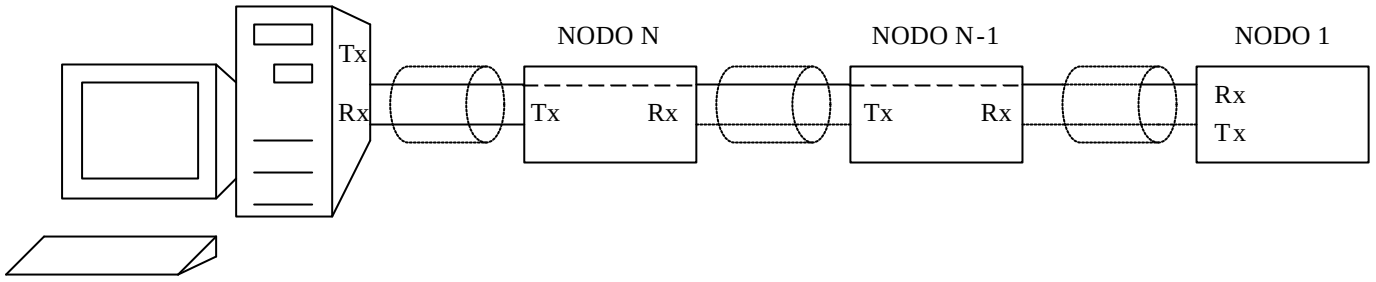


FIG 3.2.2

3.3 CONEXIONADO

A nivel mecánico, los conectores RS232 no permiten realizar las conexiones que aparecen en la figura 3.2.2, de modo que ha sido necesario implementar unos conectores especiales para separar las líneas de transmisión y recepción y conectarlas con los nodos correspondientes. El layout de dichos conectores aparece en la figura 3.3.1, donde tan sólo representamos el plano inferior, ya que el superior constituye el plano de masas con un derrame de cobre que lo cubre por completo y al que van conectados el pin 5 de cada conector.

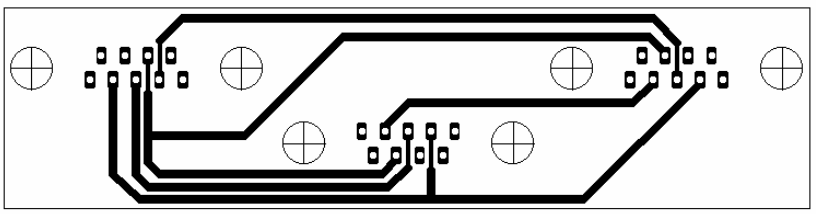


FIG 3.3.1

Estos conectores son válidos a nivel de ensayo, en que se centra este proyecto, ya que al ser totalmente pasivos, limitan la longitud máxima del perímetro del sistema a la longitud máxima de un enlace RS232 (30 m.)

El conexionado del sistema queda finalmente tal y como muestra la figura 3.3.2 :

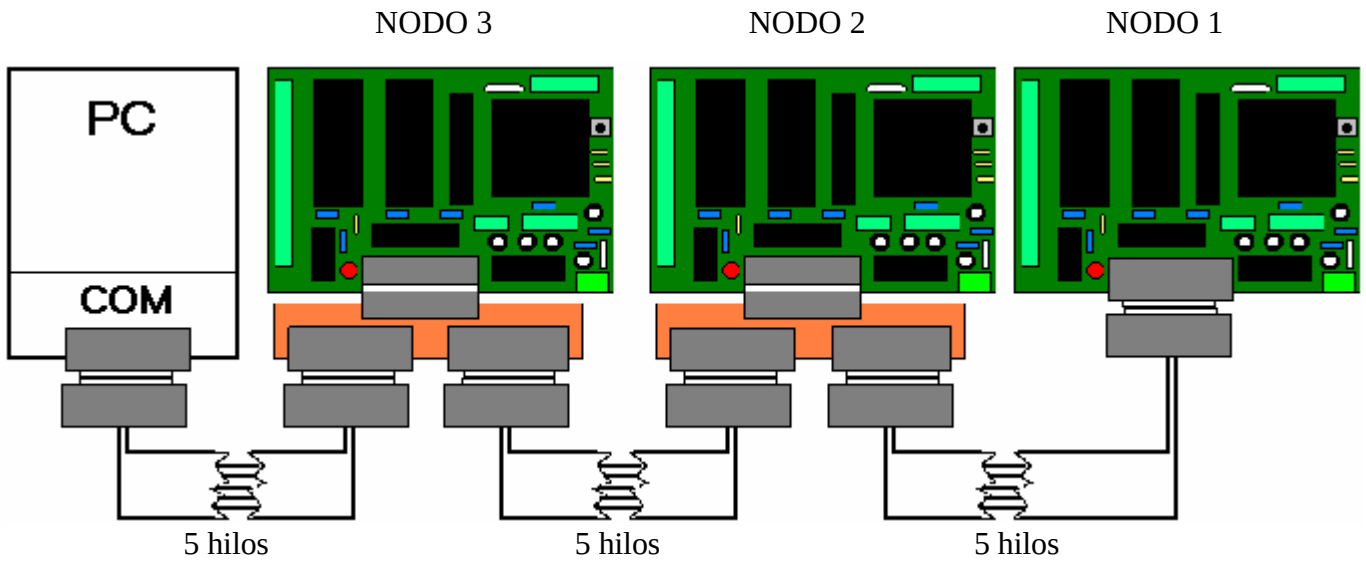


FIG. 3.3.2

3.4 MEJORAS

Una posible mejora sería la implementación de conectores activos que devolviesen la señal a ? 12V en cada conexión. Así, el perímetro máximo de la red se vería aumentado a N * 30 m. , siendo N el número de nodos del sistema.

Otra posibilidad sería la de variar el diseño de las tarjetas de evaluación del microcontrolador 68HC11, de modo que se aprovechasen los dos puertos que ofrece el integrado MAX232, uno de los cuales está desaprovechado en el diseño actual. Así sería posible dotar a la tarjeta de evaluación de dos conectores RS232 , de modo que cada uno de ellos se conectaría a un nodo diferente, y los niveles de tensión serían los que da el MAX232, cuyos dos puertos serían ahora aprovechados para generar TxD y RxD independientemente.

4. FUNCIONAMIENTO

4.1 GENERALIDADES

Una vez el sistema ha sido descrito físicamente, la siguiente cuestión que cabría hacerse es cómo intercambian información los nodos del sistema, y a un nivel superior, qué información intercambian entre si.

Ambas incógnitas quedan despejadas cuando se define cierto protocolo que rige la comunicación entre nodos. Un protocolo no es más que un conjunto de reglas que gobiernan el formato y el significado de los " paquetes " (pueden recibir el nombre de segmentos, tramas etc., en función del nivel en que se definan) intercambiados dentro de un mismo nivel.

Así pues, para hacer posible la comunicación en el sistema, se ha definido el Protocolo de Comunicación entre Nodos. En este apartado se describe cada uno de los niveles del sistema y de las unidades de información que intercambian en ellos. No obstante, **el análisis funcional completo (incluyendo campos, registros, flags etc.) se hace en el apartado 5 (Consideraciones sobre el Código).**

El fin último de un protocolo de comunicación es lograr el intercambio de información, caracterizada por un determinado formato y significado, entre dos o más nodos. Dichos nodos, a su vez se caracterizan por poseer unos recursos para establecer comunicaciones finitos y determinados.

El objetivo fundamental es conseguir, usando los recursos del sistema, que información contenida en cierto nodo origen pueda ser reproducida fielmente y procesada en el destino elegido por el origen.

Esto, que a simple vista pudiera parecer sencillo, se complica si el formato de la información es complejo y los recursos de comunicación limitados. El establecimiento de distintos niveles es en tal caso esencial.

4.2 NIVELES DEL PROTOCOLO

A la hora de determinar los niveles del protocolo, hay que tener presente cual es la información neta, es decir, los datos efectivos que se desea que los nodos intercambien entre si. Formato y significado de dicha información fijan el nivel superior del protocolo.

También hay que tener presente el nivel inferior que resulta mucho más sencillo de identificar. Este nivel, al que llamaremos Nivel Físico por analogía con numerosos protocolos existentes, ha sido descrito con detalle en el apartado correspondiente. En este apartado nos limitaremos a enumerar sus funciones.

Una vez identificados estos dos niveles límite, habrá simplemente que definir un número determinado de niveles intermedios que permitan convertir información del nivel superior en información del nivel inferior, y viceversa, de la manera más ordenada y sencilla posible. El número de niveles ha de ser el menor que permita hacerlo, ya que la introducción de capas no estrictamente necesarias, lejos de simplificar la tarea, no hacen más que añadir complejidad a la misma.

Los niveles que contempla el Protocolo de Comunicación entre Nodos son los siguientes:

- Nivel Físico
- Nivel de Red
- Nivel de Aplicación

NOTA: Estos niveles no guardan correspondencia directa con los niveles del mismo nombre del Modelo O.S.I.

La denominación de los niveles atiende a un criterio funcional, ya que pretende ser lo más descriptiva posible de la función que realiza cada una de las capas del protocolo. Estas funciones son las siguientes:

Nivel Físico: este nivel se encarga de la transmisión física de los bits entre nodos y su ensamblado en caracteres. Sus funciones son:

- Diseño eléctrico y mecánico de conectores, rango de voltajes, tiempo de bit, etc. Norma RS-232 para enlaces punto a punto.
- Topología de la red.
- Conexión entre nodos físicamente consecutivos.
- Transmisión/Recepción de caracteres completos.

Nivel de Red: este nivel se encarga de asegurar la comunicación de extremo a extremo en la red, dejándola libre de errores para el nivel superior. Sus funciones son:

- Troceado en tramas. Delimitación de los límites de las mismas.
- Corrección de pérdida o duplicidad de tramas.
- Control de tráfico y congestión.
- Autodiagnóstico de la red.
- Aislar de errores al nivel superior, y proporcionarle transparencia.
- Determinación de origen y destino de mensaje.
- Especificación del ámbito de los datos de la trama.

Nivel de Aplicación: añade funciones útiles para el usuario. Sus funciones son:

- Acceso de lectura y escritura en memoria local de los nodos.
- Transferencia y control de programas.
- Selección de Aplicación Local a ser ejecutada en los nodos

Estos niveles son específicos del Sistema de Control Multipunto objeto del proyecto, sin embargo, podría establecerse cierta correspondencia con los niveles del Modelo O.S.I. para interconexión de sistemas abiertos, por analogía con las funciones que estos comprenden. Esta correspondencia, que tan sólo tiene validez ilustrativa es la siguiente:

El Nivel Físico del Protocolo de Comunicación entre Nodos se corresponde con los Niveles 1 y 2 del Modelo O.S.I. (Nivel Físico y Nivel de Enlace).

El Nivel de Red del Protocolo de Comunicación entre Nodos recogería los Niveles 2,3 y 4 del Modelo O.S.I. (Nivel de Red y de Transporte).

El Nivel de Aplicación del Protocolo de Comunicación entre Nodos guardaría cierta correspondencia con el Nivel 7 del Modelo O.S.I. (Nivel de Aplicación) , si bien, a un nivel mucho más bajo que este.

4.3 NIVEL FÍSICO

Este nivel se encarga de la transmisión y recepción física de los bits entre los nodos que componen el sistema, y del ensamblado de los mismos en caracteres.

Comprende entre otras cosas todos los aspectos relativos a la arquitectura y conexionado del sistema descritos en el apartado Arquitectura, por lo que se repetirá aquí la discusión de dichos aspectos.

La transmisión física de los caracteres está regida por la Norma RS232 para comunicaciones serie punto a punto. En nuestro caso, los parámetros de la comunicación se han fijado de acuerdo a los siguientes datos:

- Velocidad binaria: 9600 Bps
- Bits de datos: 8
- Paridad: No
- Bits de parada: 1

El SCI en el micro 68HC11 y el puerto serie en el PC se encargan de gestionar la recepción de caracteres y de generar los eventos correspondientes cada vez que se recibe/transmite un carácter completo o varios caracteres, según se haya dispuesto que se haga.

4.4 NIVEL DE RED

4.4.1 Generalidades

Uno de los objetivos del proyecto del que, en parte, toma este nombre, es el de conseguir emular un enlace multipunto con N terminales a partir de N enlaces punto a punto. Para ello se decidió conectarlos en anillo con una transmisión en un sentido y recepción en otro. Evidentemente, las limitaciones de esta topología son enormes, ya que sólo es posible transmitir directamente al nodo inmediatamente posterior en la red, y recibir del nodo inmediatamente anterior en la red.

Es este Nivel el encargado de dotar a los nodos de la capacidad de comunicarse con cualquier otro nodo de la red, independientemente de la posición relativa que este tenga con respecto a los primeros. El Nivel de Red se ocupa por lo tanto de entregar los mensajes a sus destinatarios asegurando la comunicación de extremo a extremo en la red.

Del nivel anterior recibe los caracteres, y al nivel superior ha de entregar los paquetes de aplicación al tiempo que lo libera de cualquier error posible. Para lograr esto, el Nivel de Red lleva a cabo ciertas funciones. A continuación se detalla cada una de estas:

- Troceado en tramas: Del nivel físico recibe caracteres, de modo que estos han de ser ensamblados en tramas . Es función de este nivel la delimitación de los límites de las mismas para conseguir el sincronismo de trama y por supuesto, para la configuración de las tramas en si.

- Corrección de pérdida o duplicidad de tramas. Por diversos factores, pueden aparecer tramas perdidas en la red (no dirigidas a ningún nodo de esta) o tramas repetidas. Es misión de este nivel la de evitar que haya tramas en la red que no aporten información útil

- Control de tráfico y congestión. Este nivel se encarga de administrar los recursos para su mejor explotación, decidiendo el orden de transmisión de las tramas según su prioridad, y como ya hemos visto, eliminando tramas inútiles que pueden congestionar el sistema.

- Autodiagnóstico de la red. Detección de anomalías en la red, tales como caída de algún nodo, detección de nuevo nodo, pérdida de sincronismo de trama, etc.

- Aislar de errores al nivel superior, y proporcionarle transparencia. Durante la transmisión de una trama, algún bit de esta puede degenerarse, de modo que sus datos dejan de ser válidos. Es misión del Nivel de Red el detectar estas tramas erróneas y desecharlas, haciendo saber al emisor que su mensaje no ha llegado con éxito para que este vuelva a transmitir su trama.

- Determinación de origen y destino de mensaje. Se logra así que los mensajes vayan de extremo a extremo en la red. De este modo los nodos saben cuando tienen que procesar la trama recibida o simplemente retransmitirla al siguiente nodo en el anillo.

- Especificación del ámbito de los datos de la trama. La naturaleza de estos puede ser muy diversa (asentimiento, rechazo, información etc.), de modo que cada tipo de dato requiere un procesado diferente.

Todas estas funciones están por tanto encaminadas a conseguir que el nivel superior quede completamente libre de errores, en lo que a transmisión/recepción se refiere. Esto permite que las comunicaciones del Nivel de Aplicación se hagan de forma totalmente transparente al propio nivel, lo que abre la posibilidad de realizar ampliaciones a medida de las aplicaciones y el uso que el usuario quiera dar al sistema, sin necesidad de rediseñar o alterar el protocolo. Simplemente bastaría con añadir el código específico de las mejoras introducidas, por ejemplo como funciones de librería.

4.4.2 Trama de Nivel de Red

La transmisión de un mensaje de un nodo a otro involucra a tantos otros nodos, como terminales existen físicamente entre ambos en la red en el sentido de transmisión. Estos nodos intermedios se encuentran con el trabajo adicional de retransmitir aquellos mensajes que no van a ellos dirigidos.

Por otro lado no conviene perder de vista el objetivo del proyecto, esto es, no se trata de diseñar un Sistema Multipunto si más, sino un Sistema de Control Multipunto. Con ello se quiere resaltar el hecho de que la finalidad de instalar un sistema como este es que realice cierta función, interaccionando con el medio en el que se encuentre, lo que quiere decir que es imprescindible que cada nodo pueda ejecutar su aplicación local sin verse afectado por el Protocolo de Comunicación entre Nodos.

Por todo ello, es esencial que la carga computacional que implica la ejecución del código asociado al protocolo no sea demasiado elevada en términos relativos respecto a la de las aplicaciones locales ejecutadas en los nodos. Ello nos lleva a adoptar una solución de compromiso entre el número de veces que se ejecuta el código de protocolo por mensaje (nº de tramas por mensaje), y el intervalo de ocupación medio del procesador del nodo por trama (longitud de la trama). Es decir, hay que definir una longitud de trama.

Longitud de Trama

A este respecto la primera cuestión es decidir si esta ha de ser fija o variable, en función de los requerimientos del sistema.

La ventaja de tener longitud fija es que transmisor y receptor esperan transmitir y recibir respectivamente un número fijo de datos, de modo que no es necesario dotarlos de mecanismos para trabajar con tramas de diferente longitud.

Como inconveniente tendremos que según el tamaño fijado para la trama, si este es grande, en media se desperdiciará mucho espacio en las tramas, espacio que requerirá tiempo de procesado y tiempo de transmisión sin aportar información. Si el tamaño es pequeño, se requerirá en media un número elevado de tramas para recoger los mensajes.

Si por el contrario se tiene longitud variable, los inconvenientes de la longitud fija no afectan, pero por contra, hay que aportar mecanismos para adaptar transmisor y receptor a diferentes longitudes de trama, así como incluir información sobre su longitud en la propia trama.

La solución finalmente adoptada puede considerarse un híbrido entre ambas opciones, ya que se definen tramas de longitud variable, con un máximo de 32 bytes, para que las posiciones de memoria en que se almacenan sean fijas, y de igual tamaño al de la trama de mayor longitud, es decir, 32 bytes.

NOTA: Se ha establecido como dato unidad el byte, ya que el micro que gobierna los nodos es el 68HC11, cuyo bus de datos es de 8 bits.

Formato de Trama de Nivel de Red

Una vez definida la longitud de la trama, estamos en condiciones de determinar los campos que la forman, es decir, de establecer el formato de trama. Debemos tener en cuenta el objetivo del proyecto, el cual constituye un ensayo de la viabilidad del Sistema de Control. Por tal motivo protocolo y formato de trama han sido diseñados para un sistema bastante concreto pero de la forma más genérica posible. Por ello resulta extrapolable para sistemas no tan concretos, sin más que cambiar la longitud de los campos de la trama en algunos casos.

Explicados los motivos que han llevado a definir el formato de trama como se ha hecho, podemos introducir el formato de trama, el cual aparece en la Figura 4.4.2.1 :

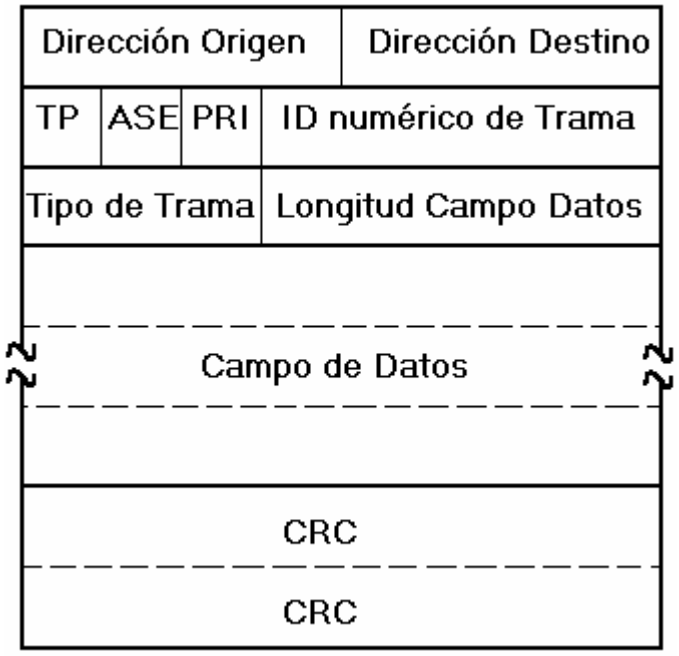


Fig. 4.4.2.1

Como se puede observar, el formato responde a la estructura que aparece en la mayoría de los protocolos: una cabecera, en este caso de longitud igual a 3 bytes, en la que se incluyen los campos relativos al control de la comunicación, un campo de datos de longitud variable, y por último un campo dedicado al control de errores en la trama.

Campos de la Trama de Nivel de Red

El significado y función de cada uno de los campos de la Trama de Nivel de Red se encuentran resumidos en el Apéndice II. A continuación se presentan en detalle:

Direcciones:

Las direcciones de los nodos del sistema, tienen una longitud de 4 bits. Ello nos permite definir 16 direcciones (entre 0 y 15), por lo que nuestro sistema podrá constar de entre 1 y 15 nodos sin incluir al nodo implementado en el PC, que no ejerce control

digital directo, sino que sirve de interfaz de comunicación con el usuario y desempeña funciones de control de tráfico y supervisión.

El valor que toma la dirección de cada nodo, es su propio ID, que como su propio nombre indica identifica al nodo. Este ID, como se verá más adelante, consta igualmente de 4 bits y es definido durante el proceso de configuración del sistema, manteniendo su valor mientras dicha configuración sea válida. A este respecto, el nodo implementado en el PC constituye un caso especial, ya que su ID toma siempre el valor 0, es decir, en binario 0000. Por este motivo a partir de ahora haremos en ocasiones referencia a este como Nodo 0.

Si los requerimientos del sistema fueran tales que implicasen la integración de un número de nodos mayor que 15, bastaría con elegir IDs y por lo tanto direcciones de mayor longitud. Por sencillez estos podrían tener una longitud múltiplo de 4 bits. Por ejemplo, con ID de 8 bits, se podrían direccionar hasta un máximo de 255 nodos.

Dirección Origen: determina el nodo que ha enviado la trama y permite al nodo que la recibe, saber “quién” se lo envió.

Dirección Destino: determina el nodo al cual va dirigida la trama, de modo que permite que este detecte la trama como propia y la procese como tal. Al mismo tiempo, permite a aquellos nodos a los que no va dirigida detectar que la trama no debe ser procesada sino simplemente retransmitida.

Dirección Multicast: no corresponde a un nuevo campo sino a un tipo de comunicación en el que el Nodo 0 transmite una trama a todos y cada uno de los nodos del sistema. Esta situación se da si tanto Dirección Origen como Dirección Destino toman el valor 0, es decir, en binario 0000. Por lo tanto, cuando una trama es multicast, el primer byte de la trama toma el valor:

DIRECCIÓN MULTICAST : 0000 0000

Bit TP

Este campo tiene de longitud 1 bit y toma su nombre de las iniciales de TRAMA PERDIDA. Cumple funciones de control para evitar que tramas cuyo campo Dirección Destino ha sufrido algún error no detectado continúen circulando indefinidamente por el sistema.

Tal situación implica que ningún nodo las detecte como tramas a ellos dirigidas, por lo que todos se limitan a retransmitirlas, lo que desperdiciaría tiempo de acceso al medio de los nodos al retransmitirlas y de procesado al gestionarlas. En el límite, al darse el caso de la existencia de un número elevado de tramas perdidas circulando por el sistema, se produciría una situación de congestión en la red que impediría el normal funcionamiento de la misma. El significado de este bit es el siguiente:

TP = 0 ; REGISTRO DE PASO POR NODO 0 = FALSO
TP = 1 ; REGISTRO DE PASO POR NODO 0 = VERDADERO

El mecanismo que permite evitar la circulación indefinida de tramas perdidas por el sistema es el siguiente:

El único nodo que tiene acceso en lectura y escritura a este campo es el Nodo 0, mientras que e l resto de los nodos tienen restringido el acceso en escritura de modo que sólo pueden escribir en este campo cuando son ellos los que emiten la trama, dándole siempre el valor 0. El valor de TP permanece inalterado retransmisión tras retransmisión en los nodos mientras la trama trata de llegar a su destino. Si este se encuentra entre el nodo que emitió la trama y el Nodo 0 el destino recibe entonces la trama y la elimina de la circulación. Si por el contrario la trama pasa por el Nodo 0 antes de llegar a su destino, este pone a 1 el bit TP y la retransmite con normalidad.

Cuando se produce un error en el campo dirección destino, y este no es detectado por otros mecanismos (CRC, detección de error en recepción del puerto serie) entonces la trama será una y otra vez retransmitida por los nodos que no detectan la trama como propia. Si el valor erróneo del campo Dirección Destino no corresponde

al ID de ningún nodo del sistema, no será eliminada de la circulación por ninguno de ellos, por lo que tenderá a circular indefinidamente por la red. Sin embargo, como hemos visto, cuando una trama con TP = 0 es recibida por el Nodo 0, este pone a TP a 1 y retransmite la trama, por lo que cuando la reciba otra vez, detectará que se trata de una trama perdida. En tal caso el Nodo 0 la elimina de la circulación sin más, por lo que el problema queda resuelto.

Bit ASE

Este campo tiene de longitud 1 bit y su nombre es la abreviatura de la palabra ASENTIMIENTO. Cumple funciones de control para determinar si una trama requiere o no asentimiento, es decir, si el nodo que emitió la trama espera asentimiento de la misma. El significado de este campo es el siguiente:

ASE = 1 ; LA TRAMA REQUIERE ASENTIMIENTO
ASE = 0 ; LA TRAMA NO REQUIERE ASENTIMIENTO

Este campo es accedido en escritura por el nodo que emite la trama, y en lectura por el nodo al que va destinada la trama y que por lo tanto la procesa. El procesado y generación de estas tramas se verá en el apartado correspondiente.

Bit PRI

Este campo tiene de longitud 1 bit y su nombre es la abreviatura de PRIORIDAD. Cumple funciones de control para determinar si una trama es o no prioritaria.

En el sistema se definen dos posibles prioridades, es decir, Prioridad 0 ó No Prioridad y Prioridad 1 ó simplemente Prioridad. A groso modo, la diferencia entre ambas se traduce en que aquellas tramas con Prioridad 1 son procesadas y transmitidas antes que aquellas que no la tienen cuando se produce un conflicto entre ambas. El significado de este campo es el siguiente:

PRI = 0 ; TRAMA NO PRIORITARIA
PRI = 1 ; TRAMA PRIORITARIA

Este campo puede ser accedido tanto en lectura como en escritura por todos los nodos en función del estado de los mismos como veremos en el apartado correspondiente.

ID Numérico de Trama

Este campo tiene de longitud 5 bits y como su propio nombre indica, cumple funciones de identificación de trama.

Su objetivo es identificar unívocamente a la trama que lo contiene, aunque en realidad no es suficiente para hacerlo. Además de este campo, para que la trama quede perfectamente identificada, se necesita conocer también el nodo emisor de la trama lo cual no es ningún problema ya que, como sabemos, esa información está contenida en el campo Dirección Origen. Como se verá en el apartado correspondiente, cada nodo posee una lista de espera de aquellas que requieren asentimiento. Son los identificadores de estas tramas los que nos ocupan, ya que permiten tener una referencia de la posición en que se guarda la trama en caso de que tenga que ser accedida por cualquier causa.

Como también veremos, estos identificadores son reutilizables, por lo que con un número limitado de identificadores el sistema funcionará correctamente durante el tiempo que sea necesario.

Estos identificadores permiten referenciar hasta un máximo de 32 tramas (entre 0 y 31), de modo que cada identificador es adjudicado a una sola trama. En el caso de los nodos, estos identificadores tendrán valores comprendidos entre 0 y 7, ya que las listas de espera que estos tienen solo cuentan con 8 posiciones, por las 32 con las que cuenta el Nodo 0. Ello es por la diferencia del número de tramas generadas por este. En resumen, la longitud de este campo viene determinada por el número de posiciones de la lista de espera del Nodo 0. Si las condiciones del sistema requiriesen una lista de espera de mayor número de posiciones, bastaría con aumentar la longitud de este campo.

Tipo de Trama

Este campo tiene una longitud de 3 bits, y como su propio nombre indica, determina el tipo de trama que caracteriza a aquella que lo contiene. Cumple funciones de especificación de ámbito de los datos del nivel superior.

La longitud de este campo, nos permite definir hasta 8 tipos de trama distintos. Estos tipos permiten clasificar las tramas en función de la naturaleza de la información contenida en el campo de datos, por lo que podría considerarse como la cabecera de los paquetes de nivel superior. Sin embargo, por orden y simplicidad, se trata en el Nivel de Red.

De los 8 posibles tipos de trama, sólo 7 han sido definidos hasta el momento, por lo que se reserva uno para usos futuros. Ello permitiría al usuario definir un nuevo tipo específico de las aplicaciones para las que se emplee el sistema. Si además de este nuevo tipo, otros fuesen necesarios, bastaría con dotar al campo Tipo de Trama de una mayor longitud, de modo que hubiese tantos valores disponibles como tipos fueran precisos, si bien, las posibilidades que ofrece el nivel superior resultan más que suficientes para dar cabida a cualquier tipo de aplicación, cualesquiera que sean los requerimientos que esta presente.

En función del ámbito de los datos contenidos en el campo de datos de la trama, el valor y significado que toma este campo son los siguientes:

TIPO DE TRAMA = 000 ; TRAMA DE INFORMACIÓN
TIPO DE TRAMA = 001 ; TRAMA DE ASENTIMIENTO
TIPO DE TRAMA = 010 ; TRAMA DE RECHAZO
TIPO DE TRAMA = 011 ; RESERVADO PARA USOS FUTUROS
TIPO DE TRAMA = 100 ; TRAMA DE ALARMA
TIPO DE TRAMA = 101 ; TRAMA DE REGISTRO DE NODO
TIPO DE TRAMA = 110 ; TRAMA DE PETICIÓN DE CONFIGURACIÓN
TIPO DE TRAMA = 111 ; TRAMA DE INICIALIZACIÓN O IDENTIFICACIÓN

El contenido y función de las tramas de cada uno de los tipos anteriores se explicarán en el apartado correspondiente.

Longitud del Campo de Datos

Este campo tiene una longitud de 5 bits, y como su nombre indica, determina la longitud, en bytes, del Campo de Datos, además de cumplir funciones de sincronismo de trama entre transmisor y receptor .

Este campo es necesario por el hecho de que la longitud de las tramas de Nivel de Red sea variable. Tal longitud, como veíamos, estaba limitada por un valor máximo de 32 bytes. Si tenemos en cuenta que la cabecera de la trama consta de 3 bytes, y que además la trama tiene un campo de 2 bytes para control de errores, la longitud máxima del campo de datos es de 27 bytes, lo que requiere un número mínimo de 5 bits para determinar dicha longitud.

Si se decidiese dotar a la Trama de Nivel de Red de una longitud máxima mayor, bastaría con incrementar la longitud del campo Longitud del Campo de Datos, en el número de bits que fuese preciso.

En el caso particular de que la trama no tuviese campo de datos, el valor que tomaría este otro campo sería 0 bytes, es decir, 00000. Dicha situación corresponde al caso en el que la trama realiza funciones puramente de control tales como rechazo o asentimiento de tramas, petición de configuración del sistema, configuración del sistema, etc. , sin aportar información del nivel superior.

Conviene resaltar lo crítico que resulta este campo, ya que además de ser una mera referencia de longitud del campo de datos, permite sincronizar transmisor y receptor, dotando a este último de la capacidad de identificar el último byte de la trama cuya recepción se encuentra en curso, y por extensión, el primer byte de la que se recibirá a continuación. Un error en la recepción del tercer byte de la trama, en el que este campo está contenido, implicaría la pérdida de sincronismo entre transmisor y receptor, lo que constituye un fallo irreversible y obliga al establecimiento de una nueva

configuración del sistema. Se ha dotado al sistema de los medios necesarios para detectar este tipo de error y avisar al usuario de la ocurrencia de este, de modo que pueda reconfigurar el sistema inmediatamente.

En el improbable caso de que el error no sea detectado inmediatamente, la falta de sincronismo producida, llevaría a un desplazamiento indeterminado en las referencias de primer y último byte de trama, por lo que la inconsistencia de los datos que los nodos " entenderían " en recepciones sucesivas acabarían por desencadenar un error del mismo tipo. De este modo se asegura la fiabilidad del sistema, que avisa siempre al usuario de la ocurrencia de errores irreversibles en la configuración.

Campo de Datos

Este campo tiene una longitud variable, de entre 0 y 27 bytes, y contiene los datos del nivel superior.

Estos datos, que se encuentran debidamente encapsulados, no corresponden al Nivel de Red, y serán tratados en el apartado correspondiente

CRC

Este campo tiene una longitud de 2 bytes, recibe su nombre de las siglas de CLYCLIC REDUNDANCY CHECK Code, ya que se obtiene aplicando una serie de cálculos matemáticos en base a un polinomio cíclico determinado. Este campo se usa para la detección de errores en la transmisión de las tramas.

El motivo de usar este tipo de polinomios es que su uso está muy extendido y su eficacia suficientemente demostrada en la detección de errores. Su potencia de detección de errores es muy superior a la que aporta un simple Checksum. Por estas razones, dado que el sistema ha sido diseñado para aplicaciones de control, se ha optado por dotar a la trama de un CRC en lugar de un simple Checksum, pese a que la carga computacional y la complejidad del primero es algo mayor.

En concreto el polinomio generador seleccionado es del código CRC-CIITT, que define una longitud de 16 bits para el campo correspondiente. Este código permite detectar errores de longitud 16 ó menor, en bits. El polinomio generador del código es el siguiente:

$$G(X) = 1 + X^5 + X^{12} + X^{16}$$

El cálculo del CRC o en su defecto, del síndrome de la palabra recibida, puede verse implementado en las subrutinas y funciones de los programas tanto en Ensamblador como en Visual Basic, y será explicado cuando se traten estas.

Estos campos son los que componen la Trama de Nivel de Red, y siempre están presentes, salvo el Campo de Datos, que puede no aparecer. Sin embargo, es posible que en determinadas situaciones no sean necesarios o requeridos. En tales circunstancias tomarán valores basura, bien arbitrarios o bien aleatorios, según sean accedidos en escritura junto con otros campos contenidos en el mismo byte (por economía del código) o transmitidos directamente con el valor que tomasen de procesos anteriores.

4.4.3 Operación del Nivel de Red

El Nivel de Físico proporciona al Nivel el Red el servicio de transmisión y recepción de caracteres, por lo que este último se encarga directamente del ensamblaje de dichos caracteres en tramas y de la gestión de la mismas. Estos caracteres pueden haber sufrido errores durante la transmisión, errores que no detecta el nivel inferior, por lo que es función del Nivel de Red el detectar estos errores y tratar de proporcionar una solución.

Hay dos técnicas básicas disponibles para el control de errores en la transmisión. La primera de ellas se denomina *Corrección de errores hacia delante*, o más comúnmente FEC (del inglés *forward-error correction)*. Esta técnica provee al receptor de los mecanismos necesarios para regenerar los datos originales cuando estos han sufrido ciertos errores en su transmisión. Este método tiene la limitación de que si

el número de errores es numeroso, el código es poco efectivo. Así pues, para conseguir una tasa de errores baja es necesario añadir un número proporcional de bits de redundancia muy alto. Esta limitación hace que nos decantemos por la segunda de las alternativas cuya relación efectividad – coste en bits se adecua más a nuestras necesidades.

Esta segunda alternativa se llama *Petición de respuesta automática*, o más comúnmente ARQ (del inglés *automatic-repeat-request*). En este tipo de sistemas, el receptor está equipado solamente para detectar errores, pero no para corregirlos. Cuando el receptor detecta una trama en la que se ha producido algún error, envía de vuelta al transmisor una trama de control especial para que le vuelva a transmitir la trama original.

Hay tres tipos básicos de ARQ, esto es *ARQ - Parada y espera*, *ARQ – Rechazo múltiple* y *ARQ – Rechazo selectivo.*

El primer tipo, que es el más sencillo, está representado en la Figura 4.4.3.1 . Consiste en que el transmisor envía una trama al receptor, de modo que no puede volver a transmitir ninguna otra hasta recibir un asentimiento de la trama enviada desde el receptor. Si en lugar de un asentimiento, lo que recibe es un rechazo, implica que el receptor detectó un error en la trama y rechazó la misma, por lo que el transmisor deberá retransmitir la trama original. Esta opción es muy simple, pero limita mucho las posibilidades de comunicación, ya que el transmisor está un tiempo inactivo hasta recibir el asentimiento, esto es, el TOC (Tiempo de Ocupación) es muy elevado. Teniendo en cuenta que en nuestro sistema deben poder establecerse comunicaciones entre dos nodos cualesquiera, y que estos pueden estar separados entre sí físicamente por hasta 15 nodos, esta solución no es ni mucho menos óptima.

El segundo tipo, representado en la Figura 4.4.3.2, es algo más complejo y ofrece más posibilidades que el anterior. El transmisor manda tramas una tras otra sin esperar a recibir asentimiento de las mismas. El receptor lo que hace es detectar tramas erróneas y mandar un rechazo al transmisor indicando el índice de la trama errónea. Este retransmite todas las tramas a partir de la trama errónea. Se asume que el retardo de

propagación y procesado de las tramas tiene tal duración que da tiempo a transmitir y recibir rechazo de una trama en el mismo tiempo en que se tarda en transmitir un número N de tramas, de modo que transmitidas esas N tramas, se supone que la anterior a estas fue recibida sin error en caso de no recibir el rechazo en ese intervalo, por lo que no es necesario que el transmisor la conserve en memoria por más tiempo. Esta técnica no es la óptima para las necesidades del sistema, ya que un nodo se puede comunicar con varios nodos, por lo que habría que tener tantas ventanas de memoria para guardar tramas como nodos tenga el sistema, número que además es variable.

El tercer y último tipo, representado en la Figura 4.4.3.3, es el elegido para el sistema. En este caso cada trama tiene un identificador único, por lo que da igual a qué nodo vaya dirigida, ya que lo que identifica a la trama es, como su nombre indica, dicho identificador. El transmisor transmite tramas sin esperar al asentimiento y mientras tanto recibe individualmente asentimiento de cada trama. Este aprovechamiento de tiempo de transmisión permite reducir notablemente el TOC. Si lo que recibe es un rechazo, solo tiene que retransmitir aquella trama que fue rechazada. En este caso sólo se necesita una sola ventana, aunque de tamaño mayor que las del tipo anterior. Como inconveniente, si la ventana se llena el transmisor no puede transmitir una nueva trama hasta que no se reciba asentimiento de una de las que ocupa una posición de la ventana, en cuyo caso esta quedaría liberada para ser ocupada por la nueva trama.

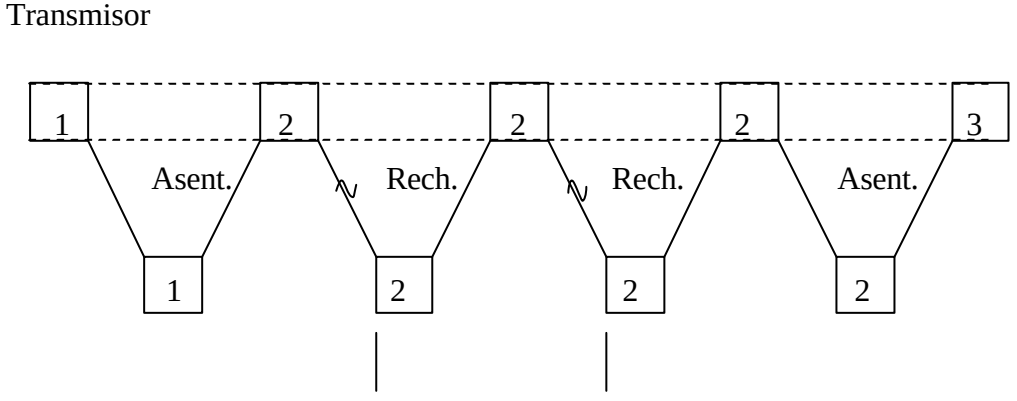


Fig. 4.4.3.1

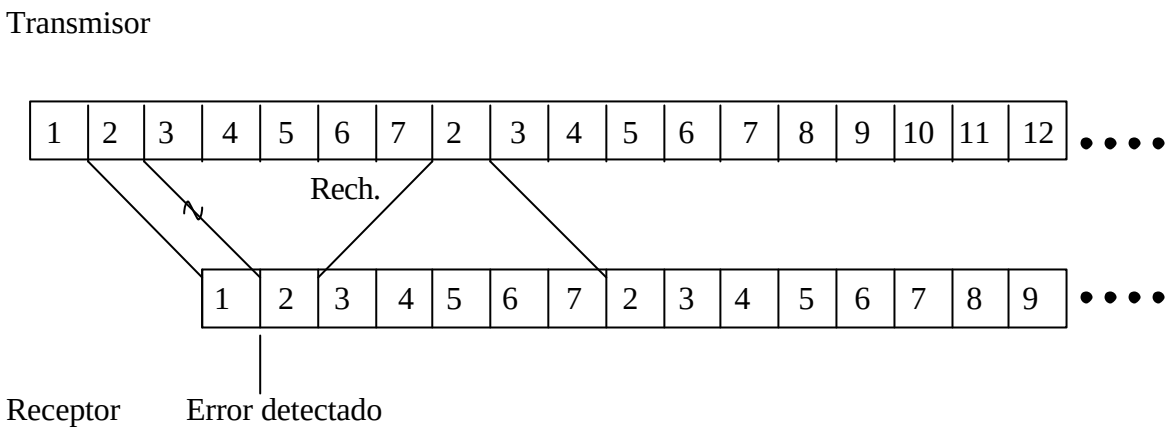


Fig. 4.4.3.2

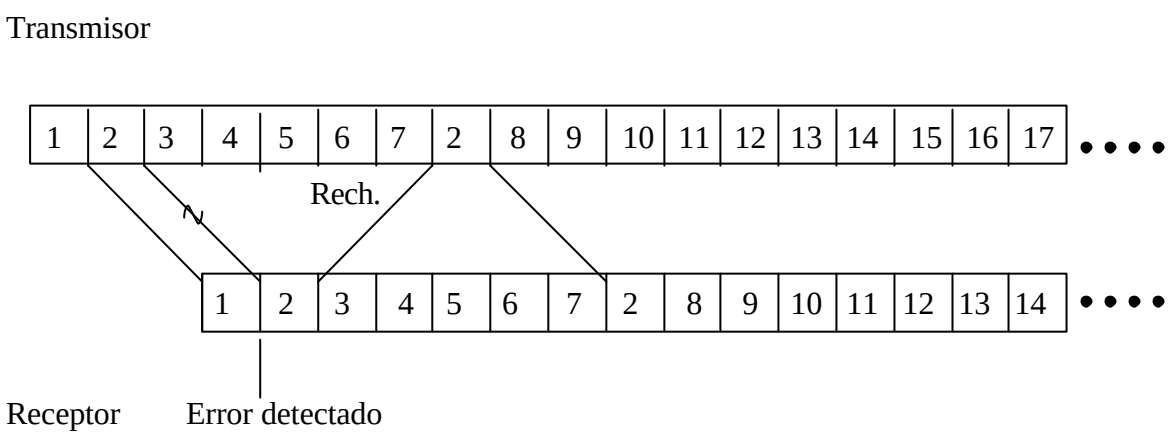


Fig. 4.4.3.3

Por lo tanto, la técnica utilizada en el sistema es ARQ - Rechazo selectivo y las tramas que contengan información relevante deberán ser asentidas. Tales tramas serán almacenadas temporalmente en memoria del nodo transmisor, de modo que si este recibe rechazo de alguna de ellas procederá a la retransmisión de la misma. Las tramas de control tales como asentimientos o rechazos no se asienten, ya que en caso de hacerse ello conduciría a un círculo vicioso que acabaría colapsando la red.

Puede darse la situación en que se produzca un error en la transmisión de un rechazo o asentimiento con lo cual el transmisor no tendría forma de saber que tiene que retransmitir la trama (si se perdió un rechazo), o eliminarla de la lista de espera de tramas en espera de asentimiento (si se perdió un asentimiento). Para solucionar esta situación lo que se hace es activar un timer para cada trama cuando esta es transmitida

en el transmisor. El valor de ese timer es tal que da tiempo a que la trama llegue a su receptor, este genere el rechazo o asentimiento, lo transmita y llegue al transmisor. Por lo tanto, si este timer ha vencido y no se ha recibido ni rechazo ni asentimiento, se asume que hay que retransmitir la trama y así se hace, iniciando nuevamente la cuenta del timer.

Para poder llevar a cabo la gestión del sistema, los nodos requieren de una serie de recursos que a continuación se presentan:

- COLAS DE PROCESADO:
  - COLA A1 para tramas con prioridad en espera de ser procesadas.
  - COLA A0 para tramas sin prioridad en espera de ser procesadas.

- COLAS DE TRANSMISIÓN:
  - COLA B1 para tramas con prioridad en espera de ser transmitidas.
  - COLA B0 para tramas sin prioridad en espera de ser transmitidas.

- LISTA DE ESPERA: para tramas en espera de asentimiento. Esta requiere de información adicional:
  - FLAGs DE OCUPACIÓN
  - TIMERS DE ASENTIMIENTO

Todas las colas son colas FIFO (First Input - First Output), de modo que su gestión la realiza también este nivel. Cada vez que llega una trama nueva, esta se almacena en la última posición de la cola correspondiente y se actualiza dicha cola.

Estos son los recursos genéricos del Nivel de Red del Protocolo de Comunicación. Dependiendo de la naturaleza y recursos del nodo, la implementación del Protocolo requerirá de recursos específicos del nodo. También dependerán de la naturaleza del nodo parámetros como por ejemplo el número de posiciones de las colas y la lista de espera. Esto se pondrá de manifiesto en el apartado en el que se explica la estructura de los programas desarrollados en Visual Basic y Ensamblador, ejecutados en el PC y los microcontroladores 68HC11 respectivamente.

La operación de este Nivel, tiene como misión dotar a los nodos de la capacidad de ensamblar y gestionar las tramas que reciban según sean los campos que estas contienen. Por otro lado, tiene también como misión permitir a los nodos transmitir las tramas que correspondan, de modo que la comunicación en el sistema se produzca de manera ordenada.

Para que ello sea posible, la operación se divide en dos partes bien diferenciadas con funciones complementarias. La primera de ellas, a la que llamaremos Encolado de Trama, está destinada a recoger los caracteres recibidos, ensamblar las tramas y prepararlas para que el nodo pueda iniciar las acciones correspondientes según sean estas y la segunda, a la que llamaremos Ciclo de Nivel de Red, tiene como cometido es llevar a cabo dichas acciones.

Es muy importante no perder de vista cual es la finalidad de los nodos. Estos se proyectan para controlar cierto proceso, de tal modo que ejecutan su aplicación de local de control (DDC), que no debe verse perturbado por el Protocolo de Comunicación entre Nodos.

El grado de perturbación no es algo abstracto, sino que puede cuantificarse mediante un índice al que llamaremos Razón de Perturbación (RP), y que ha de tomar el valor más bajo posible. Este RP responde a la siguiente fórmula:

$$RP \; ? \; \frac{TP}{TT} x100 \qquad (\%)$$

Tiempo Total de ejecución: TT = TA + TP ;

Tiempo de Aplicación: TA ; Incluye tiempo correspondiente al Ciclo de Nivel de Aplicación (aprox. constante)

Tiempo de ejecución de Protocolo: TP = TCR + TET ;

Tiempo de ejecución de ciclo nivel de red: TCR;

Tiempo de ejecución de Encolado de Tramas: TET;

RP constituye un estadístico, con lo cual conviene que el tiempo de observación para la medida de los tiempos sea lo mayor posible ya que estos tiempos no son constantes, sino que dependen del tráfico de la red (TP) y de los eventos que genere la planta (TA) .

TA depende directamente de la aplicación local de control, por lo que es imposible darle un valor este término para un estudio teórico.

No obstante, al diseñar una aplicación, este es el índice apropiado para evaluar la conveniencia del sistema para albergar esa aplicación. Si calculado este índice, resulta tener un valor elevado, digamos por encima del 10 % para el nivel medio de tráfico en la red, puede que el sistema no resulte eficiente para esa aplicación en concreto.

Otro factor a tener en cuenta es el grado de perturbación introducido por el Protocolo cuando la red está en reposo, es decir, cuando no hay transmisión/recepción ni procesado de mensajes.

Para este caso se define un nuevo índice al que llamaremos Razón de Perturbación en Vacío (RPV), que por supuesto ha de tomar también el mínimo valor posible. Este RPV responde a la siguiente fórmula:

$$RPV\ ?\ \frac{TCRV}{TTCV}x100 \qquad (\%)$$

Tiempo Total de ejecución de Ciclo en Vacío: TTCV = TCAV + TCRV ;
Tiempo de ejecución de Ciclo de Aplicación en Vacío: TCAV
Tiempo de ejecución de Ciclo de Red en Vacío: TCRV;

Este índice es más fácil de calcular que el anterior , ya que en este caso TCRV es fijo e igual al tiempo que se tarda en ejecutar el ciclo de nivel de red cuando no hay comunicaciones, es decir, cuando todas las colas están vacías al igual que la lista de

espera. En este caso el protocolo está diseñado para que se comprueben ciertos flags, de modo que si estos dictan que las colas están vacías, no se efectúa ninguna operación, de modo que se minimiza TCRV.

El valor que toma TCAV sigue dependiendo de los eventos generados por la planta aunque ahora solamente hay que medir el tiempo de un solo ciclo.

NOTA: Todos los tiempos dependen del tiempo de ejecución del micro, y del código del programa que ejecuten, por lo que se discutirá sobre dichos valores en el apartado 5.1 Consideraciones sobre el Código – Código de Programa para 68HC11.

A continuación se presentan los procesos que forman parte del funcionamiento del Nivel de Red.

Encolado de Tramas

El nodo, gracias al Nivel Físico, recibe caracteres de manera asíncrona de modo que a priori es incapaz de determinar si estos son datos, CRC o por el contrario pertenecen a la cabecera de la trama. Cada vez que se recibe un carácter o conjunto de caracteres se dispara este proceso, independientemente del momento en que se produzca, es decir, cuando se produce el evento de recepción se interrumpe la ejecución del nodo y se ejecuta el proceso de encolado de tramas.

Lo primero es conseguir ensamblar la trama a partir de los caracteres recibidos, para lo cual es necesario que haya cierto sincronismo de trama entre emisor y receptor. Este sincronismo, como habíamos visto, lo proporciona el campo Longitud de Datos, de tal modo que el nodo puede identificar el primer y último carácter de trama cada vez que estos se reciben. Ello es posible porque en el momento de recibir el tercer carácter de trama, el nodo lee el contenido del campo mencionado, de tal modo que tras recibir un número de caracteres igual al valor de este, más dos bytes de CRC, el nodo sabe que el siguiente carácter que recibirá será nuevamente el primero de una nueva trama, iniciándose otra vez el mismo proceso. De este modo el nodo es capaz de reensamblar

las tramas recibidas. La Figura 4.4.3.4 representa el diagrama del ensamblaje de las tramas a partir de los caracteres recibidos.

El punto crítico del ensamblaje, es que se produzca un error en la recepción del carácter que contiene el campo Longitud del Campo de Datos, ya que se perdería el sincronismo de trama entre emisor y receptor, lo que obliga a reconfigurar el sistema, que no puede recuperarse del error.

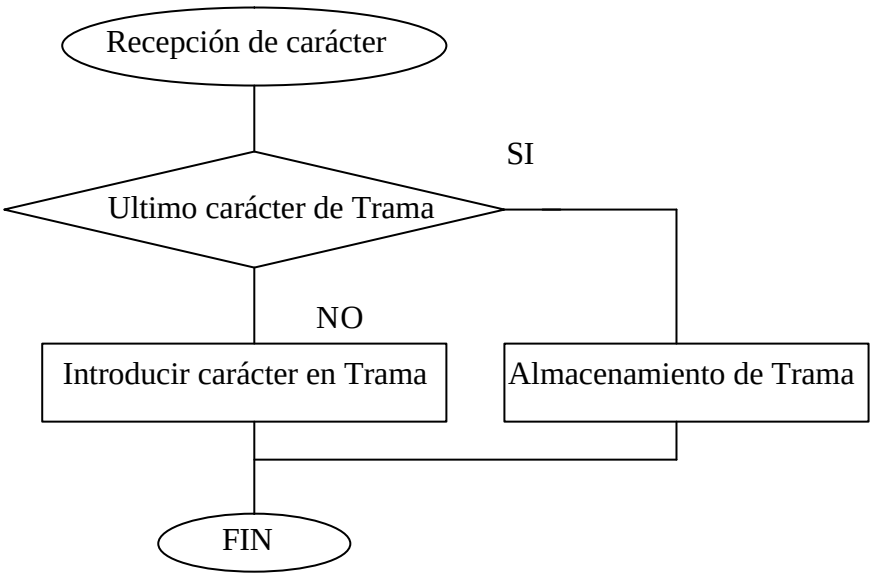


Fig. 4.4.3.4

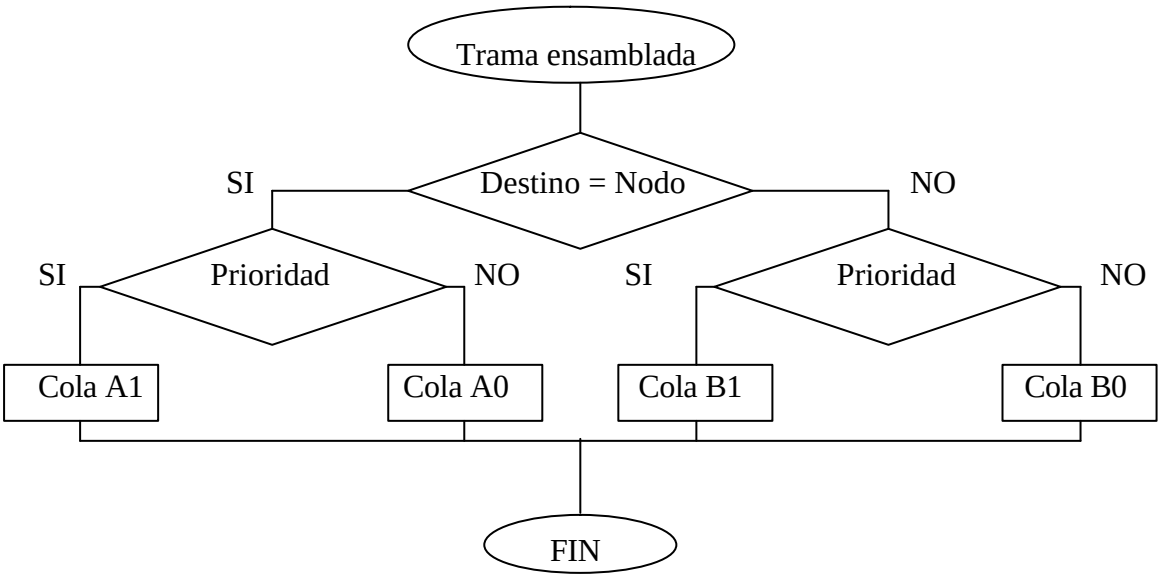


Fig. 4.4.3.5

Cuando se produce un error en la recepción de alguno de los caracteres de la trama (salvo en el caso de que sea el tercer carácter de trama), la trama es desechada, con lo cual no llega a ensamblarse ni a ser introducida en ninguna de las colas.

Una vez ensamblada la trama, es necesario almacenarla de forma ordenada según el contenido de su cabecera. Este almacenamiento atiende a dos campos, PRI y Dirección Destino, de modo que el almacenamiento se producirá en diferentes colas según sus contenidos. La figura 4.4.3.5 presenta el diagrama que rige el almacenamiento de tramas.

Dicho diagrama implica que cuando se recibe una trama, si esta va dirigida al propio nodo, se clasifica como trama para procesado, mientras que va dirigida a otro nodo, se clasifica como trama para retransmisión. Como las tramas pueden o no tener prioridad, se almacenan en colas diferentes, ya que, como se verá en el Ciclo de Nivel de Red existen diferencias en el tratamiento de tramas con y sin prioridad. El esquema de la Figura 4.4.3.6 resume el proceso de Encolado de Tramas dando una interpretación física de la misma:

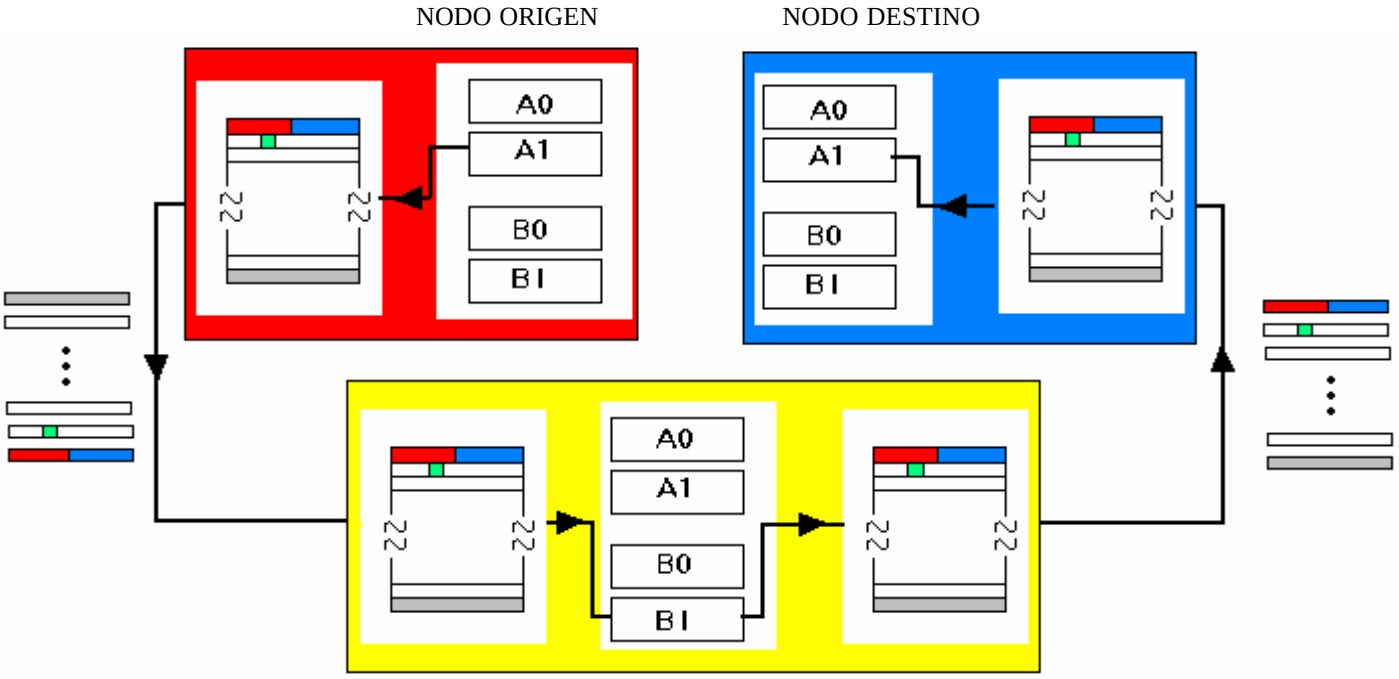


Fig. 4.4.3.6

Ciclo del Nivel de Red

Este ciclo tiene como objetivo el de gestionar la transmisión de las tramas que correspondan, así como el procesado de las tramas recibidas dirigidas al propio nodo. Tanto unas como otras han sido preparadas por el proceso de Encolado de Tramas lo que permite al ciclo acceder directamente a las tramas recibidas sabiendo qué tiene que hacer con ellas (procesado o transmisión) y en qué orden (con o sin prioridad).

Mientras que el proceso de Encolado de Tramas era disparado por la aparición del evento de recepción de carácter, este ciclo, como su nombre implica, es periódico y no es disparado por la ocurrencia de ningún evento.

El diagrama de la Figura 4.4.3.7 muestra el algoritmo ejecutado en cada nodo, incluyendo en este el Ciclo de Nivel de Red, mientras que la Figura 4.4.3.8 muestra el funcionamiento del propio ciclo.

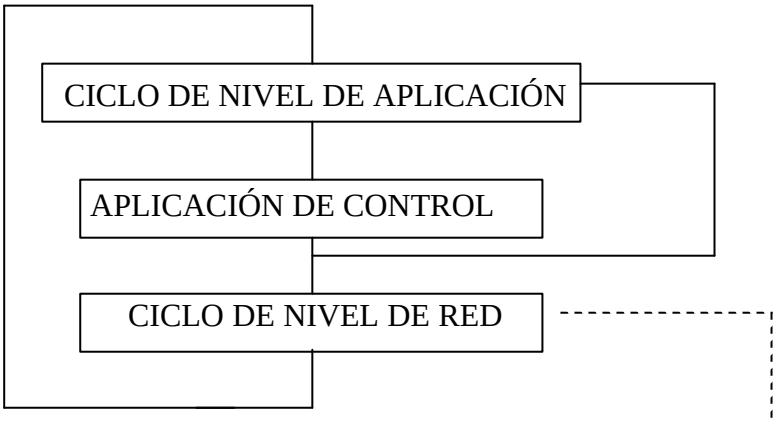


Fig. 4.4.3.7

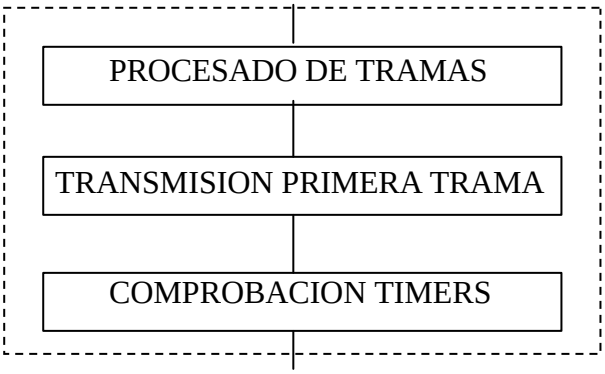


Fig. 4.4.3.8

Siguiendo adelante con el análisis funcional del Ciclo de Nivel de red, vemos ahora en qué consiste cada uno de los bloques que lo componen.

Procesado de Tramas

El diagrama de flujo de este bloque se presenta en la figura 4.4.3.9:

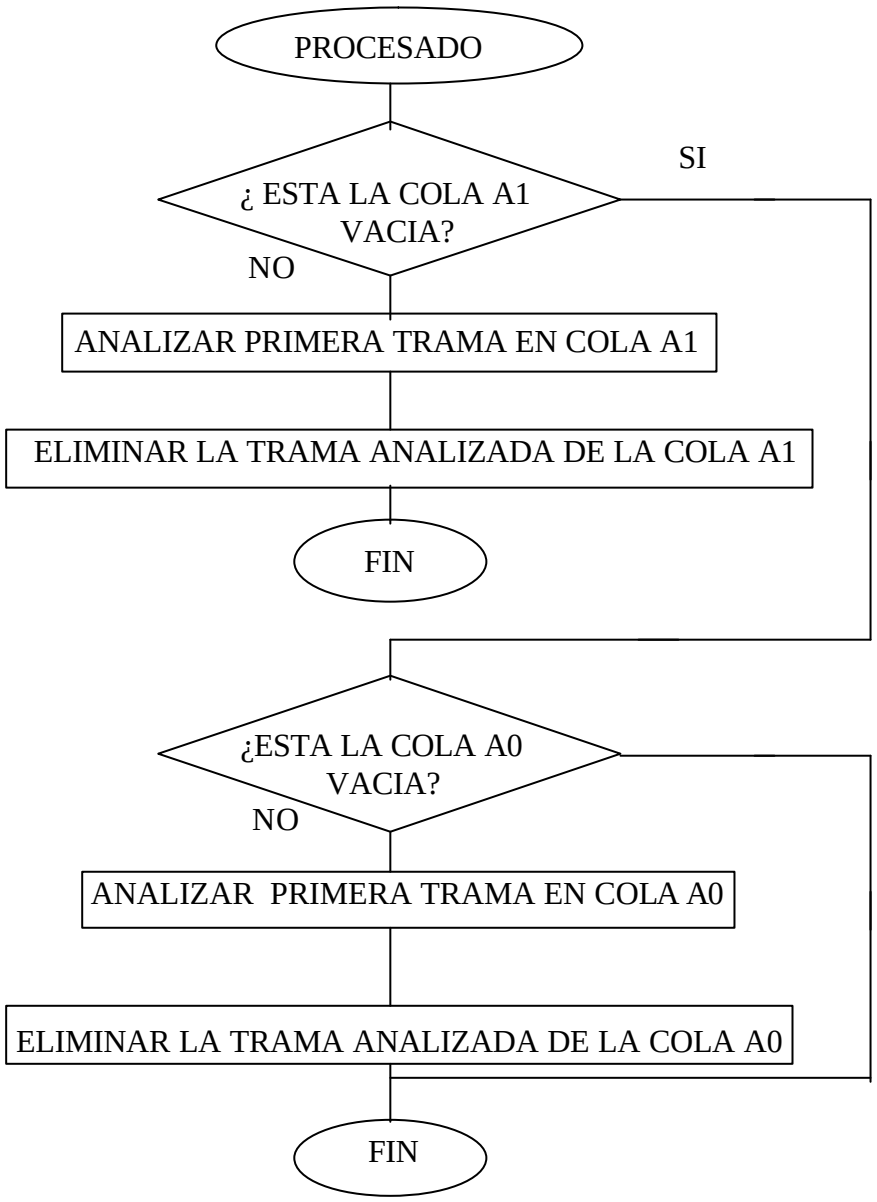


Fig 4.4.3.9

El análisis de las tramas procesadas sigue el diagrama de flujo recogido en la figura 4.4.3.10:

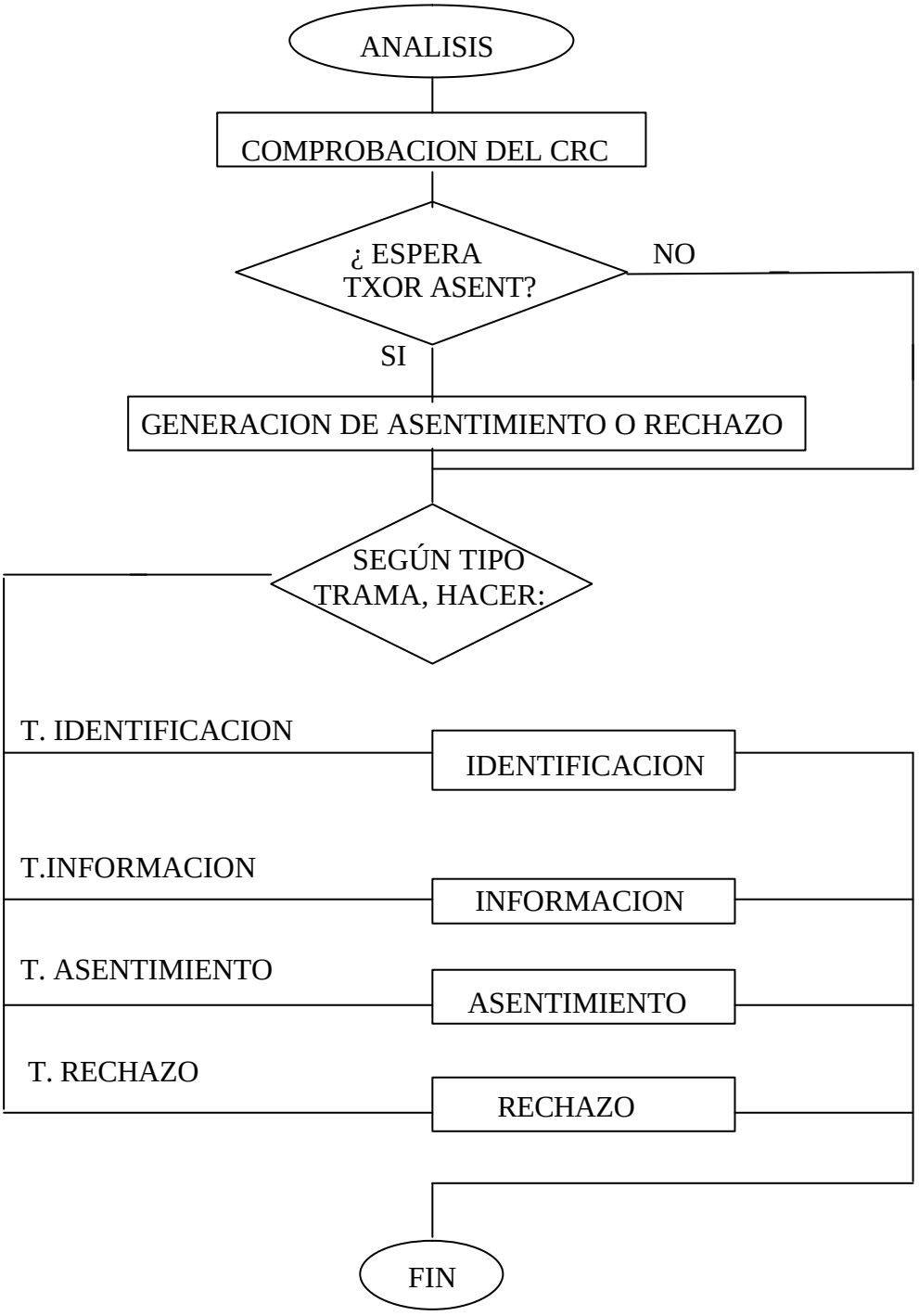


Fig. 4.4.3.10

Como se aprecia en la figura 4.4.3.9, el bloque de procesado de tramas lo que hace es comprobar si hay alguna trama que procesar en el orden que dicta la Prioridad.

En caso de que ambas Colas de Procesado estén vacías, en ese caso no se efectúa ninguna operación adicional, lo que minimiza el tiempo de ejecución. Si por el contrario Cola A1 no está vacía, se procesa la primera trama en dicha cola, y en caso de estarlo se comprueba Cola A0.

En cualquier caso, tan sólo se procesa una trama, ya que el tiempo mínimo de recepción de una trama (correspondiente a una trama sin campo de datos) es superior al tiempo de ejecución de un ciclo por largo que este resulte, ya que a la velocidad de transmisión (9600 Bps) es necesario que transcurra tiempo equivalente a varios ciclos de programa para recibir la trama completa:

Tiempo mínimo: 1/9600 s/bits x 5 bytes/trama x (8+1) bits/byte = 4,69 ms
Tiempo máximo: 1/9600 s/bits x 32 bytes/trama x (8+1) bits/byte = 30 ms

El procesado de la cola consiste en extraerla de la cola en que se encuentre y actualización de esta. Además, se activa el análisis de la trama en cuestión.

Como se puede observar en la figura 4.4.3.10, el análisis de una trama consiste en la comprobación de su CRC, generación del correspondiente Asentimiento/Rechazo si se requiere el mismo, y ejecución de diferentes secuencias de código según el tipo de trama.

En la comprobación del CRC se genera el síndrome de la trama a partir de la división binaria de la misma por el polinomio generador CRC-16 del CCITT. Si el síndrome resulta nulo significa que la trama se recibió sin errores, con lo cual su análisis puede continuar. En caso contrario significa que se produjo algún error, con lo que la trama debe ser desechada, pero antes se genera un rechazo hacia el emisor de la trama indicándole que debe retransmitirla (sólo en caso de que la trama errónea recibida requiera asentimiento).

Las secuencias de código asociadas al tipo de trama realizan las siguientes operaciones:

Identificación: Se toma el campo de datos de la trama, cuyo valor damos al ID del nodo. Se incrementa el dato y se retransmite la trama con el nuevo campo de datos y el CRC recalculado. También se genera una Trama de Registro de Nodo con información sobre la naturaleza del nodo para enviárselos al PC.

Información: Extrae los paquetes de aplicación y se los pasa al nivel superior para que este haga con ellos lo que corresponda.

Asentimiento: Se elimina la trama que ocupa la posición indicada por el Id numérico del asentimiento en la lista de espera, ya que hay seguridad de que llegó a su destino con éxito.

Rechazo: Se retransmite la trama que ocupa la posición indicada por el Id numérico del asentimiento en la lista de espera, ya que hay seguridad de que llegó a su destino con errores.

Transmisión primera trama

Las tramas que han de transmitirse están almacenadas en las Colas B1 y B0, según su prioridad. Estas tramas han de transmitirse byte a byte, de tal modo que una vez activada la transmisión del primero de los bytes que la componen, el resto se transmiten automáticamente cada vez que finaliza la transmisión del anterior. Cuando finaliza la transmisión del último de los bytes de la trama, se activa el flag correspondiente que informa de que no hay transmisión en curso.

Al igual que en el caso anterior es importante perturbar mínimamente, por lo primero que se hace es comprobar si hay o no transmisión en curso, en cuyo caso no se activa una nueva. Después se comprueban las Colas B, de modo que si están vacías no se activa transmisión. La analogía con el caso anterior es obvia.

El diagrama de flujo correspondiente se muestra en la figura 4.4.3.11:

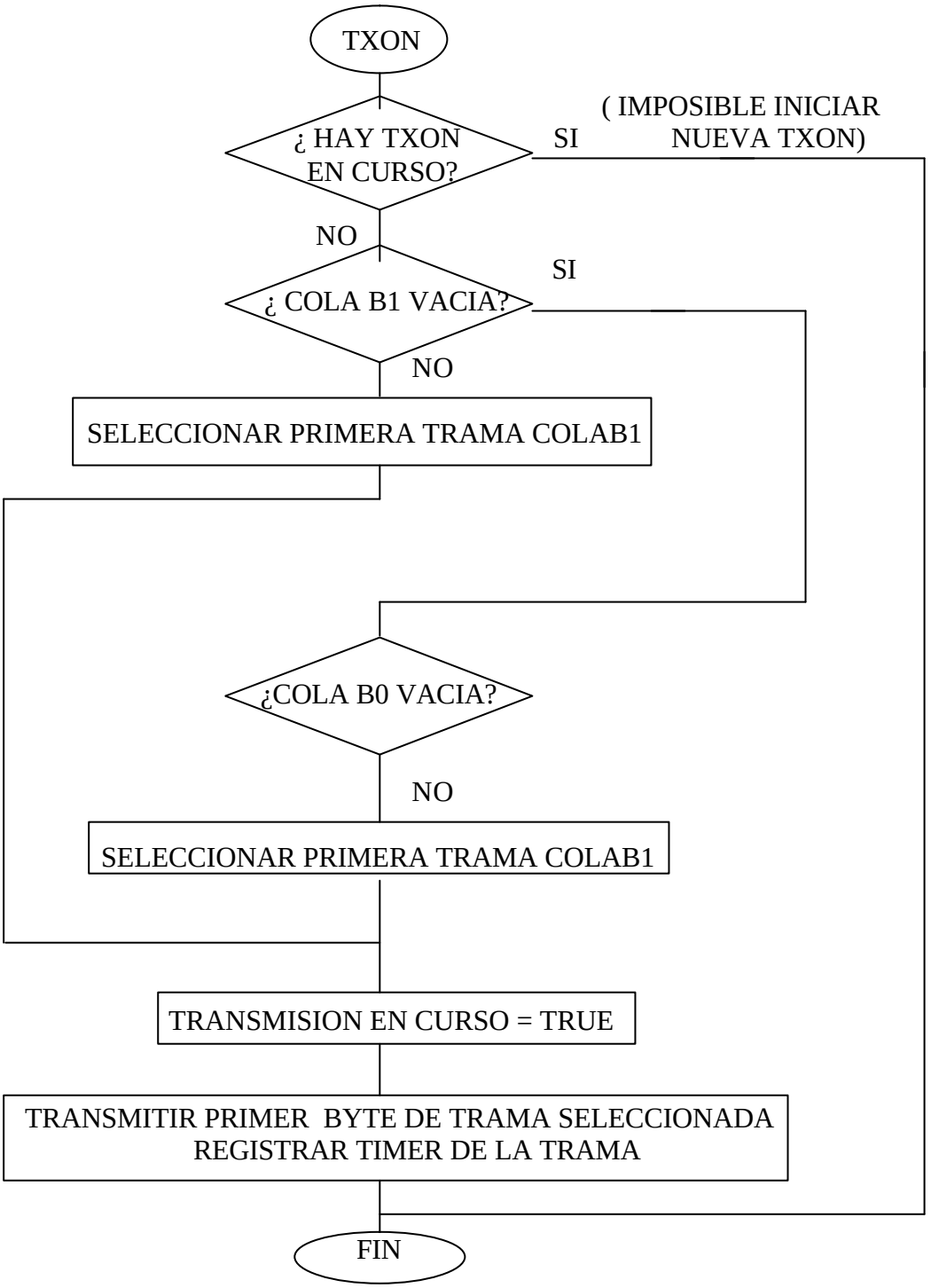


Fig. 4.4.3.11

Es necesario también registrar el timer de la trama transmitida en caso de que esta haya sido generada por el propio nodo y requiera asentimiento, con lo cual ha de encontrarse almacenada en la lista de espera.

Comprobación Timers

El diagrama de flujo es el que aparece en la figura 4.4.3.12. P es el número de posiciones de la Lista de Espera.

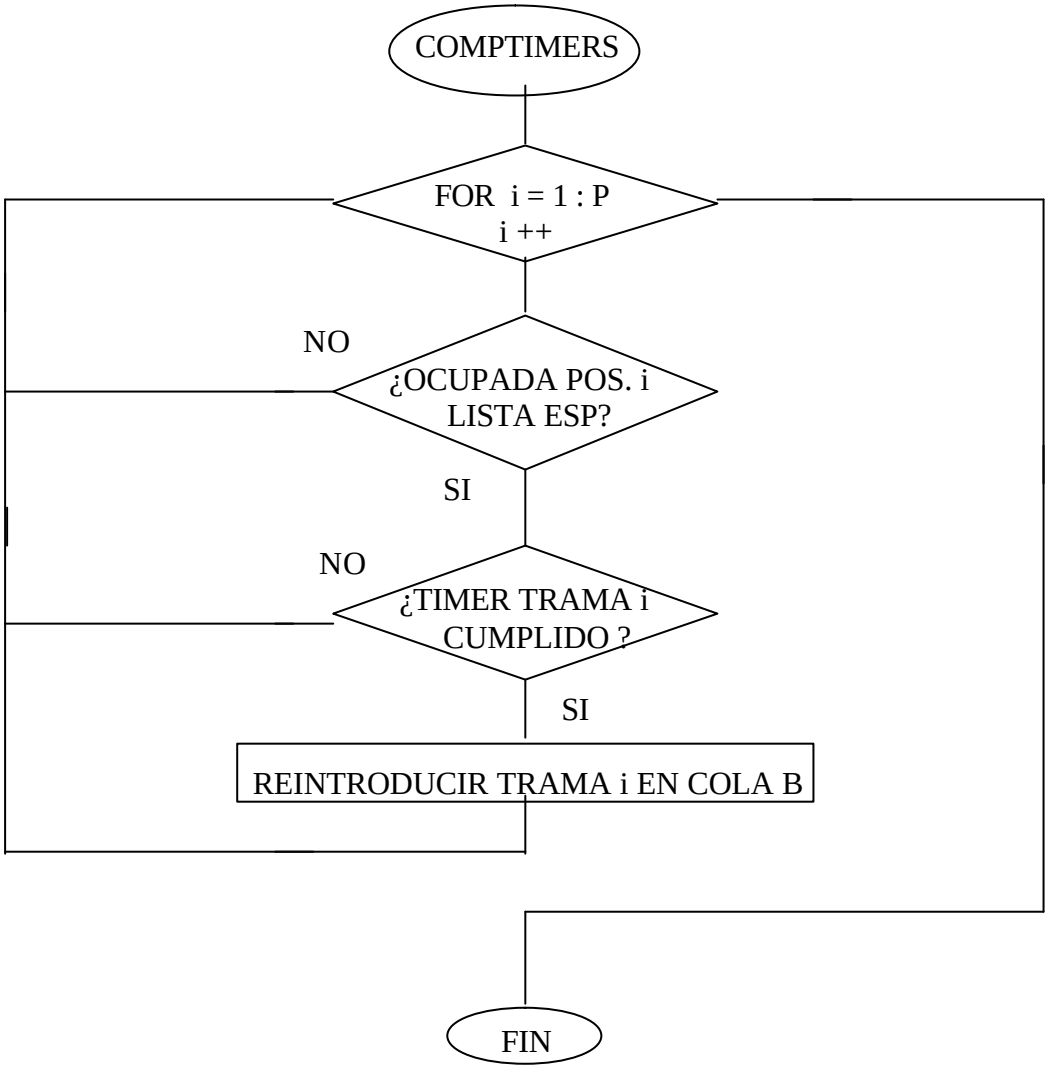


Fig. 4.4.3.12

Como vimos en el bloque anterior, cada vez que una trama generada por el propio nodo que requiere asentimiento es transmitida, su timer de asentimiento queda registrado con el valor umbral del reloj del nodo para el cual habrá que retransmitir la trama. Esto es porque se entiende que ha habido tiempo suficiente para que llegase el asentimiento esperado. El hecho de que no se haya recibido implica que o la trama o su asentimiento se han perdido, en cuyo caso es necesario retransmitir la trama.

Este bloque lo que hace es comprobar cada posición de la lista de espera. En caso de que estén ocupadas, se comprueba si su timer ha vencido, y si lo ha hecho, se retransmite la trama en cuestión, es decir, se reintroduce en colas de transmisión. En el

caso más sencillo tan sólo se comprueban las posiciones, no haciéndose nada más al no encontrarse ocupadas.

4.4.4 Tipos de Trama de Nivel de Red

Para la definición de los Tipos de Trama de Nivel de Red contamos con 3 bits, lo que nos proporciona 8 tipos posibles, de los que sólo se usan 7 de ellos, lo que deja 1 como reserva para futuras ampliaciones. Estos tipos son los siguientes:

TIPO DE TRAMA = 000 ; TRAMA DE INFORMACIÓN
TIPO DE TRAMA = 001 ; TRAMA DE ASENTIMIENTO
TIPO DE TRAMA = 010 ; TRAMA DE RECHAZO
TIPO DE TRAMA = 011 ; RESERVADO PARA USOS FUTUROS
TIPO DE TRAMA = 100 ; TRAMA DE ALARMA
TIPO DE TRAMA = 101 ; TRAMA DE REGISTRO DE NODO
TIPO DE TRAMA = 110 ; TRAMA DE PETICIÓN DE CONFIGURACIÓN
TIPO DE TRAMA = 111 ; TRAMA DE INICIALIZACIÓN O IDENTIFICACIÓN

Trama de información

Este tipo de trama encapsula paquetes de aplicación, del nivel superior, en su campo de datos. Para su análisis, se extraen dichos paquetes y se entregan al nivel superior para que genere la secuencia de operaciones correspondiente. Estas tramas, por defecto y salvo decisión contraria del programador, requieren asentimiento, ya que llevan información neta de origen a destino y si no llega a este ocasiona pérdida de prestaciones en el sistema.

Trama de asentimiento

Estas tramas las generan los nodos cuando reciben tramas a ellos dirigidas sin errores (síndrome nulo), y sirven para informar al emisor de dicha recepción.

Cuando llega una trama sin errores que requiere asentimiento (ASE = 1), se genera el asentimiento dirigido al nodo emisor de la trama procesada, con dirección de origen la del nodo que la procesa.

El asentimiento lleva implícita su información en el ID numérico de la trama, ya que identifica la posición de la lista de espera del nodo al que va dirigido el asentimiento que almacena la trama asentida y que por lo tanto es posible eliminar de la lista.

Al generarse el asentimiento, al Id numérico del mismo se le da directamente el valor del de la trama procesada.

Estas tramas no requieren nunca asentimiento, del mismo modo que su campo Longitud del Campo de Datos toma siempre el valor 0, lo que implica que la trama carece de Campo de Datos.

Trama de rechazo

Estas tramas las generan los nodos cuando reciben tramas a ellos dirigidas en las que se ha detectado algún error (síndrome no nulo), y sirven para informar al emisor de dicha recepción no exitosa.

Cuando llega una trama con errores que requiera asentimiento (ASE = 1), se genera el rechazo dirigido al nodo emisor de la trama procesada, con dirección de origen la del nodo que la procesa.

El rechazo lleva implícita su información en el ID numérico de la trama, ya que identifica la posición de la lista de espera del nodo al que va dirigido el rechazo que almacena la trama rechazada y que por lo tanto es necesario que se retransmita.

Al generarse el rechazo, al Id numérico del mismo se le da directamente el valor del de la trama procesada.

Estas tramas no requieren nunca asentimiento, del mismo modo que su campo Longitud del Campo de Datos toma siempre el valor 0, lo que implica que la trama carece de Campo de Datos.

Trama de alarma

Estas tramas encapsulan mensajes de alarma dirigidas a la Estación de Operación desde cualquier nodo. Estos mensajes, de 27 caracteres a lo sumo, se insertan directamente en el campo de datos de la trama carácter a carácter (en código ASCII). En la Estación de Operación simplemente se reagrupan los bytes para originar la cadena de caracteres que contiene el texto de alarma.

Esta configuración es susceptible de cualquier cambio que el usuario desee hacer, ya que las función encargada del análisis de este tipo de trama es totalmente independiente y puede ser sustituida con suma facilidad. De igual modo pueden introducirse funciones de librería para el análisis de las tramas de alarma de acuerdo a las necesidades de la aplicación de usuario.

Las tramas de alarma son generadas automáticamente en los nodos cuando se produce algún evento que desencadena una secuencia de operaciones entre las cuales está la de generación de alarma. Este evento implica normalmente alguna anomalía en los nodos o en la planta.

Estas tramas requieren asentimiento y son prioritarias debido a que la información que llevan es normalmente crítica..

Trama de Registro de Nodo

Estas tramas son enviadas por los nodos al PC durante el proceso de configuración del sistema y contienen información sobre la naturaleza del nodo (E/S

activas, direccionalidad de ciertos pines bidireccionales y función de dichas E/S). Esta información se corresponde con una ristra de 7 bytes que se introducen directamente en el campo de datos de la trama y que se encuentran almacenados en memoria local no volátil. Sobre ello se discutirá en el capítulo 5, correspondiente a las Consideraciones sobre el Código.

Estas tramas se generan cuando el nodo ha recibido una Trama de Inicialización o Identificación, de modo que son enviadas al PC para que este lo registre en la configuración del sistema. Ello permite que en la estación de operación haya información detallada de cada uno de los nodos del sistema en lo referente a posición y señales de control de cada nodo.

Estas tramas requieren asentimiento, y siempre toman el Id numérico de trama 00000, ocupando la posición 0 de la Lista de Espera, ya que es la primera trama que requiere asentimiento que genera cada nodo.

Trama de Petición de Configuración

Tramas dirigidas siempre al PC por cualquiera de los nodos. Informan al PC de que se ha producido algún error fatal en el sistema (pérdida de sincronismo de trama, nodo no registrado etc.) para que este adopte las medidas automáticas oportunas, esto es, informar al usuario y anular la configuración vigente que resulta ya inútil.

Son tramas fijas, es decir, todos sus campos toman siempre el mismo valor, ya que la dirección origen de la trama toma un valor basura fijo, por lo que no depende del nodo que emita la trama. Carecen de campo de datos, y al ser fijas, no requieren cálculo de CRC, ya que este es conocido.

Trama de Inicialización o Identificación

Son tramas multicast (dirigidas a todos los nodos del sistema) emitidas por el PC. Su campo de datos lo compone únicamente 1 byte. El nodo que recibe esta trama toma su ID de este dato. Cuando el PC la genera, el campo de datos toma el valor 1 (ID

del primer nodo que recibirá la trama), y cuando vuelve a él tras recorrer todos los nodos toma un valor igual al número de nodos del sistema más 1.

Al tratarse de una trama multicast no requiere asentimiento y por otro lado, es una trama prioritaria.

Reserva

Como hemos visto, el sistema dispone de 1 tipo de trama de reserva. Este puede ser utilizado por el usuario a su antojo por ejemplo para definir otro tipo de alarmas diferente al existente (mensaje de texto).

No obstante, si más tipos fueran necesarios, bastaría con aumentar la longitud en bits del campo Tipo de Trama, de modo que por cada bit en que se aumente esta, el número de tipos se duplica.

Lo aconsejable sin embargo no es recurrir a la solución anterior, ya que lo que se pretende es que el usuario no descienda al Nivel de Red. Por ello se ha dispuesto que haya una amplia reserva de instrucciones del Nivel de Aplicación a disposición del operador, como veremos en el capítulo correspondiente.

4.5 NIVEL DE APLICACIÓN

4.5.1 Generalidades

Este es el más alto nivel del Protocolo, y el que añade todas las funciones útiles al usuario, es decir, este nivel define el grado de manipulabilidad de los nodos del sistema por parte del usuario u operador del mismo.

Del nivel inferior recibe los paquetes de aplicación totalmente libre de errores, de tal modo que sólo hay que preocuparse ya de dar formato a esos paquetes de acuerdo a las operaciones que se desee realizar. También es transparente a este nivel la forma que tiene el sistema de hacer llegar las instrucciones a su destino, ya que veíamos que era misión del nivel inferior.

Es el Nivel de Aplicación el que admite un mayor grado de intervención del usuario para adaptar el sistema a su aplicación específica. En el apartado Juego de Instrucciones veremos como es ello posible.

Entre sus funciones se encuentra también la selección de la Aplicación Local a ser ejecutada en los nodos del sistema.

4.5.2 Paquete de Nivel de Aplicación

El Paquete de Nivel de Aplicación constituye la unidad de datos fundamental de dicho nivel, si bien este no es ni mucho menos tan complejo como la Trama de Nivel de Red.

Formato del Paquete de Aplicación

El paquete consta de tan sólo 2 campos, tal y como refleja la figura 4.5.2.1, aunque el campo de datos puede o no estar presente como se verá más adelante. Además el campo de datos incluirá a su vez sus propios campos que dependerán de la instrucción. Ello implica que dicho campo tiene una longitud variable.

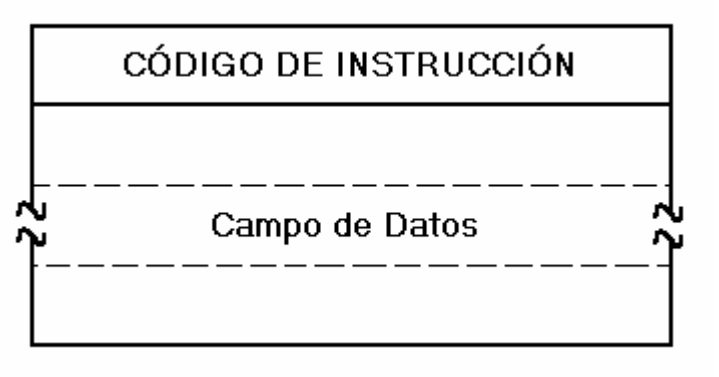


Fig. 5.4.2.1

Longitud del Paquete de Aplicación

Esta longitud, como hemos visto en el párrafo anterior es variable, y toma valores en un rango de 0 (carece de campo de datos) a 26 bytes. Sin embargo en la cabecera del paquete no consta información explícita relativa a la longitud del campo de datos.

Para tener delimitados perfectamente los límites del paquete, se cuenta con la información implícita en el Código de Instrucción, ya que cada uno de ellos lleva asociada una longitud del campo de datos determinada. En este caso se ha optado por esta opción ya que cada instrucción tiene una función para su análisis que incluye código para delimitar el paquete. Esto nos permite ahorrarnos un campo en el paquete, lo que aumenta la eficiencia del protocolo.

4.5.3 Operación del Nivel de Aplicación

Para generar paquetes de aplicación este nivel toma el código de instrucción de la operación que se quiere llevar a cabo y lo introduce en el primer campo. Luego configura el resto de los campos dependiendo de la instrucción que sea y los añade a continuación, entregándole el paquete al nivel inferior que es el que se encarga de encapsularlos adecuadamente en tramas para su transmisión. Al recibir un paquete actúa

de forma inversa, es decir, lee el código y opera con los datos de acuerdo al código leído.

Como la operación depende en gran medida del tipo de instrucción, será en el siguiente apartado donde se trate el modo de operación asociado a cada instrucción.

El aspecto hasta ahora discutido de la operación del Nivel de Aplicación se limita al tratamiento de los paquetes (generación y recepción), sin embargo, una parte importante de este nivel está dedicada a la selección de la Aplicación Local a ser ejecutada en cada nodo. Para ello cada ciclo de Aplicación del programa ejecutado localmente por cada microcontrolador consta a su vez de un Ciclo de Nivel de Aplicación cuyo diagrama de flujo aparece en la figura 4.5.3.1.

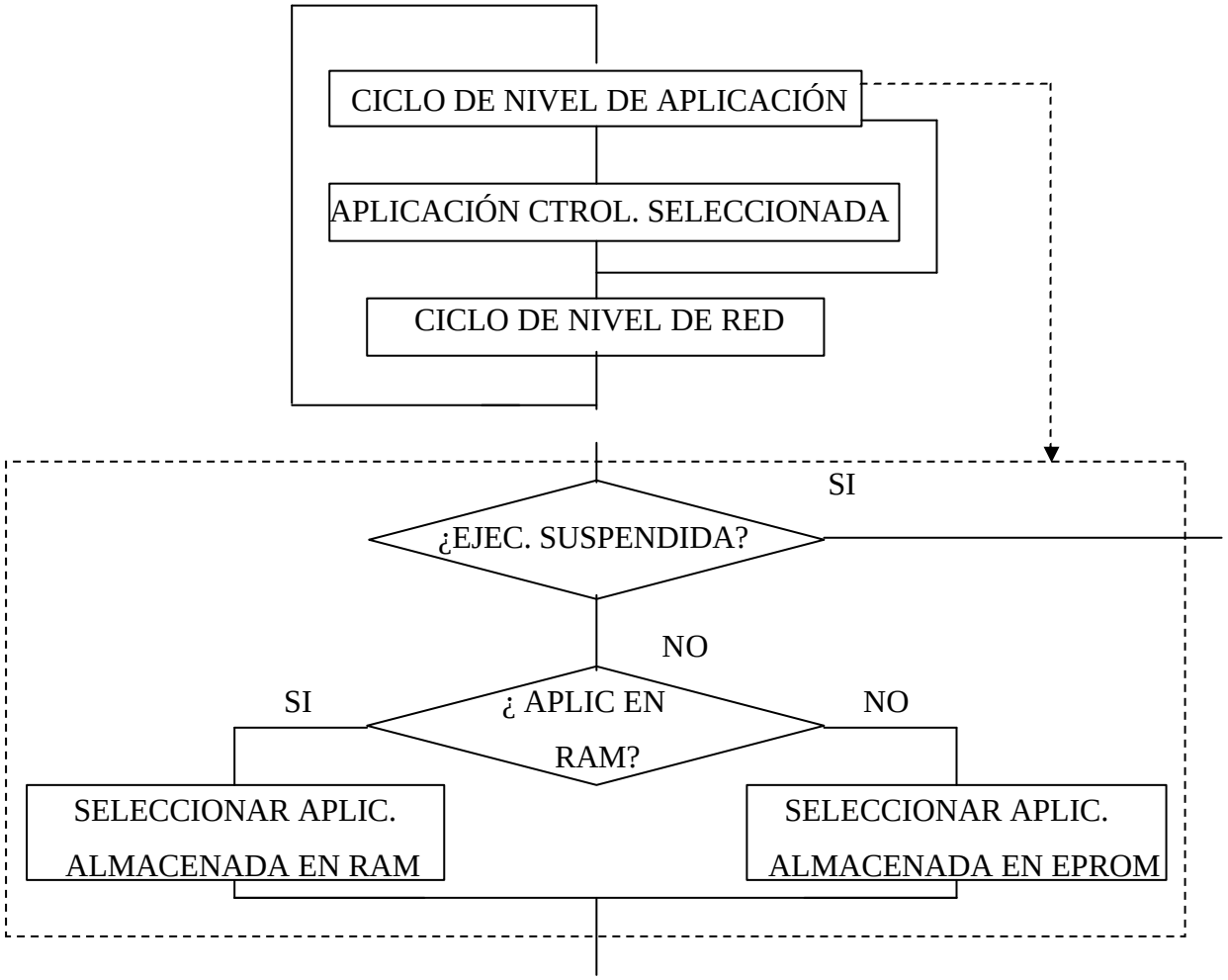


Fig. 4.5.3.1

Para la hacer la selección hay instrucciones disponibles, como veremos en el siguiente apartado. Estas instrucciones permiten modificar los Flags de EJECUCIÓN SUSPENDIDA y EJECUTAR PROGRAMA EN RAM, de tal modo que cada vez que comienza un ciclo de programa estos flags son chequeados según hemos visto en el diagrama de flujo.

Según los valores de estos flags se ejecuta una de entre dos posibles secciones de código almacenado en diferentes posiciones en el mapa de memoria del micro. También puede saltarse la ejecución de la aplicación local (Ejecución Suspendida) de modo que el micro no ejecuta ninguna aplicación de control pero sí que ejecuta el protocolo, lo que permite mantener la red en servicio sin necesidad de recablearla cuando se decide prescindir del control ejercido por tal nodo.

De las dos opciones posibles, la correspondiente a Aplicación en RAM, permite que la Aplicación de Control pueda ser sustituida tantas veces como sea necesario. Las nuevas aplicaciones se cargan desde la Estación de Operación mediante las instrucciones correspondiente que veremos más adelante. La Aplicación en RAM que se encuentre en memoria, desaparece al dejar de alimentar al nodo, en cuyo caso es necesario volver a cargarla desde el PC antes de activar su ejecución al configurar nuevamente el sistema.

La otra opción, la Aplicación en EPROM, es fija y ha de ser programada en memoria no volátil junto con el Protocolo antes de realizar la instalación del sistema. Esta aplicación no desaparece al dejar de alimentar al nodo, y es la aplicación seleccionada por defecto al configurar el sistema.

No obstante, sea cual sea la opción elegida, la ejecución está por defecto suspendida al configurar el sistema. Ello obliga a activarla para que el nodo actúe sobre el proceso que controla. Esta solución se ha adoptado para evitar fenómenos indeseados durante la configuración del sistema.

4.5.4 Juego de Instrucciones

Como hemos visto, los paquetes cuentan con un campo de cabecera que contiene el código de la instrucción a la que corresponde el propio paquete. Dicho código de instrucción tiene una longitud de 8 bits, lo que nos proporciona 256 posibles instrucciones. Estas instrucciones son las siguientes:

CÓDIGO DE INSTRUCCIÓN = 0000 0001 = 1 ; LEER DATO
CÓDIGO DE INSTRUCCIÓN = 0000 0010 = 2 ; ESCRIBIR DATO
CÓDIGO DE INSTRUCCIÓN = 0000 0100 = 4 ; EJECUTAR PROGRAMA EN RAM
CÓDIGO DE INSTRUCCIÓN = 0000 1000 = 8 ; COPIAR PROGRAMA EN RAM
CÓDIGO DE INSTRUCCIÓN = 0000 1001 = 9 ; FIN COPIAR PROG. EN RAM
CÓDIGO DE INSTRUCCIÓN = 0000 1010 = 10 ; PROGRAMA COPIADO EN RAM
CÓDIGO DE INSTRUCCIÓN = 0000 1011 = 11 ; PRIMERA COPIA EN RAM
CÓDIGO DE INSTRUCCIÓN = 0001 0000 = 16 ; EJECUTAR PROG. EN EPROM
CÓDIGO DE INSTRUCCIÓN = 0010 0000 = 32 ; SUSPENDER EJECUCIÓN
CÓDIGO DE INSTRUCCIÓN = 0100 0000 = 64 ; ACTIVAR EJECUCIÓN
CÓDIGO DE INSTRUCCIÓN = 1000 0000 = 128 ; DATO LEIDO

Por lo tanto hay 11 instrucciones definidas, con lo cual quedan 245 códigos a disposición del usuario para que incluya las instrucciones que estime oportunas.

Para el procesado de nuevas instrucciones basta con escribir nuevas rutinas que pueden incluirse fácilmente como funciones de librería, lo que dota al sistema de gran flexibilidad ya que permite al usuario satisfacer cualquier necesidad que pueda aparecer en su aplicación.

NOTA: En la definición de varias de las instrucciones se aludirá a posiciones de memoria local de los nodos. Dado que el Protocolo se ha diseñado para tarjetas de evaluación del micro controlador 68HC11 de Motorola, se considerará que las posiciones de memoria tendrán una longitud de 1 byte (bus de datos de 8 bits), y las direcciones de dichas posiciones una longitud de 2 bytes (bus de direcciones de 16 bits).

Veamos ahora una a una las instrucciones definidas y el modo de operación que requieren.

Leer Dato

Esta instrucción permite que un nodo solicite a otro el contenido de una determinada posición de memoria local de este último. El formato del paquete de aplicación correspondiente a esta instrucción aparece en la figura 4.5.4.1.

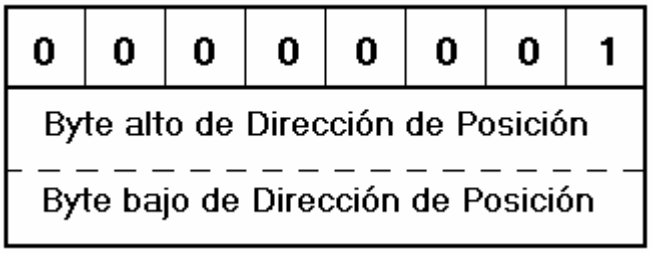


Fig. 4.5.4.1

Su campo de datos consta de un solo subcampo, de longitud 2 bytes, que contiene la dirección de la posición de memoria local del nodo destino cuyo contenido se desea leer.

Cuando un nodo necesita conocer el contenido de una determinada posición de otro nodo, el Nivel de Aplicación del primero genera un paquete con una instrucción 'Leer Dato', en la que incluye la dirección de la posición requerida, para el Nivel de Aplicación del segundo de los nodos.

Cuando el nodo destino recibe este paquete, al leer el código de instrucción sabe que este consta de un subcampo de datos de longitud 2 bytes. Además genera un paquete con una instrucción 'Dato Leído' para el nodo emisor, en la que ha de incluir el dato requerido así como la dirección del mismo (Ver instrucción 'Dato Leído').

Escribir Dato

Esta instrucción permite que un nodo modifique el contenido de una determinada posición de memoria local de otro nodo. El formato del paquete de aplicación correspondiente a esta instrucción aparece en la figura 4.5.4.2.

Fig. 4.5.4.2

Su campo de datos consta de dos subcampos. El primero de ellos tiene longitud 2 bytes y contiene la dirección de la posición de memoria local del nodo destino cuyo contenido se desea escribir. El segundo tiene longitud 1 byte y almacena el dato que se quiere escribir en la posición representada por el campo anterior.

Cuando un nodo necesita escribir cierto dato en una determinada posición de otro nodo, el Nivel de Aplicación del primero genera un paquete con una instrucción 'Escribir Dato', en la que incluye la dirección de la posición y el dato que desea almacenar en esta, para el Nivel de Aplicación del segundo de los nodos.

Cuando el nodo destino recibe este paquete, al leer el código de instrucción sabe que este consta de dos subcampos de datos de longitud 3 bytes en total. Además almacena el dato recibido en la posición de memoria cuya dirección se toma del primero de los subcampos.

Evidentemente para que esto sea posible, la dirección de memoria cuyo contenido se desea modificar debe apuntar a una posición del mapa de memoria del

nodo destino de memoria RAM, ya que de otro modo sería imposible escribir en esta. En nuestro caso, el rango de direcciones de RAM va de $0000 a $7FFF.

Existen mecanismos de protección automática que evitan esta situación , ya que cuando un usuario pretende escribir fuera de dicho rango, el sistema se lo impide y le avisa del error en el que está incurriendo. (Ver apartado 4.6.1, Operaciones de Control y Supervisión – Lectura y Escritura).

Ejecutar Programa en RAM

Esta instrucción permite al operador del sistema seleccionar la Aplicación de Control almacenada en memoria RAM, que previamente ha sido descargada desde el PC, para ser ejecutada como Aplicación Local de Control en el nodo al que va destinado el paquete. Realmente cualquier nodo puede emitir paquetes con esta instrucción si ello fuera preciso.

El formato del paquete de aplicación correspondiente a esta instrucción aparece en la figura 4.5.4.3 y como puede observarse carece de campo de datos.

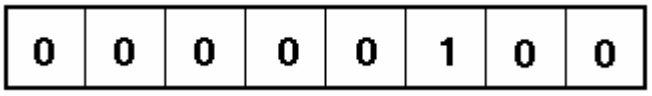


Fig. 4.5.4.3

Cuando el nodo destino recibe este paquete, al leer el código de instrucción sabe que este carece de campo de datos. Además actualiza el valor del Flag EJECUTAR PROGRAMA EN RAM = VERDADERO, de modo que en el próximo ciclo de programa se ejecutará el código almacenado en RAM salvo si la ejecución está suspendida. El Flag conservará su valor hasta que el nodo reciba un paquete de instrucción ‘Ejecutar Programa en EPROM’. El valor por defecto del Flag al configurar el sistema es Falso, para evitar fenómenos indeseados.

Copiar Programa en RAM

Esta instrucción permite al operador del sistema cargar instrucciones de una aplicación cualquiera como instrucciones de Aplicación de Control almacenada en memoria RAM del nodo al que va destinado el paquete. El formato del paquete asociado a esta instrucción aparece en la figura 4.5.4.4.

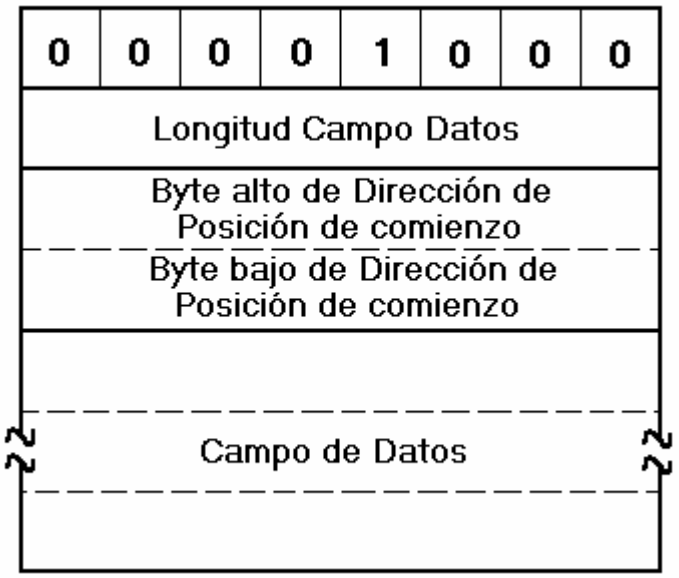


Fig. 4.5.4.4

El paquete consta de 3 subcampos de longitud total variable para aumentar la eficiencia de este tipo de instrucciones ya que lo habitual será transferir gran cantidad de datos que han de almacenarse consecutivamente.

El primero de los subcampos, de longitud 1 byte, contiene la longitud del tercero de ellos y toma valores entre 1 y 23. Permite al Nivel de Aplicación mantener el sincronismo de paquete.

El segundo campo, de nombre Dirección de la Posición de Comienzo, tiene longitud 2 bytes y contiene la dirección de la posición de memoria local RAM del nodo destino donde el primero de los bytes del subcampo campo de datos debe ser almacenado.

El tercer y último subcampo corresponde al campo de datos y contiene un número variable (entre 1 y 23) de datos que han de ser almacenados consecutivamente en memoria local RAM del nodo destino a partir de la posición indicada por el contenido del campo Dirección de la Posición de Comienzo.

Estos paquetes son generados por el PC cada vez que el operador quiere transferir un programa a algún nodo, como se verá en el apartado 4.6.3 Operaciones de Control y Supervisión – Copia de Programa.

Cuando un nodo recibe un paquete de este tipo lee la dirección de comienzo y a partir de esta almacena los datos uno a uno, consecutivamente a partir del primero. Para saber donde terminan los datos cuenta con la longitud del campo de datos.

Normalmente la transferencia completa de un programa requiere de la transmisión de un elevado número de instrucciones de este tipo que constituyen lo que podríamos llamar una secuencia de copia de programa. Por este motivo estas instrucciones se encapsulan en tramas sin prioridad. Esta solución se ha adoptado para evitar que estas tramas interrumpan la comunicación de información crítica a través de la red durante todo el tiempo que dure la transferencia del programa. Dicha información crítica ha de ser transmitida en tramas prioritarias, que son transmitidas en cada sección del anillo antes que las no prioritarias en espera de ser transmitidas.

Fin Copiar Programa en RAM

Esta instrucción permite al nodo que está recibiendo una secuencia de instrucciones de copia de programa conocer cuando puede dar esta por concluida. Ello le habilita para informar al PC de que la transferencia del programa ha concluido con éxito mediante la transmisión de un paquete de instrucción de 'Programa Copiado en RAM'. El formato del paquete asociado a la instrucción 'Fin Copiar Programa en RAM' aparece en la figura 4.5.4.5.

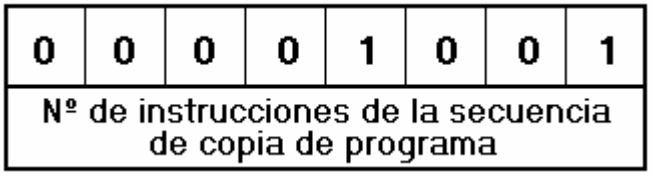


Fig. 4.5.4.5

El paquete consta en este caso de 1 solo subcampo de longitud 1 byte que contiene el número de instrucciones de 'Copia de Programa en RAM' asociadas a la secuencia de copia del programa que ha sido previamente transferido.

El PC genera el paquete asociado a esta instrucción cuando ha terminado de transmitir todas las instrucciones de copia requeridas para la transferencia de un programa. A medida que ha ido haciéndolo, ha incrementado un contador dispuesto a tal efecto de modo que al finalizar conoce perfectamente el número de instrucciones que componen la secuencia. Envía este dato al nodo destino.

Cuando un nodo recibe esta instrucción comprueba que ha recibido un número de instrucciones de copia igual al indicado por el campo de datos. Si lo ha hecho, envía al PC una instrucción de 'Programa Copiado en RAM', y si no lo ha hecho, permanece a la espera de recepción de nuevas instrucciones de copia hasta alcanzar el valor dado, momento en el que envía la instrucción anteriormente citada.

Programa Copiado en RAM

Esta instrucción permite al PC conocer cuando una transferencia de programa ha concluido con éxito, de modo que dicho programa está ya correctamente almacenado en memoria local RAM del nodo origen de la instrucción dispuesto para ser seleccionado para ser ejecutado. El formato del paquete asociado a esta instrucción aparece en la figura 4.5.4.6.

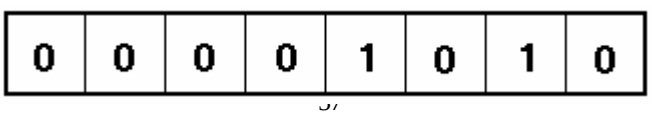


Fig.4.5.4.6

El paquete asociado a esta instrucción carece de campo de datos, ya que toda la información va implícita en el propio código de la instrucción.

Ya hemos visto como operan tanto los nodos como el PC con estas instrucciones en la descripción de la instrucción anterior por lo que no es necesario repetirla.

Primera Copia en RAM

Esta instrucción es exactamente igual a la instrucción 'Copiar Programa en RAM', salvo por el código de instrucción. Este lleva implícito el hecho de que la presente instrucción de copia de programa es la primera de una secuencia de copia de programa. Su formato aparece en la figura 4.5.4.7.

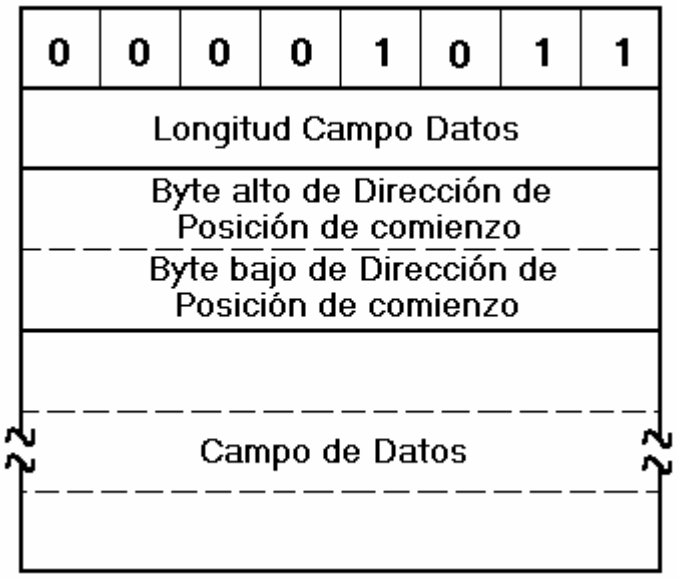


Fig. 4.5.4.7

Cuando el PC comienza una transferencia de programa, la primera instrucción de copia es de este tipo. Cuando el nodo destino la recibe, sabe que va a empezar a

recibir una secuencia completa, por lo que activa el contador que le permitirá saber cuando finaliza la secuencia.

Otra diferencia es que la trama que encapsula esta instrucción es prioritaria para permitir que la activación del contador en el nodo destino se lleve a cabo con seguridad y prontitud, ya que las tramas prioritarias llegan antes a su destino y tienen menor probabilidad de perderse por congestión en la red.

A esta solución se llegó por razones empíricas, ya que se comprobó que problemas que de otro modo aparecían en las copias de programa, desaparecían al adoptar esta solución.

Ejecutar Programa en EPROM

Esta instrucción es la inversa a la instrucción 'Ejecutar Programa en RAM'. Permite al operador del sistema seleccionar la Aplicación de Control almacenada en memoria EPROM, que fue programada junto con el protocolo, para ser ejecutada como Aplicación Local de Control en el nodo al que va destinado el paquete. Realmente cualquier nodo puede emitir paquetes con esta instrucción si ello fuera preciso.

El formato del paquete de aplicación correspondiente a esta instrucción aparece en la figura 4.5.4.8 y como puede observarse carece de campo de datos.

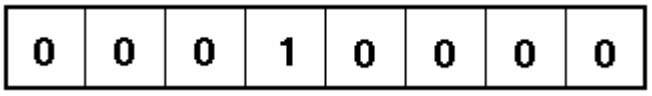


Fig. 4.5.4.8

Cuando el nodo destino recibe este paquete, al leer el código de instrucción sabe que este carece de campo de datos. Además actualiza el valor del Flag EJECUTAR PROGRAMA EN RAM = FALSO, de modo que en el próximo ciclo de programa se ejecutará el código almacenado en EPROM salvo si la ejecución está

suspendida. El Flag conservará su valor hasta que el nodo reciba un paquete de instrucción 'Ejecutar Programa en RAM'. El valor por defecto del Flag al configurar el sistema es Falso, para evitar fenómenos indeseados como vimos en su momento.

Suspender Ejecución

Esta instrucción permite al operador del sistema (o a cualquier nodo de forma genérica) suspender la Ejecución de Aplicación Local de Control en el nodo al que da dirige la instrucción. Es en el siguiente ciclo de programa cuando ya no se ejecutará el código que estuviese seleccionado al recibir la instrucción (RAM o EPROM) sino que este es saltado de modo que tan sólo se ejecuta el código correspondiente al Protocolo de Comunicación. El formato del paquete asociado a esta instrucción aparece en la figura 4.5.4.9.

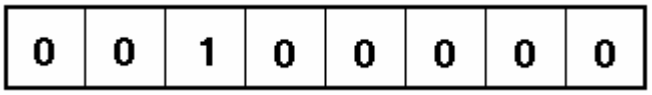


Fig 4.5.4.9

Como se puede apreciar, el paquete carece de campo de datos, ya que toda su información la lleva implícita en el propio código de instrucción.

Cuando el operador o cualquier otro nodo desean suspender la ejecución de la aplicación local que está siendo ejecutada en otro nodo, no tienen más que enviarles una instrucción de este tipo.

Cuando un nodo recibe un paquete como este, ejecuta la instrucción actualizando el valor del Flag EJECUCIÓN SUSPENDIDA = VERDADERO, que conservará su valor hasta que el nodo no reciba una instrucción 'Activar Ejecución'. De este modo el micro, al ejecutar el Ciclo de Nivel de Aplicación genera el salto correspondiente que evita ejecutar la aplicación local.

Esta instrucción resulta muy útil para hacer una transferencia controlada de programa local, para pasar de RAM a EPROM y viceversa o para un cambio de RAM a RAM.

Activar Ejecución

Esta instrucción permite al operador del sistema (o a cualquier nodo de forma genérica) activar la Ejecución de Aplicación Local de Control en el nodo al que da dirige la instrucción, es decir, es la instrucción inversa a la anterior. Es en el siguiente ciclo de programa cuando se ejecutará el código que estuviese seleccionado al recibir la instrucción (RAM o EPROM). El formato del paquete asociado a esta instrucción aparece en la figura 4.5.4.10.

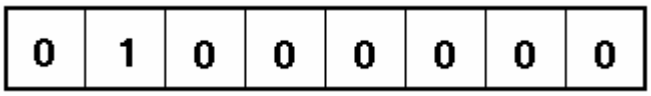


Fig. 4.5.4.10

Como era de suponer, esta instrucción carece también de campo de datos, ya que igual que en el caso de la anterior, toda la información va implícita en el código de instrucción.

Cuando esta instrucción es ejecutada por un nodo, el valor del Flag correspondiente es actualizado: EJECUCIÓN SUSPENDIDA = FALSO, lo que se traduce en que en el próximo ciclo de programa se ejecutará la aplicación local seleccionada.

El valor por defecto al configurar el sistema es EJECUCIÓN SUSPENDIDA = FALSO para cada nodo, lo que implica que el operador debe enviar una instrucción de este tipo a los nodos si quiere que estos ejecuten su aplicación local de control.

4.6 OPERACIONES DE CONTROL Y SUPERVISIÓN

4.6.1 Generalidades

El sistema desarrollado dota al usuario u operador de la capacidad para ejercer el mando y supervisión que estime oportunos sobre la planta a través del control digital directo (DDC) ejercido por los nodos del sistema.

Dicha capacidad de mando y supervisión ha de permitirle de forma sencilla e intuitiva dar consignas y visualizar datos. Para lograr esa sencillez es preciso definir un juego de operaciones con un grado mínimo de complejidad y que no presupongan conocimientos sobre la arquitectura y funcionamiento del sistema en el usuario.

Estas operaciones y sus resultados serán accesibles mediante una interfaz visual que también se encargará de la representación de alarmas, gráficos, históricos y demás elementos adicionales que desee el operador. Es decir, el Software a instalar en la Estación de Operación (PC) constituye en sí mismo un pequeño SCADA (Supervisory Control and Data Acquisition) y será ampliamente discutido en el apartado 5.2 'Consideraciones sobre el Código – Código de Programa para PC'.

En este apartado nos limitaremos a definir las operaciones diseñadas en la actualidad para el mando y supervisión en las que el operador participa activamente, quedando pues excluidas de esta discusión aquellas relacionadas con la representación de datos, alarmas, configuración del sistema etc. , que se describirán detalladamente en el apartado al que se alude en el párrafo anterior.

Cabe resaltar el empeño que se ha puesto durante la etapa de diseño en dotar al sistema de una gran flexibilidad que le permita incluir fácilmente nuevas operaciones para así satisfacer los requerimientos de cualquier proceso(s) que quiera(n) controlarse con el sistema. No obstante las operaciones actualmente definidas tienen una gran potencia y permiten al operador tener un conocimiento pleno y capacidad absoluta de control sobre la planta.

4.6.2 Lectura/Escritura

En este apartado se presenta la operación que aglutina instrucciones ‘Leer dato’ y ‘Escribir dato’. Como habíamos visto, estas instrucciones permitían a cualquier nodo en general, y al PC en particular, obtener y modificar el contenido de la posición deseada de memoria local de otro nodo.

Esta operación da al usuario la posibilidad de emitir estas instrucciones a través de una sencilla secuencia de órdenes que puede dar sin más que pulsar ciertos botones. La configuración de los paquetes de aplicación y a más bajo nivel, de las tramas de nivel de red, es totalmente transparente al operador. Este se limitará a introducir los datos necesarios (dirección, y dato en caso de ser necesario), y seleccionar el tipo de operación deseada (lectura o escritura) y su prioridad (con ó sin prioridad).

Será posible realizar tantas operaciones simultáneas (una sola trama de nivel de red) como permita el campo de datos de la trama., teniendo en cuenta que cada paquete de instrucción ‘Leer datos’ requiere 3 bytes, cada paquete de instrucción ‘Escribir dato’ 4 bytes , y el espacio total disponible es de 27 bytes. Existe sin embargo otra restricción sobre las órdenes de lectura, y es que las respuestas a estas (instrucciones ‘Dato leído’ de 4 bytes) deben encapsularse en la misma trama, con lo cual, sólo 6 tramas de lectura simultáneas se permiten como máximo. El mayor aprovechamiento de la trama permite el envío simultáneo, por lo tanto, de 6 instrucciones de lectura (18 bytes) y 2 de escritura (8 bytes).

Todas las órdenes simultáneas se encapsulan en la misma trama de nivel de red cuya prioridad ha sido previamente determinada por el operador que, una vez configurada la trama no tiene más que pulsar un botón para que la trama llegue a su destino.

A partir de ese momento puede considerarse que las órdenes de escritura se han hecho efectivas. Si se realizó también alguna lectura se recibirá , con un mínimo retardo que dependerá del tráfico de la red y prioridad seleccionada, la respuesta del nodo destino con la dirección y dato de las posiciones leidas.

4.6.3 Modificación de ejecución

En este apartado se presenta la operación que aglutina instrucciones 'Activar Ejecución', 'Suspender Ejecución', 'Ejecutar Programa en RAM' y 'Ejecutar Programa en EPROM'. Como habíamos visto, estas instrucciones permitían a cualquier nodo en general, y al PC en particular, modificar los parámetros de ejecución de la aplicación local de control de cualquier otro nodo.

Esta operación da al operador la posibilidad de emitir estas instrucciones a través de una sencilla secuencia de órdenes que puede dar sin más que pulsar ciertos botones.

La configuración de los paquetes de aplicación y a más bajo nivel, de las tramas de nivel de red, es totalmente transparente al operador. Este se limitará a seleccionar el/los tipo/s de operación deseada/s (activar, suspender, RAM, EPROM).

Será posible realizar todas las operaciones simultáneamente (una sola trama de nivel de red), si bien, resulta absurdo enviar órdenes contradictorias, por lo que en caso de que ello suceda, el sistema ejecutará consecutivamente todas ellas, de modo que tan sólo se harán finalmente efectivas las seleccionadas en último lugar de entre las parejas Suspender/Activar, EPROM/RAM.

Todas las órdenes simultáneas se encapsulan en la misma trama de nivel de red, que tiene prioridad y, una vez configurada la trama no tiene más que pulsar un botón para que la trama llegue a su destino.

A partir de ese momento puede considerarse que las órdenes se han hecho efectivas, de modo que en el siguiente ciclo de programa la ejecución de la aplicación local de control en el nodo destino se llevará a cabo según los nuevos parámetros. El nodo de destino no envía mensaje de confirmación de estas operaciones, si bien, el usuario puede consultar los parámetros vigentes en cada momento leyendo el registro correspondiente a los Flags de EJECUCIÓN SUSPENDIDA y EJECUTAR PROGRAMA EN RAM.

4.6.4 Copia de Programa

En este apartado se presenta la operación que aglutina instrucciones ‘Copia de programa en RAM’, ‘Primera copia de programa en RAM’ y ‘Fin de copia en RAM’. Como habíamos visto, estas instrucciones permitían a cualquier nodo en general, y al PC en particular, escribir datos de programa (código) en memoria local RAM de otro nodo.

Esta operación permite transferir un programa completo a memoria RAM de forma transparente al usuario, que tan sólo debe seleccionar el fichero ‘.s19’ correspondiente y enviarlo al nodo deseado.

5. CONSIDERACIONES SOBRE EL CÓDIGO

5.1 CÓDIGO DE PROGRAMA PARA 68HC11

5.1.1 Generalidades

El objeto de este apartado es describir detalladamente el software instalado en cada nodo que posibilita el funcionamiento del sistema de control.

El código fuente del protocolo, así como de las aplicaciones locales de control de los nodos, han sido escritas en Lenguaje Ensamblador de Motorola para M68HC11. El código fuente constituye una unidad, es decir, todo el código fuente, incluyendo aplicación local y protocolo, está contenido en un solo archivo con extensión '.s'. Dicho código es ensamblado por el programa AS11 de Motorola el cual genera el código ejecutable en formato S19 de Motorola (Anexo I) contenido en un archivo del mismo nombre que el primero , pero con extensión '.s19' que es transferido a la memoria EPROM local de la tarjeta de evaluación del micro correspondiente.

Los micros funcionan en modo normal expandido (MODA= 1 ; MODB= 1), con lo cual cada vez que reciben un reset, ejecutan el código almacenado en memoria externa EPROM. A esta acceden a través de los buses de datos y direcciones mediante los puertos B y C que quedan por ello sin utilidad para las aplicaciones locales de control.

El código ha sido escrito de modo que puede absorber de forma sencilla cualquier cambio que se quiera introducir sin más que añadir o suprimir subrutinas, ya que se ha procurado que cada función fuese totalmente independiente del resto.

Se ha tratado también independizar en la medida de lo posible lo relativo al protocolo y lo relativo a la aplicación, de modo que las variables de uno y otro ocupan rangos de memoria diferentes. Esto no es necesario con el código ejecutable, ya que la aplicación de control se define como una subrutina o conjunto de subrutinas inmerso en el código único, aunque fácilmente diferenciable de este.

El mapa de memoria de los nodos del sistema aparece en la figura 5.1.1.1 .

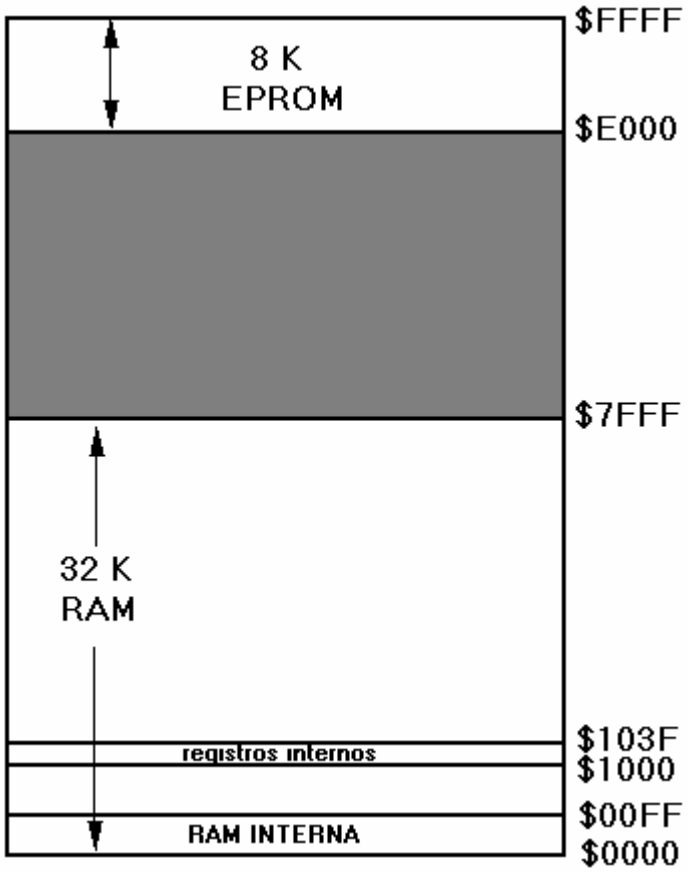


Fig. 5.1.1.1

En memoria RAM tendremos, como veremos, variables de protocolo y aplicación (tanto de EPROM como de RAM), y en EPROM tendremos el código máquina de programa y protocolo.

El reparto de memoria está totalmente estandarizado, para permitir al programador ver rápidamente de qué consta cualquier programa (código, variables etc.) o realizar la programación de forma fácilmente entendible por otros programadores.

Para entender como se lleva a cabo este reparto, basta con ver el mapa de memoria RAM y memoria EPROM con detenimiento. Estos están representados en la figura 5.1.1.2 y 5.1.1.3 respectivamente.

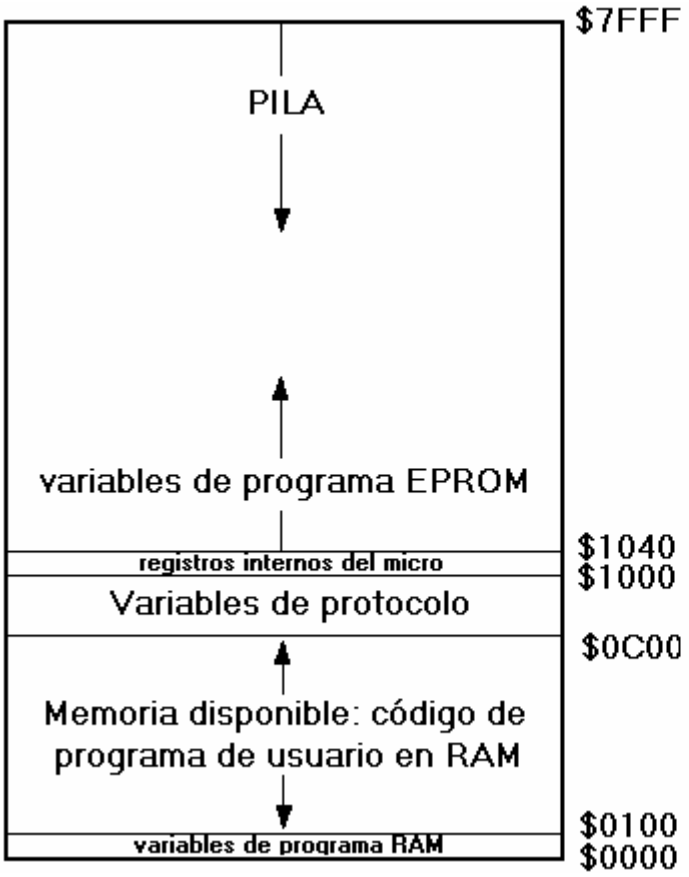


Fig. 5.1.1.2

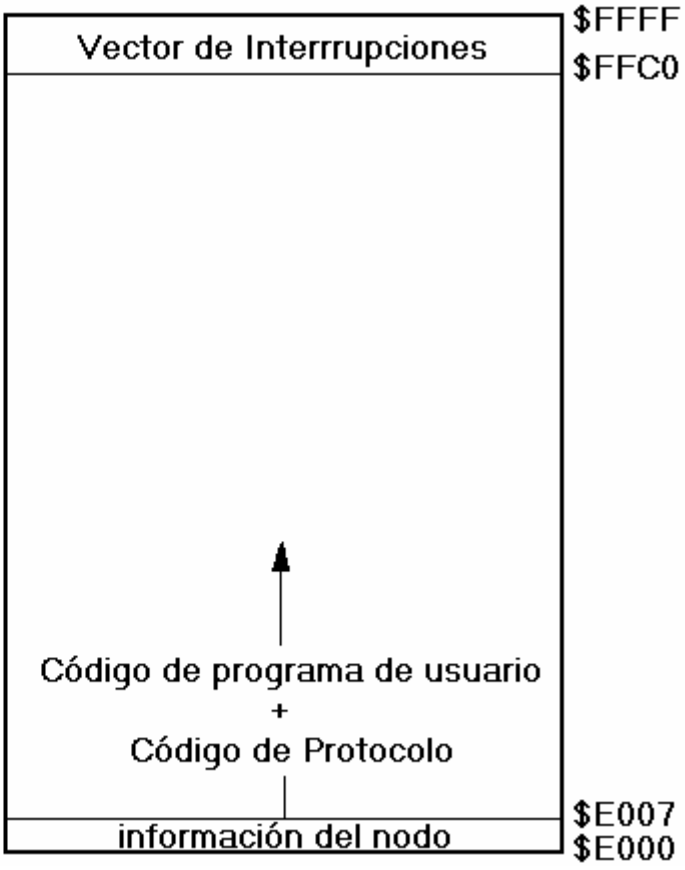


Fig. 5.1.1.3

En los siguientes apartados veremos con más detalle cuáles son los campos que aparecen en los anteriores mapas, de qué se componen y qué funciones cumplen.

Las tarjetas están preparadas para disponer de una memoria EPROM de 32K, en cuyo caso la dirección de comienzo de esta estaría en la posición $8000, sin que ello suponga ningún disturbio para nuestro diseño.

5.1.2 Protocolo: Variables

Antes de entrar a discutir el código del protocolo vemos cuales son los recursos del sistema que este necesita en términos de memoria. Habíamos visto el mapa de memoria de los micros, en los que se distinguían algunos campos asociados al protocolo (código de programa y variables). Los registros y bloques de memoria RAM para almacenamiento de variables, tramas, flags, etc. Van desde la posición $0C00 hasta $1000 si bien, sólo se usan las posiciones que llegan hasta $0F81, quedando las 126 posiciones restantes libres para futuros usos.

Habíamos visto que en el Nivel de Red eran necesarias dos colas de transmisión (con y sin prioridad), dos colas de procesado (con y sin prioridad) y una lista de espera. Adoptando una solución de compromiso entre requerimientos de memoria, tiempo etc. y eficiencia del sistema, se definen colas de 4 posiciones y lista de espera de 8 posiciones, de tal modo que el bloque de memoria correspondiente a las mismas presenta el siguiente aspecto que recoge la figura 5.1.2.1.

En las direcciones binarias de las direcciones base de cada una de las posiciones de las colas/lista aparecen ciertos bits en color rojo. Dichos bits corresponden al índice que referencia la posición en la cola o lista correspondiente, con lo que se usarán para la gestión de las colas circulares y para recorrer la lista de espera como se verá en las subrutinas correspondientes.

En la Lista de Espera: El índice i se mueve entre 0 y 7 para definir 8 posiciones y aparece en color rojo en la figura 5.1.2.1. Desplazando adecuadamente dicho índice (5 posiciones a la izda., realimentando con 0`s por la derecha), y sumando el offset obtenido a la dirección base de la Lista de Espera se puede calcular la dirección base de la posición correspondiente.

En colas circulares: el índice se mueve entre 0 y 3 para definir 4 posiciones. De forma análoga al caso anterior pueden calcularse las direcciones base de las posiciones de las colas.

En la figura 5.1.2.1 aparecen también dos bloques de memoria de 32 bytes cada uno. Dichos bloques, cuyas direcciones base son BASERESP y BASEALARM, tienen capacidad para almacenar una trama completa, y se usan para almacenar, mientras están siendo construidas, las tramas de respuesta y alarma respectivamente que posteriormente serán transferidas a la cola de transmisión correspondiente (según prioridad) para ser transmitidas físicamente.

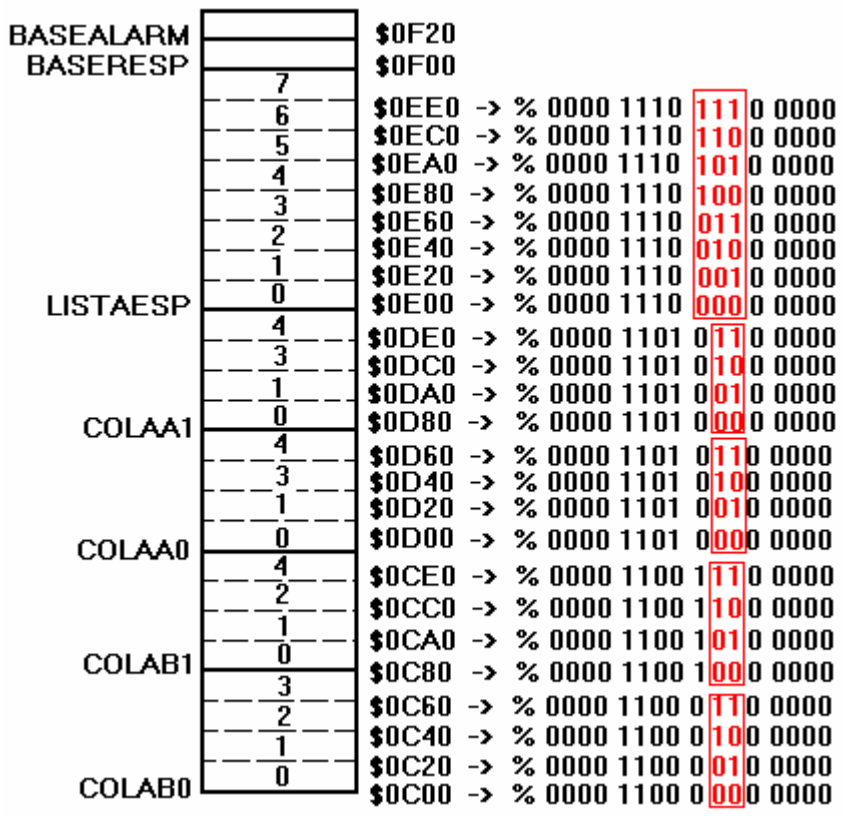


Fig. 5.1.2.1

Además de las colas/lista el protocolo necesita ciertos registros de control, flags, variables etc. Estos registros pueden corresponder al Nivel de Red (las colas/lista corresponden a este nivel), o al Nivel de Aplicación. Por economía del código existen registros mixtos, es decir, contienen campos de Nivel de Red y campos de Nivel de Aplicación. Las variables de Protocolo correspondientes al Nivel de Red y Nivel de Aplicación se presentan a continuación, siguiendo el orden que tienen en la memoria local RAM del nodo:

NIVEL DE RED

- **PRCTEA – Protocolo: Reg. de Control de Tramas en Espera de Asentimiento**

Tabla de 8 posiciones que contienen el estado de la posición correspondiente de la Lista de Espera (ocupada = 255 / no ocupada = 0). PRCTEA es la dirección base de la tabla, de modo que PRCTEA + i es la dirección de la posición que contiene el estado de la posición i en Lista de Espera.

- **PRCHB1 / B0 – Protocolo: Reg. de Control de Huecos en ColaB1 / ColaB0**

Tablas de 4 posiciones que determinan si la posición correspondiente de la cola es un hueco (hueco = 255 / no hueco = 0). Un hueco se produce cuando una trama que está siendo recibida en colas B por ser de retransmisión pura sufre algún fallo y tiene que ser desechada. Si en la siguiente posición en la cola se ha introducido alguna otra trama, nuestra trama deja un hueco en la cola al ser desechada. Si por el contrario en la posición siguiente no se ha introducido una nueva trama, nuestra trama no deja ningún hueco, ya que la siguiente trama a ser introducida ocupara la posición liberada. Estos huecos requieren un tratamiento especial, de ahí que sea necesario mantener un estricto control de los mismos.

PRCHBX es la dirección base de la tabla, de modo que PRCHBX + i es la dirección de la posición que determina si la posición i en la cola X es o no un hueco.

- **PRIA1 – Primero en ColaA1 – Análogamente PRIA0, PRIB1, PRIB0**

Indice de la posición de ColaA1 que contiene la primera trama en cola circular. Este índice contendrá un número entre 0 y 3, y se corresponde con los bits de color rojo en la figura 5.1.2.1 .

- **ULTA1 – Ultimo en ColaA1 – Análogamente ULTA0, ULTB0, ULTB1**

Indice de la posición de ColaA1 que contiene la última trama en cola circular. Este índice contendrá un número entre 0 y 3, y se corresponde con los bits de color rojo en la figura 5.1.2.1 .

- **STCOLAA1 – Estado de ColaA1 – An. STCOLAA0,STCOLAB1, STCOLAB0**

Registro que contiene el estado de ColaA1 (vacía = 0 / ni vacía ni llena = 170 / llena = 255). Es muy útil para el tratamiento de las colas circulares.

- **PRCD – Protocolo: Reg. de Control de Desbordamientos**

Contiene el reloj global del nodo. Cada vez que el temporizador del micro se desborda, el contenido de PRCD es incrementado en 1. Es usado para temporización de asentimientos.

- **CNTTX – Contador de Transmisión**

Registro que contiene el contador de bytes transmitidos hasta el momento de la trama. Sirve para sincronismo de trama en transmisión.

- **CNTRX – Contador de Recepción**

Registro que contiene el contador de bytes recibidos de la trama hasta el momento. Sirve para sincronismo de trama en recepción.

- **CNTPROC – Contador de Procesado**

Registro que contiene el contador de bytes procesados hasta el momento de la trama. Sirve para sincronismo de trama en procesado.

- **CNTRESP – Contador de Respuesta**

Registro que contiene el contador de bytes de respuesta de la misma trama generados hasta el momento. Sirve para sincronismo de trama en construcción de trama de respuesta.

- **CNTCOP – Contador de Copia**

Registro que contiene el contador de instrucciones de copia de programa ejecutadas hasta el momento. Sirve para sincronismo de transferencia de programa.

- **CONTCRC – Contador de CRC**

Contador de bytes para el cálculo del CRC que se van introduciendo uno a uno en la división binaria correspondiente.

- **PRCC – Protcolo: Registro de Control de la Comunicación**

Este registro contiene flags que controlan la comunicación a través del puerto serie. Estos flags aparecen en la figura 5.1.2.2 .

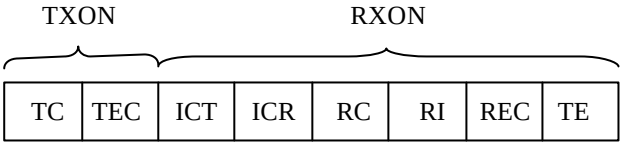


Fig 5.1.2.2

TC: Transmisión completa. TC =1 : txón completa; TC = 0 txón no completa.
Cuando una transmisión se completa con éxito, este flag se activa para permitir la actualización de las colas de Transmisión.

TEC: Transmisión en curso. TEC = 1 : txón en curso; TEC = 0 no hay txón en curso.
Cuando una transmisión comienza se activa este flag, que permanece así hasta que finaliza y las colas de txón son actualizadas. Inhibe el comienzo de una nueva transmisión.

ICT: Insertando en cola de transmisión. ICT = 1 insert, en cola Txón; ICT = 0 otro caso.
Cuando se recibe una trama para retransmisión pura se inserta directamente en colas de transmisión según se va recibiendo bytes a byte. En ese caso se activa este flag.

ICR: Insertando en cola de procesado. ICR = 1 insert. En cola Proc; ICT = 0 otro caso.
Cuando se recibe una trama para procesado se inserta obviamente en colas de procesado según se va recibiendo bytes a byte. En ese caso se activa este flag.

RC: Recepción completa. RC = 1: Rxón completa; RC = 0: Rxón no completa
Cuando una recepción se completa con éxito, este flag se activa para permitir la actualización de las colas de Recepción.

RI: Recepción incompleta. RI = 1: Rxón incompleta; RI = 0: Rxón no incompleta.
Este flag se activa para permitir la actualización de las colas de Recepción siempre que se produce un fallo en la recepción de una trama que obliga a desecharla.

REC: Recepción en curso. REC = 1: Rxón en curso; REC = 0 : No hay Rxón en curso
Este flag se activa cuando se inicia una recepción de trama y monitoriza el estado del receptor.

TE: Trama errónea. TE = 1: Trama errónea; TE = 0: Trama no errónea
Este flag se activa cuando se detecta un error en la recepción de un byte. Evita el almacenamiento de los siguiente byte de la trama, de modo que el receptor se limita a esperar a que dicha recepción finalice sin leer los datos que reciba.

- **PRCN: Protocolo Registro de Control de Nodo**

Registro mixto que contiene flags de Nivel de Red y Aplicación. Aquí presentamos los que corresponden al más bajo de los dos niveles. El registro aparece en la figura 5.1.2.3.

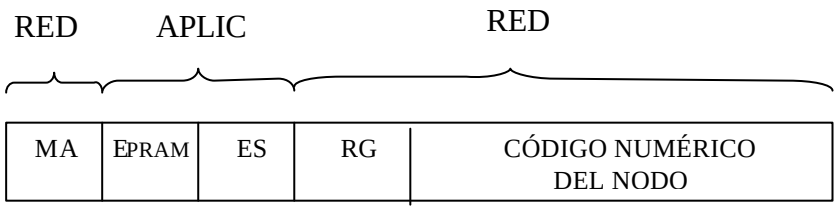


Fig 5.1.2.3

MA: Micro activo. MA = 0 : Micro no activo; MAC = 1: Micro activo.
Este flag se activa cuando el nodo se registra en el sistema, es decir, durante el proceso de configuración de la red. Se usa para la detección de nodos no registrados en el sistema cuya aparición obliga a reconfigurarlo. Es decir, cumple funciones de autodiagnóstico del sistema.

RG: Respuesta generada. RG = 1: Se ha generado una respuesta; RG = 0: otro caso
Cuando se genera una respuesta por parte del micro, se activa este flag, de modo que el Nivel de red sabe que tiene que insertar la trama generada en Cola de transmisión cuando quede configurada completamente.

CÓDIGO NUMÉRICO DEL NODO: Campo de 4 bits (0-15) que constituye la dirección de Red del nodo correspondiente. Permite definir un sistema de 15 nodos (1 – 15), ya que el código 0000 está reservado al Nodo 0, Pc ó Estación de Operación.

- **CNTCOPMAX**

Valor máximo de CNTCOP. Cuando alcanza este valor, la transferencia de programa se puede dar por concluida.

- **CNTTXMAX**

Valor máximo de CNTTX. Cuando alcanza este valor, la transmisión se puede dar por concluida.

- **CNTRXMAX**

Valor máximo de CNTRX. Cuando alcanza este valor, la recepción se puede dar por concluida.

- **CNTPROCMAX**

Valor máximo de CNTPROC. Cuando alcanza este valor, el procesado de trama se puede dar por concluida.

- **CRCOK**

Indica si el CRC de la trama procesada es correcto CRCOK = 0 ó por el contrario es incorrecto CRCOK = 255, lo que implica que ha habido algún error en la trama recibida.

- **BYTECRC**

Contiene el byte que está siendo introducido bit a bit en la división binaria para el cálculo del síndrome o CRC de la trama recibida o generada.

- **STINCOL**

Refleja el tipo de trama que está siendo introducido en cola de Transmisión, para determinar su ubicación. Toma los siguientes valores:
STINCOL = 0 : Insertando trama de alarma (ubicada en BASEALARM)

STINCOL = 255 : Insertando trama para retransmisión (ubicada en Lista de Espera)
Otro caso: Insertando trama de respuesta (ubicada en BASERESP)

- **PRIMBYTE**

Variable auxiliar que almacena el primer byte de trama recibida, a la espera de que el segundo sea recibido y pueda determinarse la cola en que almacenar la trama.

- **TIMERBASE**

Tabla de 8 posiciones que contienen el timer de vencimiento de la trama que ocupa la posición correspondiente de la Lista de Espera . TIMERBASE es la dirección base de la tabla, de modo que TIMERBASE + i es la dirección de la posición que contiene el timer de la trama que ocupa la posición i en Lista de Espera.

- **BASETX**

Variable de 2 bytes que contiene la dirección base del bloque de 32 bytes donde está contenida la trama que se está transmitiendo, es decir, contiene la dirección base de la posición correspondiente de las colas de transmisión de la extraer los bytes que hay que transmitir físicamente.

- **BASERX**

Variable de 2 bytes que contiene la dirección base del bloque de 32 bytes donde se está recibiendo una trama, es decir, contiene la dirección base de la posición correspondiente de las colas en la que almacenar los bytes de la trama que se está recibiendo

- **BASEPROC**

Variable de 2 bytes que contiene la dirección base del bloque de 32 bytes donde está contenida la trama que se está procesando, es decir, contiene la dirección base de la posición correspondiente de las colas de procesado de la extraer los bytes que hay que procesar.

- **BASERTXON**

Variable de 2 bytes que contiene la dirección base del bloque de 32 bytes donde está contenida la trama que hay que retransmitir, es decir, contiene la dirección base de la posición correspondiente de la lista de espera de donde extraer los bytes que hay que transferir a la cola de transmisión correspondiente.

- **BASECRC**

Variable de 2 bytes que contiene la dirección base de la trama cuyo CRC ó síndrome ha de calcularse. Alternativamente se usa para devolver el valor del CRC ó síndrome obtenidos.

- **TEXTOALARMA**

Variable de 2 bytes que contiene la dirección base de la cadena de caracteres que hay que introducir en el campo de datos de la trama de alarma que está siendo generada.

NIVEL DE APLICACIÓN

- **PRCN: Protocolo Registro de Control de Nodo**

Como habíamos visto, este es un registro mixto. Los flags que lo componen aparecen en la figura 5.1.2.3, de modo que los flags correspondientes a este nivel son los siguientes:

EPRAM: Ejecutar Programa en Ram.
EPRAM = 1: Ejec. código de programa local almacenado en RAM descargado desde Pc
EPRAM = 0: Ejec. código de programa local almacenado en EPROM.

ES: Ejecución suspendida
ES = 1: Ejecución suspendida. El nodo no ejecuta programa local de control.
ES = 0: Ejecución activada. El nodo ejecuta la aplicación local seleccionada, según el contenido de EPRAM.

CLASENODO

No es una variable y tampoco está mapeado en memoria RAM, sino que está en memoria EPROM. Identifica la naturaleza del nodo a nivel operativo, es decir, indica los pines que este tiene activos, su función y cuando es necesario su direccionalidad. Evidentemente es único para cada nodo, y debe ser determinado al programar la memoria EPROM del nodo.

Es una tabla de 7 bytes, de tal modo que los tres primeros hacen referencia a si el pin correspondiente está activo (1) o no (0). Los tres bytes siguientes hacen referencia a la funcionalidad de los pines activos (E/S digital de propósito general = 0 ó función alternativa = 1). El último byte indica, para aquellos pines activos de propósito general bidireccionales, su direccionalidad (Entrada = 0 ; Salida = 1). La figura 5.1.2.4 presenta las posibilidades de CLASENODO.

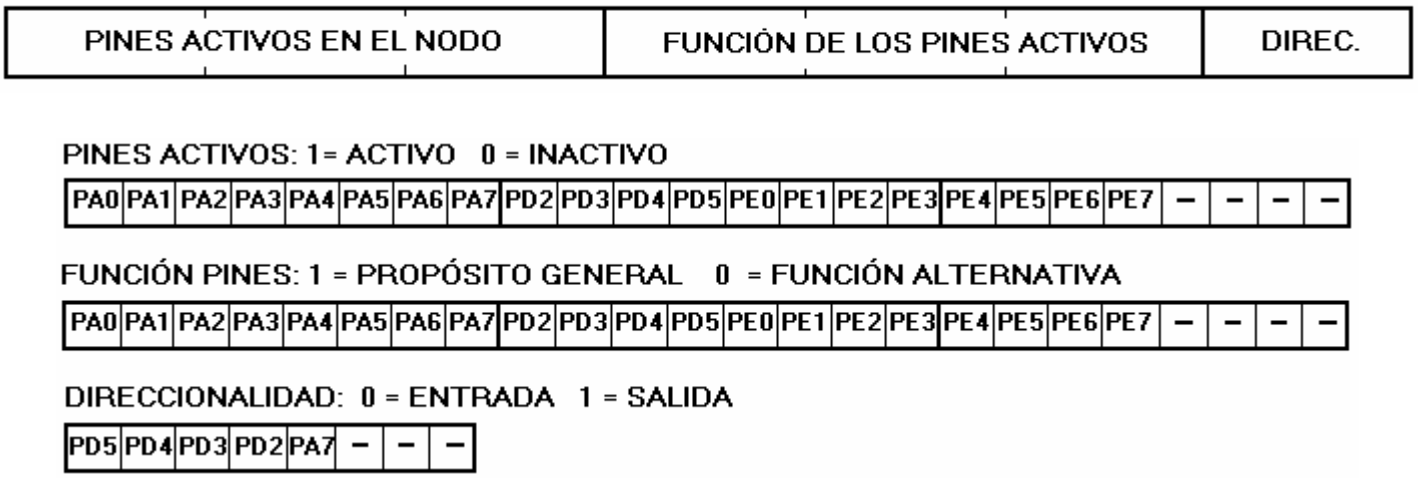


Fig. 5.1.2.4

En el apartado 3.1 Equipos, vimos cuáles eran las funciones alternativas y demás información relativa a las tarjetas de evaluación del micro 68HC11. Los registros de control se programarán en el bloque de inicialización correspondiente según la información contenida CLASENODO. No obstante la naturaleza puede ser dinámica, de modo que para activar pines en tiempo de ejecución, basta con escribir los datos que correspondan en los registros asociados a los pines que se quieran activar.

5.1.3 Protocolo: Principales subrutinas

El protocolo, como veíamos en el apartado correspondiente a su funcionamiento, se basa en dos tipos de rutinas, las periódicas y las disparadas por eventos.

En el caso del 68HC11 para implementar disparos por eventos contamos con las interrupciones, en cambio para generar rutinas periódicas lo más sencillo es introducirlas en el bucle infinito del programa principal, de tal modo que la parte asociada al protocolo se ejecuta una sola vez por ciclo de programa.

Como el funcionamiento del protocolo ha sido descrito ampliamente, en este apartado nos centraremos en los aspectos relativos a cómo se ha conseguido implementar el funcionamiento con las limitaciones del lenguaje ensamblador. No obstante, el código se encuentra ampliamente comentado en el Anexo IV correspondiente al código fuente en ensamblador, por lo que para una correcta comprensión del código lo más conveniente sería consultar dicho anexo.

PROGRAMA PRINCIPAL

Tras recibir un reset, el contador de programa del micro salta a la dirección INICIO. A partir de esta dirección se ejecutan una serie de instrucciones destinadas a la inicialización de variables de protocolo y registros del micro, tras las cuales se entra en el 'bucle forever' de programa. El diagrama de aparece en la figura 5.1.3.1.

Vemos que al comienzo del módulo principal se resetea el perro guardián para monitorizar el correcto funcionamiento del micro. Luego se comprueban los flags de Nivel de Aplicación (EPRAM y ES) para seleccionar el salto adecuado que dar y así ejecutar la aplicación de control seleccionada por el operador tal y como muestra el diagrama de flujo de la figura 5.1.3.1.

Tras ejecutar la aplicación seleccionada, se producen dos saltos a sendas subrutinas encargadas de l Ciclo de Nivel de Red. Tras este se salta al comienzo del bucle PRINCIPAL nuevamente.

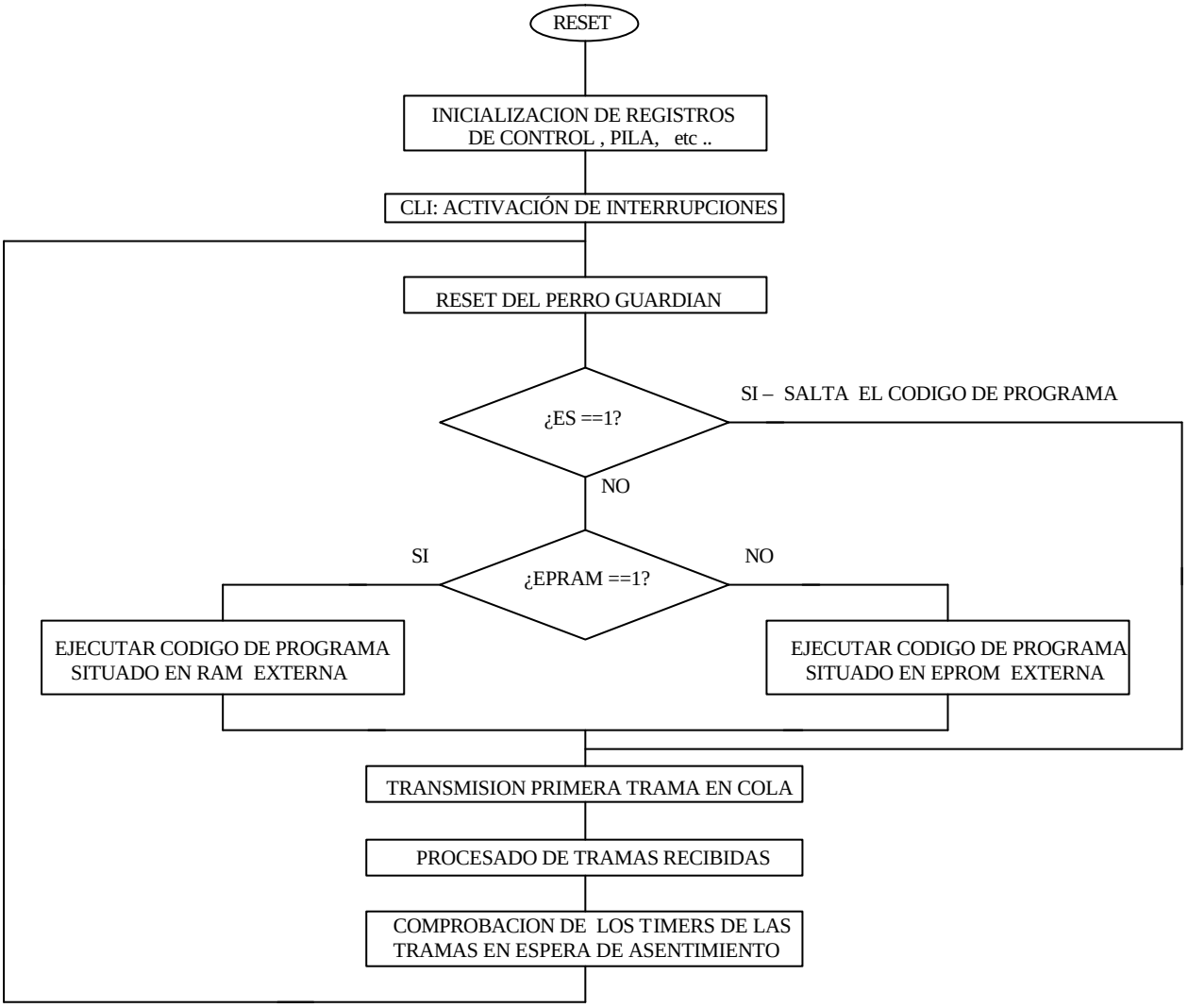


Fig. 5.1.3.1

Las subrutinas APLICACIÓN y RAMBS contienen el código de programa de la aplicación local de control almacenada en EPROM y RAM respectivamente.

TXON

Esta subrutina es la encargada de iniciar la transmisión de una nueva trama cuando corresponda, es decir, cuando no haya transmisión en curso y exista al menos una trama en espera de ser transmitida.

La subrutina Txon tan sólo se encarga de iniciar la transmisión mediante la inicialización de contadores de transmisión, dirección base de transmisión y transmisión física del primer bytes de trama. El diagrama de flujo de la subrutina Txon aparece en la figura 5.1.3.2.

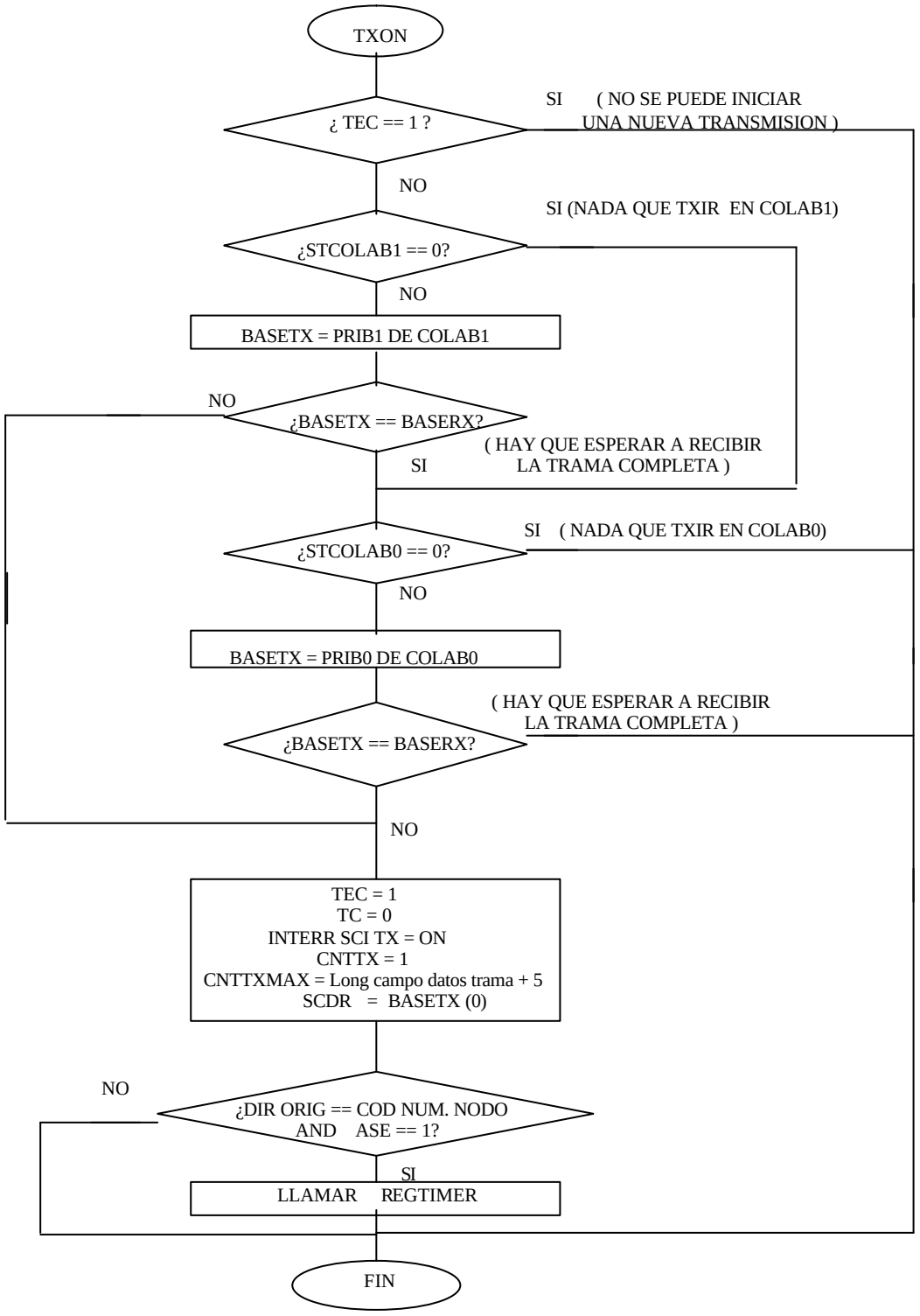


Fig. 5.1.3.2

El resto de los bytes se transmiten de forma automática mediante el disparo de interrupción SCI cada vez que finaliza la transmisión de un byte. De ese modo, la interrupción actualiza los contadores y transmite el siguiente byte de trama hasta que esta se ha completado, en cuyo caso la interrupción resetea los flags correspondientes a transmisión, dejando el SCI preparado para iniciar una nueva. Para que esto pueda darse, la subrutina Txon activa interrupción SCI por transmisión completa.

La subrutina Txon, a la hora de iniciar una nueva transmisión, lo hace de acuerdo a la prioridad establecida. Si ya hay una transmisión en curso, no inicia una nueva. Si no la hay, comprueba primero la cola con prioridad (ColaB1). Si no está vacía, establece ColaB1 + PRIB1 (desplazado) como dirección base de transmisión. Si ColaB1 está vacía comprueba ColaB0 del mismo modo, de modo que si esta está también vacía, no se inicia una nueva transmisión.

Puede darse el caso de que se esté recibiendo en la posición que se supone contiene la trama a transmitir, en cuyo caso es imposible iniciar la transmisión de esa trama, con lo que se toma la cola en cuestión como vacía y se continúa haciendo la selección de trama.

Una vez la trama se ha seleccionado para ser transmitida, es necesario comprobar si su timer ha de ser registrado. La condición para registrar el timer es que la trama haya sido emitida por el propio nodo (dirección origen de la trama = = Código numérico del nodo) y que la trama requiera asentimiento (ASE = = 1). En caso de que así sea, se llama a la subrutina encargada de registrar el timer (JSR REGTIMER).

PROCESADO

Esta subrutina se encarga del procesado de las tramas dirigidas al propio nodo que se encuentran almacenadas por consiguiente en colas A de procesado. Toma la primera trama que corresponda, según prioridades (comprueba ColaA1, y si esta está vacía comprueba ColaA0) y la somete a análisis. Posteriormente elimina la trama procesada y actualiza la cola. El diagrama de flujo aparece en la figura 5.1.3.3 .

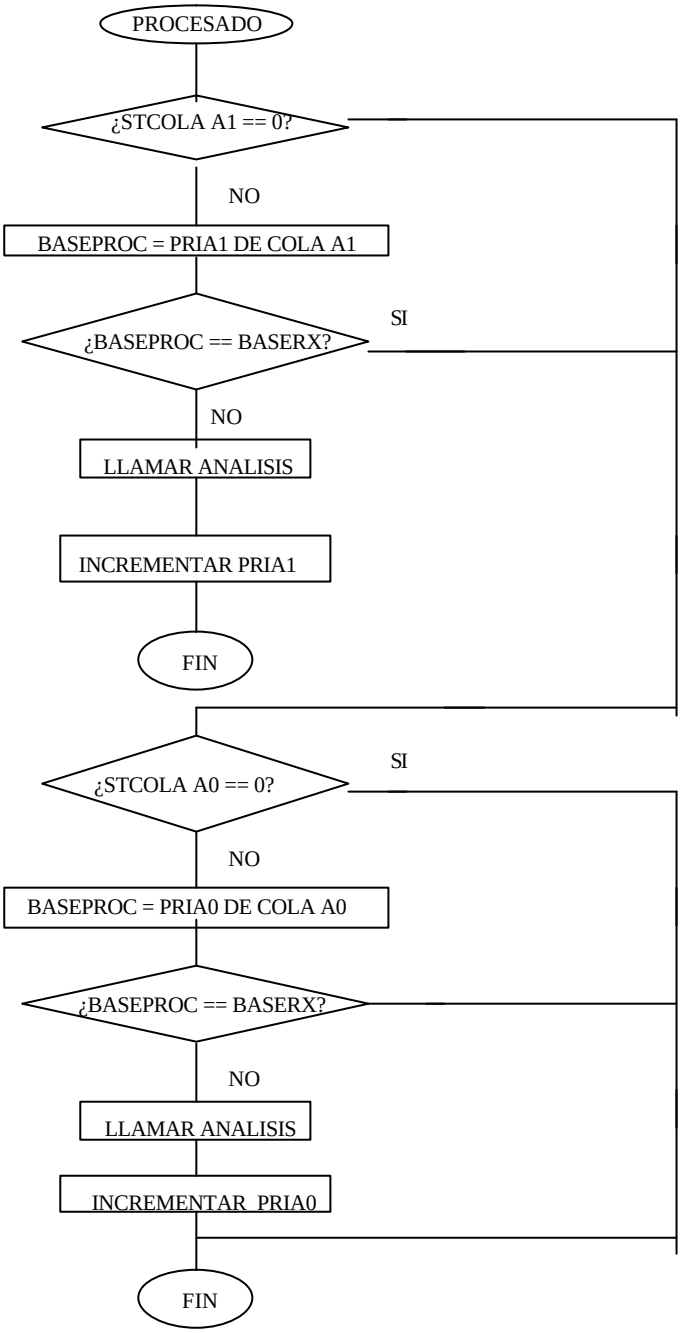


Fig. 5.1.3.3

La subrutina procesado, a la hora de procesar una nueva trama, lo hace de acuerdo a la prioridad establecida. Comprueba primero la cola con prioridad (ColaA1). Si no está vacía, establece ColaA1 + PRIA1 (desplazado) como dirección base de procesado. Si ColaA1 está vacía comprueba ColaA0 del mismo modo, de modo que si esta está también vacía, no se procesa ninguna trama.

Puede darse el caso de que se esté recibiendo en la posición que se supone contiene la trama a procesar, en cuyo caso es imposible procesar esa trama, con lo que se toma la cola en cuestión como vacía y se continúa haciendo la selección de trama.

Tras procesar la trama correspondiente se encarga de actualizar la cola circular correspondiente, para lo cual llama a la subrutina encargada de incrementar el índice PRIAX correspondiente.

Por otro lado, esta subrutina no analiza las tramas, sino que sirve únicamente para el tratamiento de las colas de procesado, de modo que llama a otra subrutina (ANALISIS) genérica para el análisis de las tramas que ya no se preocupa de las colas.

COMPTIMERS

Esta subrutina se encarga de la comprobación de los timers de las tramas que están almacenadas en la Lista de Espera, a la espera de recibir asentimiento. Recorre la lista de modo que cuando una posición está ocupada lee su timer y lo compara con el reloj del sistema (PRCD). Si el valor contenido en el timer coincide con el del reloj, implica que esa trama ya debería haber recibido el asentimiento, y que si no lo ha hecho es porque ha surgido algún problema en la comunicación y la trama debe ser retransmitida.

En ese caso, la subrutina llama a la subrutina INTCOLTX, encargada de introducir tramas en colas de transmisión para reintroducir la trama en la cola B de transmisión que corresponda según prioridad, y de actualizar su timer con un valor basura que evite que sea reintroducida nuevamente en cola de transmisión antes de que se inicie la retransmisión física. Ese valor basura de timer es TIMER = 0, de modo que dicho valor es evitado por PRCD, que pasa de 255 a 1.

Antes de llamar a la subrutina INTCOLTX debe indicar a esta subrutina que la trama que debe retransmitir se trata de una trama de retransmisión (STINCOL = 255) y su ubicación, para lo cual desplaza convenientemente el índice i y lo suma a la dirección base de la Lista de Espera, almacenando la dirección calculada en BASERTXON. Esta

dirección es la dirección base de la posición que ocupa la trama a retransmitir en la Lista de Espera.

El diagrama de flujo de la subrutina aparece en la figura 5.1.3.4 de la página siguiente.

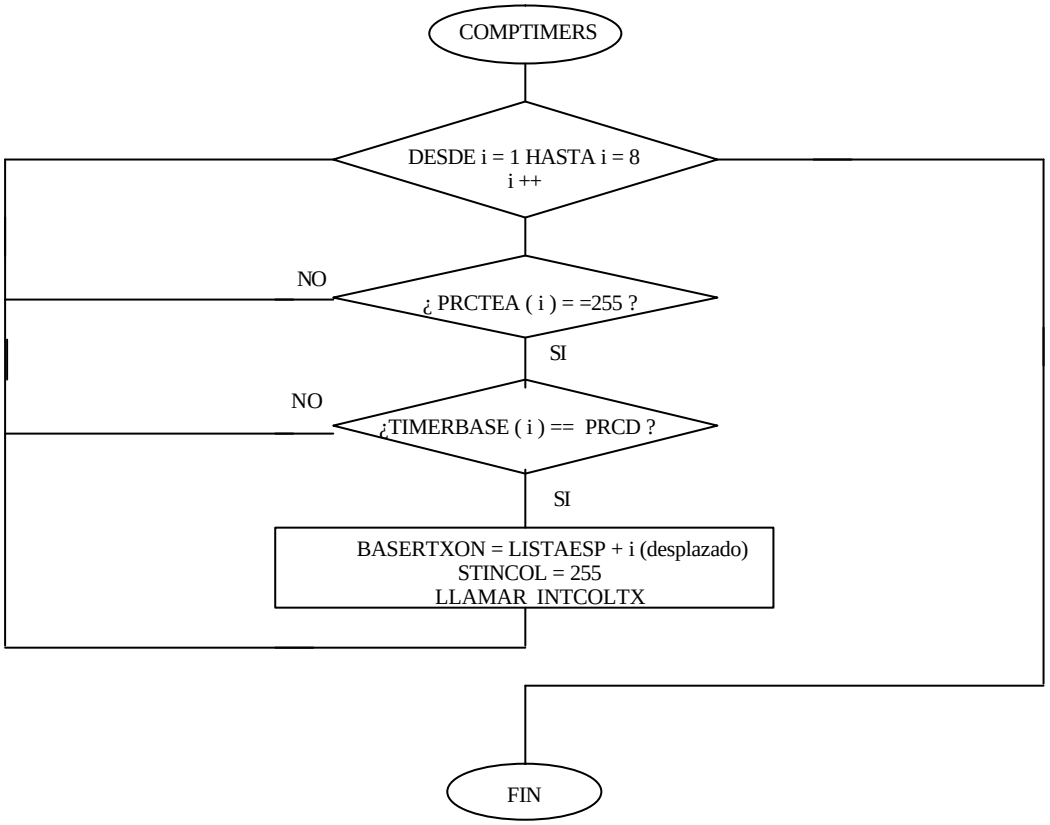


Fig. 5.1.3.4

ANALISIS

Esta subrutina recibe la trama que tiene que analizar tal cual, sin preocuparse de la cola de la que proceda. Tiene las siguientes funciones: llama a la subrutina encargada de calcular el síndrome de la trama, y si esta requiere asentimiento o rechazo, llama a la función encargada de generarlo. Si el síndrome calculado es 0 (trama sin errores), analiza la trama, llamando a la subrutina que corresponde según el tipo de trama.. En caso contrario, la trama es simplemente desechada.

Esta subrutina, realmente lo que hace es gestionar el análisis de la trama repartiendo el trabajo entre diversas subrutinas especializadas. Su diagrama de flujo aparece en la figura 5.1.3.5.

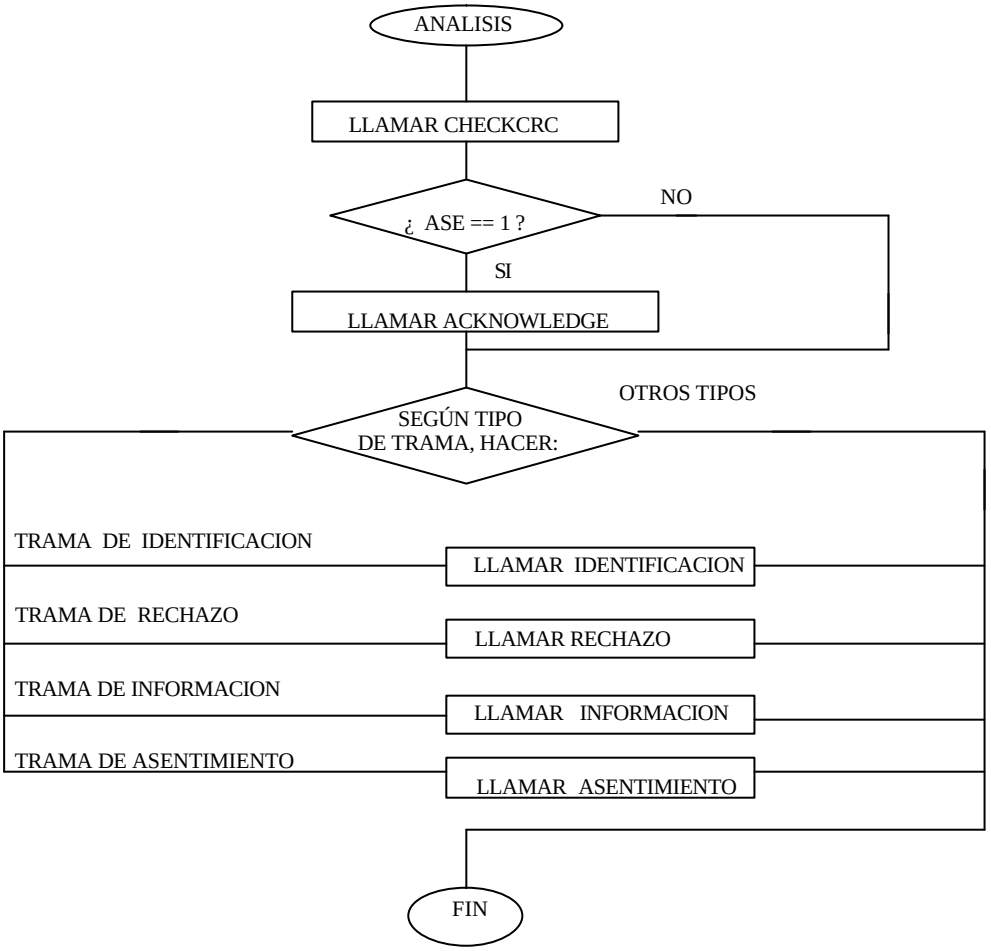


Fig. 5.1.3.5

ACKNOWLEDGE

Esta subrutina se encarga de generar el asentimiento o rechazo de las tramas que lo requieran en función de si dicha trama se recibió con (síndrome <> 0) ó sin errores (síndrome = 0). La trama es generada en la posición que tiene como dirección base BASERESP.

La trama que genera es una de asentimiento o una de rechazo, cuya cabecera configura según la cabecera de la trama original. Cuando tiene la cabecera configurada (estas tramas carecen de campo de datos), llama a las subrutinas encargadas del cálculo

del CRC (CALCULOCRC), y de la introducción de la misma en cola de transmisión (INTCOLTX). Para hacer saber a estas subrutinas que la trama sobre la que tienen que trabajar tiene como dirección base BASERESP, da el valor 170 a STINCOL. El diagrama de flujo correspondiente a la subrutina aparece en la figura 5.1.3.6 .

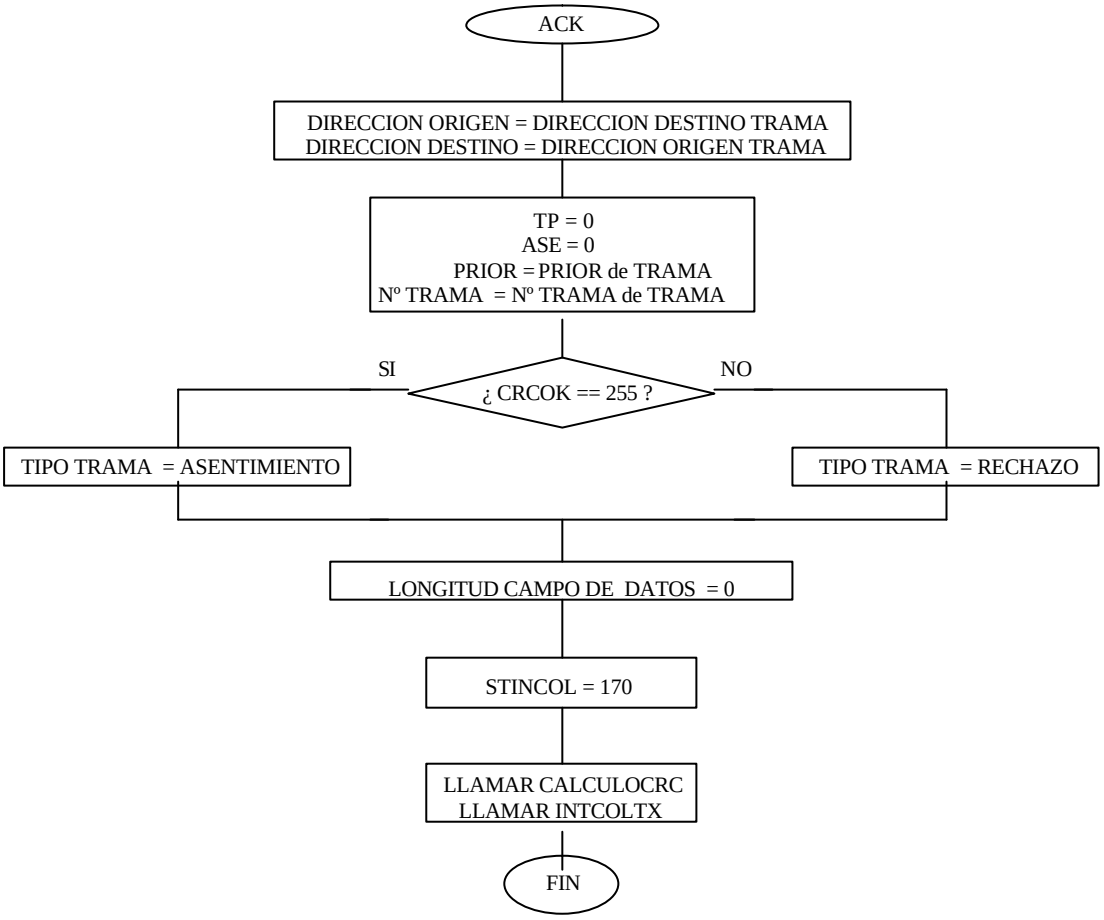


Fig. 5.1.3.6

ASENTIMIENTO

Esta subrutina tiene como misión realizar las operaciones oportunas tras la recepción de un asentimiento, esto es, dar por exitosa la transmisión de la trama eliminando esta de la Lista de Espera.

Para ello toma el identificador numérico del asentimiento recibido e introduce un 0 en la posición PRCTEA + ID numérico. Es decir, registra la posición correspondiente de la Lista de Espera como no ocupada ($00).

RECHAZO

Esta subrutina se encarga de realizar las operaciones oportunas para que una trama que ha sufrido errores en su transmisión , errores que detectó el nodo destino, sea retransmitida. Para ello llama a otra subrutina encargada de introducir tramas en colas de transmisión (INTCOLTX) a la que pasa como parámetros el tipo de trama a introducir (STINCOL = 255) y la dirección base de la trama (BASERTXON = LISTAESPERA + ID num (desplazado)).

IDENTIFICACIÓN

Esta subrutina se encarga de registrar el número de nodo que le ha correspondido a la tarjeta en el proceso de configuración del sistema, así como de enviar al PC la información acerca de la naturaleza del nodo recogida en el campo CLASENODO.

El ID de nodo lo toma directamente del campo de datos de la trama de identificación recibida y lo introduce en el registro PRCN. Este identificador único será desde ese instante la “huella dactilar” del nodo.

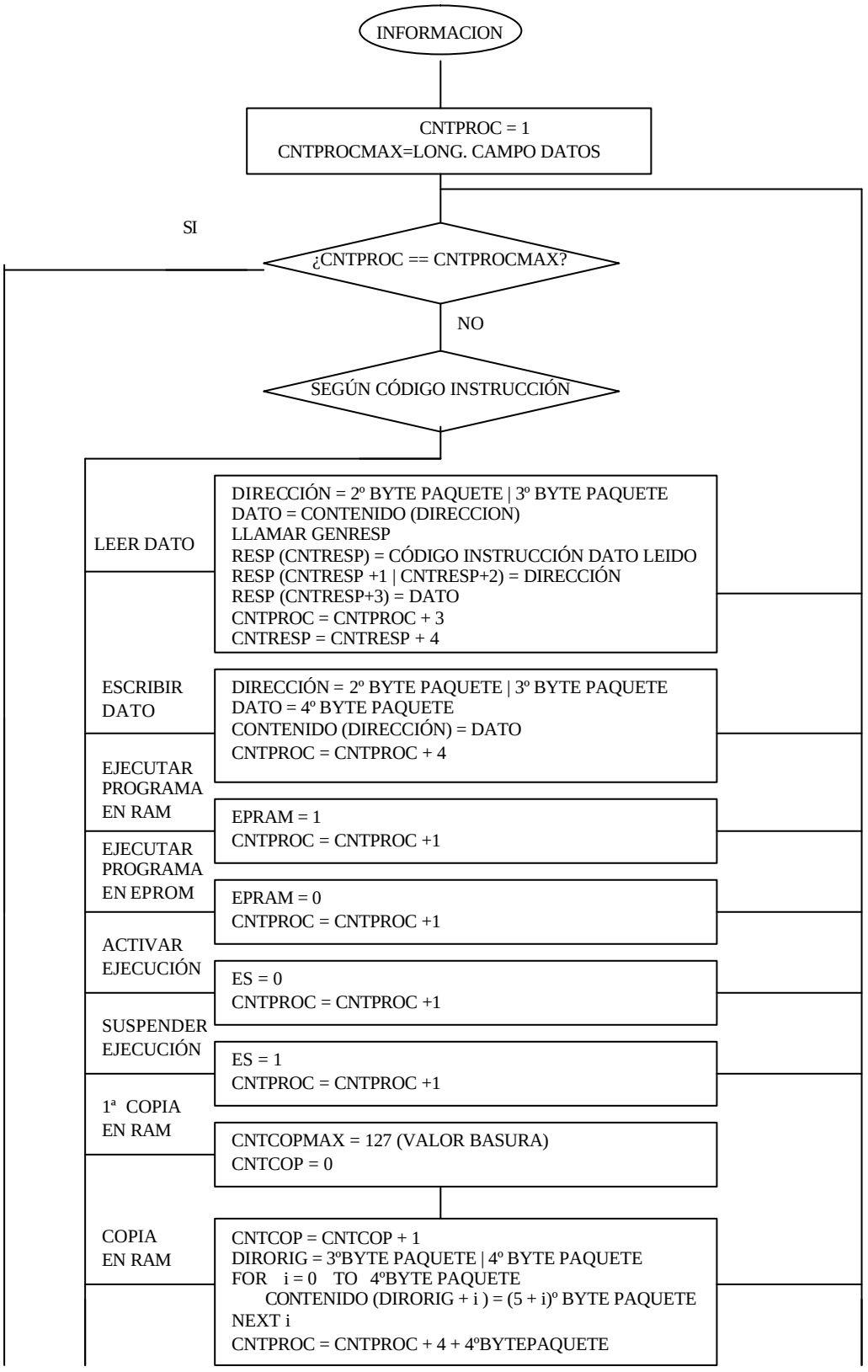
Además configura una trama idéntica a la recibida, salvo por el campo de datos, que ha de tomar el valor correspondiente al que será el ID del siguiente nodo en el sistema. Por este motivo incrementa el campo de datos recibido en 1. Con el nuevo campo recalcula el CRC de la trama (llama a CALCULOCRC) y la introduce en cola de transmisión (llama a INTCOLTX).

Tras haberse identificado, para que el nodo esté totalmente registrado en el sistema, debe emitir una trama de registro de nodo. La subrutina se encarga también de la configuración de esta trama. Por ser la primera trama emitida por el nodo que necesita asentimiento, se configura directamente en la posición 0 de la Lista de Espera, por lo que tiene como dirección base LISTAESP, y como Identificador de trama 0.

El campo de datos lo configura copiando el contenido de los 7 bytes de CLASENODO. Por último llama a CALCULOCRC e INTCOLTX para que la trama sea transmitida. Además ha de dar al timer de esta trama un valor basura ($00) como ya habíamos visto.

INFORMACIÓN

Esta subrutina se encarga de ejecutar las instrucciones de nivel de aplicación que llegan encapsuladas en tramas de información. Para comprender el funcionamiento lo mejor es ver el diagrama de flujo de la subrutina, presente en la figura 5.1.3.7 .



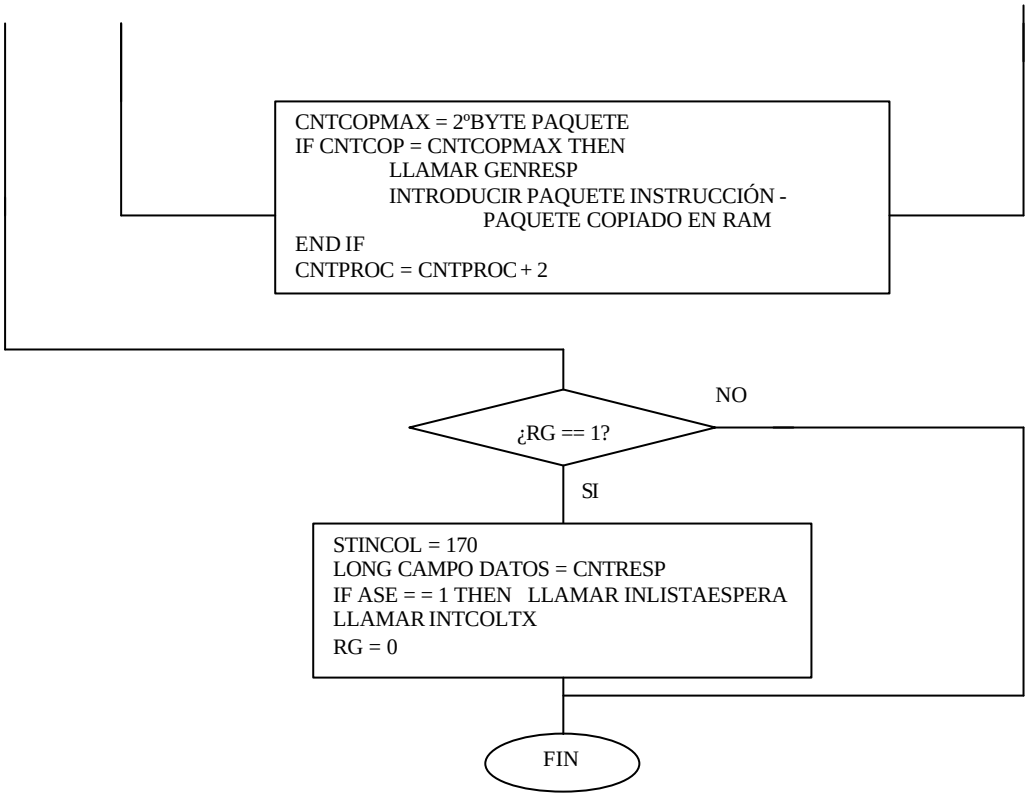


Fig. 5.1.3.7

Como se puede apreciar en el diagrama de flujo, la subrutina extrae uno a uno los paquetes de aplicación, y según el código de instrucción de estos, desemcadena una secuencia de operaciones que ejecutan la instrucción del paquete.

En el diagrama se hacen llamadas a la subrutina GENRESP. Esta subrutina se encarga de configurar la cabecera de la trama de respuesta según corresponda e inicializa el contador CNTRESP de datos de respuesta a 0. Esta trama tiene como dirección base BASERESP.

CALCULOCRC – CHECKCRC – OBTENERCRC

Esta terna de subrutinas se encargan del control de errores en las transmisiones en base a un polinomio cíclico CRC (Cyclic Redundancy Check), concretamente el CRC-16 del CCITT.

CALCULOCRC recibe una trama recién configurada y la prepara para que su CRC sea calculado poniendo este a 0. Luego llama a la subrutina OBTENERCRC.

CHECKCRC recibe una trama que está siendo procesada y llama a la subrutina OBTENERCRC a la que indica que se está chequeando una trama. En función del resultado da valores a la variable CRCOK que indica si la trama contiene o no errores (Síndrome = = 0 -> CRCOK = 255) (Síndrome <> 0 -> CRCOK = 0).

OBTENERCRC es la subrutina genérica, encargada del cálculo del CRC ó síndrome de una palabra de código (en nuestro caso la trama de 32 bytes). Genera una salida de 16 bits (el CRC ó síndrome) que es el resto de la división binaria de la palabra de código y el polinomio generador del CRC-16 del CCITT.

Para ejecutar la división binaria, se trabaja con 2 bytes contenidos en el acumulador D (compuesto por los acumuladores A (byte alto) y B(byte bajo)), que finalmente contendrá el resto como resultado. Es necesaria una variable auxiliar BYTECRC que contiene el byte que se va introduciendo bit a bit por la derecha en el dividendo. Es necesario introducir tantos bytes como bytes componen la trama menos 2. Por cada una de estas iteraciones habrá que introducir 8 bits por lo que se repetirá un bucle 8 veces por cada iteración. Para tener una idea más clara de ello, lo mejor es ver el diagrama de flujo de la figura 5.1.3.8 .

Realmente no se ejecuta ninguna instrucción de división binaria de palabras tan largas, ya que el juego de instrucciones del micro es limitado y no lo permite. Por ello ha habido que implementar esta división con operaciones sencillas y repetitivas que además dejasen libres los acumuladores ya que estos resultan imprescindibles para manejar la gran cantidad de datos necesarios para el cálculo del CRC.

El tiempo de ejecución de esta subrutina es elevado, pero necesario para el control de errores, sobre todo si se tiene en cuenta que el sistema está pensado para aplicaciones de control. No obstante esta subrutina solamente se ejecuta una vez por trama, y si tenemos en cuenta que estas en media no serán muy largas, el tiempo de ejecución no resulta tan elevado.

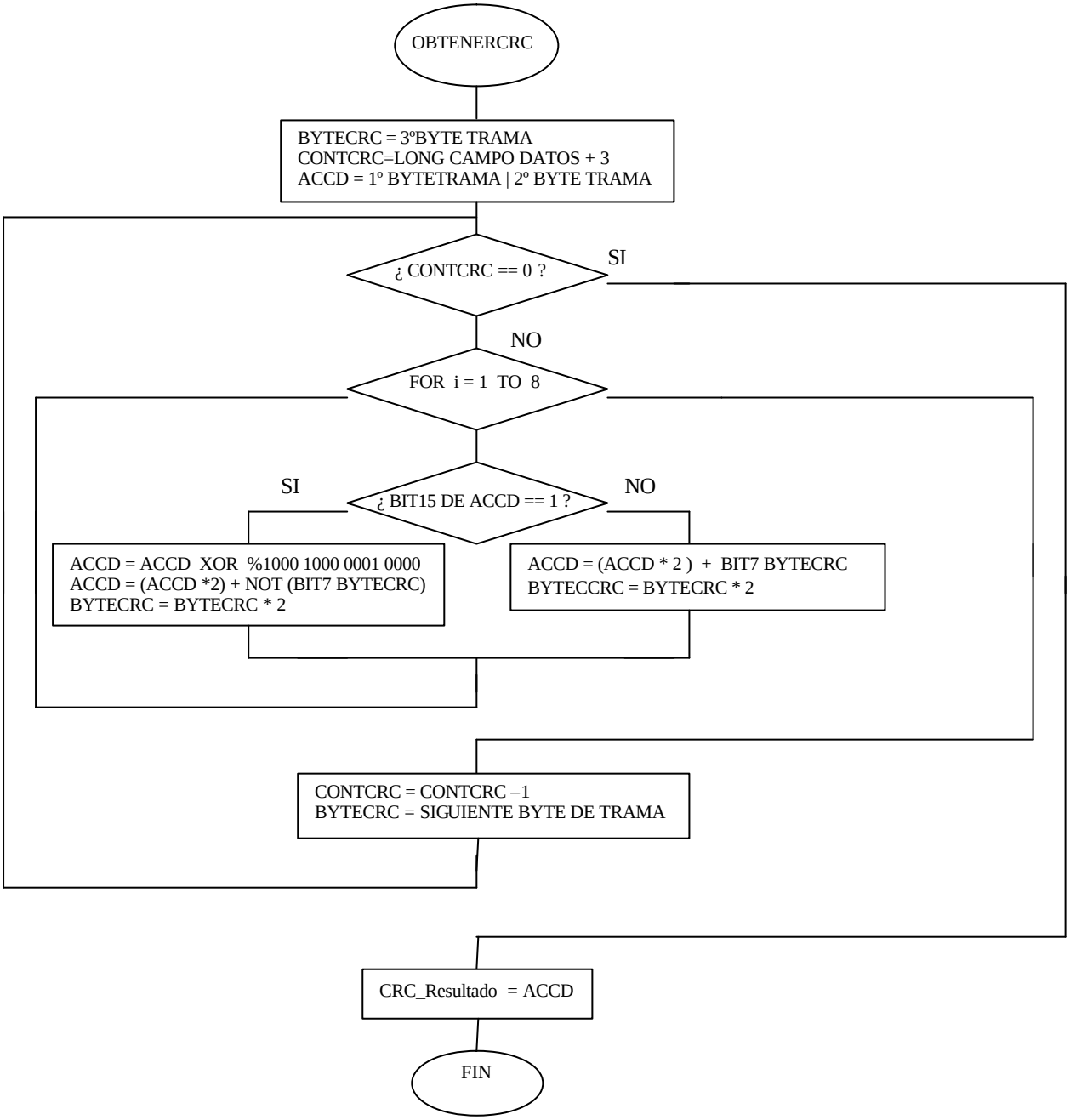


Fig. 5.1.3.7

INTERRUPCIÓN AVISARPC (SWI – FALLO COP)

Esta subrutina de interrupción no enmascarable se utiliza para la generación de un mensaje de error fatal en la configuración del sistema hacia el PC, para permitir a este reconocerlo y reconfigurar inmediatamente el sistema..

La interrupción se dispara cada vez que se ejecuta la instrucción SWI o cada vez que el Watch Dog detecta un fallo. Estas instrucciones están estratégicamente distribuidas a lo largo del código, de modo que cada vez que se produce un error fatal, la interrupción es disparada.

Estos errores pueden ser debidos a ciertos factores:

- Desbordamiento del Watch Dog (Perro guardián), lo que implica que se ha producido un fallo local en el nodo correspondiente al que hay que resetear. Ello obliga a dar un reset general al sistema y reconfigurarlo.

- Error detectado en la recepción del tercer byte de trama, ya que puede que este afecte al campo longitud del campo de datos, lo que hace que se pierda el sincronismo de trama. Ello hace necesario reconfigurar el sistema.

- Error no detectado en el tercer byte de trama de modo que el valor recibido para el campo de datos sea superior al máximo (27 bytes). Por el mismo motivo que en el caso anterior se hace necesario reconfigurar el sistema.

- Recepción de una trama con dirección no multicast cuando el nodo no está registrado en el sistema (MA, micro activo, = 0). La primera trama que recibe siempre un nodo tras el Reset, es la trama multicast de inicialización. Por este motivo, cuando una trama con otro tipo de dirección es recibida por un nodo no activo, implica que ha surgido algún problema (nuevo nodo en la red, pérdida de configuración etc..) y el sistema debe ser reconfigurado.

Esta interrupción es totalmente portable, y puede ser disparada cuando se estime oportuno sin más que introducir la instrucción SWI donde corresponda. Puede resultar útil para futuras ampliaciones del sistema.

La subrutina de interrupción AVISARPC se encarga de configurar una trama de petición de configuración (TIPO = 110) e introducirla en colas de transmisión enviársela al PC. Esta trama es fija, por lo que no es necesario calcular su CRC.

A partir del momento en que la subrutina de interrupción ha sido ejecutada, el micro deja de recibir por el SCI, ya que la subrutina desactiva interrupciones en recepción para enviar la trama de petición de configuración cuanto antes, de modo que trata de evitarse ejecutar código no asociado a la transmisión de dicha trama.

INTERRUPCIÓN INTERRUPTIMER (TIMER OVERFLOW)

Esta interrupción se dispara cada vez que el TCNT desborda. En ese momento el contador global del nodo (PRCD: Protocolo Registro de Control de Desbordaminetos) se incrementa en 1. De este modo conseguimos un reloj de paso igual al tiempo de desbordamiento del TCNT y de desbordamiento igual a 256 por el desbordamiento de TCNT. Este reloj se adecua totalmente a los parámetros temporales que rigen el protocolo para asentimiento de tramas.

INTERRUPCIÓN INTERRUPSCI (SCI SERIAL SYSTEM)

Esta interrupción se encarga de la gestión de las comunicaciones a través del SCI. Hay tres posibles causas de disparo: recepción carácter, transmisión de carácter y error en recepción de carácter. La interrupción comprueba los flags correspondientes para saber cual ha sido la causa del disparo en cada ocasión, y en función de la misma, ejecuta distintos sectores de código.

Las causas de disparo pueden activarse/desactivarse de manera independiente en función de las necesidades del sistema, así por ejemplo, la interrupción de transmisión completa de carácter está desactivada mientras no haya transmisión en curso, de forma que es activada cuando la transmisión es iniciada (subrutina TXON), y se desactiva en el último disparo de la interrupción asociado a la transmisión de una trama, esto es, cuando la transmisión del último byte se ha completado con éxito.

Para ver como funciona esta interrupción lo mejor es ver directamente los diagramas de flujo que aparece en las figuras 5.1.3.8 , 5.1.3.9 y 5.1.3.10 correspondientes a Transmisión completa, Error en Recepción de carácter y Recepción completa respectivamente. Realmente constituyen partes de un mismo diagrama, pero por simplicidad se ha decidido presentarlos de forma independiente.

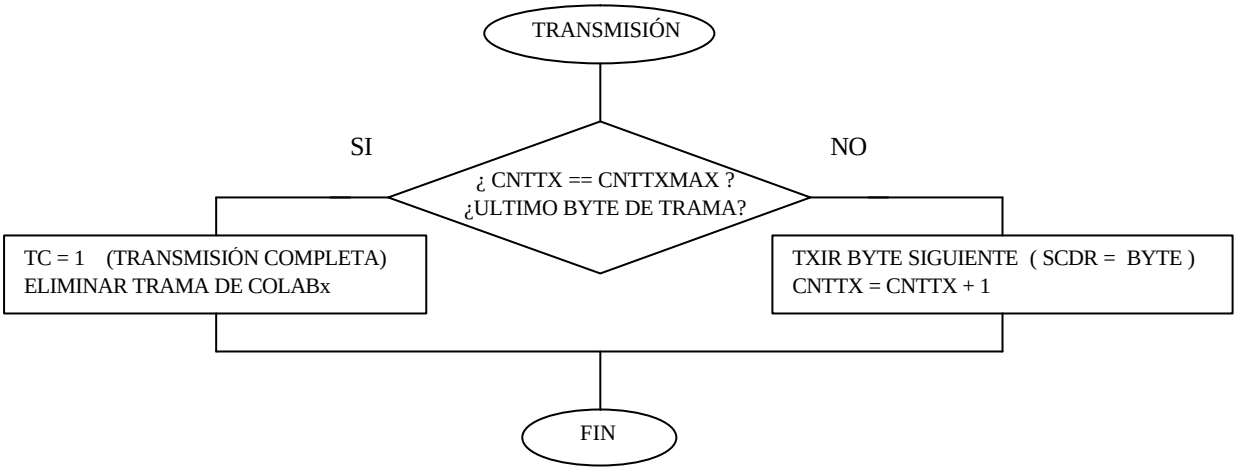


Fig. 5.1.3.8

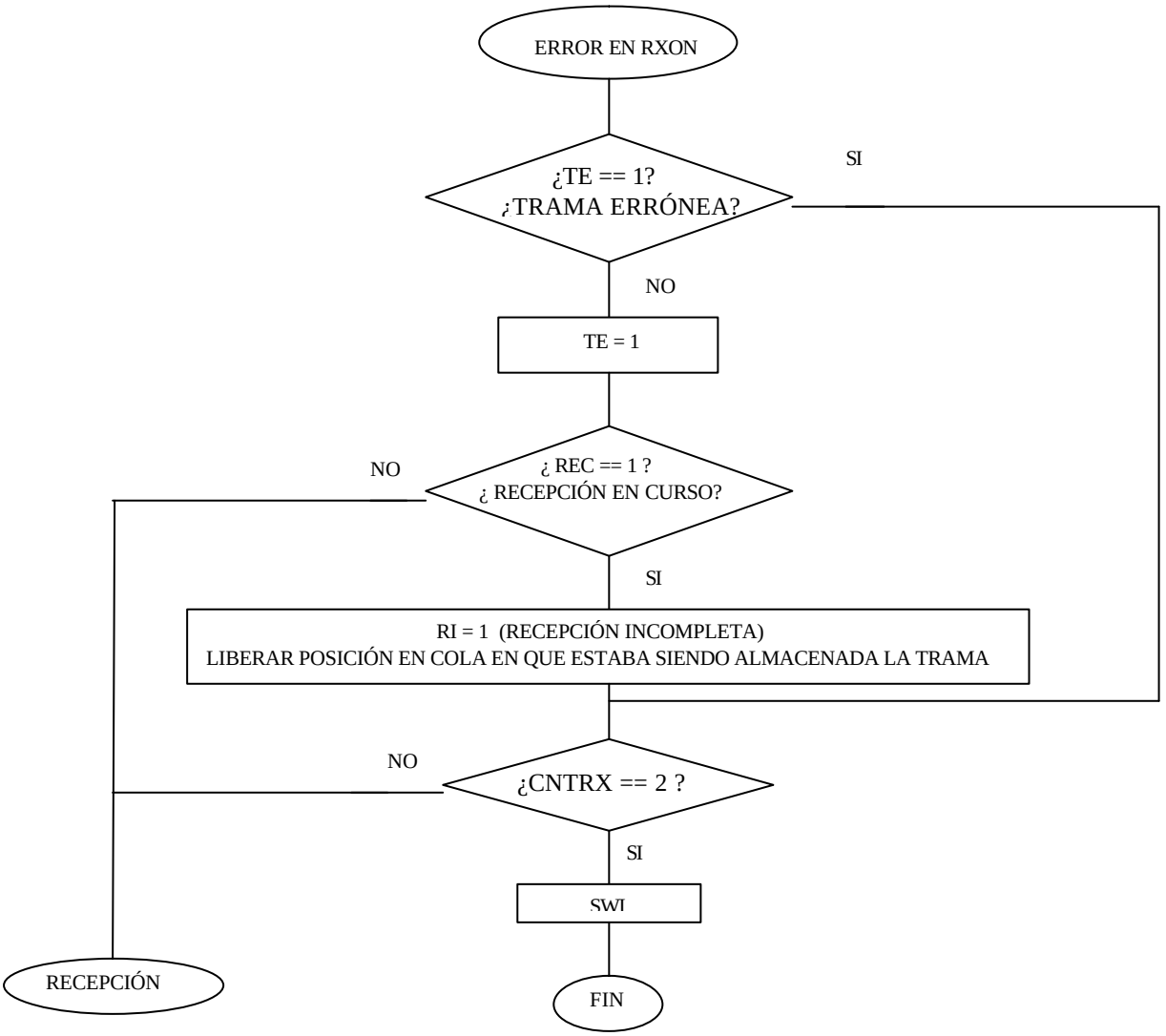
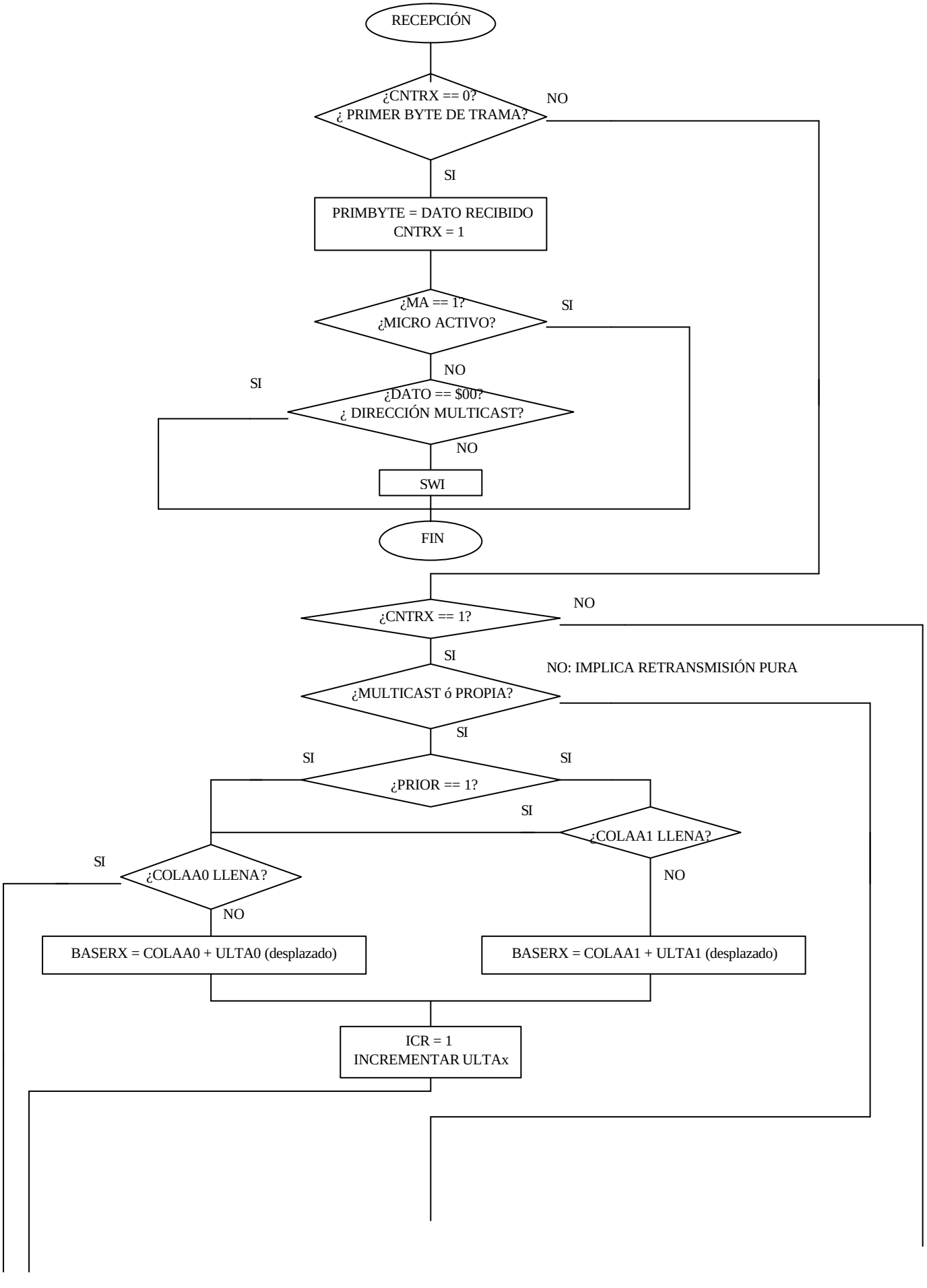
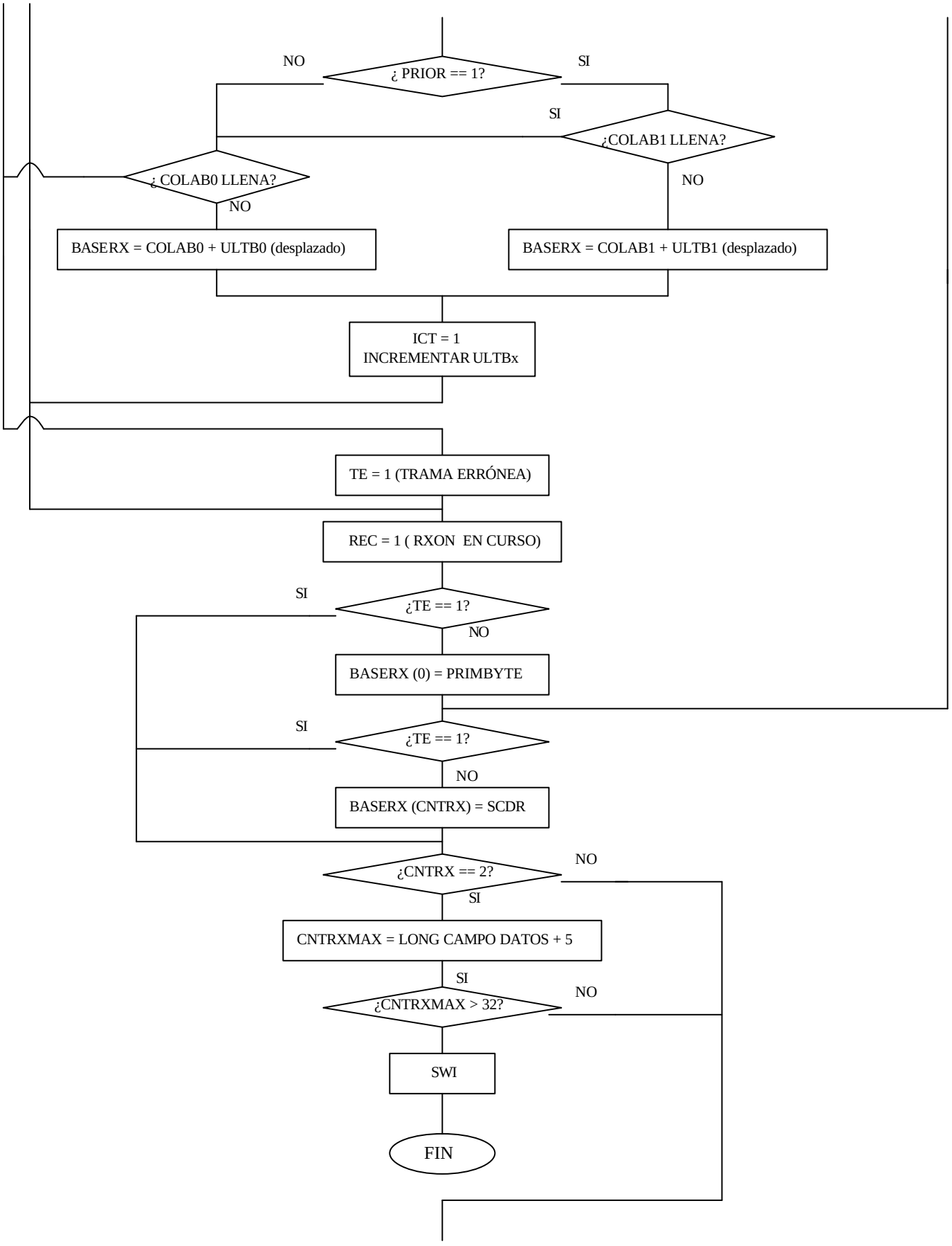


Fig. 5.1.3.9


RECEPCIÓN
¿CNTRX == 0?
¿ PRIMER BYTE DE TRAMA?
NO
SI
PRIMBYTE = DATO RECIBIDO
CNTRX = 1
¿MA == 1?
¿MICRO ACTIVO?
SI
NO
SI
¿DATO == $00?
¿ DIRECCIÓN MULTICAST?
NO
SWI
FIN
¿CNTRX == 1?
NO
SI
NO: IMPLICA RETRANSMISIÓN PURA
¿MULTICAST ó PROPIA?
SI
SI
SI
¿PRIOR == 1?
SI
¿COLAA1 LLENA?
SI
¿COLAA0 LLENA?
NO
NO
BASERX = COLAA0 + ULTA0 (desplazado)
BASERX = COLAA1 + ULTA1 (desplazado)
ICR = 1
INCREMENTAR ULTAx


¿ PRIOR == 1?
NO
SI
¿COLAB1 LLENA?
SI
NO
¿ COLAB0 LLENA?
NO
BASERX = COLAB0 + ULTB0 (desplazado)
BASERX = COLAB1 + ULTB1 (desplazado)
ICT = 1
INCREMENTAR ULTBx
TE = 1 (TRAMA ERRÓNEA)
REC = 1 (RXON EN CURSO)
¿TE == 1?
SI
NO
BASERX (0) = PRIMBYTE
¿TE == 1?
SI
NO
BASERX (CNTRX) = SCDR
¿CNTRX == 2?
NO
SI
CNTRXMAX = LONG CAMPO DATOS + 5
SI
¿CNTRXMAX > 32?
NO
SWI
FIN

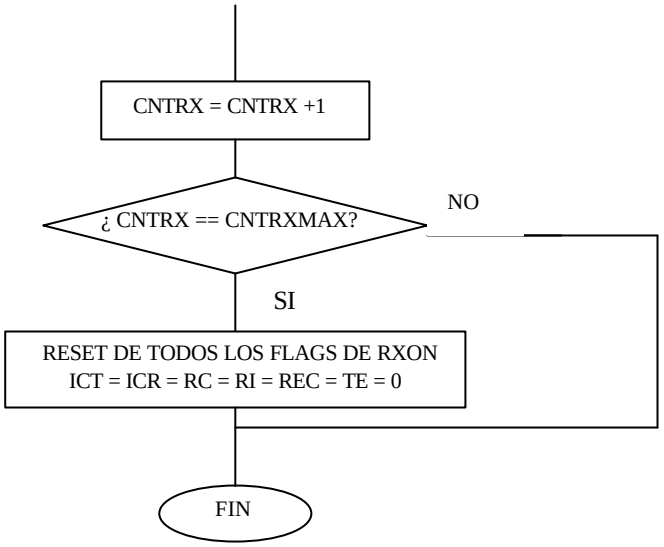


Fig. 5.1.3.10

CICLO DE NIVEL DE APLICACIÓN – CICLO DE NIVEL DE RED

Una vez presentadas las diferentes subrutinas (las principales) que componen el código del protocolo, podemos evaluar la carga computacional que este acarrea. Habíamos presentado con anterioridad diversos índices que permitían cuantificar la eficiencia del protocolo., si bien habíamos dicho que era difícil evaluarlos ya que dependían en gran medida de las condiciones del tráfico de la red, aplicación local etc.

Lo que si que podemos evaluar en términos absolutos son los parámetros a los que llamábamos TCRV (Tiempo de ciclo de nivel de red en vacío) y la parte correspondiente al ciclo de nivel de aplicación del TCAV (Tiempo de ciclo de aplicación en vacío). Estos parámetros se evalúan cuando no hay tráfico en la red y cuando las colas y la lista de espera están vacías.

TCAV - Ciclo aplicación: 25 ciclos de instrucción, lo que se traduce en 12,5 us.

TCRV : aproximadamente 300 ciclos de instrucción, es decir 150 us.

5.1.4 Aplicación de usuario.

La aplicación de usuario es el programa local ejecutado en cada uno de los nodos para control digital directo de un proceso determinado.

Como habíamos visto, esta aplicación puede estar almacenada en memoria EPROM ó ser descargado a memoria RAM desde el PC, para lo cual es necesario haber ensamblado el programa antes, de modo que la información se transfiere desde el fichero .s19 correspondiente que contiene el programa en formato s19 de Motorola.

En todos los casos el programa debe tener la misma estructura, esto es, estar definido como una subrutina, ya que se llama desde el programa principal (JSR APLICACIÓN para EPROM y JSR RAMBS para RAM):

```
                (ORG   $0100)     (Sólo para aplicaciones RAM)
(APLICACIÓN)    PSHA              ; La etiqueta sólo corresponde a aplicaciones
                .....             ; EPROM.
                PULA
                RTS         ; Absolutamente indispensable
```

Es imprescindible terminar la subrutina con la instrucción RTS, ya que en caso contrario no se devolvería el control al programa principal y el sistema acabaría fallando (fallo detectado por Watch Dog).

Independientemente de estar definido como una subrutina, el programa puede contener tantas subrutinas como sea preciso.

Variables

Con objeto de estandarizar el sistema para que las aplicaciones de usuario puedan ser fácilmente entendidas por programadores diferentes al autor de las mismas, se ha establecido un rango de memoria para las variables de aplicación local, de modo que deben ser declaradas adecuadamente:

Variables de aplicación RAM: se ha reservado para estas un bloque de memoria de 256 posiciones (desde $0000 hasta $00FF). En caso de necesitar más posiciones, las variables correspondientes pueden ser declaradas justo a continuación del código de programa.

Variables de aplicación EPROM: se declararán a partir de la posición $1040, es decir, justo a continuación de los registros de control del micro, lo que permite aprovechar la memoria de forma eficiente.

En definitiva, el código de una aplicación RAM tendrá la siguiente estructura:

```
                ORG   $0000
VARIABLE1       RMB   1
    ........
VARABLE2        RMB   1
                ORG $0100
                PSHA
                 ........
                PULA
                RTS
```

El código de una aplicación EPROM se inserta en el código fuente del protocolo, en la dirección etiquetada por APLICACIÓN. Para declarar las variables es necesario hacerlo de la siguiente manera:

```
                ORG   $1040
VARIABLE1       RMB   1
VARIABLE2       RMB   1
    .......
```

5.1 CÓDIGO DE PROGRAMA PARA PC

5.1.1 Generalidades

El programa para la gestión del Sistema de Control Multipunto ha sido escrito en Visual Basic por la sencillez que presenta para la programación de una aplicación en un entorno visual para Windows. Este programa constituye la interfase hombre-maquina, de modo que resulta de gran utilidad que esta sea visual.

Para la etapa de diseño del programa fue necesario el uso del programa Visual Basic 6.0 del paquete Microsoft Visual Studio 6.0, en su versión profesional, ya que el control MSCOMM32.OCX, que facilita la comunicación a través del puerto serie, no está disponible en otras versiones. Asimismo, es necesaria la utilización de dicha versión para la compilación del código y generación del fichero ejecutable correspondiente.

El programa incluye todas las funciones de protocolo que hemos visto en el apartado anterior, al tiempo que incluye otras para gestión de la red (autodiagnóstico, control de tráfico etc.) y aquellas funciones útiles para el usuario.

5.1.2 Protocolo: Variables

El lenguaje Visual Basic permite definir variables de diferentes ámbitos, de tal modo que estas sólo son accesibles desde las funciones y procedimientos que permita dicho ámbito. De este modo tendremos variables privadas a la función o procedimiento donde están definidas que se utilizarán para cálculos u operaciones locales a dicha función o procedimiento. También tendremos variables privadas al módulo en que están definidas que podrán ser usadas por todas las funciones y procedimientos pertenecientes a ese módulo. Por último tendremos variables públicas accesibles por todas las funciones y procedimientos del programa. Son estas últimas las que nos interesa detallar en este apartado por ser las que contendrán información global del sistema. Están declaradas en el Módulo-módulo y son las siguientes:

- **ColaA0 – ColaA1 – ColaB0 – ColaB1**

Son estructuras que incluyen las colas de transmisión ó procesado, que en este caso tienen 32 posiciones que contienen un tabla de 32 bytes para los caracteres de trama, así como todos las variables de control asociadas (primero, último, flag de cola vacía y flag de cola llena).

- **ListaEspera**

Es una tabla de 32 posiciones, cada una de las cuales tiene otra tabla de 32 bytes (caracteres de trama) y variable longitud de la trama.

- **TramaRecibida**

Es una tabla de 32 posiciones (bytes) para almacenar las tramas recibidas

- **Sistema**

Estructura que contiene información sobre el sistema como el número de nodos que lo componen, flag que indica si está o no configurado, y una tabla de 15 posiciones (una por cada posible nodo del sistema), cada una de las cuales contiene los 7 bytes de información sobre la naturaleza del nodo correspondiente (pines activos, direccionalidad y función).

- **Respuestas - Alarmas**

Son colas circulares (32 mensajes disponibles en cada una de ellas) de cadenas de caracteres que contienen los mensajes de respuesta y alarma recibidos por el PC.

Además de estas variables hay otras variables como flags de control de las comunicaciones, CRC, etc... Al tratarse el lenguaje de programación Visual Basic de un lenguaje de alto nivel y estar el código suficientemente comentado, remitimos a este para una mejor comprensión del programa.

5.2.2 Protocolo: Principales características

En este apartado se describirá el criterio seguido para la estructuración del programa, pero no se comentará el código en sí, ya que como se ha descrito anteriormente, con la información contenida en el propio código fuente es suficiente para comprender el funcionamiento del programa.

Este lenguaje tiene su base en los eventos, de modo que para que se ejecute cualquier función o procedimiento es preciso que se produzca un evento. Para aquellos procedimientos que han de dispararse periódicamente se han dispuesto timers que generan eventos a intervalos fijos de tiempo.

El programa consta de cuatro formularios para la presentación de operaciones específicas, un formulario principal y por último un módulo que contiene aquellos procedimientos, funciones y variables públicos que utilizan las funciones y procedimientos del resto de los formularios.

Cada formulario contiene varios controles que permiten generar diferentes eventos. El programa está formado por las secuencias de código asociadas a cada uno de estos controles. En concreto el control Mscomm es el que mayor interés tiene para nosotros, ya que es el que nos permite establecer la comunicación con el sistema.

FORMULARIO PRINCIPAL

Este formulario aparece cuando arranca la aplicación y presenta tres diálogos diferentes. El primero de ellos permite configurar el sistema, el segundo es un diálogo de espera mientras dura la configuración y el tercero se activa una vez se ha configurado el sistema y permite acceder a los formularios de operación. Este formulario aparece en la figura 5.2.2.1 .

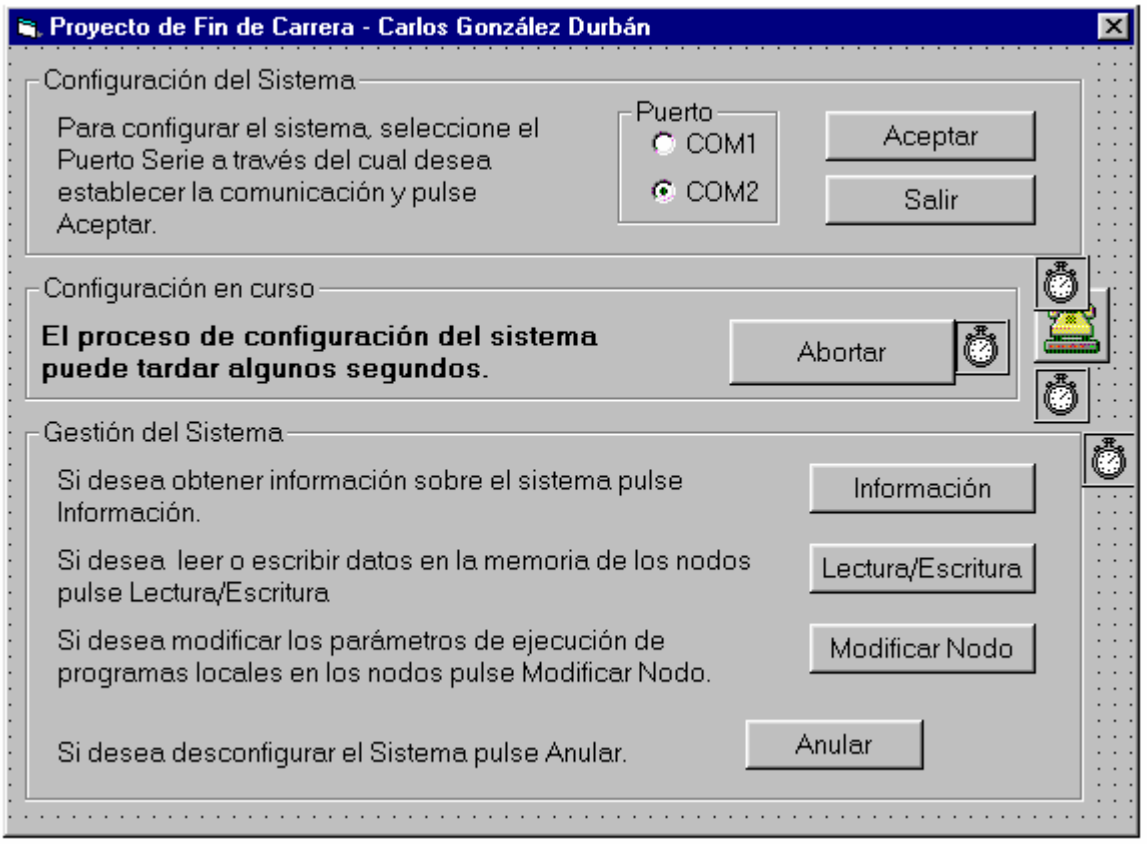


Fig. 5.2.2.1

Los procedimientos principales de este formulario son aquellas asociadas a los controles Time y MSComm.

AceptarConfiguracion_Click

Este procedimiento, que se ejecuta cuando el usuario pica el control correspondiente (Aceptar) , se encarga de configurar los parámetros del control MSComm que es a su vez el que permite establecer comunicaciones a través del puerto serie. Los parámetros del puerto serie toman los siguientes valores:

- Velocidad binaria = 9600 bps
- Bits de datos = 8
- Paridad = Off
- Bits de stop =1

Para poder configurar el sistema, este procedimiento se encarga de mandar un

reset a todos los nodos. Lo hace poniendo un nivel alto en RTS durante un pequeño intervalo, devolviendo la línea a nivel bajo después.

Desde este procedimiento se llama a otro (EnvioInicialización) encargado de configurar y enviar una trama de identificación a los nodos del sistema.

MSComm1.OnComm

Este procedimiento se ejecuta cada vez que se produce cualquier evento en el puerto de comunicaciones.

Se encarga de recibir/transmitir las tramas a través del puerto serie, al tiempo que ejerce funciones de autodiagnóstico de la red (detecta falta de sincronismo de trama).

La principal diferencia con la subrutina de interrupción SCI del código ensamblador visto en el apartado correspondiente, es que este evento permite ser programada para dispararse tras la recepción/trasmisión de un número cualquiera de bytes.

De este modo, para la transmisión de una trama basta con copiarla completamente en el buffer de transmisión y para la recepción de una trama, sólo se ejecuta el procedimiento MSComm en dos ocasiones:

- La primera de ellas se encarga de recibir la cabecera, de longitud fija (3 bytes). Se programa el controlador para que se dispare el evento cuando se hayan recibido 3 bytes en el buffer de entrada. Al recibir la cabecera, se extrae el campo longitud del campo de datos (n), de modo que se programa el controlador para que el siguiente evento de recepción se dispare cuando se hayan recibido n+2 bytes.

- La segunda se encarga de recibir el campo de datos y el CRC de la trama de longitud variable (entre 2 y 29 bytes). El evento se dispara según lo descrito en el párrafo anterior. Al finalizar, se reprograma el controlador para que el evento en recepción se vuelva a disparar tras recibir 3 bytes (cabecera).

Para ello es necesario tener flags que indiquen si el próximo evento será recepción de cabecera o de datos. Cuando se ha recibido tanto la cabecera como los datos+CRC, se integran todos los bytes en una sola trama y se almacena en la cola correspondiente.

TimerAsentimientos_Timer

Es el procedimiento equivalente a la subrutina Comptimers del código ensamblador, pero en esta ocasión se ejecuta periódicamente, ya que se trata de un control Timer. Estos controles generan un evento cada cierto intervalo que es fijado por el programador.

Recorre la lista de espera, comprobando que el timer de las posiciones ocupadas no haya vencido. En caso de haberlo hecho, reintroduce la trama correspondiente en cola de transmisión.

TimerTransmisiones_Timer

Es el procedimiento equivalente a la subrutina Txon del código ensamblador, pero en esta ocasión se ejecuta periódicamente, ya que se trata de un control Timer. Estos controles generan un evento cada cierto intervalo que es fijado por el programador.

Comprueba que el flag ListaEnviar está activo, lo que implica que puede transmitir una nueva trama, y en caso de que haya trama a la espera, la transmite introduciéndola en buffer de transmisión.

Son muchos los procedimientos que contiene este formulario, si bien estos se encuentran ampliamente comentados en el propio código fuente, al que remitimos para un completo conocimiento del formulario.

FORMULARIO INFORMACIÓN

Este formulario presenta el cuadro de diálogo que permite al usuario obtener información sobre el sistema acerca del número y la naturaleza de los nodos que lo componen. El formulario aparece en la figura 5.2.2.2 .

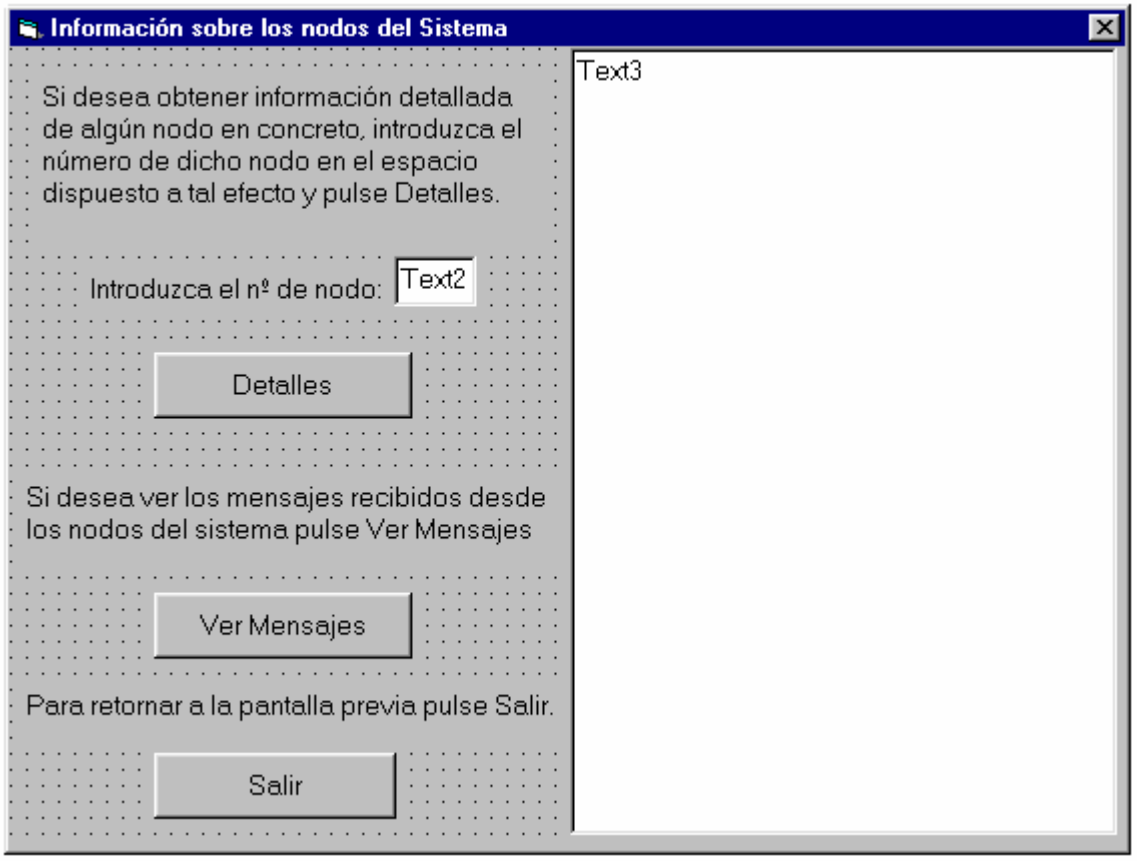


Fig. 5.2.2.2

Los procedimientos que incorpora este formulario manejan información almacenada en variables globales, y se limitan exclusivamente a dar formato a esa información.

DarInformacion

Este es el más importante procedimiento del formulario. Se encarga de dar formato a la información codificada en 7 bytes que sobre los nodos se tiene.

Convierte esa información en una larga cadena de caracteres que luego presenta por pantalla. Para ello, cada bit de esos 7 bytes lleva asociada una cadena de caracteres

en función de si está o no a nivel alto. Lo que el procedimiento hace es comprobar bit a bit si está o no a 1, y añade la cadena de caracteres correspondiente a la total.

El tratamiento que hace es completamente numérico, de tal modo que divide los 7 bytes en 3 bloques (3-3-1 bytes), calculando el valor numérico asociado. Para ver si está a 1 el bit correspondiente a la posición i, hace lo siguiente:

Num AND (2 exp i) = 0 -> bit i =0; Num AND (2 exp i) = 2 exp i -> bit i =1

Recorre uno a uno los bits de cada uno de los bloques, añadiendo la cadena de caracteres que corresponde en cada caso según esté a 1 ó a 0 el bit asociado.

FORMULARIO MENSAJES

Este formulario muestra por pantalla los últimos mensajes de alarma y respuesta recibidos por el PC. Para ello dispone de dos largas cadenas de caracteres que muestra a través de sendos controles TextBox. El formulario aparece en la figura 5.2.2.3.

Los mensajes están almacenados en dos tablas de 32 cadenas de caracteres cada una, que constituyen listas circulares.

Form_Load

Este procedimiento se ejecuta cada vez que se carga el formulario, y es el encargado de ensamblar todos los mensajes de respuesta y de alarma en dos cadenas de caracteres que presenta a través de sendos controles TextBox.

Para ello añade uno a uno, de forma decreciente, todos los mensajes de alarma y respuesta desde la posición del que sería el siguiente mensaje a ser introducido.

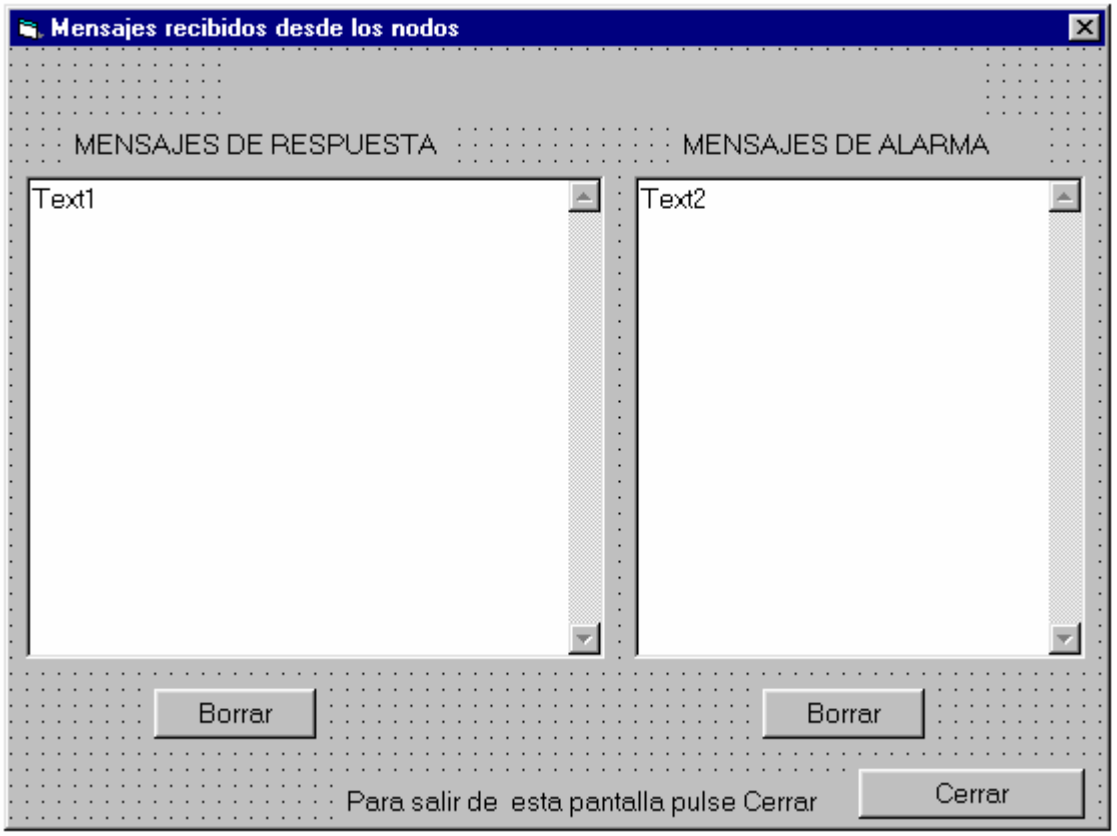


Fig. 5.2.2.3

FORMULARIO LECESC

Este formulario presenta el cuadro de diálogo que permite al usuario realizar operaciones de lectura y escritura sobre la memoria local de los nodos del sistema.

Para ello configura tramas en las que encapsula los paquetes de aplicación correspondientes a las instrucciones de lectura y escritura que el usuario haya decidido hacer. Como es de suponer, una trama sólo puede contener instrucciones dirigidas a un mismo nodo, ya que es en la cabecera de la trama donde se incluyen las direcciones, en concreto la dirección del nodo destino.

El formulario es el que aparece en la figura 5.2.2.4.

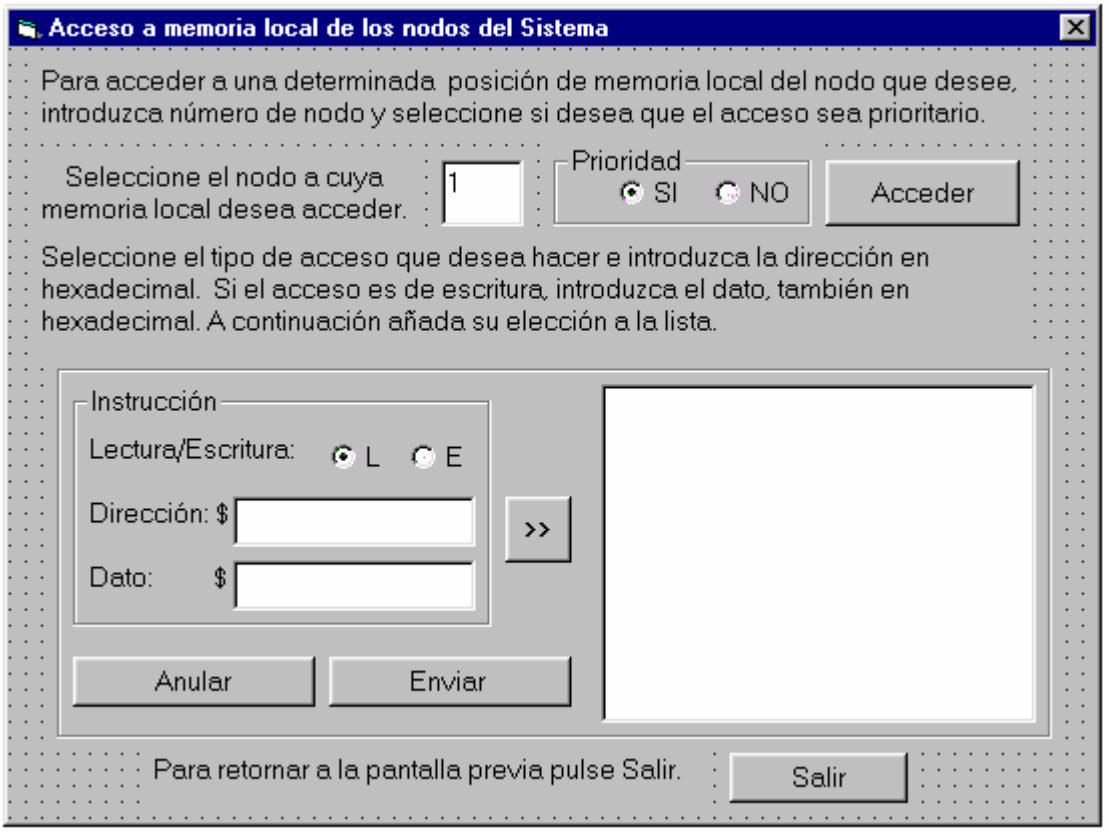


Fig.5.2.2.4

Command1_Click

Este procedimiento chequea la elección de nodo que ha hecho el ususario, de modo que lee el número que este ha introducido como cadena de caracteres a través del control TextBox. Lo convierte a formato numérico y comprueba que sea menor o igual que el número de nodos del sistema dando un mensaje de error en caso contrario.

Una vez ha comprobado la correcta elección del nodo, configura la cabecera de la trama que contendrá las futuras instrucciones de lectura y escritura. Lo hace de modo que la Dirección de Origen es $00 (PC) y la de Destino es la del nodo seleccionado, con lo cual basta con hacer 1º byte de trama = numero de nodo.Para el resto de los campos de la cabecera, asigna valores fijo:

- TP = 1; Esta trama no debe volver al PC. Si lo hace este deberá eliminarla

- ASE = 1; Requiere asentimiento, ya que lleva información relevante para el usuario.
- PRIOR = Según selección del usuario; Comprueba el valor seleccionado (Controles Option).
- ID num de trama = 00000; Esto es provisionalmente hasta que sea introducida en la lista de espera.
- TIPO = 000; Trama de información.
- Longitud del campo de datos = 00000; Provisionalmente hasta que los datos sean introducidos.

Command2_Click

Este procedimiento permite introducir un nuevo paquete de apliacción con la instrucción seleccionada en el campo de datos de la trama que está siendo configurada.

Para ello lee los datos que el usuario ha introducido como una cadena de caracteres, y llama a las funciones globales encargadas de convertir esos datos a un formato numérico adecuado. Comprueba también que estos datos son coherentes y da un mensaje de error en caso contrario.

Para introducir los paquetes, comprueba que hay espacio suficiente en el campo de datos de la trama (contador de datos + num. bytes instrucción <= 27), configura estos adecuadamente (código inst. dirección, dato en caso de escritura) y los copia en el campo de datos de la trama, actualizando los contadores correspondientes (número de instrucciones de lectura introducidos, número de datos de trama introducidos).

Conviene reseñar que, pese a haber espacio suficiente para ello, no es posible enviar más de 6 instrucciones de lectura consecutivas, esto es, encapsuladas en una misma trama. Ello es porque la respuesta a esas instrucciones de lectura (instrucciones de dato leído) requiere 1 byte más y el sistema está diseñado para que todas las instrucciones encapsuladas en una misma trama tengan su respuesta también en una única trama. Por este motivo, la restricción la introducen las respuestas y no las propias instrucciones de lectura.

FORMULARIO MODIFICAR

Este formulario presenta los cuadros de diálogo que permiten al usuario modificar los parámetros de ejecución de las aplicaciones locales de control de los nodos, así como transferir programas a memoria local RAM de estos. El formulario aparece en la figura 5.2.2.5 .

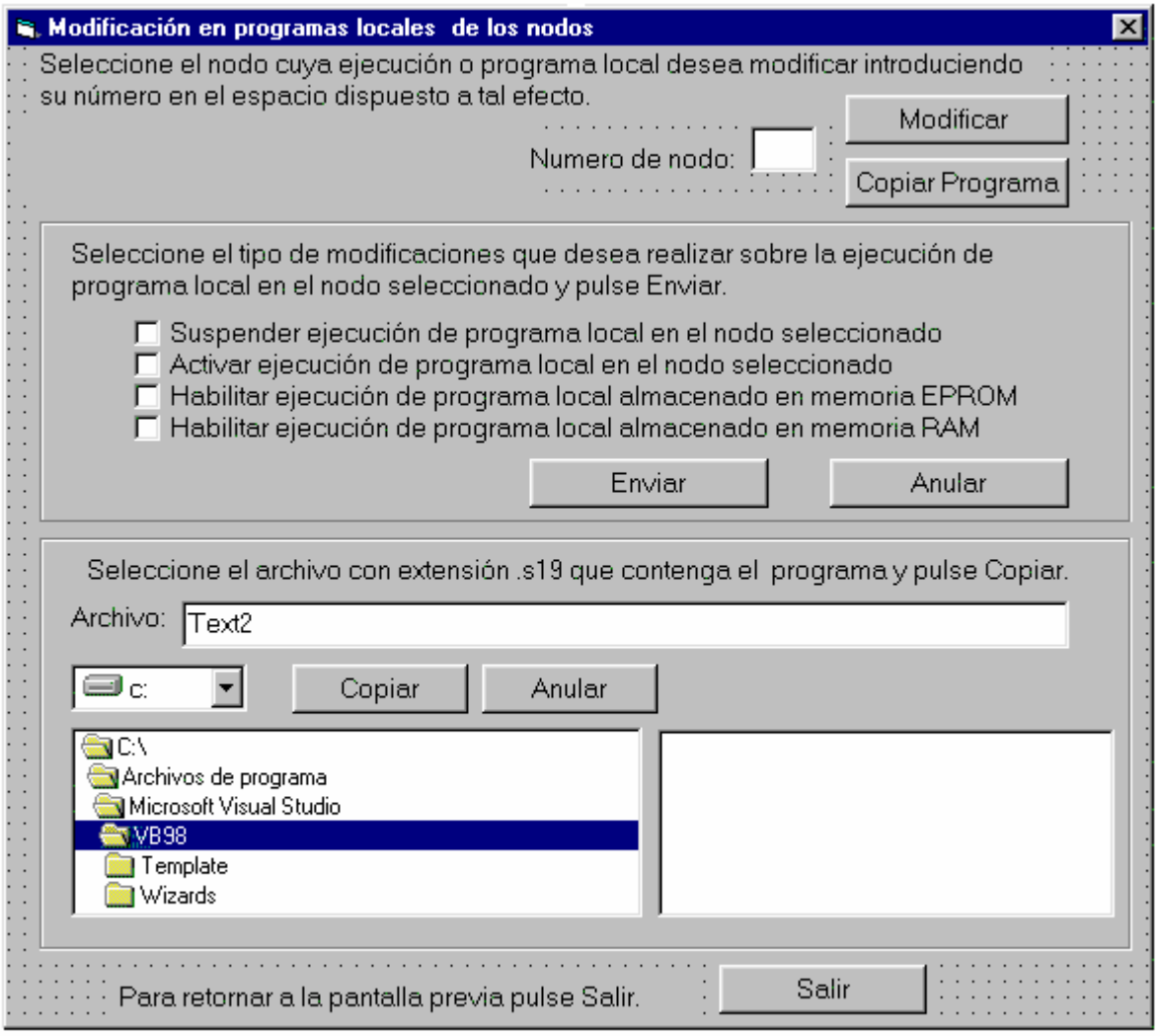


Fig. 5.2.2.5

La parte correspondiente a la modificación de los parámetros de ejecución es relativamente sencilla, y discusión sobre la configuración de la trama correspondiente es similar al formulario anterior. Nos centraremos en la transferencia de programas.

Command6_Click

Este procedimiento se encarga de configurar la(s) trama(s) necesaria(s) para transferir el programa seleccionado a memoria local RAM del nodo que se desee.

El código de programa se encuentra almacenado en formato s19 de Motorola (Ver anexo correspondiente) en el archivo correspondiente cuyo path lo determina el propio control FileBox automáticamente. El procedimiento opera del siguiente modo:

Abre el fichero de texto '.s19' seleccionado, donde se encuentra el programa en formato s19 de Motorola. Este fichero contendrá una serie de líneas de extensión máxima habitual 16 bytes de datos. Estos datos tienen que ser almacenados consecutivamente en memoria a partir de la dirección indicada en el campo Direccción de la línea correspondiente. Puede ser que dos ó más líneas distintas contengan datos consecutivos. El procedimiento se encarga de detectar esta situación, de modo que 'ensambla' todas las líneas en esta situación en una sola línea, con Dirección la de la primera de ellas y longitud la suma de todas.

De este modo se logra tener el programa en tantas líneas de texto como directivas de ensamblador ORG tuviese el código fuente original. Ello hace que el tratamiento y encapsulado de los datos sea más sencillo y eficiente.

Por cada línea se tienen los siguientes datos: Dirección de comienzo en formato numérico, Longitud en formato numérico (num bytes). Los datos se tienen en formato texto, de modo que hay que convertirlos a formato numérico antes de introducirlos en las tramas, como se hizo previamente con la Dirección y la Longitud.

Para introducir los datos se ejecuta el algoritmo cuyo diagrama de flujo aparece en la figura 5.2.2.6. Para ello hay que tener en cuanta que una instrucción de copia necesita:

- 1 byte de código de instrucción (0000 1000 ó 0000 1011)
- 1 byte de longitud del campo de datos
- 2 bytes de dirección de comienzo
- n bytes de datos

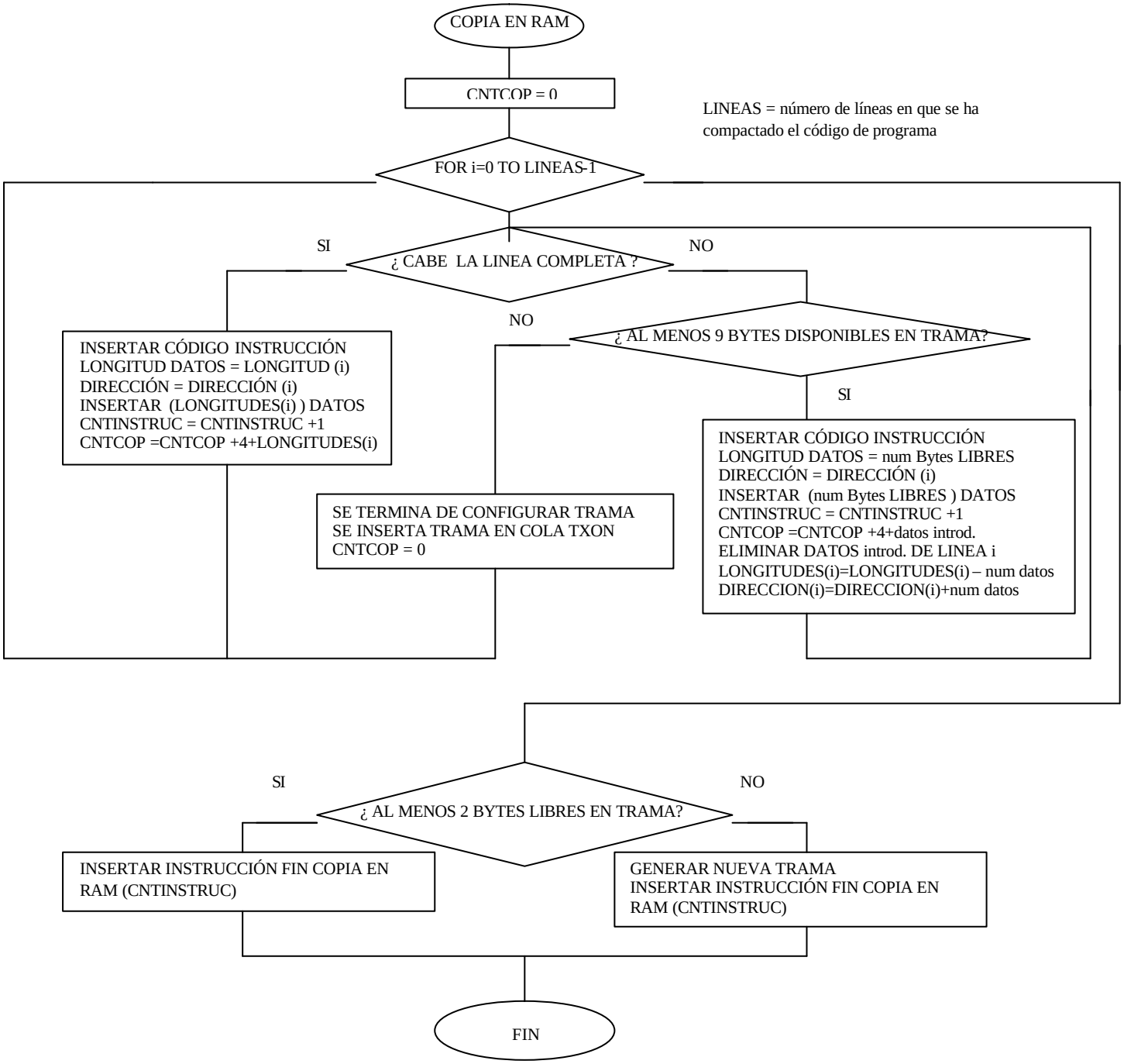


Fig. 5.2.2.6

Las tramas que contienen instrucciones de copia de programa son enviadas sin prioridad, ya que generalmente implican un tráfico denso que en caso de ser prioritario puede ocasionar retardos significativos en la transmisión de información crítica en el sistema por parte de los nodos o el propio PC.

La trama que contiene sin embargo la primera instrucción de copia si que es prioritaria, a que a su recepción en los nodos es cuando se inicializan todos los contadores relativos a la transferencia del programa.

PRINCIPALES DIFERENCIAS CON EL CÓDIGO DE LOS NODOS

Como ya se ha comentado en párrafos anteriores, el programa desarrollado en Visual Basic para PC es en esencia similar al código de Protocolo comentado para los nodos , con la salvedad de que está escrito en un lenguaje de mucho más alto nivel que el ensamblador, lo que hace más fácil su comprensión para cualquier programador que desee conocer el código.

Es por esta razón por la que la discusión hecha en este apartado no ha sido tan amplia como la del código ensamblador, ya que es de suponer que con los comentarios que incluye el código es suficiente.

Sin embargo, el código para el PC presenta ciertas diferencias respecto al código ensamblador. Estas diferencias se asocian al carácter centralizador del nodo implementado en este. Dicho carácter le atribuye funciones de control y gestión de red de la que los nodos carecen, al tiempo que le hace incluir facilidades adicionales de autodiagnóstico del sistema.

Estas variantes pueden resumirse en las siguientes:

Autodiagnósticos propios del PC

- Configuración del sistema. Cuando el PC inicia el proceso de configuración del sistema enviando una trama de inicialización a los nodos activa un timer de espera. Si antes de que venza ese timer el PC no recibe la trama que envió de vuelta, así como todas las tramas de registro de cada uno de los nodos que integran el sistema, entonces el proceso de configuración se aborta al darse por fallido.

Existe una variable global booleana 'Sistema.Configurado' que indica cuando el sistema está siendo o ha sido configurado. Mientras está siendo configurado, los procedimientos que analizan las tramas recibidas son especiales, y permiten almacenar la información del sistema de forma ordenada.

- Caída de nodo. Un nodo se considera que ha caído cada vez que el PC intenta comunicarse con él sin éxito. Se considera que la comunicación ha fracasado cuando tras 5 retransmisiones de una trama, no se recibe asentimiento de esta. Ello obliga a desechar la configuración actual del sistema, así que el PC aborta dicha configuración y muestra un mensaje de error por pantalla para que el operador tome las acciones oportunas.

  Cada vez que una trama es insertada en cola de transmisión, el PC incrementa el valor del contador de retransmisiones de dicha trama, y al detectar que llega a un valor determinado se dispara el mecanismo de protección.

Gestión de red

El PC controla que no haya tramas perdidas circulando por la red. Para ello comprueba el valor del bit TP (Trama Perdida) de cada una de las tramas que recibe. Si dicho bit está a 1, simplemente desecha la trama, mientras que si está a 0 lo pone a 1.

Gracias a este mecanismo se evita que tramas que han sufrido un error en el campo de dirección destino circulen indefinidamente por la red y terminen por congestionarla.

6. MANUAL DE USUARIO

Como se ha visto a lo largo de la descripción del Protocolo de Comunicación, es posible que los nodos que componen el Sistema de Control Multipunto intercambien información entre si. Esta información que intercambian los nodos es en principio inaccesible para el usuario, al que no se le presumen conocimientos sobre sistemas digitales.

El programa Sistema Control para la gestión del Sistema de Control Multipunto permite al usuario, de manera transparente e intuitiva configurar, gestionar y obtener información del sistema de forma centralizada a través de un PC. De este modo, los nodos, que pueden estar separados entre sí por una distancia apreciable, pueden ser gestionados unitariamente desde un mismo puesto de control, el PC.

Tal forma de control centralizado no es incompatible con que dichos nodos sean completamente autónomos en lo que respecta a la aplicación local que ejecutan, es decir, no requieren de la intervención del usuario para funcionar normalmente. Sin embargo admiten dicha intervención para cambiar los parámetros de ejecución o simplemente para que el usuario pueda obtener información de los nodos si ello fuera preciso.

El programa, además de permitir la comunicación entre nodos, presenta una interfaz con el usuario intuitiva y sencilla. Dicha interfaz es susceptible de actualizaciones que se adapten a aplicaciones concretas que puedan ejecutarse en los nodos. Ello es posible porque esta ha sido diseñada de la forma más genérica posible, de modo que pueda admitir aplicaciones de diversa naturaleza.

Por último el programa actúa, a efectos de comunicación, como lo hace el código del Protocolo de Comunicación que se ejecuta en el resto de los nodos, sirviendo de " repetidor " de aquellos mensajes que no van a él dirigidos sino de un nodo a otro. Además realiza ciertas funciones de control de tráfico en la red, así como de autodiagnóstico del anillo.

A continuación se presenta el funcionamiento " externo " del programa, al tiempo que se detallan las funciones adicionales presentes en el funcionamiento " interno " del mismo, que son específicas del nodo implementado en el PC.

FUNCIONAMIENTO DEL PROGRAMA A NIVEL USUARIO

Para arrancar el programa basta con hacer clic sobre el icono del fichero ejecutable Sistema de Control, o, bajo el directorio raíz del CD-ROM ejecutar " Programa\SistemadeControl.exe " .

Configuración del Sistema

Inmediatamente después de arrancar el programa, un cuadro de diálogo aparecerá ante el usuario. Este cuadro de diálogo se muestra en la figura 6.1 .

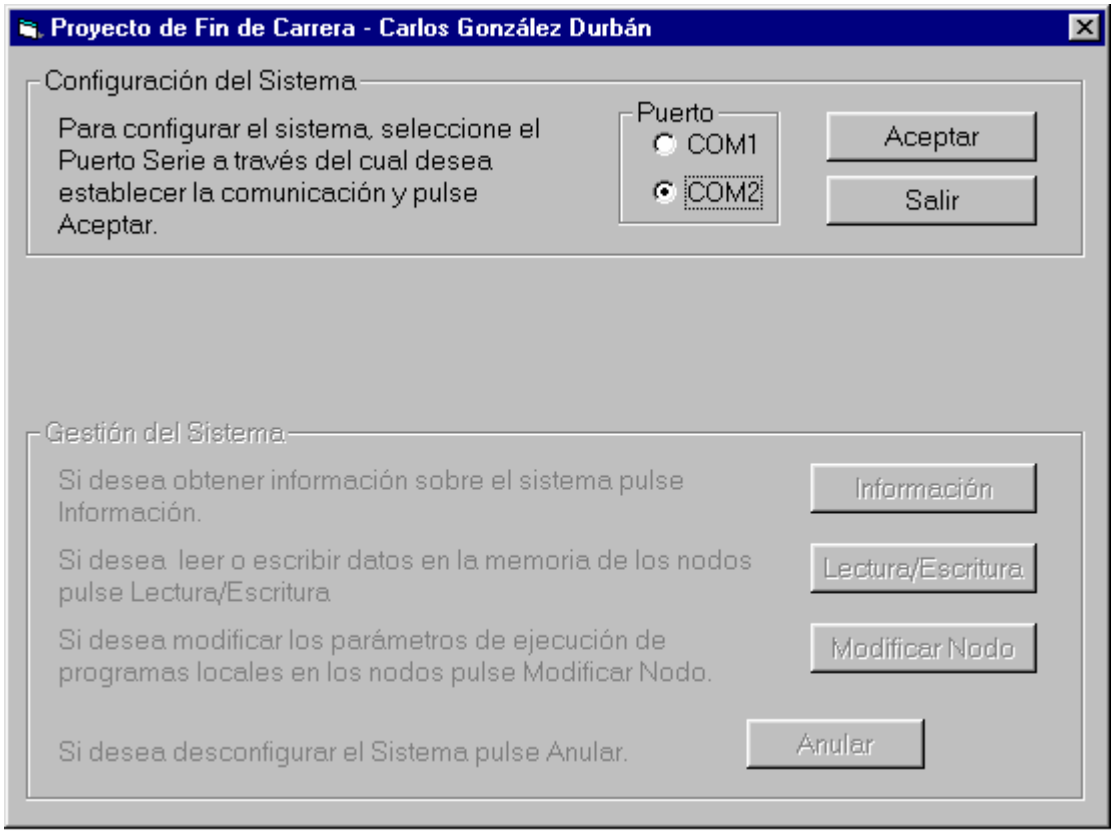


Fig. 6.1

El puerto de comunicaciones serie que aparece seleccionado por defecto es COM2, por lo que, a no ser que el usuario haga otra selección, ese será el puerto a través del que se establezca la comunicación con el sistema. Los parámetros de la comunicación son: 9600 baudios, y tramas de 8 bits de datos, 1 bit de stop, sin paridad.

Para configurar el sistema el usuario debe seguir los siguientes pasos:

- Seleccionar el puerto de comunicaciones serie que desee.
- Conectar el conector RS-232 del sistema a la salida del puerto serie seleccionado.
- Pulsar el botón Aceptar.

Una vez se haya pulsado Aceptar, el proceso de configuración habrá comenzado, y el cuadro de diálogo tomará el aspecto mostrado en la figura 6.2 .

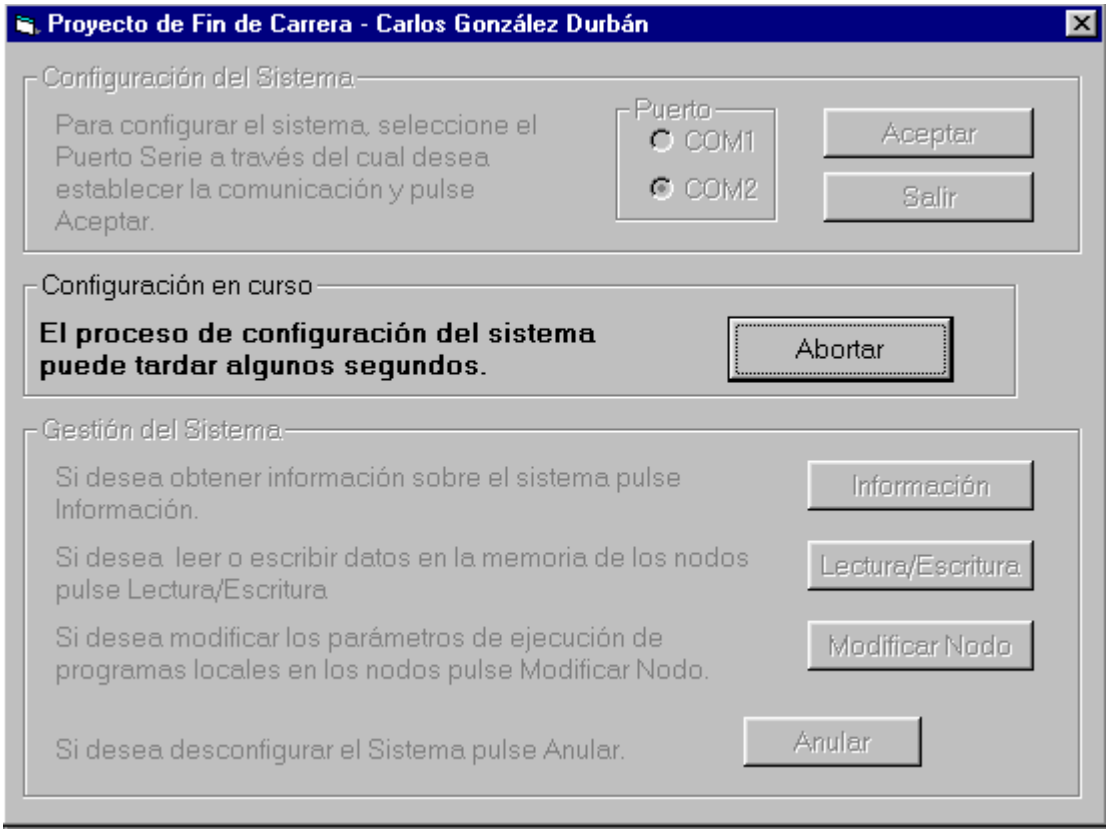


Fig. 6.2

El proceso de configuración es totalmente automático, y durante el mismo, el programa recibe información detallada sobre los nodos que integran el sistema. Una vez finalizado dicho proceso el sistema estará configurado, y funcionando correctamente.

NOTA: Si tras cierto periodo de tiempo, no se ha logrado configurar el sistema, el programa volverá al diálogo inicial, mostrando un mensaje de error por pantalla. La causa más probable de error es la presencia de algún problema en las conexiones, por lo que se recomienda revisar las mismas e iniciar un nuevo proceso de configuración.

Una vez que el sistema está configurado, un nuevo cuadro de diálogo aparecerá en pantalla. Este cuadro, que aparece en la figura 6.3, ofrece al usuario varias posibilidades, en función del tipo de acción que quiera ejercer sobre el sistema. Estas posibles acciones son: obtener información sobre el sistema, leer o escribir en posiciones de memoria de los nodos, o modificar la aplicación que en estos se ejecuta localmente.

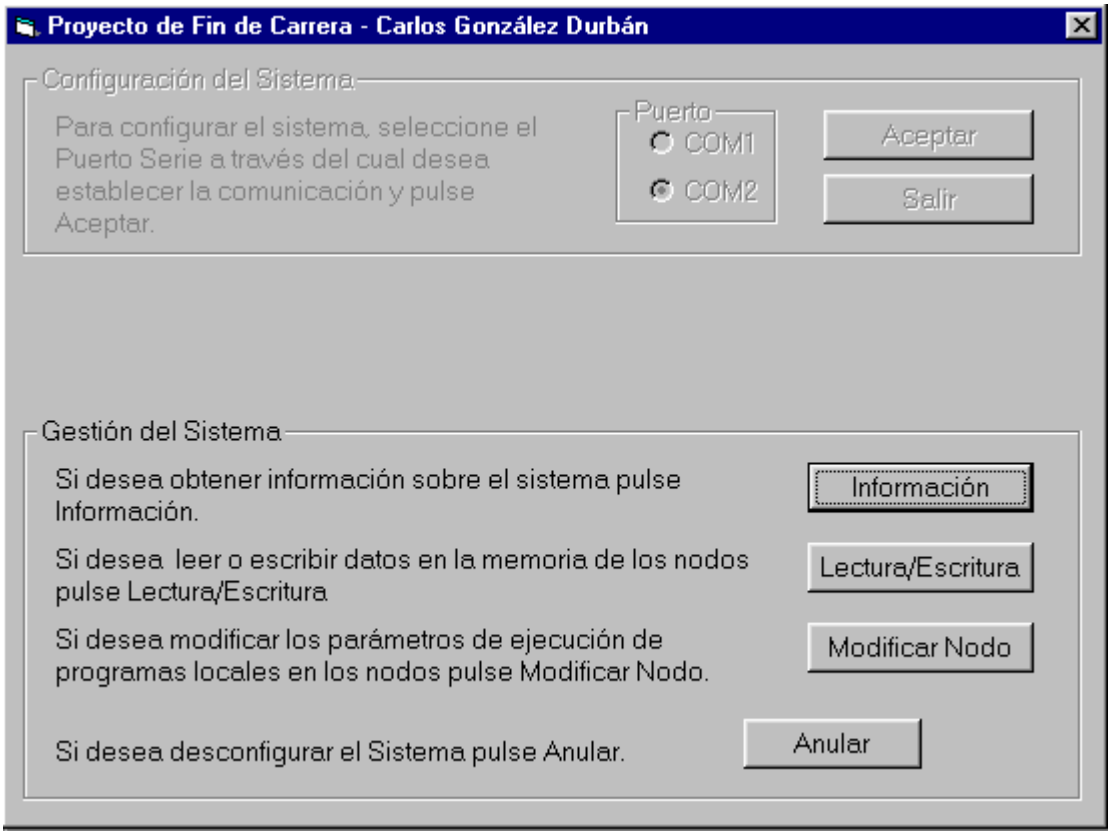


Fig. 6.3

Para acceder a cada una de las posibilidades, basta con pulsar el botón correspondiente.

Si por el contrario se desea desconfigurar el sistema, habría que pulsar el botón Anular, en cuyo caso, el programa cerrará el puerto de comunicaciones serie a través del que se establecía la comunicación con los nodos, y devolverá al usuario al diálogo inicial.

Información del Sistema

El usuario, al disponer de esta opción, puede acceder a la información que se ha obtenido del sistema durante el proceso de configuración del mismo, así como mensajes que hayan llegado de los nodos durante el tiempo que el sistema haya estado configurado.

Cuando el usuario acceda a la opción de Información, el cuadro mostrado en la figura 6.4 aparecerá en pantalla. Permite obtener la siguiente información:

- Información sobre los nodos. Esta pantalla muestra a través del cuadro de texto el número de nodos que componen el sistema, y permite visualizar así mismo información sobre los pines de cada uno de los nodos del sistema (pines activos, función y direccionalidad). Para ello basta con seleccionar el nodo deseado y pulsar el botón Detalles.

  Tan sólo aparecerá información de aquellos pines activos en el nodo. Se entiende que aquellos pines que no figuren en el listado no están activos en el nodo seleccionado.

- Mensajes de alarma y respuesta. El botón Mensajes abre una nueva pantalla que muestra los mensajes de alarma y respuesta que el PC ha recibido.

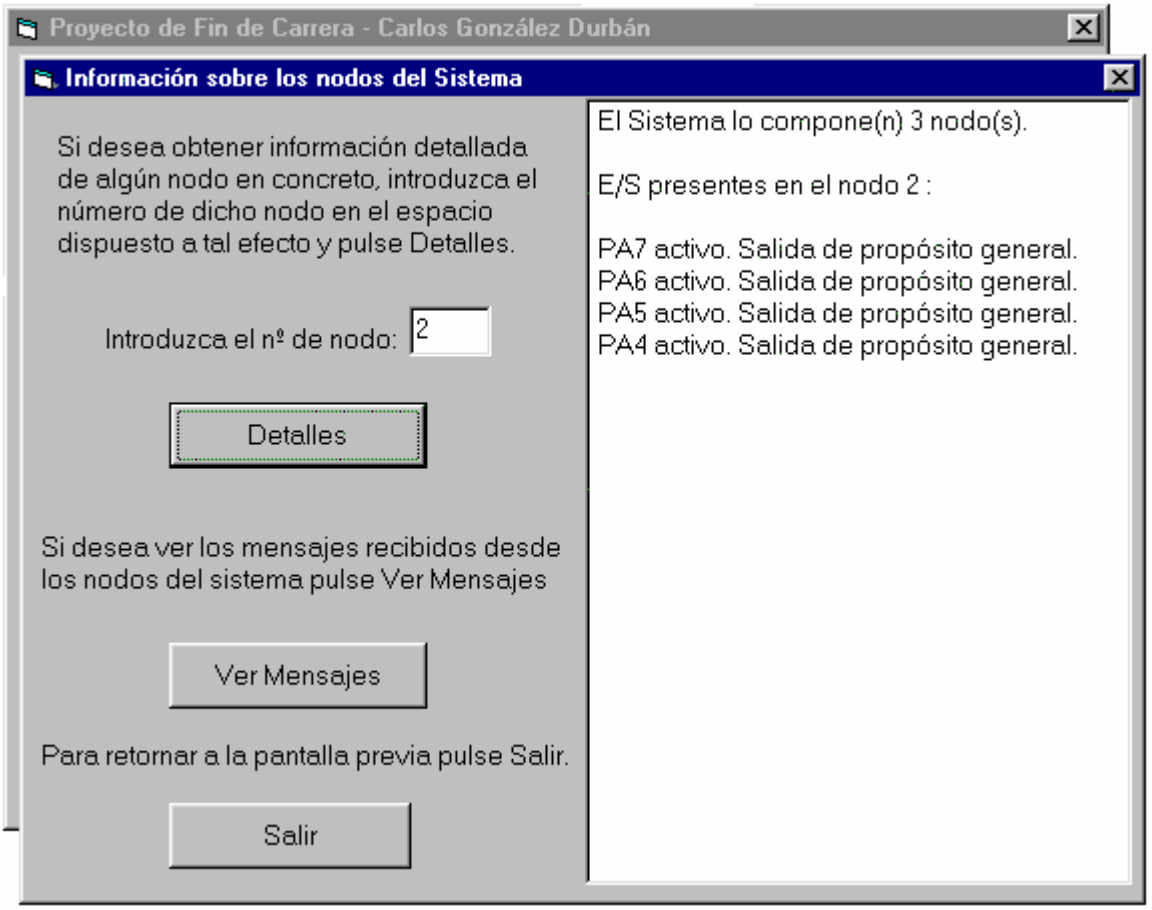


Fig. 6.4

Mensajes de alarma y respuesta

Veíamos en los párrafos anteriores que desde la pantalla de información podía abrirse una pantalla que muestra los mensajes de alarma y respuesta. Esta pantalla, que aparece en la figura 6.5, se abre también de manera automática cada vez que un (os) nuevo(s) mensaje(s) de alarma o respuesta llega(n) desde algún nodo.

El sistema tiene capacidad para almacenar hasta 32 mensajes de cada tipo. Los últimos mensajes son los guardados, de modo que cuando no hay capacidad para más mensajes, son los antiguos los que se pierden.

El formato de los mensajes de respuesta puede ser configurado por el usuario para que tome el formato deseado. Actualmente toma el siguiente:

- Nodo Origen: nodo que emitió el mensaje
- Respuesta a lectura de dato

  Dirección: posición de memoria del dato leído

  Dato: Valor almacenado en la posición de Dirección.
- Respuesta a copia de programa

  "Instalación de Programa local en RAM completada con éxito"

Los mensajes de alarma son textos de 27 caracteres como mucho, de modo que al ser mostrados adoptan el siguiente formato:

- Nodo Origen: nodo que emitió la alarma
- Texto de alarma

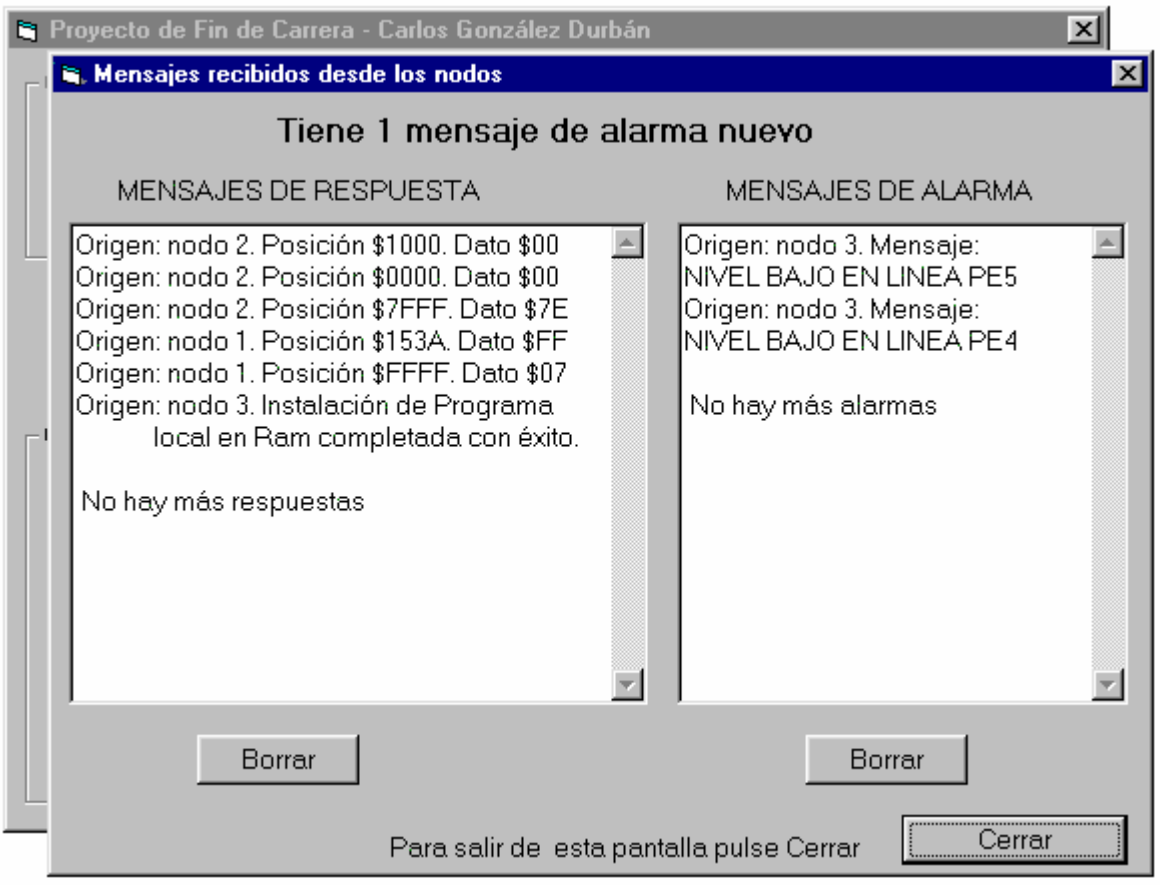


Fig. 6.5

Cuando la pantalla aparece automáticamente tras la recepción de nuevos mensajes, el sistema informa sobre el número y tipo de los mensajes que acaban de recibirse.

Los mensajes pueden ser borrados de la memoria sin más que pulsar los botones Borrar asociados a cada tipo de mensaje.

Lectura / Escritura

Esta opción es la que permite al usuario acceder individualmente a cualquier posición del mapa de memoria de los nodos, ya sea en lectura (todo el mapa) o en escritura (RAM).

La pantalla correspondiente aparece en la figura 6.6 (selección de órdenes). Esta presenta 2 cuadros de diálogo, para selección de nodo y selección de órdenes.

Selección de nodo:

Introducir el número de nodo que ha de ser menor o igual al número de nodos del sistema. En caso contrario el programa da un mensaje de error.

Seleccionar la prioridad que desea para la trama que encapsulará sus órdenes de lectura/escritura

Para acceder al diálogo de selección de órdenes pulsar Acceder.

Selección de órdenes:

Seleccionar el tipo de orden (lectura/escritura) deseada. Introducir la dirección y el dato en hexadecimal (el programa da error de formato si no se hace así) y pulsar '>' para añadir la orden. Las órdenes de escritura sólo se pueden hacer en

posiciones RAM ($0000 a $7FFF) de modo que el programa da error si se intenta acceder en escritura fuera de este rango.

Pulsar Enviar para transmitir la trama que contiene todas las órdenes dadas. Si se ha dado alguna orden de lectura se recibirá un mensaje de respuesta de forma casi inmediata.

Sólo es posible realizar hasta un máximo de 6 operaciones de lectura consecutivas, y hay un número máximo de ordenes a realizar que depende del tipo de estas. El sistema avisa cuando ya no es posible añadir nuevas órdenes. En ese caso basta con enviar las ordenes ya introducidas y configurar una nueva trama para el resto.

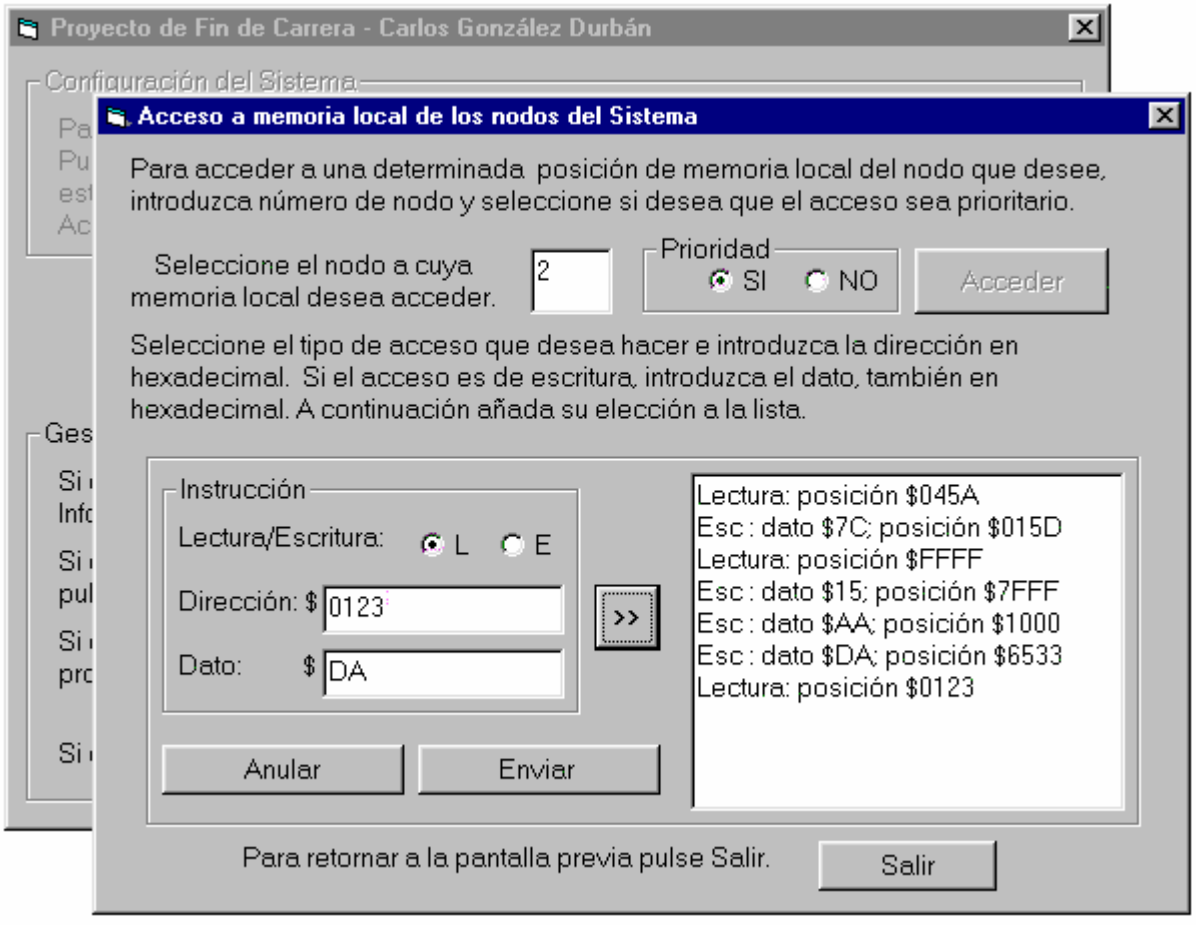


Fig. 6.6

Modificar Nodo

Esta opción dota al usuario de un absoluto control sobre la aplicación local de control ejecutada en cada nodo en términos de elección de programa y estado de ese (activado / desactivado).

Esta pantalla consta de tres cuadros de diálogo diferentes: elección de nodo y operación, modificación de parámetros de ejecución y transferencia de programas.

Elección de nodo y operación.

Permite seleccionar el nodo sobre el que actuar y el tipo de operación a realizar sobre este (modificación parámetros de ejecución ó transferencia de programa). El usuario debe introducir el número de nodo en el espacio correspondiente, y pulsar sobre el tipo de operación deseado, lo que activará el cuadro de diálogo asociado.

Modificación de los parámetros de ejecución.

Permite seleccionar la orden deseada y enviársela al nodo seleccionado. Basta con marcar la(s) opción(es) deseada(s) y pulsar Enviar. La pantalla correspondiente aparece en la figura 6.7 .

Las opciones son las siguientes:

- Activar ejecución de aplicación local: El nodo ejecuta la aplicación local que tenga seleccionada.
- Suspender ejecución de aplicación local: El nodo no ejecuta ninguna aplicación local. Selección por defecto al configurar el sistema
- Habilitar ejecución de programa local almacenado en EPROM. La aplicación seleccionada es aquella almacenada en EPROM. Selección por defecto al configurar el sistema

- Habilitar ejecución de programa local almacenado en RAM. La aplicación seleccionada es aquella almacenada en RAM. Previamente ha tenido que ser transferido algún programa a memoria RAM. En caso contrario se produciría un error en el nodo correspondiente.

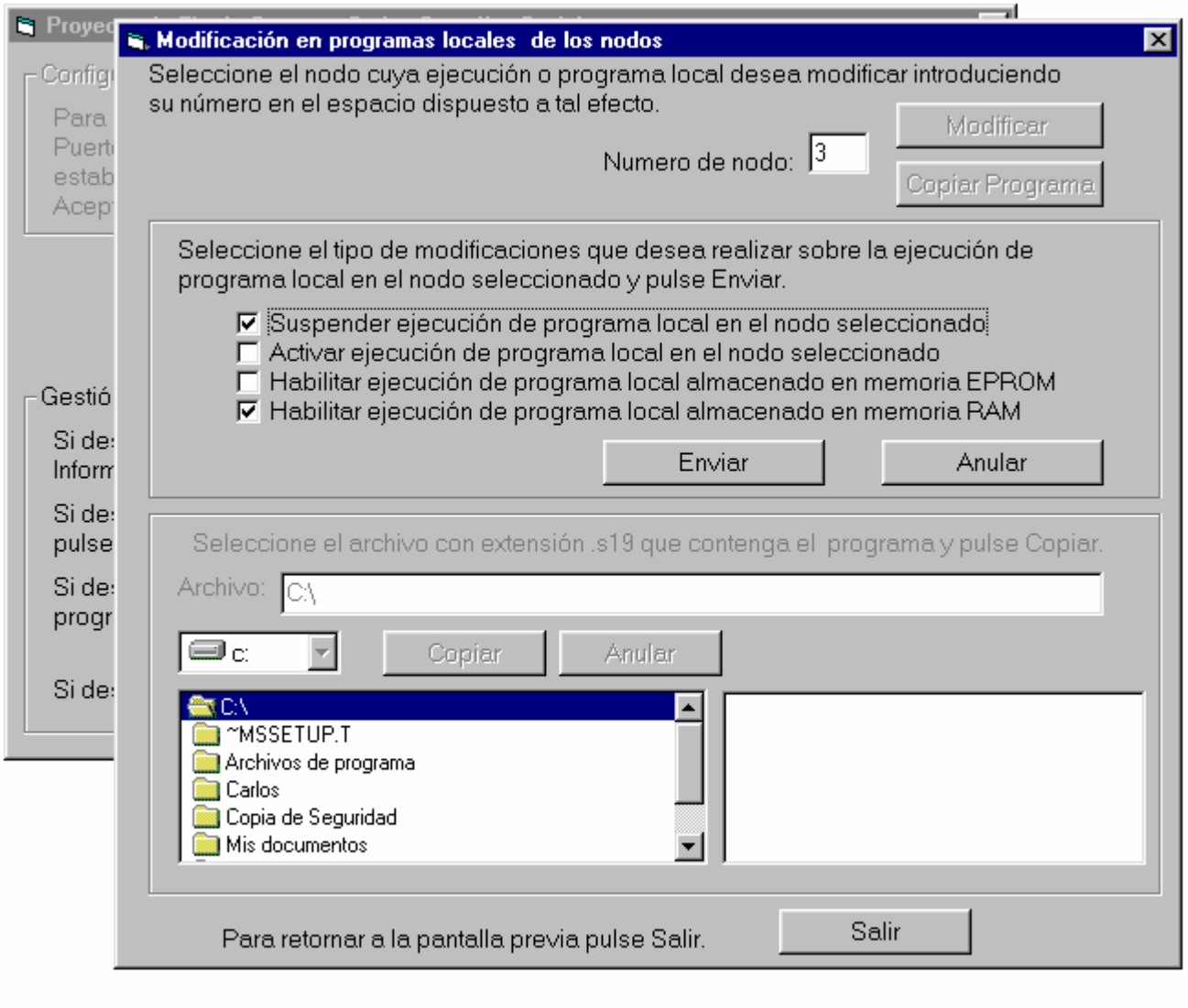


Fig. 6.7

Transferencia de programa.

El usuario puede transferir programas a los nodos para que estos sean ejecutados como aplicación local de control en estos. Ello es posible gracias a los controles

que presenta la pantalla de la figura 6.8 . Los programas a transferir deben estar en formato s19 de Motorola, almacenados en ficheros '.s19'.

Para transferirlos basta con seleccionar el fichero correspondiente y pulsar Copiar. Pasados unos segundos aparecerá por pantalla un mensaje de respuesta del nodo correspondiente en el que se indicará que la transferencia de programa ha concluido con éxito, tras lo cual el usuario podrá habilitar la ejecución de programa local almacenado en RAM.

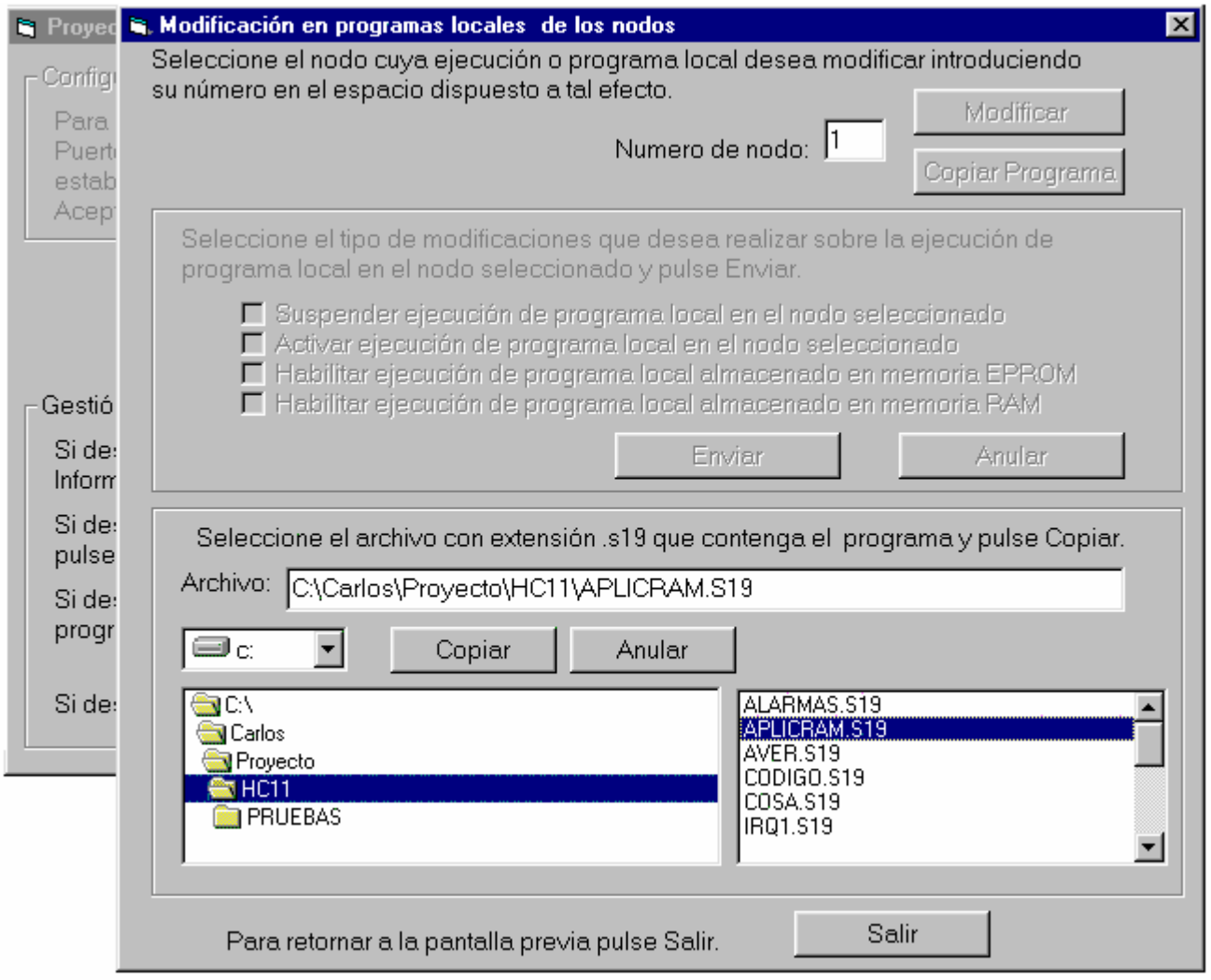


Fig. 6.8

CONSEJOS PRÁCTICOS

Para el correcto funcionamiento del sistema, conviene seguir los siguientes consejos:

- Cuando se trata de configurar el sistema por primera vez después de que este haya estado sin alimentar durante cierto tiempo suele requerir ciertos intentos. Para que nuestro sistema funcione perfectamente es conveniente realizar varias operaciones de lectura en cada nodo. Puede que al principio los nodos no respondan adecuadamente o el PC genere respuestas no deseadas. Este fenómeno desaparece al realizar las lecturas reseñadas.

- Antes de activar la ejecución de aplicación local conviene habilitar la aplicación deseada (EPROM o RAM) para evitar fenómenos indeseados.

- A la hora de transferir programas a memoria RAM asegúrese de que la ejecución está suspendida o bien está activa pero la aplicación seleccionada es EPROM.

- Si desea activar pines de salidas digitales o leer pines de entradas digitales no es necesario que escriba un programa para ello, basta con leer/escribir directamente sobre los registros asociados a cada uno de los puertos.

Si sigue los consejos que acabamos de presentar podrá sacar obtener un gran rendimiento del sistema y le permitirá controlar y supervisar todos aquellos procesos que desee en su entorno de trabajo.

7. CONCLUSIONES

Tal y como se ponía de manifiesto en la declaración de objetivos del proyecto, este pretende ser un ensayo sobre la viabilidad del sistema propuesto. A este respecto, es obvio que el ensayo ha resultado un éxito, no sólo en lo que concierne a viabilidad, sino también a potencia y eficiencia.

La viabilidad ha quedado plenamente demostrada al haber sido implementado físicamente el sistema, constando este de 3 nodos, y funcionar perfectamente. El número de nodos implementados es suficiente para considerar el ensayo un éxito, ya que un posible aumento en el mismo tan sólo ocasionaría cambios en el tiempo de respuesta pero no perturbaría el funcionamiento del sistema en modo alguno.

El sistema cumple con todos los requerimientos de funcionalidad. Permite al usuario acceder en lectura y escritura a la memoria local de los nodos mientras estos ejecutan el programa local correspondiente. También es posible la comunicación desde los nodos al PC, sin necesidad de que este la solicite, lo cual proporciona al sistema capacidad para la gestión de alarmas.

En cuanto a seguridad, se ha dotado al sistema de las facilidades de autodiagnóstico necesarias para detectar cualquier anomalía así como de mecanismos de recuperación ante fallos y de protección frente a pérdida o duplicidad de tramas.

El estilo de diseño seguido se ha mantenido totalmente independiente de las posibles aplicaciones del sistema. Ello se ha traducido en un producto de una potencia muy elevada, de modo que permite satisfacer las necesidades de cualquier aplicación cuyos requerimientos temporales estén dentro del rango de nuestro sistema. A este respecto el sistema se caracteriza por:

- **Escalabilidad:** El sistema puede tener un número máximo de nodos cualquiera, sin más que incrementar la longitud de determinados campos en la trama.

- **Versatilidad:** El sistema permite ser adaptado fácilmente a cualquier aplicación de usuario, de modo que existe una reserva de instrucciones y tipos de trama a disposición de este.

- **Robustez:** Al tratarse de un sistema de control, este posee los mecanismos necesarios para liberar al usuario de las tareas de diagnóstico del propio sistema. Además se recupera automáticamente ante los fallos que puedan aparecer en las comunicaciones.

Para poner de manifiesto la versatilidad del sistema se ha añadido la facilidad de copia de programa en memoria local RAM de los nodos, así como de modificación de ejecución (Activa/Inactiva; EPROM/RAM) del programa local de estos. Ello, unido a la capacidad para visualizar el mapa de memoria de los nodos, convierten al sistema en una potente **Herramienta de Desarrollo** para el diseño y ejecución de cualquier otro tipo de aplicación.

En resumen, el resultado del proyecto ha sido plenamente satisfactorio y deja la puerta abierta a una amplia gama de posibilidades de uso y ampliación del mismo.

8. BIBLIOGRAFÍA

" Real-Time Computer Control, an Introduction" Stuart Bennet. Prentice Hall International.

"Communication Systems" , 4ª Ed. , S.Haykin. John Wiley & Sons, Inc.

"Redes de Comunicación. Conceptos fundamentales y arquitecturas básicas" Alberto León-García – Indra Widjaja. McGraw-Hill

"Data Networks" , Dimitri Bertsekas / Robert Gallager. Prentice Hall, Inc.

"Sistemas Digitales, Ingeniería de los microprocesadores 68000", Antonio Guerra - Colección ETSI de Telecomunicaciones (UPM). Editorial Centro de Estudios Ramón Areces, S.A.

"Serial Port Complete", Jan Axelson. Lakeview Research 1998.

ANEXOS

El presente capítulo de anexos incluye los siguientes documentos:

ANEXO I – FORMATO S19 DE MOTOROLA

ANEXO II – PROPIEDADES DEL NIVEL DE RED

ANEXO III – PROPIEDADES DEL NIVEL DE APLICACIÓN

ANEXO IV – CÓDIGO FUENTE ENSAMBLADOR

ANEXO V – CÓDIGO FUENTE VISUAL BASIC

ANEXO I

ARCHIVOS EN FORMATO-S MOTOROLA

Los archivos en Formato-S de Motorola son un medio de codificación de ficheros objeto binarios en forma de ficheros ASCII que contienen además información sobre direcciones y checksum para control de errores. Estos archivos pueden por lo tanto ser usados para transferencia de ficheros entre sistemas. Este formato es usado por el programa GESTIÓN DEL SISTEMA DE CONTROL MULTIPUNTO para la transferencia de programas a memoria RAM local en los nodos del sistema.

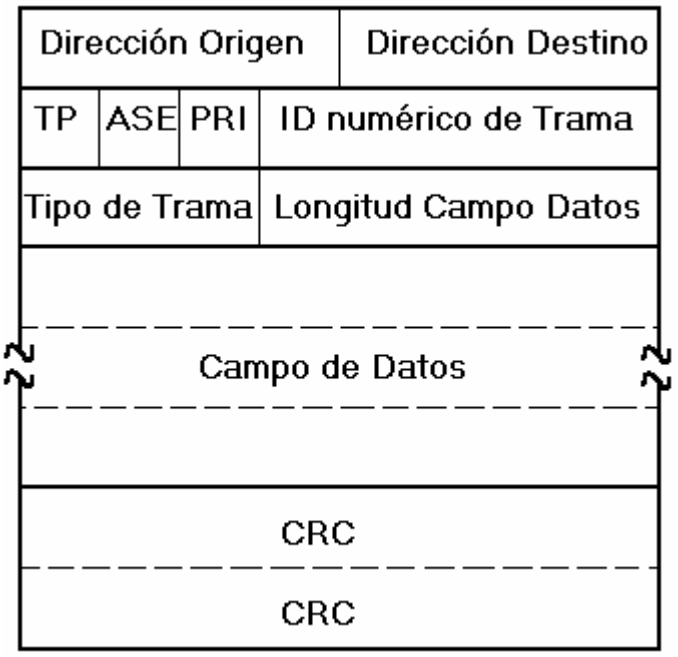
S	es simplemente el código ASCII del carácter ‘ S ’.
n	es el tipo de línea: 0 = línea de cabecera; contiene el nombre del archivo. 1 = línea de datos para direcciones de 16 bits. 2 = línea de datos para direcciones de 24 bits. 3 = línea de datos para direcciones de 32 bits. 7 = línea de fin de fichero para direcciones de 32 bits. 8 = línea de fin de fichero para direcciones de 24 bits. 9 = línea de fin de fichero para direcciones de 16 bits.
Longitud	indica el número de bytes (en hexadecimal) en la línea incluyendo dirección, datos y checksum.
Dirección	contiene la dirección de la posición de memoria del primero de los datos. El resto son consecutivos.
Datos	es el bloque de datos binarios en código ASCII. Cada línea contiene normalmente 16 bytes de datos, pero puede contener más o menos bytes.
Checksum	es el complemento a 1 de la suma de los campos Longitud, Dirección y Datos. En una transferencia de ficheros se usaría en el receptor para controlar la ocurrencia de errores durante la transmisión.

Cada línea es seguida por un retorno de carro y nueva línea . Los archivos que contienen información con este formato son aquellos con extensión .S19 .

ANEXO II

PROPIEDADES DEL NIVEL DE RED

Formato de Trama de nivel de red



Tipos de trama

TIPO DE TRAMA = 000 ; TRAMA DE INFORMACIÓN

TIPO DE TRAMA = 001 ; TRAMA DE ASENTIMIENTO

TIPO DE TRAMA = 010 ; TRAMA DE RECHAZO

TIPO DE TRAMA = 011 ; RESERVADO PARA USOS FUTUROS

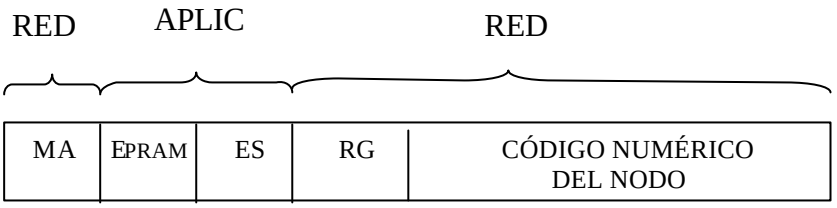
TIPO DE TRAMA = 100 ; TRAMA DE ALARMA

TIPO DE TRAMA = 101 ; TRAMA DE REGISTRO DE NODO

TIPO DE TRAMA = 110 ; TRAMA DE PETICIÓN DE CONFIGURACIÓN

TIPO DE TRAMA = 111 ; TRAMA DE INICIALIZACIÓN O IDENTIFICACIÓN

PRCN: Protocolo Registro de Control de Nodo



PRCC: Protocolo Registro de Control de Comunicación

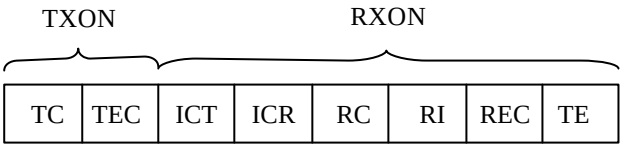
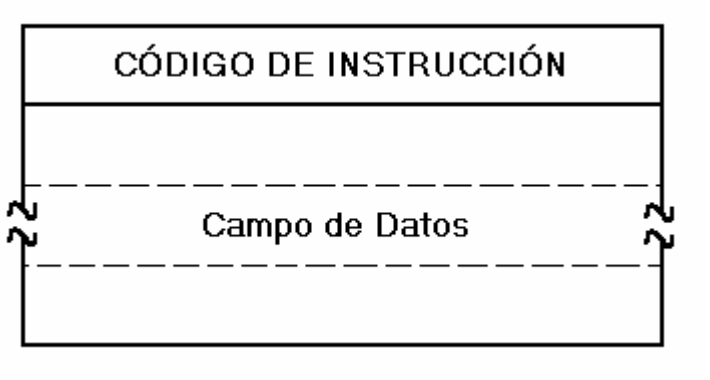


Fig 5.1.2.2

ANEXO III

PROPIEDADES DEL NIVEL DE APLICACIÖN

Formato de paquete de nivel de aplicación



Juego de instrucciones de nivel de aplicación

CÓDIGO DE INSTRUCCIÓN = 0000 0001 = 1 ; LEER DATO
CÓDIGO DE INSTRUCCIÓN = 0000 0010 = 2 ; ESCRIBIR DATO
CÓDIGO DE INSTRUCCIÓN = 0000 0100 = 4 ; EJECUTAR PROGRAMA EN RAM
CÓDIGO DE INSTRUCCIÓN = 0000 1000 = 8 ; COPIAR PROGRAMA EN RAM
CÓDIGO DE INSTRUCCIÓN = 0000 1001 = 9 ; FIN COPIAR PROG. EN RAM
CÓDIGO DE INSTRUCCIÓN = 0000 1010 = 10 ; PROGRAMA COPIADO EN RAM
CÓDIGO DE INSTRUCCIÓN = 0000 1011 = 11 ; PRIMERA COPIA EN RAM
CÓDIGO DE INSTRUCCIÓN = 0001 0000 = 16 ; EJECUTAR PROG. EN EPROM
CÓDIGO DE INSTRUCCIÓN = 0010 0000 = 32 ; SUSPENDER EJECUCIÓN
CÓDIGO DE INSTRUCCIÓN = 0100 0000 = 64 ; ACTIVAR EJECUCIÓN
CÓDIGO DE INSTRUCCIÓN = 1000 0000 = 128 ; DATO LEIDO

ANEXO IV

CÓDIGO FUENTE ENSAMBLADOR

```
*********************************************************************
************          Fichero: CODFUENTE.S           ****************
*********************************************************************

* Proyecto de Fin de Carrera - Carlos Gonzalez Durban – Sept. de 2002
* Titulo: " Dise¤o de un sistema de control multipunto a traves del
puerto
*           serie y del microcontrolador 68HC11 "

* El siguiente codigo fuente corresponde al Protocolo de Comunicacion
que
* permite el intercambio de informacion entre los nodos del sistema


*------------------*
*   Constantes     *
*------------------*

REGBS           EQU     $1000       ; Comienzo de registros de control
del micro
COPRST          EQU     REGBS+$3A   ; Registro de reset del COP (perro
guardian)
TCNT            EQU     REGBS+$0E   ; Registro contador para
temporizacion
TMSK2           EQU     REGBS+$24   ; Registro de mascaras 2 para
temporizacion
TFLG2           EQU     REGBS+$25   ; Registro 2 de estado para
temporizacion
SCCR1           EQU     REGBS+$2C   ; Registros de control del SCI
SCCR2           EQU     REGBS+$2D
BAUD            EQU     REGBS+$2B   ; Registro de control de velocidad
del SCI
SCSR            EQU     REGBS+$2E   ; Registro de estado del SCI
SCDR            EQU     REGBS+$2F   ; Registro de datos del SCI


RAMBS           EQU     $0100   ; Direccion de comienzo de codigo
memoria RAM
PILA            EQU     $7FFF   ; Direccion de comienzo de la pila
COLAB0          EQU     $0C00   ; Cola de transmision de tramas sin
prioridad
COLAB1          EQU     $0C80   ; Cola de transmision de tramas con
prioridad
COLAA0          EQU     $0D00   ; Cola de procesado de tramas sin
prioridad
COLAA1          EQU     $0D80   ; Cola de procesado de tramas con
prioridad
LISTAESP        EQU     $0E00   ; Lista de tramas en espera de
asentimiento
```

```
BASERESP        EQU     $0F00   ; Trama de respuesta
BASEALARM       EQU     $0F20   ; Trama de alarma


*---------------------------------------*
*   Variables globales del Protocolo    *
*---------------------------------------*
* Las siguientes variables son las que utiliza el protocolo de
comunicacion

                ORG     $0F40

PRCTEA          RMB     8       ; Informacion sobre el contenido de la
lista
PRCHB1          RMB     4       ; Informacion sobre COLAB1
PRCHB0          RMB     4       ; Informacion sobre COLAB0
PRIA1           RMB     1       ; Indice del primer elemento de COLAA1
ULTA1           RMB     1       ; Indice de la primera posicion libre
en COLAA1
PRIA0           RMB     1       ; Indice del primer elemento de COLAA0
ULTA0           RMB     1       ; Indice de la primera posicion libre
en COLAA0
PRIB1           RMB     1       ; Indice del primer elemento de COLAB1
ULTB1           RMB     1       ; Indice de la primera posicion libre
en COLAB1
PRIB0           RMB     1       ; Indice del primer elemento de COLAB0
ULTB0           RMB     1       ; Indice de la primera posicion libre
en COLAB0
STCOLAA1        RMB     1       ; Estado de COLAA1
STCOLAA0        RMB     1       ; Estado de COLAA0
STCOLAB1        RMB     1       ; Estado de COLAB1
STCOLAB0        RMB     1       ; Estado de COLAB0
PRCD            RMB     1       ; Contador de desbordamientos. Tiempo
global
CNTTX           RMB     1       ; Contador de bytes transmitidos de
una trama
CNTRX           RMB     1       ; Contador de bytes recibidos de una
trama
CNTPROC         RMB     1       ; Contador de bytes procesados de una
trama
CNTRESP         RMB     1       ; Contador de bytes de respuesta de
una trama
CNTCOP          RMB     1       ; Contador de instrucciones de copia
de programa
CONTCRC         RMB     1       ; Contador de bytes para el calculo
del CRC
PRCC            RMB     1       ; Registro de control de comunicacion
PRCN            RMB     1       ; Registro de Control del nodo de la
red
CNTCOPMAX       RMB     1       ; Contador tope de instrucciones de
copia
CNTTXMAX        RMB     1       ; Numero maximo de bytes en trama
transmitida
CNTRXMAX        RMB     1       ; Numero maximo de bytes en trama
recibida
CNTPROCMAX      RMB     1       ; Numero maximo de bytes en trama
procesada

CRCOK           RMB     1       ; Resultado del analisis del CRC de
una trama
```

```
BYTECRC         RMB     1       ; Byte cuyos bits se usan en el
calculo del CRC
STINCOL         RMB     1       ; Tipo de trama insertada en cola de
transmision
PRIMBYTE        RMB     1       ; Primer byte de trama recibida

TIMERBASE       RMB     8       ; Timers de las tramas en espera de
asentimiento
BASETX          RMB     2       ; Direccion base de transmision
BASERX          RMB     2       ; Direccion base de recepcion
BASEPROC        RMB     2       ; Direccion base de procesado
BASERTXON       RMB     2       ; Direccion base de retansmision
BASECRC         RMB     2       ; Direccion base de trama para calculo
CRC
TEXTOALARMA     RMB     2       ; Direccion base de texto de alarma

*-------------------------------------------------*
*   Variables globales de la Aplicacion (Eprom)  *
*-------------------------------------------------*
* Las siguientes variables son las que utiliza la Aplicacion en
memoria Eprom
* del usuario

                ORG     $1040

* EJEMPLO       RMB     3
* EJEMPLO       FCB     7
* EJEMPLO       FCC     ' ESTO NO ES MAS QUE UN EJEMPLO '
*               FCB     0


*---------------------------------*
*   Vector de interrupciones      *
*---------------------------------*

                ORG     $FFFE
                FDB     INICIO          ; Vectorizacion del reset

                ORG     $FFFA
                FDB     AVISARPC        ; Interrupcion de fallo en el
nodo

                ORG     $FFF6
                FDB     AVISARPC        ; Interrupcion de fallo en el
nodo

                ORG     $FFDE
                FDB     INTERRUPTIMER   ; Interrupcion de
desbordamiento TCNT

                ORG     $FFD6
                FDB     INTERRUPSCI     ; Interrupcion SCI



*-------------------------------------------------------*
*    Modulo Inicial (Configuracion de dispositivos)     *
*-------------------------------------------------------*
```

```
                ORG     $E000           ; Eprom de 8K comienza en
$E000

CLASENODO       FCB     0,0,0,0,0,0,0   ; ID de la naturaleza del nodo

INICIO          LDS     #PILA           ; Inicializacion de la pila

                LDAA    #$30            ; Configuracion del SCI a 9600
baudios
                STAA    BAUD
                LDAA    #$00            ; Formato de trama:
                STAA    SCCR1           ; bit start - 8 bits datos -
bit stop
                LDAA    #$2C
                STAA    SCCR2           ; Activacion SCI e
interrupciones SCI
                LDAA    #$80
                STAA    SCSR

                LDAA    #%10000011      ; Configuracion del contador
global e
                STAA    TMSK2           ; interrupciones del mismo

                LDX     #$0F40          ; Se inicializan las variables
globales
                LDAA    #1              ; del protocolo
                LDAB    #0

PRININIC        CMPA    #37             ; Lista de Espera y Colas de
Procesado y
                BGE     FININIC         ; Transmision vacias
                STAB    0,X             ; Inicializacion de PRCC a $00
es decir,
                INX                     ; no hay comunicaciones
activas
                INCA                    ; Contadores de
RX,TX,Procesado y Copia
                BRA     PRININIC        ; inicializados a $00

FININIC         LDAB    #47             ; A los delimitadores de fin
de RX,TX,
                STAB    0,X             ; Procesado y Copia, se les da
un valor
                LDAB    #255            ; basura de seguridad
                STAB    1,X
                STAB    2,X
                STAB    3,X
                STAB    4,X

                LDAA    #%00101111      ; Inicializacion PRCN : nodo
inactivo
                STAA    PRCN            ; codigo en EPROM, ejecucion
suspendida

                CLI                     ; CCR (I=0), activacion
interrupciones



*-------------------------*
```

```
*    Modulo Principal     *
*--------------------------*

PRINCIPAL       LDAA    #$55        ; Reset COP (perro guardian)
                STAA    COPRST
                LDAA    #$AA
                STAA    COPRST

                LDAA    PRCN        ; Si Flag de Ejecucion Suspendida
== TRUE
                ANDA    #%00100000  ; entonces se salta el codigo de
aplicacion
                CMPA    #0          ; anulando asi su ejecucion
                BNE     ET002A
                LDAA    PRCN        ; Si Flag de Ejecucion en Ram ==
TRUE se
                ANDA    #%01000000  ; ejecuta codigo de aplicacion con
direc
                CMPA    #0          ; base en RAMBS ( en memoria Ram).
En caso
                BEQ     ET001A      ; contrario el codigo de
aplicacion que se
                JSR     RAMBS       ; ejecuta tiene direccion base
APLICACION
                BRA     ET002A      ; ( en memoria Eprom)
ET001A          JSR     APLICACION
ET002A          JSR     TXON        ; Transmision de primera trama en
cola
                JSR     COMPTIMERS  ; Gestion espera asentimientos
                JSR     PROCESADO   ; Procesado de tramas recibidas
                BRA     PRINCIPAL   ; Comienzo del siguiente proceso


*----------------------------*
*   Subrutinas normales      *
*----------------------------*

* Subrutina Aplicacion: el codigo de la aplicacion que el usuario
quiere que
*                      ejecute el micro, debe ser incluido en este
espacio como
*                      una subrutina normal, de modo que es
imprescindible para
*                      el buen funcionamiento del nodo que la
aplicacion termine
*                      con una instruccion RTS

APLICACION      PSHA
                PSHB
                PSHX
                PSHY

* Aqui se incluiria el codigo de la Aplicacion del usuario

                PULY
                PULX
                PULB
                PULA
                RTS     ; Esta instruccion es imprescindible
```

```
* Subrutina Procesado: esta subrutina se encarga del procesado de las
tramas
*                     dirigidas al propio nodo, o tramas multicast (
dirigidas
*                     a todos los nodos) que estan en cola de
procesado

PROCESADO       PSHB
                PSHX
                LDAB    STCOLAA1
                CMPB    #0
                BEQ     ET003A          ; Si colaA1 vacia -> colaA0
                LDAB    PRIA1           ; Si colaA1 tiene trama,
analiza la
                ASLB                    ; trama que ocupa la primera
posicion
                ASLB                    ; de colaA1
                ASLB
                ASLB                    ; Calcula la direccion base de
dicha
                ASLB                    ; trama a partir del indice
priA1 y la
                LDX     #COLAA1         ; direccion base de la colaA1
                ABX
                STX     BASEPROC        ; Si se esta recibiendo en esa
posicion
                CPX     BASERX          ; no se analiza esa trama
                BEQ     ET003A
                JSR     ANALISIS        ; Analiza la trama
correspondiente
                JSR     INCPRIA1        ; Se elimina dicha trama de la
cola
                LDX     #$0000
                STX     BASEPROC        ; Valor basura de seguridad en
BASEPROC

                PULX
                PULB
                RTS

ET003A          LDAB    STCOLAA0
                CMPB    #0
                BEQ     ET004A
                LDAB    PRIA0           ; Si colaA0 tiene trama,
analiza la
                ASLB                    ; trama que ocupa la primera
posicion
                ASLB                    ; de colaA0
                ASLB
                ASLB                    ; Calcula la direccion base de
dicha
                ASLB                    ; trama a partir del indice
priA0 y la
                LDX     #COLAA0         ; direccion base de la colaA0
                ABX
                STX     BASEPROC        ; Si se esta recibiendo en esa
posicion
                CPX     BASERX          ; no se analiza esa trama
```

```
                BEQ     ET004A
                JSR     ANALISIS        ; Analiza la trama
correspondiente
                JSR     INCPRIA0        ; Se elimina dicha trama de la
cola
                LDX     #$0000
                STX     BASEPROC        ; Valor basura de seguridad en
BASEPROC

ET004A          PULX
                PULB
                RTS



* Subrutina Txon: esta subrutina se encarga de iniciar la transmision
de la
*                 primera trama en cola de transmision


TXON            PSHB
                PSHX

                LDAB    PRCC            ; Se comprueba si hay transmision
en curso
                ANDB    #%01000000      ; Si TEC == TRUE, no se inicia
una nueva
                CMPB    #%01000000      ; y se abandona la rutina
                BNE     ET005A          ; Si TEC == FALSE se salta a la
etiqueta
                PULX
                PULB
                RTS

ET005A          LDAB    STCOLAB1
                CMPB    #0
                BEQ     ET006A          ; Si colaB1 esta vacia se
comprueba colaB0
                LDAB    PRIB1
                ASLB                    ; ColaB1 no vacia. Se establece
como direc
                ASLB                    ; base de transmision la
direccion de la
                ASLB                    ; posicion indicada por PriB1 en
ColaB1
                ASLB
                ASLB
                LDX     #COLAB1
                ABX                     ; Se comprueba que no se esta
recibiendo
                CPX     BASERX          ; en esa posicion de ColaB1. Si
se esta
                BNE     ET008A          ; recibiendo, no se inicia la
Transmision

ET006A          LDAB    STCOLAB0        ; Si ColaB0 esta vacia no se
inicia Txon
                CMPB    #0
```

```
                BNE     ET007A          ; Si ColaB0 no vacia, se ejecuta
el salto
                PULX
                PULB
                RTS

ET007A          LDAB    PRIB0           ; ColaB0 no vacia. Se establece
como direc
                ASLB                    ; base de transmision la
direccion de la
                ASLB                    ; posicion indicada por PriB0 en
ColaB0
                ASLB
                ASLB
                ASLB
                LDX     #COLAB0
                ABX                     ; Se comprueba que no se esta
recibiendo
                CPX     BASERX          ; en esa posicion de ColaB0. Si
se esta
                BNE     ET008A          ; recibiendo, no se inicia la
Transmision
                PULX
                PULB
                RTS

ET008A          LDAB    PRCC
                ORAB    #%01000000
                STAB    PRCC            ; Transmision en curso = TRUE
                STX     BASETX          ; Se guarda la direccion base de
Txon
                LDAB    #1
                STAB    CNTTX           ; Inicializacion del contador de
Txon
                LDAB    2,X
                ANDB    #%00011111      ; N§bytes de trama = long campo
datos + 5
                ADDB    #5              ; limite superior de bytes de la
trama
                STAB    CNTTXMAX
                PSHA
                LDAA    0,X             ; Se transmite fisicamente el
primer byte
                JSR     OUTPUT          ; de trama. El resto de los bytes
se txten
                PULA                    ; gestionados por interrupciones
SCI
                LDAB    SCCR2
                ORAB    #%10000000      ; Se activa interrupcion SCI por
Txon
                STAB    SCCR2

ET008A1         LDAB    1,X             ; Se comprueba si la trama
requiere asent
                ANDB    #%01000000      ; en cuyo caso habra sido
guardada en Lista
                CMPB    #%01000000      ; de espera, y habra que
registrar su timer
                BNE     ET009A
```

```
                LDAB    0,X             ; Si la trama es multicast, no es
necesario
                CMPB    #0              ; registrar su timer
                BEQ     ET009A
                ASRB
                ASRB
                ASRB
                ASRB
                EORB    PRCN            ; Si emisor de trama no es el
propio nodo
                ANDB    #%00001111      ; no hay que registrar su timer
                CMPB    #0
                BNE     ET009A          ; Si se han dado todas las
condiciones para
                JSR     REGTIMER        ; ello, registramos timer de txon

ET009A          PULX
                PULB
                RTS


* Subrutina Output: esta subrutina se encarga de introducir
adecuandamente un
*                   byte en el registro de desplazamiento
correspondiente del
*                   SCI para que se efectue su transmision

OUTPUT          EQU     *
                PSHB
OUTPUT1         LDAB    SCSR        ; Espera activa hasta que el
periferico
                BITB    #$80        ; indique que se puede transmitir
un nuevo
                BEQ     OUTPUT1     ; dato sin machacar el anterior
                STAA    SCDR        ; Txon del byte
                PULB
                RTS


* Subrutina Comptimers:esta subrutina se encarga de comprobar si los
timers de
*                     las tramas en espera de asentimiento han
vencido, con lo
*                     cual habria que retransmitir dichas tramas


COMPTIMERS      PSHA
                PSHB
                PSHX

                LDAB    PRCD            ; El valor 0 para PRCD esta
reservado, por
                CMPB    #0              ; lo que no se efectua
comprobacion
                BEQ     ET012A
                LDAB    #0              ; Comprobacion de todas las
posiciones de
ET010A          CMPB    #8              ; la Lista de Espera
                BEQ     ET012A
```

```
                LDX     #PRCTEA       ; Si la posicion no esta ocupada,
no se
                ABX                   ; comprueba el timer
                LDAA    0,X
                CMPA    #0
                BEQ     ET011A
                LDX     #TIMERBASE    ; Si hay trama, se comprueba si
el timer
                ABX                   ; ha vencido. Si no lo ha hecho,
no se
                LDAA    0,X           ; hace nada
                CMPA    PRCD
                BNE     ET011A        ; Si ha vencido, se da al timer
un valor
                LDAA    #0            ; basura para no ser introducida
en cola
                STAA    0,X           ; de transmision otr vez antes de
ser
                LDX     #LISTAESP     ; fisicamente transmitida
                TBA
                ASLB                  ; Se calcula la direccion base de
la trama
                ASLB                  ; en Lista de Espera, y se
almacena como
                ASLB                  ; direccion para retransmision
                ASLB
                ASLB
                ABX
                STX     BASERTXON
                TAB
                LDAA    #%11111111   ; STINCOL = RETRANSMISION
                STAA    STINCOL      ; Se le da un codigo a la rutina
INTCOLTX
                JSR     INTCOLTX     ; indicando que la trama a
introducir se
ET011A          INCB                 ; asocia a una retransmision, y se
llama
                BRA     ET010A       ; a la rutina INTCOLTX

ET012A          PULX
                PULB
                PULA
                RTS


* Subrutina Analisis: esta subrutina se encarga de analizar las tramas
que se
*                    encuentran en cola de procesado, y que la
subrutina de
*                    procesado se ha encargado de preparar para su
analisis

ANALISIS        PSHA
                PSHX

                JSR     CHECKCRC      ; Comprobacion del CRC de trama
de trama
                LDX     BASEPROC      ; procesada
                LDAA    1,X
```

```
                ANDA    #%01000000    ; Si la trama requiere
asentimiento, se
                CMPA    #%01000000    ; genera asent/rechazo segun
corresponda
                BNE     ET013A        ; Eso lo hace la rutina
ACNOWLEDGE
                JSR     ACKNOWLEDGE
ET013A          LDAA    CRCOK
                CMPA    #0
                BEQ     ET017A        ; Si CRCOK == TRUE, se analiza
la trama
                LDAA    2,X           ; Si no, se desecha la trama
                ANDA    #%11100000
                CMPA    #%00000000    ; Si la trama es de informacion
se somete
                BNE     ET014A        ; a analisis de trama de
informacion
                JSR     INFORMACION
                PULX
                PULA
                RTS

ET014A          CMPA    #%00100000    ; Si la trama es de asentimiento
se somete
                BNE     ET015A        ; analisis de trama de
asentimiento
                JSR     ASENTIMIENTO
                PULX
                PULA
                RTS

ET015A          CMPA    #%01000000    ; Si la trama es de rechazo se
somete
                BNE     ET016A        ; a analisis de trama de rechazo
                JSR     RECHAZO
                PULX
                PULA
                RTS

ET016A          CMPA    #%11100000       ; Si la trama es de
identificacion
                BNE     ET017A           ; se somete a analisis de
trama de
                JSR     IDENTIFICACION   ; identificacion
ET017A          PULX
                PULA
                RTS


* Subrutina Acknowledge : esta subrutina se encarga de generar el
asentimiento
*                        o rechazo de tramas que asi lo requieran, en
funcion
*                        de si su CRC es correcto o no

ACKNOWLEDGE     PSHA
                PSHX
                PSHY
```

```
                LDX     BASEPROC    ; Se establecen direcciones base
para las
                LDY     #BASERESP   ; tramas de procesado y respuesta
                LDAA    #0
                LDAB    0,X         ; Se invierten las direcciones de
origen y
                ASLD                ; destino, y se introducen en el
primer
                ASLD                ; byte de la trama de acknowledge
                ASLD
                ASLD
                ABA                 ; Bits de trama perdida y
asentimiento
                STAA    0,Y         ; de la trama de acknowledge a 0,
y el
                LDAA    1,X         ; control numerico de trama igual
que
                ANDA    #%00111111  ; el de la trama procesada
                STAA    1,Y
                LDAA    CRCOK
                CMPA    #255
                BEQ     ET019A
                LDAA    #%01000000  ; Si CRC mal -> rechazo
                BRA     ET020A
ET019A          LDAA    #%00100000  ; Si CRC Ok -> asentimiento
ET020A          STAA    2,Y         ; Longitud campo datos = 0
                LDAA    #%10101010
                STAA    STINCOL     ; Codigo trama de respuesta
                JSR     CALCULOCRC  ; Se calcula el CRC de trama
                JSR     INTCOLTX    ; Se inserta en cola de
transmision

                PULY
                PULX
                PULA
                RTS



* Subrutina Intcoltx : esta subrutina se encarga de insertar en cola
de txon
*                     las tramas que asi lo requieran segun sean de
alarma,
*                     respuesta o retransmision

INTCOLTX        PSHA
                PSHB
                PSHX
                PSHY

                LDAA    STINCOL
                CMPA    #255
                BEQ     ET021A          ; Se establece direccion base
de la
                CMPA    #0              ; trama a ser introducida en
cola de
                BEQ     ET022A          ; transmision segun esta sea de
                LDX     #BASERESP       ; respuesta, retransmision o
alarma
```

```
                BRA     ET023A
ET021A          LDX     BASERTXON
                BRA     ET023A
ET022A          LDX     #BASEALARM
ET023A          LDAA    1,X
                ANDA    #%00100000      ; Se comprueba la prioridad de
trama
                CMPA    #%00100000      ; y en funcion de esta, se
introduce
                BNE     ET027A          ; en ColaB1 o ColaB0
ET024A          LDAA    STCOLAB1
                CMPA    #0
                BEQ     ET025A          ; Si es prioritaria, se
introduce la
                LDAA    PRIB1           ; trama en cola prioritaria
ColaB1
                INCA
                ANDA    #%00000011      ; Si hay por lo menos dos
huecos en la
                CMPA    ULTB1           ; cola, se introduce la trama,
y si no
                BEQ     ET025A          ; pierde su prioridad en este
nodo. Es
                INCA                    ; introducida en cola no
prioritaria
                ANDA    #%00000011
                CMPA    ULTB1
                BNE     ET027A
ET025A          JSR     INCULTB1        ; Se actualiza ColaB1
                LDAB    ULTB1
                DECB
                ANDB    #%00000011      ; Se calcula la posicion de
ColaB1
                ASLB                    ; donde introducir la trama
para que
                ASLB                    ; sea transmitida
                ASLB
                ASLB
                ASLB
                LDY     #COLAB1
                ABY
                LDAB    2,X
                ANDB    #%00011111
                ADDB    #5
ET026A          LDAA    0,X             ; Se copia la trama byte a byte
desde
                STAA    0,Y             ; su direccion base a ColaB1
                INX
                INY
                DECB
                CMPB    #0
                BNE     ET026A
                PULY
                PULX
                PULB
                PULA
                RTS

ET027A          LDAA    STCOLAB0
                CMPA    #0
```

```
                BEQ     ET028A          ; Si la trama no tiene
prioridad, se
                LDAA    PRIB0           ; introduce en cola sin
prioridad, ColaB0
                INCA
                ANDA    #%00000011
                CMPA    ULTB0
                BEQ     ET028A
                INCA
                ANDA    #%00000011      ; Cuando no hay al menos dos
huecos en
                CMPA    ULTB0           ; la cola, se produce espera
activa
                BEQ     ET028A          ; hasta que se producen
                LDAA    #$55            ; Reset COP (perro guardian)
                STAA    COPRST
                LDAA    #$AA
                STAA    COPRST
                JSR     TXON            ; Se lanza una posible
transmision
                BRA     ET023A


ET028A          JSR     INCULTB0        ; Se actualiza ColaB0
                LDAB    ULTB0
                DECB
                ANDB    #%00000011      ; Calculo de la direccion de
posicion
                ASLB                    ; en ColaB0 desde donde
transmitir la
                ASLB                    ; trama
                ASLB
                ASLB
                ASLB
                LDY     #COLAB0
                ABY
                LDAB    2,X
                ANDB    #%00011111      ; Se copia la trama byte a byte
en la
                ADDB    #5              ; posicion de ColaB0 desde
donde sera
ET029A          LDAA    0,X             ; transmitida
                STAA    0,Y
                INX
                INY
                DECB
                CMPB    #0
                BNE     ET029A
                PULY
                PULX
                PULB
                PULA
                RTS


* Subrutina Regtimer: esta subrutina se encarga de registrar los
timers de las
*                     tramas en espera de asentimiento cuya
transmision acaba
*                     de iniciarse
```

```
REGTIMER        PSHA
                PSHX

                LDX     BASETX          ; Se obtiene numero de trama de
la trama
                LDAB    1,X             ; transmitida
                ANDB    #%00000111
                LDX     #TIMERBASE
                ABX                     ; Se recoge el estado del
contador global
                LDAB    PRCD            ; para el que ya se debiera
haber
                ADDB    #90             ; recibido asentimiento, y se
anota en
                CMPB    #0              ; el regisro correspondiente
                BNE     ET029A1
                INCB
ET029A1         STAB    0,X

                PULX
                PULB
                RTS



* Subrutina IncpriA1: esta subrutina se encarga de la actualizacion de
colaA1
*                   cuando la primera posicion de esta ha sido
procesada

INCPRIA1        PSHA

                LDAA    PRIA1           ; PriA1 = PriA1 + 1
                INCA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    PRIA1
                CMPA    ULTA1           ; Si PriA1 == UltA1 -> cola
vacia
                BNE     ET030A
                LDAA    #%00000000
                BRA     ET031A
ET030A          LDAA    #%10101010      ; Si PriA1 != UltA1 -> cola no
llena
ET031A          STAA    STCOLAA1

                PULA
                RTS


* Subrutina IncpriA0: esta subrutina se encarga de la actualizacion de
colaA0
*                   cuando la primera posicion de esta ha sido
procesada

INCPRIA0        PSHA

                LDAA    PRIA0           ; PriA0 = PriA0 + 1
                INCA
```

```
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    PRIA0
                CMPA    ULTA0           ; Si PriA0 == UltA0 -> cola
vacia
                BNE     ET032A
                LDAA    #%00000000
                BRA     ET033A
ET032A          LDAA    #%10101010      ; Si PriA0 != UltA0 -> cola no
llena
ET033A          STAA    STCOLAA0
                PULA
                RTS


* Subrutina IncpriB1: esta subrutina se encarga de la actualizacion de
colaB1
*                    cuando la primera posicion de esta ha sido
procesada

INCPRIB1        PSHA

                LDAA    PRIB1           ; PriB1 = PriB1 + 1
                INCA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    PRIB1
                CMPA    ULTB1           ; Si PriB1 == UltB1 -> cola
vacia
                BNE     ET030A1
                LDAA    #%00000000
                BRA     ET031A1
ET030A1         LDAA    #%10101010      ; Si PriB1 != UltB1 -> cola no
llena
ET031A1         STAA    STCOLAB1
                PULA
                RTS


* Subrutina IncpriB0: esta subrutina se encarga de la actualizacion de
colaB0
*                    cuando la primera posicion de esta ha sido
procesada

INCPRIB0        PSHA

                LDAA    PRIB0           ; PriB0 = PriB0 + 1
                INCA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    PRIB0
                CMPA    ULTB0           ; Si PriB0 == UltB0 -> cola
vacia
                BNE     ET032A1
                LDAA    #%00000000
                BRA     ET033A1
ET032A1         LDAA    #%10101010      ; Si PriB0 != UltB0 -> cola no
llena
ET033A1         STAA    STCOLAB0
                PULA
                RTS
```

```
* Subrutina IncultB1: esta subrutina se encarga de la actualizacion de
colaB1
*                    cuando se ha insertado una nueva trama en dicha
cola

INCULTB1        PSHA

                LDAA    ULTB1           ; UltB1 = UltB1 + 1
                INCA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    ULTB1
                CMPA    PRIB1           ; Si UltB1 == PriB1 -> cola
llena
                BNE     ET034A
                LDAA    #%11111111
                BRA     ET035A
ET034A          LDAA    #%10101010      ; Si UltB1 != PriB1 -> cola no
vacia
ET035A          STAA    STCOLAB1
                PULA
                RTS


* Subrutina IncultB0: esta subrutina se encarga de la actualizacion de
colaB0
*                    cuando se ha insertado una nueva trama en dicha
cola

INCULTB0        PSHA

                LDAA    ULTB0           ; UltB0 = UltB0 + 1
                INCA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    ULTB0
                CMPA    PRIB0           ; Si UltB0 == PriB0 -> cola
llena
                BNE     ET036A
                LDAA    #%11111111
                BRA     ET037A
ET036A          LDAA    #%10101010      ; Si UltB0 != PriB0 -> cola no
vacia
ET037A          STAA    STCOLAB0
                PULA
                RTS


* Subrutina IncultA1: esta subrutina se encarga de la actualizacion de
colaA1
*                    cuando se inserta una nueva trama en dicha cola

INCULTA1        PSHA

                LDAA    ULTA1           ; UltA1 = UltA1 + 1
                INCA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    ULTA1
                CMPA    PRIA1           ; Si UltA1 == PriA1 -> cola
llena
                BNE     ET038A
                LDAA    #%11111111
```

```
                BRA     ET039A
ET038A          LDAA    #%10101010      ; Si UltA1 != PriA1 -> cola no
vacia
ET039A          STAA    STCOLAA1

                PULA
                RTS


* Subrutina IncultA0: esta subrutina se encarga de la actualizacion de
colaA0
*                    cuando se ha insertado una nueva trama en dicha
cola

INCULTA0        PSHA

                LDAA    ULTA0           ; UltA0 = UltA0 + 1
                INCA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    ULTA0
                CMPA    PRIA0           ; Si UltA0 == PriA0 -> cola
llena
                BNE     ET040A
                LDAA    #%11111111
                BRA     ET041A
ET040A          LDAA    #%10101010      ; Si UltA0 != PriA0 -> cola no
vacia
ET041A          STAA    STCOLAA0

                PULA
                RTS



* Subrutina DecultA1: esta subrutina se encarga de la actualizacion de
colaA1
*                    cuando la trama que se estaba recibiendo en esa
posicion
*                    ha sido desechada por sufrir algun error

DECULTA1        PSHA

                LDAA    ULTA1           ; UltA1 = UltA1 - 1
                DECA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    ULTA1
                CMPA    PRIA1           ; Si UltA1 == PriA1 -> cola
vacia
                BNE     ET042A
                LDAA    #%00000000
                BRA     ET043A
ET042A          LDAA    #%10101010      ; Si UltA1 != PriA1 -> cola no
llena
ET043A          STAA    STCOLAA1
                PULA
                RTS


* Subrutina DecultA0: esta subrutina se encarga de la actualizacion de
colaA0
```

```
*                    cuando la trama que se estaba recibiendo en esa
posicion
*                    ha sido desechada por sufrir algun error

DECULTA0        PSHA

                LDAA    ULTA0           ; UltA0 = UltA0 - 1
                DECA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    ULTA0
                CMPA    PRIA0           ; Si UltA0 == PriA0 -> cola
vacia
                BNE     ET044A
                LDAA    #%00000000
                BRA     ET045A
ET044A          LDAA    #%10101010      ; Si UltA0 != PriA0 -> cola no
llena
ET045A          STAA    STCOLAA0
                PULA
                RTS



* Subrutina DecultB1: esta subrutina se encarga de la actualizacion de
colaB1
*                    cuando la trama que se estaba recibiendo en esa
posicion
*                    ha sido desechada por sufrir algun error

DECULTB1        PSHA

                LDAA    ULTB1           ; UltB1 = UltB1 - 1
                DECA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    ULTB1
                CMPA    PRIB1           ; Si UltB1 == PriB1 -> cola
vacia
                BNE     ET046A
                LDAA    #%00000000
                BRA     ET047A
ET046A          LDAA    #%10101010      ; Si UltB1 != PriB1 -> cola no
llena
ET047A          STAA    STCOLAB1
                PULA
                RTS


* Subrutina DecultB0: esta subrutina se encarga de la actualizacion de
colaB0
*                    cuando la trama que se estaba recibiendo en esa
posicion
*                    ha sido desechada por sufrir algun error

DECULTB0        PSHA

                LDAA    ULTB0           ; UltB0 = UltB0 - 1
                DECA
                ANDA    #%00000011      ; Tratamiento de cola circular
                STAA    ULTB0
```

```
                CMPA    PRIB0           ; Si UltB0 == PriB0 -> cola
vacia
                BNE     ET048A
                LDAA    #%00000000
                BRA     ET049A
ET048A          LDAA    #%10101010      ; Si UltB0 != PriB0 -> cola no
llena
ET049A          STAA    STCOLAB0
                PULA
                RTS



* Subrutina Actcolas: esta subrutina se encarga de gestionar la
actualizacion
*                    de las colas cuando sucede algun evento, tal
como la
*                    finalizacion de la transmision de una trama,
error en
*                    recepcion de algun byte de trama,o comienzo de
recepcion
*                    de trama

ACTCOLAS        PSHB
                PSHX

                LDAB    PRCC
                ANDB    #%10000000
                CMPB    #%10000000
                BNE     ET053A          ; Si transmision completa,
segun se haya
                LDX     #BASETX         ; transmitido desde ColaB1 o
ColaB0:
                INX
                LDAB    0,X
                ANDB    #%10000000
                CMPB    #%10000000
                BNE     ET051A
ET050A          JSR     INCPRIB1
                LDAB    PRIB1           ; Desde ColaB1  entonces:
                LDX     #PRCHB1         ; se incrementa PriB1, y si
hay algun
                ABX                     ; hueco en la cola, se elimina
                LDAB    0,X
                CMPB    #%11111111
                BNE     ET052A
                LDAB    #%00000000
                STAB    0,X
                BRA     ET050A
ET051A          JSR     INCPRIB0         ; Desde ColaB0 entonces:
                LDAB    PRIB0            ; se incrementa PriB0, y si
hay algun
                LDX     #PRCHB0          ; hueco en la cola, se
elimina
                ABX
                LDAB    0,X
                CMPB    #%11111111
                BNE     ET052A
                LDAB    #%00000000
                STAB    0,X
```

```
                BRA     ET051A
ET052A          LDX     #$0000          ; Se da a la direccion base
de
                STX     BASETX          ; transmision un valor basura
                LDAB    PRCC            ; Transmision Completa =
FALSE
                ANDB    #%00111111      ; Transmision en Curso =
FALSE
                STAB    PRCC
                PULX
                PULB
                RTS

ET053A          LDX     BASERX          ; Se ha producido un evento
en Rxon
                LDAB    PRCC
                ANDB    #%00000100
                CMPB    #%00000100
                BEQ     ET059A          ; Salto si Recepcion
Incompleta
                LDAB    PRCC
                ANDB    #%00010000
                CMPB    #%00010000
                BNE     ET056A          ; Salto si se recibe en colas
B
                LDAB    1,X
                ANDB    #%00100000
                CMPB    #%00100000
                BNE     ET054A          ; Produciendose una recepcion
:
                JSR     INCULTA1        ; Si se recibe en ColaA1 ->
                BRA     ET055A          ; UltA1 = UltA1 + 1
ET054A          JSR     INCULTA0        ; Si se recibe en colaA0 ->
ET055A          PULX                    ; UltA0 = UltA0 + 1
                PULB
                RTS


ET056A          LDAB    1,X
                ANDB    #%00100000
                CMPB    #%00100000      ; Produciendose una recepcion
:
                BNE     ET057A          ; Si se recibe en ColaB1 ->
                JSR     INCULTB1        ; UltB1 = UltB1 + 1
                BRA     ET058A          ; Si se recibe en ColaB0 ->
ET057A          JSR     INCULTB0        ; UltB0 = UltB0 + 1
ET058A          PULX
                PULB
                RTS

ET059A          LDAB    PRCC
                ANDB    #%00010000
                CMPB    #%00010000
                BNE     ET062A          ; Abortar la recepcion en
colas
                LDAB    1,X             ; A0/A1 por fallo
                ANDB    #%00100000
                CMPB    #%00100000
                BNE     ET060A
                JSR     DECULTA1        ; Si se recibia en ColaA1 ->
```

```
                BRA     ET061A          ; UltA1 = UltA1 - 1
ET060A          JSR     DECULTA0        ; Si se recibia en ColaA0 ->
ET061A          LDX     #$0000          ; UltA0 = UltA0 - 1
                STX     BASERX          ; Se da a la direccion base
de
                PULX                    ; recepcion un valor basura
                PULB
                RTS

ET062A          PSHA                    ; Abortar la recepcion en
colas
                LDD     #BASERX         ; B0/B1 por fallo
                ANDB    #%01100000
                ASRB                    ; Calculo del indice de cola
de
                ASRB                    ; la posicion de ColaBi en la
                ASRB                    ; que se esta recibiendo
                ASRB
                ASRB
                INCB
                ANDB    #%00000011
                LDAA    1,X
                ANDA    #%00100000
                CMPA    #%00100000
                BNE     ET065A          ; Si se recibia ultima
posicion
                PULA
                CMPB    ULTB1           ; de ColaB1 -> UltB1 = UltB1
- 1
                BNE     ET063A
                JSR     DECULTB1
                BRA     ET064A          ; Si no era la ultima
posicion
ET063A          DECB                    ; implica que se ha producido
un
                ANDB    #%00000011      ; hueco, y se registra como
tal
                LDX     #PRCHB1
                ABX
                LDAB    #%11111111
                STAB    0,X
ET064A          LDX     #$0000          ; Se da a la direccion base
de
                STX     BASERX          ; recepcion un valor basura
                PULX
                PULB
                RTS

ET065A          PULA
                CMPB    ULTB0           ; Si se recibia ultima
posicion
                BNE     ET066A          ; de ColaB0 -> UltB0 = UltB0
- 1
                JSR     DECULTB0
                BRA     ET067A          ; Si no era la ultima
posicion
ET066A          DECB                    ; implica que se ha producido
un
                ANDB    #%00000011      ; hueco, y se registra como
tal
```

```
                LDX     #PRCHB0
                ABX
                LDAB    #%11111111
                STAB    0,X
ET067A          LDX     #$0000          ; Se da a la direccion base
de
                STX     BASERX          ; recepcion un valor basura
                PULX
                PULB
                RTS



* Subrutina Asentimiento: esta subrutina se encarga del analisis de
las tramas
*                       de asentimiento, eliminando de la lista de
tramas en
*                       espera de asentimiento a aquella trama que
esperaba
*                       el asentimiento recibido

ASENTIMIENTO    PSHB
                PSHX

                LDX     BASEPROC
                LDAB    1,X             ; Acumulador B = numero de
trama
                ANDB    #%00000111
                LDX     #PRCTEA         ; Registra esa posicion en
lista
                ABX                     ; de espera como vacia
                LDAB    #%00000000
                STAB    0,X
                PULX
                PULB
                RTS

* Subrutina Rechazo: esta subrutina se encarga de retransmitir aquella
trama que
*                  estando almacenada en la lista de tramas en
espera de
*                  asentimiento, ha sido rechazada

RECHAZO         PSHA
                PSHB
                PSHX

                LDX     BASEPROC
                LDAB    1,X             ; Acumulador A = Acumulador B =
n§ trama
                ANDB    #%00000111
                TBA
                ASLB                    ; Calculo de la direccion de la
posicion
                ASLB                    ; en la lista de espera de la
trama
                ASLB                    ; rechazada que hay que
retransmitir
                ASLB
```

```
                ASLB
                LDX     #LISTAESP
                ABX
                STX     BASERTXON       ; BASERTXON = direccion de
dicha posicion
                LDAB    #%11111111
                STAB    STINCOL         ; STINCOL = Retransmision
                JSR     INTCOLTX        ; Se introduce la trama a
retransmitir en
                TAB                     ; cola de transmision
                LDX     #TIMERBASE
                ABX                     ; Se da al timer de la trama un
valor de
                LDAA    #0              ; vencimiento basura para que
no vuelva
                STAA    0,X             ; que no sea fisicamente
transmitida
                PULX
                PULB
                PULA
                RTS


* Subrutina Identificacion: esta subrutina se encarga de registrar el
numero
*                         de nodo que le ha correspondido a la
tarjeta en
*                         el proceso de identificacion de la red,
asi como
*                         de enviar al PC informacion acerca del
tipo de nodo

IDENTIFICACION  PSHA
                PSHX
                PSHY

                LDX     BASEPROC        ; En campo de datos de la trama
de
                LDAA    3,X             ; identificacion esta el n§ de
nodo
                ORAA    #%10100000
                STAA    PRCN            ; Se registra dicho numero y se
                LDAA    3,X
                INCA
                STAA    3,X             ; establece MICRO ACTIVO,
EJECUCION
                LDY     #BASERESP       ; PROGRAMA EN EPROM, EJECUCION
NO
                LDAA    #%00000000      ; SUSPENDIDA
                STAA    0,Y
                LDAA    1,X             ; Se incrementa en 1 el
contenido
                STAA    1,Y             ; del campo de datos, y se
vuelve
                LDAA    2,X             ; a introducir la trama en cola
de
                STAA    2,Y             ; transmision
                LDAA    3,X
                STAA    3,Y
                LDAA    #%10101010
```

```
                STAA    STINCOL
                JSR     CALCULOCRC      ; Se recalcula el CRC
                JSR     INTCOLTX
                LDX     #LISTAESP
                STX     BASERTXON       ; Se genera a su vez una trama
para
                LDAA    PRCN            ; el PC
                ASLA
                ASLA
                ASLA
                ASLA
                STAA    0,X
                LDAA    #%01100000      ; Se le da el codigo numerico 0
es
                STAA    1,X             ; prioritaria y requiere
asentimiento
                LDAA    #%10100111      ; Se trata de una trama de
registro
                STAA    2,X             ; de nodo, de longitud fija
                LDY     #CLASENODO
                LDAA    0,Y             ; Clasenodo = direccion base de
una
                STAA    3,X             ; posicion de 7 bytes de memoria
no
                LDAA    1,Y             ; volatil
                STAA    4,X
                LDAA    2,Y             ; Tal informacion define
univocamente
                STAA    5,X             ; al nodo en lo referente a
entradas
                LDAA    3,Y             ; y salidas de este (
existencia,
                STAA    6,X             ; funcionalidad y
direccionalidad )
                LDAA    4,Y
                STAA    7,X             ; Se introduce esa informacion
en el
                LDAA    5,Y             ; campo de datos de la trama
dirigida
                STAA    8,X             ; al PC
                LDAA    6,Y
                STAA    9,X
                LDAA    #%11111111
                STAA    STINCOL
                JSR     CALCULOCRC      ; Calculo del CRC de la trama
                JSR     INTCOLTX
                LDX     #TIMERBASE
                LDAA    #0              ; Se da al timer de la trama un
valor
                STAA    0,X             ; de vencimiento basura, hasta
que
                PULY                    ; cuando sea transmitida se
registre
                PULX                    ; el valor efectivo del mismo
                PULA
                RTS


* Subrutina Informacion: esta subrutina se encarga del analisis de las
tramas
```

```
*                       de informacion, generando las acciones
oportunas de
*                       acuerdo con las instrucciones encapsuladas en
los
*                       paquetes de datos de tales tramas


INFORMACION     PSHA
                PSHB
                PSHX
                PSHY

                LDX     BASEPROC
                LDAB    2,X             ; Cntprocmax = Long campo datos
                ANDB    #%00011111      ; de la trama de informacion
                STAB    CNTPROCMAX
                LDAB    #3
                ABX
                LDAB    #1              ; Contador = 1
ET068A          CMPB    CNTPROCMAX      ; Mientras Contador =< Cntpromax
                BLE     ET068A1         ; se ejecuta el bucle
                JMP     ET076A
ET068A1         LDAA    0,X
                CMPA    #%00000001
                BNE     ET068A2
                JMP     ET074A
ET068A2         CMPA    #%00000010
                BNE     ET068A3
                JMP     ET073A          ; Segun el codigo de instruccion
ET068A3         CMPA    #%00000100      ; del paquete que se este
leyendo
                BNE     ET068A4
                JMP     ET072A          ; se ejecuta una u otra
subrutina
ET068A4         CMPA    #%00001000
                BEQ     ET071A
                CMPA    #%00001011
                BEQ     ET070A1
                CMPA    #%00001001
                BEQ     ET075A
                CMPA    #%00010000
                BEQ     ET070A
                CMPA    #%00100000
                BEQ     ET069A

* CMPA    #%01000000
* Futuras instrucciones se insertarian aqui

                LDAA    PRCN
                ANDA    #%11011111
                STAA    PRCN            ; Ejecutar  Activar Ejecucion
                INCB                    ; Contador = Contador + 1
                INX
                BRA     ET068A          ; Volver al comienzo del bucle

ET069A          LDAA    PRCN
                ORAA    #%00100000
                STAA    PRCN            ; Ejecutar Suspender Ejecucion
                INCB                    ; Contador = Contador + 1
                INX
```

```
                BRA     ET068A          ; Volver al comienzo del bucle

ET070A          LDAA    PRCN
                ANDA    #%10111111
                STAA    PRCN            ; Ejecutar Programa en Eprom
                INCB                    ; Contador = Contador + 1
                INX
                BRA     ET068A          ; Volver al comienzo del bucle

ET070A1         LDAA    #127            ; Cuando comienza la copia de
                STAA    CNTCOPMAX       ; un programa, se inicializan
                LDAA    #0              ; los contadores
correspondientes
                STAA    CNTCOP
ET071A          LDY     2,X             ; Ejecutar Copiar Programa en
Ram
                LDAA    CNTCOP
                INCA                    ; Numero de instrucciones de
copia
                STAA    CNTCOP          ; se incrementa en 1
                ADDB    1,X
                ADDB    #4              ; Se actualiza contador de bytes
                LDAA    1,X             ; ya procesados, asi como el
                INX                     ; puntero correspondiente
                INX
                INX
                INX
                PSHB                    ; Se salva el valor del contador
ET071A1         CMPA    #0              ; Mientras no se hayan copiado
                BEQ     ET071A2         ; todos los bytes, se copia el
                LDAB    0,X             ; byte actual y se actualizan
                STAB    0,Y             ; tanto contadores como punteros
                INX
                INY
                DECA
                BRA     ET071A1         ; Byte actual = Siguiente byte
ET071A2         PULB
                LDAA    CNTCOP
                ANDA    #%10000000
                CMPA    #%10000000
                BEQ     ET071A3
                JMP     ET068A
ET071A3         LDAA    CNTCOP          ; Si se han ejecutado todas las
                CMPA    CNTCOPMAX       ; instrucciones de copia, y
llego
                BEQ     ET076A1         ; fin de copia, se salta
                JMP     ET068A          ; Volver al comienzo del bucle
ET075A          LDAA    1,X             ; Ejecutar Fin de Copia Programa
                INX                     ; Se actualizan tanto puntero
como
                INX                     ; contador de procesado
                ADDB    #2              ; Si se han ejecutado todas las
                CMPA    CNTCOP          ; instrucciones de copia, se
                BEQ     ET076A1         ; salta, y si no, se registra el
                STAA    CNTCOPMAX       ; fin de copia
                LDAA    CNTCOP
                ORAA    #%10000000
                STAA    CNTCOP
                JMP     ET068A          ; Volver al comienzo del bucle
```

```
ET072A          LDAA    PRCN
                ORAA    #%01000000
                STAA    PRCN            ; Ejecutar Programa en Ram
                INCB                    ; Contador = Contador + 1
                INX
                JMP     ET068A          ; Volver al comienzo del bucle

ET073A          LDY     1,X             ; Ejecutar Escribir Dato
                LDAA    3,X
                STAA    0,Y             ; Se incrementa el contador en
                ADDB    #4              ; 4 bytes, y se actualiza el
                INX                     ; puntero
                INX
                INX
                INX
                JMP     ET068A          ; Volver al comienzo del bucle

ET074A          LDAA    PRCN
                ANDA    #%00010000
                CMPA    #%00010000
                BEQ     ET074A1         ; Ejecutar Leer_Dato
                JSR     GENRESP         ; Si se empieza a generar
respuesta, se
ET074A1         LDY     #BASERESP       ; ejecuta la subrutina
                TBA
                LDAB    CNTRESP
                ADDB    #3
                ABY
                TAB
                LDAA    #%10000000      ; Se escribe codigo de
instruccion de
                STAA    0,Y             ; respuesta de lectura
                LDAA    1,X             ; A continuacion se escribe la
direccion
                STAA    1,Y             ; donde se encuentra el dato
leido y por
                LDAA    2,X             ; ultimo el dato solicitado
                STAA    2,Y
                PSHX
                LDX     1,Y
                LDAA    0,X
                STAA    3,Y
                PULX
                LDAA    CNTRESP         ; Actualizacion de contadores de
trama
                ADDA    #4              ; de respuesta y de procesado
                STAA    CNTRESP
                ADDB    #3
                INX
                INX
                INX
                JMP     ET068A          ; Volver al comienzo del bucle


ET076A1         JSR     GENRESP         ; Se genera trama de respuesta
al
                LDAA    #%00001010      ; nodo que envia el programa
                LDY     #BASERESP       ; Se introduce la instruccion de
                STAA    3,Y             ; programa copiado en Ram
                LDAA    1,Y
```

```
                ORA     #%00100000
                STAA    1,Y
                LDAA    CNTRESP
                INCA
                STAA    CNTRESP
                JMP     ET068A

ET076A          LDAB    PRCN
                ANDB    #%00010000
                CMPB    #%00010000
                BNE     ET078A
                LDX     #BASERESP       ; Si se  genero trama respuesta
                LDAB    2,X             ; Longitud campo de datos =
Bytes de
                ORAB    CNTRESP         ; respuesta generados
                STAB    2,X
                LDAB    #%10101010
                STAB    STINCOL
                LDAB    1,X
                ANDB    #%01000000
                CMPB    #%01000000
                BNE     ET077A            ; Si la respuesta requiere
                JSR     INLISTAESPERA     ; asentimiento -> lista
espera
ET077A          JSR     INTCOLTX          ; Se inserta en cola de txon
                LDAA    PRCN
                ANDA    #%11101111
                STAA    PRCN            ; Reset flag respuesta generada

ET078A          PULY
                PULX
                PULB
                PULA
                RTS



* Subrutina Generar Respuesta : esta subrutina es la encargada de
generar una
*                               trama de respuesta

GENRESP         PSHA
                PSHB
                PSHX
                PSHY

                LDX     BASEPROC        ; Si no se empezo a generar
respuesta
                LDY     #BASERESP       ; anteriormente, se establece
direccion
                LDAA    #0              ; de tramas de procesado y
respuesta
                LDAB    0,X             ; Invierte las direcciones de
origen y
                ASLD                    ; destino, y las introduce en el
primer
                ASLD                    ; byte de la trama de
acknowledge
                ASLD
                ASLD
```

```
                ABA                     ; Bit de trama perdida a 0 y bit
de
                STAA    0,Y             ; asentimiento a 1, y el de
prioridad
                LDAA    1,X             ; igual que el de la trama
procesada
                ANDA    #%00100000      ; Por el momento se le da num de
trama 0
                ORAA    #%01000000
                STAA    1,Y
                LDAA    #0
                STAA    2,Y             ; Tipo de trama = Informacion
                STAA    CNTRESP
                LDAA    PRCN
                ORAA    #%00010000
                STAA    PRCN

                PULY
                PULX
                PULB
                PULA
                RTS


* Subrutina Inlistaespera: esta subrutina se encarga de buscar en la
lista de
*                        tramas en espera de asentimiento un hueco y
de
*                        introducir en este la trama generada


INLISTAESPERA   PSHA
                PSHB
                PSHX
                PSHY
                LDAA    STINCOL
                CMPA    #0
                BEQ     ET079A
                LDX     #BASERESP       ; Introduccion de alarma
                BRA     ET080A
ET079A          LDX     #BASEALARM      ; Introduccion de respuesta
ET080A          LDY     #PRCTEA
                LDAB    #0
ET081A          CMPB    #8              ; Se busca un hueco en las
                BGE     ET084A          ; posiciones de la lista
                LDAA    0,Y
                CMPA    #0
                BEQ     ET082A
                INCB
                INY
                BRA     ET081A
ET082A          LDAA    #255
                STAA    0,Y             ; Se registra la posicion como
ocupada
                LDY     #TIMERBASE
                ABY
                LDAA    #0              ; Se da al timer de la posicion
un
                STAA    0,Y             ; valor de vencimiento basura
```

```
                LDAA    1,X             ; Registro del identificador
numerico
                ANDA    #%11100000      ; de la trama en campo
correspondiente
                ABA
                STAA    1,X
                JSR     CALCULOCRC      ; Se calcula su CRC
                LDY     #LISTAESP
                ASLB
                ASLB
                ASLB
                ASLB
                ASLB
                ABY
                LDAB    2,X             ; Calculo del numero de bytes
que hay
                ANDB    #%00011111      ; que transferir a la lista de
espera
                ADDB    #5
ET083A          CMPB    #0
                BEQ     ET085A
                LDAA    0,X             ; Una vez encontrado, se copia
la trama
                STAA    0,Y             ; de alarma o respuesta byte a
byte
                INX
                INY
                DECB
                BRA     ET083A

ET084A          LDAA    1,X
                ANDA    #%10111111
                STAA    1,X             ; Si no se encontro hueco, se
transmite
                JSR     CALCULOCRC      ; como trama que no requiere
asentimiento
ET085A          PULY                    ; con lo cual se pierden
prestaciones
                PULX
                PULB
                PULA
                RTS




* Subrutina CalculoCRC: esta subrutina se encarga de calcular el CRC
de la
*                       trama generada, usando el polinomio generador
del
*                       CRC-CCITT de orden 16

CALCULOCRC      PSHA
                PSHB
                PSHX

                LDAA    STINCOL
                CMPA    #0              ; Se establece la direccion
base de la
```

```
                BEQ     ET087A          ; trama segun esta sea de
retransmision,
                CMPA    #255            ; respuesta o alarma
                BEQ     ET086A
                LDX     #BASERESP
                BRA     ET088A
ET086A          LDX     BASERTXON
                BRA     ET088A
ET087A          LDX     #BASEALARM      ; Una vez determinada, se
almacena como
ET088A          STX     BASECRC         ; direccion base para calculo de
CRC
                LDAB    2,X
                ANDB    #%00011111      ; Como lo que se quiere hacer es
calcular
                ADDB    #3              ; el CRC, los dos ultimos bytes
de trama
                ABX                     ; se ponen a 0
                LDD     #$0000
                STD     0,X             ; Una vez preparada la trama, se
llama a
                JSR     OBTENERCRC      ; la rutina encargada de la
obtencion del
                LDD     BASECRC         ; CRC
                STD     0,X             ; Se almacena el CRC obtenido en
los campos
*                                         correspondientes
                PULX
                PULB
                PULA
                RTS


* Subrutina CheckCRC: esta subrutina se encarga de comprobar si la
trama que
*                     esta siendo procesada, se recibio correctamente

CHECKCRC        PSHA
                PSHB
                PSHX
                LDX     BASEPROC
                STX     BASECRC         ; Se establece direccion base
                JSR     OBTENERCRC      ; Se llama a la subrutina que
devolvera
                LDD     BASECRC         ; el sindrome de la trama
                CPD     #$0000          ; Si sindrome == $0000 -> CRCOK
= TRUE
                BNE     ET089A          ; Si no -> CRCOK = FALSE
                LDAA    #%11111111
ET089A          STAA    CRCOK
                PULX
                PULB
                PULA
                RTS

* Subrutina ObtenerCRC: esta subrutina se encarga de calcular el CRC o
el
*                     sindrome de las tramas a transmitir o a
procesar
*                     respectivamente
```

```
OBTENERCRC      PSHA
                PSHB
                PSHX
                PSHY

                LDY     BASECRC         ; Direccion base de trama ->
acumulador Y
                LDAA    2,Y
                STAA    BYTECRC         ; ByteCRC = tercer byte de la
trama
                ANDA    #%00011111
                ADDA    #3              ; ContCRC = Long campo datos + 3
= n§ de
                STAA    CONTCRC         ; iteraciones para la obtencion
del CRC
                LDD     0,Y             ; D = 2 primeros bytes de la
trama
                INY
                INY
ET090A          PSHA
                LDAA    CONTCRC
                CMPA    #0
                PULA
                BEQ     ET090A1
                BRA     ET090A2
ET090A1         JMP     ET123A
ET090A2         BITA    #%10000000      ; Si primer bit en el dividendo
== 0
                BNE     ET091A          ; se desplaza el dividendo una
posicion a
                ROL     BYTECRC         ; la derecha, introduciendo el
bit desde
                ROLB                    ; ByteCRC
                ROLA
                BRA     ET094A          ; Si primer bit en dividendo ==
1
ET091A          EORA    #%10001000      ; Se efectua la division entre
el
                EORB    #%00010000      ; polinomio generador CRC-CCITT
                ROL     BYTECRC         ; Se desplaza resto una posicion
a la dcha
                BCS     ET092A          ; El bit introducido es el
negado del que
                SEC                     ; llega desde ByteCRC, debido al
factor 1
                BRA     ET093A          ; en el polinomio CRC-CCITT
ET092A          CLC
ET093A          ROLB
                ROLA

ET094A          BITA    #%10000000      ; Segundo bit de byte
                BNE     ET095A          ; Ver comentarios entre
etiquetas ET090A
                ROL     BYTECRC         ; y ET094A
                ROLB
                ROLA
                BRA     ET098A
ET095A          EORA    #%10001000
                EORB    #%00010000
```

```
                ROL     BYTECRC
                BCS     ET096A
                SEC
                BRA     ET097A
ET096A          CLC
ET097A          ROLB
                ROLA

ET098A          BITA    #%10000000    ; Tercer bit de byte
                BNE     ET099A        ; Ver comentarios entre
etiquetas ET090A
                ROL     BYTECRC       ; y ET094A
                ROLB
                ROLA
                BRA     ET102A
ET099A          EORA    #%10001000
                EORB    #%00010000
                ROL     BYTECRC
                BCS     ET100A
                SEC
                BRA     ET101A
ET100A          CLC
ET101A          ROLB
                ROLA

ET102A          BITA    #%10000000    ; Cuarto bit de byte
                BNE     ET103A        ; Ver comentarios entre
etiquetas ET090A
                ROL     BYTECRC       ; y ET094A
                ROLB
                ROLA
                BRA     ET106A
ET103A          EORA    #%10001000
                EORB    #%00010000
                ROL     BYTECRC
                BCS     ET104A
                SEC
                BRA     ET105A
ET104A          CLC
ET105A          ROLB
                ROLA

ET106A          BITA    #%10000000    ; Quinto bit de byte
                BNE     ET107A        ; Ver comentarios entre
etiquetas ET090A
                ROL     BYTECRC       ; y ET095A
                ROLB
                ROLA
                BRA     ET110A
ET107A          EORA    #%10001000
                EORB    #%00010000
                ROL     BYTECRC
                BCS     ET108A
                SEC
                BRA     ET109A
ET108A          CLC
ET109A          ROLB
                ROLA

ET110A          BITA    #%10000000    ; Sexto bit de byte
```

```
                BNE     ET111A          ; Ver comentarios entre
etiquetas ET090A
                ROL     BYTECRC         ; y ET095A
                ROLB
                ROLA
                BRA     ET114A
ET111A          EORA    #%10001000
                EORB    #%00010000
                ROL     BYTECRC
                BCS     ET112A
                SEC
                BRA     ET113A
ET112A          CLC
ET113A          ROLB
                ROLA

ET114A          BITA    #%10000000      ; Septimo bit de byte
                BNE     ET115A          ; Ver comentarios entre
etiquetas ET090A
                ROL     BYTECRC         ; y ET095A
                ROLB
                ROLA
                BRA     ET118A
ET115A          EORA    #%10001000
                EORB    #%00010000
                ROL     BYTECRC
                BCS     ET116A
                SEC
                BRA     ET117A
ET116A          CLC
ET117A          ROLB
                ROLA

ET118A          BITA    #%10000000      ; Octavo y ultimo bit de byte
                BNE     ET119A          ; Ver comentarios entre
etiquetas ET090A
                ROL     BYTECRC         ; y ET095
                ROLB
                ROLA
                BRA     ET122A
ET119A          EORA    #%10001000
                EORB    #%00010000
                ROL     BYTECRC
                BCS     ET120A
                SEC
                BRA     ET121A
ET120A          CLC
ET121A          ROLB
                ROLA

ET122A          INY                     ; Se introduce en ByteCRC el
siguiente
                PSHA                    ; byte para que sea introducido
bit a bit
                LDAA    0,Y             ; en la division
                STAA    BYTECRC
                PULA
                DEC     CONTCRC         ; Se decrementa el n§ de
iteraciones que
```

```
                JMP     ET090A          ; quedan del bucle. Salto al
principio

ET123A          STD     BASECRC         ; BaseCRC = sindrome o CRC
(segun caso)
                PULY
                PULX
                PULB
                PULA
                RTS

* Subrutina Genalarma:  esta subrutina se encarga de generar un
mensaje de
*                       alarma hacia el PC cuando le es solicitado.

GENALARM        PSHA
                PSHB
                PSHX
                PSHY
                LDX     TEXTOALARMA     ; Se establece la direccion
base del
                LDY     #BASEALARM      ; texto de alarma
                LDAA    PRCN
                ASLA
                ASLA
                ASLA
                ASLA                    ; Direccion Destino = PC
                STAA    0,Y             ; Direccion Origen = ID del
nodo
                LDAA    #96             ; Prioridad = TRUE
                STAA    1,Y             ; Requiere Asentimiento = TRUE
                LDAA    #128            ; Tipo de Trama = Alarma
                STAA    2,Y
                INY
                INY
                INY
                LDAB    #0              ; Se introduce el texto de
alarma en el
ET124A          LDAA    0,X             ; campo de datos de la trama
                CMPA    #0              ; La copia finaliza al detectar
un 0, que
                BEQ     ET125A          ; indica fin de texto o al
estar completo
                STAA    0,Y             ; el campo de datos de la
trama, es decir
                INX                     ; el texto puede constar como
maximo de
                INY                     ; 27 caracteres
                INCB
                CMPB    #27
                BEQ     ET125A
                BRA     ET124A
ET125A          LDY     #BASEALARM
                LDAA    2,Y
                ABA
                STAA    2,Y
                LDAB    STINCOL       ; Se salva el valor de STINCOL,
para poder
                LDAA    #0            ; establecee STINCOL = ALARMA sin
machacar
```

```
                STAA    STINCOL      ; otro proceso de introduccion de
trama
                JSR     INLISTAESPERA      ; Se inserta trama en Lista
de Espera
                JSR     INTCOLTX           ; Despues se inserta la
trama en Cola
                STAB    STINCOL            ; de Transmision

                PULY
                PULX
                PULB
                PULA
                RTS


*-------------------------------*
*   Subrutinas de interrupcion  *
*-------------------------------*

* Subrutina AvisarPc: esta subrutina se activa cuando se ha producido
algun
*                     error fatal en la red, ya sea por fallo de
comunicacion
*                     de conexion o cualquier otro tipo de fallo. El
nodo
*                     genera una trama para el PC, solicitando
reconfiguracion
*                     de la red


AVISARPC        LDAB    SCCR2            ; Desactivacion de
interrupciones
                ANDB    #%10001000       ; de recepcion SCI
                STAB    SCCR2
                LDX     #BASEALARM
                LDAB    #%11110000       ; Direccion destino = PC
                STAB    0,X
                LDAB    #%00100000       ; Trama prioritaria
                STAB    1,X
                LDAB    #%11000000       ; Tipo trama = peticion
configuracion
                STAB    2,X
                LDAB    #%00111100       ; CRC de esta trama es fijo.
                STAB    3,X
                LDAB    #%11111000       ; CRC = 0011110011111000
                STAB    4,X
                LDAA    STINCOL
                LDAB    #0
                STAB    STINCOL
                JSR     INTCOLTX         ; Se introduce la trama en
cola de
                STAA    STINCOL          ; transmision
                RTI                      ; Salida de la interrupcion
```

```
* Subrutina Interrupcion Timer: esta rutina es la encargada de
incrementar en
*                              un paso el contador global, cada vez
que se
*                              desborda el TCNT

INTERRUPTIMER   LDAA    PRCD          ; PRCD = PRCD + 1
                INCA                  ; Se registra por lo tanto un
nuevo
                STAA    PRCD          ; desbordamiento
                LDAA    #$80
                STAA    TFLG2         ; Reset flag TOF del registro
TFLG2
                RTI


* Subrutina Interrupcion SCI: esta rutina es la encargada de gestionar
los
*                             eventos SCI que dan lugar a
interrupcion, como
*                             son la recepcion de un byte, la
finalizacion de
*                             transmision de un byte, o la ocurrencia
de error
*                             en la recepcion de un byte


INTERRUPSCI     LDAA    SCSR
                ANDA    #%00100000
                CMPA    #%00100000
                BEQ     ET004B          ; Se ha transmitido un byte
                LDAB    CNTTX
                CMPB    CNTTXMAX
                BEQ     ET003B          ; No era el ultimo byte de
trama
                LDX     BASETX
                ABX
                LDAA    0,X
                LDAB    SCSR
                STAA    SCDR            ; Se transmite el siguiente
byte de
                LDAB    CNTTX           ; trama y se incrementa el
contador
                INCB                    ; de bytes transmitidos
                STAB    CNTTX
                RTI

ET003B          LDAA    PRCC            ; Era el ultimo byte de trama
por lo que
                ORAA    #%10000000      ; se actualizan las colas
                STAA    PRCC            ; Transmision Completa = TRUE
                JSR     ACTCOLAS
                LDAA    SCCR2
                ANDA    #%00101100
                STAA    SCCR2
                RTI

ET004B          LDAA    SCSR            ; Se ha recibido un byte en
SCI
                ANDA    #%00001110
```

```
                BEQ     ET007B          ; Salto si no ha habido error
en rxon
                LDAA    PRCC
                ANDA    #%00000001      ; Salto si ya se habia
producido algun
                CMPA    #%00000001      ; error en la recepcion de la
trama
                BEQ     ET005B
                LDAA    PRCC            ; Si la trama no era erronea
entonces
                ORAA    #%00000001      ; se hace Trama erronea = TRUE
                STAA    PRCC
                ANDA    #%00000010
                CMPA    #%00000010
                BNE     ET007B          ; Si se esta recibiendo trama
entonces
                LDAA    PRCC            ; se hace Recepcion Incompleta
= TRUE
                ORAA    #%00000100
                STAA    PRCC
                JSR     ACTCOLAS        ; Se libera la posicion en que
ET005B          LDAA    CNTRX           ; se estaba recibiendo trama
                CMPA    #2
                BNE     ET007B       ; Si byte erroneo = 2§ byte ->
error total
                SWI                  ; se avisa al PC de esta
circunstancia
ET006B          RTI

ET007B          LDAB    CNTRX
                CMPB    #0
                BNE     ET009B       ; Si se ha recibido el primer
byte de trama
                LDAA    SCSR         ; Reset de los flags de rxon y
error
                LDAA    SCDR         ; Lectura del byte recibido
                STAA    PRIMBYTE     ; Primbyte = byte recibido
                INCB                 ; CntRx = 1 (proximo byte a
recibir)
                STAB    CNTRX
                LDAB    PRCN
                ANDB    #%10000000
                CMPB    #%10000000
                BEQ     ET008B
                CMPA    #$00
                BRA     ET008B  ; Si Micro Activo == FALSE y trama no
multicast
                SWI             ; error total, avisar al PC de tal
circunstancia
ET008B          RTI

ET009B          CMPB    #1
                BEQ     ET009B1
                JMP     ET017B
ET009B1         LDAA    PRIMBYTE     ; Se ha recibido el segundo byte
de trama
                CMPA    #0
                BEQ     ET009B2
                EORA    PRCN         ; Si la trama es multicast o va
dirigida
```

```
                ANDA    #%00001111   ; al nodo se almacena en Colas A
(Proces)
                CMPA    #0
                BNE     ET011B

ET009B2         LDAA    SCSR         ; Reset de Flags de Rxon
                LDAA    SCDR
                ANDA    #%00100000
                CMPA    #%00100000
                BNE     ET010B       ; Si la trama es prioritaria y
ColaA1 no
                LDAA    STCOLAA1     ; llena, se calcula la direccion
base de
                CMPA    #255         ; primera posicion libre en
ColaA1
                BEQ     ET010B
                LDAB    ULTA1
                ASLB
                ASLB
                ASLB
                ASLB
                ASLB
                LDX     #COLAA1
                ABX
                STX     BASERX       ; BaseRx = dicha posicion
                BRA     ET014B

ET010B          LDAA    STCOLAA0
                CMPA    #255
                BEQ     ET015B       ; Si trama no prioritaria y
ColaA0 no llena
                LDAB    ULTA0        ; se calcula la direccion base de
primera
                ASLB                 ; posicion libre de ColaA0
                ASLB
                ASLB
                ASLB
                ASLB
                LDX     #COLAA0
                ABX
                STX     BASERX       ; BaseRx = dicha posicion
                BRA     ET014B

ET011B          LDAA    SCSR         ; Reset de los Flags de Rxon
                LDAA    SCDR
                ANDA    #%00100000
                CMPA    #%00100000
                BNE     ET012B       ; Trama de retransmision pura
                LDAB    STCOLAB1
                CMPB    #255
                BEQ     ET012B       ; Si la trama es prioritaria y
ColaB1 no
                LDAB    ULTB1        ; llena entonces se calcula
direccion base
                ASLB                 ; de primera posicion libre de
ColaB1
                ASLB
                ASLB
                ASLB
                ASLB
```

```
                LDX     #COLAB1
                ABX
                STX     BASERX          ; BaseRx = dicha direccion
                BRA     ET013B

ET012B          LDAA    STCOLAB0
                CMPA    #255
                BEQ     ET015B          ; Si trama no es prioritaria y
ColaB0 no
                LDAB    ULTB0           ; llena entonces se calcula
direcion base
                ASLB                    ; de primera posicion libre de
ColaB0
                ASLB
                ASLB
                ASLB
                ASLB
                LDX     #COLAB0
                ABX
                STX     BASERX          ; BaseRx = dicha direccion

ET013B          LDAA    PRCC
                ORAA    #%00100000
                STAA    PRCC            ; Insertando en Cola de Txon =
TRUE
                JSR     ACTCOLAS        ; Actualizacion de colas
                BRA     ET016B

ET014B          LDAA    PRCC
                ORAA    #%00010000
                STAA    PRCC            ; Insertando en Cola de Rxon =
TRUE
                JSR     ACTCOLAS        ; Actualizacion de Colas
                BRA     ET016B

ET015B          LDAA    PRCC
                ORAA    #%00000001      ; Trama Erronea = TRUE
                STAA    PRCC
ET016B          LDAA    PRCC
                ORAA    #%00000010
                STAA    PRCC            ; Recepcion en Curso = TRUE
                ANDA    #%00000001
                CMPA    #%00000001
                BEQ     ET018B          ; Si trama no erronea, primer
byte de
                LDAA    PRIMBYTE        ; posicion de recepcion = primer
byte
                LDX     BASERX
                STAA    0,X

ET017B          LDAB    PRCC
                ANDB    #%00000001
                CMPB    #%00000001
                BEQ     ET018B          ; Si trama no erronea byte en
lugar
                LDAB    CNTRX           ; indicado por CntRx en posicion
de
                LDX     BASERX          ; recepcion = Byte recibido por
SCI
                ABX
```

```
                LDAB    SCSR
                LDAB    SCDR
                STAB    0,X

ET018B          LDAB    #2              ; Si byte recibido == 3§ byte de
trama
                CMPB    CNTRX
                BNE     ET019B
                LDAB    SCDR            ; CntRxMax = longitud campo datos
+ 5
                ANDB    #%00011111
                ADDB    #5
                STAB    CNTRXMAX        ; Si CntRxMax > 32 bytes -> error
total
                CMPB    #32
                BLE     ET019B
                SWI                     ; Se avisa de tal circunstancia
al PC
                RTI

ET019B          LDAA    CNTRX           ; CntRx = CntRx + 1
                INCA
                STAA    CNTRX
                CMPA    CNTRXMAX
                BEQ     ET020B
                RTI

ET020B          LDAA    PRCC
                ANDA    #%11000000
                STAA    PRCC            ; CntRx == CnRxMax -> Recepcion
completa
                LDX     #$0000          ; Reset de todos los flags de
recepcion
                STX     BASERX          ; BaseRx toma un valor basura
                LDAA    #0              ; CntRx = 0
                STAA    CNTRX
                RTI

                END
```

ANEXO V

CÓDIGO FUENTE VISUAL BASIC

Módulo – modulo

```
' Este módulo contiene las variables, funciones y procedimientos globales que
' usa el programa
Option Explicit

' DEFINICION DE LAS ESTRUCTURAS QUE SE USARÁN EN EL PROGRAMA

    Public Type PosicionCola    ' Posición de memoria en cuallquiera de las colas
      trama(31) As Byte      ' Matriz de 32 bytes. Recogen los bytes de trama
      Longitud As Byte       ' Longitud = nº de bytes de la trama
    End Type

    Public Type Cola              ' Representa una cola
      Posicion(31) As PosicionCola ' Son las 32 posiciones de la Cola en cuestión
      PrimeroCola As Byte         ' Indice de primera posición ocupada de la Cola
      UltimoCola As Byte          ' Indice de primera posición libre de la Cola
      ColaVacia As Boolean        ' Flag que indica si la Cola está vacía
      ColaLlena As Boolean        ' Flag que indica si la Cola está llena
    End Type

    Public Type PosicionLista   ' Representa una posición en una lista de tramas
      trama(31) As Byte      ' Octetos de la trama
      Longitud As Byte       ' Nº de octetos de la trama
      Ocupada As Boolean     ' Flag que indica si la posición está o no ocupada
      TimerTrama As Date     ' Contador de vencimiento de la trama
      Transmisiones As Byte  ' Contador de retransmisiones de la trama
    End Type

    Public Type infonodo       ' Información sobre el tipo de nodo
      informacion(6) As Byte  ' Dicha información se codifica en 7 bytes
    End Type

    Public Type infosistema        ' Información sobre el Sistema
```

```
    Numeronodos As Byte        ' Nº de nodos del Sistema
    tiponodos(14) As infonodo  ' Array de información de cada nodo
    Configurado As Boolean     ' Flag que indica si el sistema está configurado
End Type

Public Type mensaje        ' Información recibida desde los nodos
    Texto(31) As String    ' Ultimos 32 mensajes recibidos desde los nodos
    siguiente As Byte      ' Posición donde guardar el siguiente mensaje
End Type


' VARIABLES GLOBALES DEL PROGRAMA

Public ColaA0 As Cola                ' Cola de procesado sin prioridad
Public ColaA1 As Cola                ' Cola de procesado con prioridad
Public ColaB0 As Cola                ' Cola de transmisión sin prioridad
Public ColaB1 As Cola                ' Cola de transmisión con prioridad
Public ListaEspera(31) As PosicionLista  ' Lista de tramas en espera de asentimiento

Public TramaRecibida(31) As Byte       ' Recoge la trama recién recibida
Public Sistema As infosistema          ' Contiene información sobre el sistema
Public ListoParaEnviar As Boolean      ' Indica si hay transmisión en curso
Public RecibiendoCabecera As Boolean   ' Indica si los próximos caracteres a recibir son de
datos o cabecera
Public ErrorTramaRx As Boolean         ' Indica si la trama que se está recibiendo sufrió algún
tipo de error
Public configurando As Boolean         ' Indica si el Sistema está configurado o no
Public Alarmas As mensaje              ' Almacena los mensajes de alarma recibidos desde los
micros
Public Respuestas As mensaje           ' Almacena los mensajes de respuesta recibidos desde los
micros



' DECLARACIÓN DE LAS FUNCIONES Y PROCEDIMIENTOS GLOBALES

' Este procedimiento permite obtener el CRC o el síndrome de una trama que se va a transmitir o
que se ha recibido
Public Sub calculocrc(ByRef matriz() As Byte)
    Dim bytesiguiente As Byte, i As Byte, j As Byte
```

```
        Dim crc As Long
        Dim bit1 As Boolean
        crc = matriz(0) * 2 ^ 8  ' Se almacena en la variable crc el contenido de los dos
        crc = crc + matriz(1)    ' primeros bytes de la trama
        For i = 0 To ((matriz(2) And 31) + 2)  ' Se itera hasta que se llega al último byte de la trama
          bytesiguiente = matriz(i + 2)      ' bytesiguiente actúa como registro de desplazamiento de
          For j = 0 To 7                     ' de los bits que van entrando en la división binaria
            If bytesiguiente < 128 Then
              bit1 = False             ' Siguiente bit del dividendo es 0
            Else
              bit1 = True                        ' Siguiente bit del dividendo es 1 pero
              bytesiguiente = bytesiguiente And 127  ' se pone a 0 por razones de calculo
            End If
            bytesiguiente = bytesiguiente * 2    ' Se desplaza el siguiente bit del dividendo (bit 7)
            If crc < 32768 Then                  ' Se efectúa la division binaria, siendo el divisor
              crc = crc * 2                      ' el polinomio generador del CRC-16 del CCITT
              If bit1 = True Then crc = crc + 1   ' Si el primer bit del resto (provisional) es 0,
            Else                                 ' el dividendo es el resto desplazado 1 bit a la izda,
              crc = (crc Xor 34832) * 2          ' realimentado por la dcha con el bit correspondiente
              If bit1 = False Then crc = crc + 1  ' Si el primer bit del resto(provisional) es 1, se divide
            End If                               ' por el divisor, se desplaza hacia la izda 1 bit y se
          Next j                                 ' realimenta con el bit correspondiente
        Next i
        matriz((matriz(2) And 31) + 3) = crc \ 256      ' El crc calculado, de 16 bits, se almacena en los
dos
        matriz((matriz(2) And 31) + 4) = crc Mod 256    ' últimos bytes de la trama
      End Sub


      ' Esta función devuelve el valor numérico de un numero hexadecimal expresado como una
cadena de caracteres
      Public Function Valor(ByVal Texto As String, ByVal NumCaracteres As Integer) As Currency
        Dim i As Byte
        Dim j As String
        Valor = 0
          For i = 1 To NumCaracteres    ' Para cada carácter calcula su valor numérico y lo
            j = Mid(Texto, i, 1)      ' pondera por la potencia de 16 conveniente
            Select Case j
            Case "A"
              Valor = Valor + 10 * (16 ^ Abs(i - NumCaracteres))
```

```
            Case "B"
                Valor = Valor + 11 * (16 ^ Abs(i - NumCaracteres))
            Case "C"
                Valor = Valor + 12 * (16 ^ Abs(i - NumCaracteres))
            Case "D"
                Valor = Valor + 13 * (16 ^ Abs(i - NumCaracteres))
            Case "E"
                Valor = Valor + 14 * (16 ^ Abs(i - NumCaracteres))
            Case "F"
                Valor = Valor + 15 * (16 ^ Abs(i - NumCaracteres))
            Case "0"
                Valor = Valor + 0 * (16 ^ Abs(i - NumCaracteres))
            Case "1"
                Valor = Valor + 1 * (16 ^ Abs(i - NumCaracteres))
            Case "2"
                Valor = Valor + 2 * (16 ^ Abs(i - NumCaracteres))
            Case "3"
                Valor = Valor + 3 * (16 ^ Abs(i - NumCaracteres))
            Case "4"
                Valor = Valor + 4 * (16 ^ Abs(i - NumCaracteres))
            Case "5"
                Valor = Valor + 5 * (16 ^ Abs(i - NumCaracteres))
            Case "6"
                Valor = Valor + 6 * (16 ^ Abs(i - NumCaracteres))
            Case "7"
                Valor = Valor + 7 * (16 ^ Abs(i - NumCaracteres))
            Case "8"
                Valor = Valor + 8 * (16 ^ Abs(i - NumCaracteres))
            Case "9"
                Valor = Valor + 9 * (16 ^ Abs(i - NumCaracteres))
            Case Else
                Valor = 65536    ' En caso de que haya un carácter que no se corresponda
                Exit Function    ' con ningún digito hexadecimal, se registra el error dando
        End Select               ' el valor basura que aparece como salida
    Next i
End Function

' Esta función devuelve una cadena de caracteres que representa en hexadecimal al número
' que se le pasa como parámetro
Public Function Conversion(ByVal numero As Currency) As String
```

```
    Dim i As Byte
    For i = 0 To 3     ' Tenemos números hexadecimales de 4 cifras
      Select Case numero \ (16 ^ Abs(i - 3))  ' Se factoriza el numero como suma de distintas potencias de 16
      Case 15                    ' ponderadas por diferentes coeficientes(entre 0 y 15), que son
        Conversion = Conversion & "F"       ' los que se codifican como un "carácter hexadecimal"
      Case 14
        Conversion = Conversion & "E"
      Case 13
        Conversion = Conversion & "D"
      Case 12
        Conversion = Conversion & "C"
      Case 11
        Conversion = Conversion & "B"
      Case 10
        Conversion = Conversion & "A"
      Case Else
        Conversion = Conversion & numero \ (16 ^ Abs(i - 3))
      End Select
      numero = numero Mod (16 ^ Abs(i - 3))
    Next i
  End Function


  ' Este procedimiento envía a los nodos una trama de inicialización del sistema
  Public Sub EnvioInicializacion()
    Dim trama(5) As Byte
    Dim Salida As Variant
    ' Se definen los campos de la trama de inicialización
    trama(0) = 0      ' Direccion Multicast
    trama(1) = 32     ' No requiere asentimiento, con prioridad
    trama(2) = 225    ' Trama de inicialización, 1 byte de datos
    trama(3) = 1      ' Datos = 1
    trama(4) = 181    ' CRC
    trama(5) = 100    ' CRC
    Salida = trama()
    ' Se transmite la trama de inicialización
    frmPantalla.MSComm1.Output = Salida
  End Sub
```

```
' Este procedimiento permite la gestión y encolado de una trama recibida
Public Sub RecepcionTrama(ByRef trama() As Byte)
   Dim i As Byte
   ' Si la trama recibida es una trama perdida, se elimina de la
   ' circulación sin más y se abandona el procedimiento
   If (trama(1) And 128) = 128 Then
      Exit Sub
   End If
   ' Si la trama no es una trama perdida se ejecuta el siguiente código:
   If (trama(0) And 15) = 0 Then
   ' En este caso la trama tiene como destino al PC
      If (trama(1) And 32) = 32 Then
      ' En este caso la trama es prioritaria -> ColaA1
         If ColaA1.ColaLlena = False Then
         ' Si hay espacio en ColaA1 se introduce la trama
            For i = 0 To ((trama(2) And 31) + 4)
               ' Se copia cada byte de trama en la posicion correspondiente
               ColaA1.Posicion(ColaA1.UltimoCola).trama(i) = trama(i)
            Next i
            ' Se da al campo longitud el valor de la longitud del campo de
            ' datos de la trama
            ColaA1.Posicion(ColaA1.UltimoCola).Longitud = (trama(2) And 31)
            ' Se actualiza la cola circular
            ColaA1.UltimoCola = (ColaA1.UltimoCola + 1) And 31
            If ColaA1.UltimoCola = ColaA1.PrimeroCola Then ColaA1.ColaLlena = True
            ColaA1.ColaVacia = False
            Exit Sub
         End If
      End If
      ' Aqui se llega si la trama no es prioritaria, o si siéndolo
      ' no hay espacio libre en ColaA1
      ' Si ColaA0 está llena, la trama se pierde
      If ColaA0.ColaLlena = True Then Exit Sub
      ' Si hay espacio en ColaA0 se introduce la trama
      For i = 0 To ((trama(2) And 31) + 4)
         ' Se copia cada byte de trama en la posicion correspondiente
         ColaA0.Posicion(ColaA0.UltimoCola).trama(i) = trama(i)
      Next i
```

```
        ' Se da al campo longitud el valor de la longitud del campo de
        ' datos de la trama
        ColaA0.Posicion(ColaA0.UltimoCola).Longitud = (trama(2) And 31)
        ' Se actualiza la cola circular
        ColaA0.UltimoCola = (ColaA0.UltimoCola + 1) And 31
        If ColaA0.UltimoCola = ColaA0.PrimeroCola Then ColaA0.ColaLlena = True
        ColaA0.ColaVacia = False
        Exit Sub
    Else
    ' La trama es para retransmisión pura, así que se aciva el bit de
    ' trama perdida y se introduce la trama en cola de transmisión
        trama(1) = trama(1) Or 128          ' Trama perdida = TRUE
        trama((trama(2) And 31) + 3) = 0    ' Se prepara el CRC para ser recalculado
        trama((trama(2) And 31) + 4) = 0
        Call calculocrc(trama())            ' Se recalcula el CRC
        If (trama(1) And 32) = 32 Then
            ' En este caso la trama es prioritaria -> ColaB1
            If ColaB1.ColaLlena = False Then
            ' Si hay espacio en ColaB1 se introduce la trama
                For i = 0 To ((trama(2) And 31) + 4)
                    ' Se copia cada byte de trama en la posicion correspondiente
                    ColaB1.Posicion(ColaB1.UltimoCola).trama(i) = trama(i)
                Next i
                ' Se da al campo longitud el valor de la longitud del campo de
                ' datos de la trama
                ColaB1.Posicion(ColaB1.UltimoCola).Longitud = (trama(2) And 31)
                ' Se actualiza la cola circular
                ColaB1.UltimoCola = (ColaB1.UltimoCola + 1) And 31
                If ColaB1.UltimoCola = ColaB1.PrimeroCola Then ColaB1.ColaLlena = True
                ColaB1.ColaVacia = False
                Exit Sub
            End If
        End If
        ' Aqui se llega si la trama no es prioritaria, o si siéndolo
        ' no hay espacio libre en ColaB1
        ' Si ColaB0 está llena, la trama se pierde
        If ColaB0.ColaLlena = True Then Exit Sub
        ' Si hay espacio en ColaB0 se introduce la trama
        For i = 0 To ((trama(2) And 31) + 4)
            ' Se copia cada byte de trama en la posicion correspondiente
```

```
            ColaB0.Posicion(ColaB0.UltimoCola).trama(i) = trama(i)
        Next i
        ' Se da al campo longitud el valor de la longitud del campo de
        ' datos de la trama
        ColaB0.Posicion(ColaB0.UltimoCola).Longitud = (trama(2) And 31)
        ' Se actualiza la cola circular
        ColaB0.UltimoCola = (ColaB0.UltimoCola + 1) And 31
        If ColaB0.UltimoCola = ColaB0.PrimeroCola Then ColaB0.ColaLlena = True
        ColaB0.ColaVacia = False
    End If
End Sub


' Este procedimiento se ejecuta cada vez que se recibe una trama durante
' el proceso de configuración del sistema
Public Sub ConfiguracionenCurso(ByRef trama() As Byte)
    Static contador As Byte
    Dim Ack(4) As Byte
    Dim SalidaAck As Variant
    Dim i As Byte
    Dim ID As Byte
    ' Si se trata de trama perdida, simplemente se desecha
    If (trama(1) And 128) = 128 Then Exit Sub
    ' Se calcula el síndrome de la trama recibida
    Call calculocrc(trama)
    ' Según el tipo de trama se ejecuta cierta sección de código
    Select Case (trama(2) And 224)
        Case 224
        ' Si se trata de la trama de inicialización
        If (trama(4) = 0) And (trama(5) = 0) Then
            ' Si el síndrome es 0 (trama recibida sin errores),
            ' la configuración del sistema se lleva a cabo
            ' El campo de datos de la trama es el número de nodos que
            ' componen el sistema + 1
            Sistema.Numeronodos = (trama(3) And 31) - 1
            ' Se inicializa a 0 el contador de mensajes de los nodos en los
            ' que informan de su naturaleza
            contador = 0
            ' Se establece el estado "configurando", que implica que el PC se
            ' dispone a recibir tramas de registro de nodo
```

```
            configurando = True
        Else
            ' Si se produjo error en la trama, se aborta el proceso de configuración
            ' y se da un mensaje de alarma por pantalla
            MsgBox " Se ha producido un error durante la configuración. " & Chr(10) & "      Vuelva
a configurar el sistema. ", vbExclamation
            Call frmPantalla.AbandonarGestion_Click
        End If

        Case 160
        ' Si se trata de una trama de registro de nodo
        ' Si el PC no se encontraba en estado "configurando", se desecha la trama, para
        ' evitar que tramas de un intento de configuración anterior perturben el sistema
        If configurando = False Then Exit Sub
        ' Se asiente o rechaza la trama de registro de nodo
        Ack(0) = trama(0) \ 16
        ID = trama(0) \ 16
        Ack(1) = 160
        If (trama(10) = 0) And (trama(11) = 0) Then
            ' Si la trama se recibió sin errores, se asiente
            Ack(2) = 32
        Else
            ' Si la trama se recibió con errores, se rechaza
            Ack(2) = 64
        End If
        ' Se calcula el CRC de la trama de asentimiento/rechazo generada, y se envía
        Call calculocrc(Ack)
        SalidaAck = Ack()
        frmPantalla.MSComm1.Output = SalidaAck
        If Ack(2) = 32 Then
            ' Si la trama se recibió sin errores, se registra información contenida en
            ' su campo de datos relativa a la naturaleza del nodo que la emitió
            For i = 0 To 6
                Sistema.tiponodos(ID).informacion(i) = trama(i + 3)
            Next i
            ' Se incrementa el contador de nodos registrados
            contador = contador + 1
            If contador = Sistema.Numeronodos Then
                ' Si todos los nodos del sistema se han registrado, el sistema queda
                ' configurado con éxito y se habilita la gestión del sistema
```

```
                    Call IniciarGestion
                End If
            End If
            Case Else
            ' Si se recibe cualquier otro tipo de trama antes de que el sistema esté configurado,
            ' implica que se ha producido algún error que puede ser catastrófico para la configuración,
            ' por lo que se aborta la misma y se da un mensaje de alarma por pantalla
                MsgBox " Se ha producido un error durante la configuración. " & Chr(10) & "      Vuelva
a configurar el sistema. ", vbExclamation
                Call frmPantalla.AbandonarGestion_Click
            End Select
        End Sub



        ' Este procedimiento permite iniciar la gestión del sistema después de que este haya
        ' sido configurado con éxito
        Public Sub IniciarGestion()
            Sistema.Configurado = True
            ' Se prepara la interface con el usuario para que este pueda acceder a los
            ' formularios que permiten la gestión del sistema
            frmPantalla.TimerEspera.Enabled = False
            frmPantalla.FrameEspera.Visible = False
            frmPantalla.AbortarEspera.Enabled = False
            frmPantalla.FrameGestion.Enabled = True
            frmPantalla.Label1Gestion.Enabled = True
            frmPantalla.Label2Gestion.Enabled = True
            frmPantalla.Label3Gestion.Enabled = True
            frmPantalla.Label4Gestion.Enabled = True
            frmPantalla.InformacionGestion.Enabled = True
            frmPantalla.LecEscGestion.Enabled = True
            frmPantalla.ModificarGestion.Enabled = True
            frmPantalla.AbandonarGestion.Enabled = True
            frmPantalla.TimerAsentimientos.Enabled = True
            frmPantalla.TimerTransmisiones.Enabled = True
            frmPantalla.TimerProcesado.Enabled = True
        End Sub

        ' Este procedimiento se encarga del procesado de las tramas recibidas. Es llamado por otro
        ' procedimiento periódicamente
```

```
Public Sub ProcesarTramas()
    Dim auxiliar As Long
    Dim i As Byte, CRCposicion As Byte
    ' Chequea el estado de las colas A de procesado, de modo que si están vacías, este
    ' procedimiento no hace nada
    If ColaA1.ColaVacia = False Then
        ' Si ColaA1 no está vacía, se procesa la primera trama en cola
        ' Se calcula el síndrome de esta trama
        Call calculocrc(ColaA1.Posicion(ColaA1.PrimeroCola).trama())
        If ((ColaA1.Posicion(ColaA1.PrimeroCola).trama(1) And 64) = 64) Then
            ' Si la trama requiere asentimiento, se genera el asent/rechazo
            Call AsentirRechazar(ColaA1.Posicion(ColaA1.PrimeroCola).trama())
        End If
        auxiliar =
ColaA1.Posicion(ColaA1.PrimeroCola).trama(ColaA1.Posicion(ColaA1.PrimeroCola).Longitud + 3)
        auxiliar = auxiliar +
ColaA1.Posicion(ColaA1.PrimeroCola).trama(ColaA1.Posicion(ColaA1.PrimeroCola).Longitud + 4)
        If (auxiliar <> 0) Then
            ' Si la trama que se recibió tenía errores, no se procesa, sino que se desecha
            ' Se actualiza la cola circular correspondiente
            ColaA1.ColaLlena = False
            ColaA1.PrimeroCola = (ColaA1.PrimeroCola + 1) And 31
            If ColaA1.PrimeroCola = ColaA1.UltimoCola Then ColaA1.ColaVacia = True
            Exit Sub
        End If
        ' Si la trama se recibió sin errores, se procesa según sea su tipo
        Select Case (ColaA1.Posicion(ColaA1.PrimeroCola).trama(2) And 224)
            Case 0
                ' Si la trama es de información se llama al procedimiento encargado de procesarla
                Call GestionInformacion(ColaA1.Posicion(ColaA1.PrimeroCola).trama())
            Case 32
                ' Si la trama es de asentimiento, se libera la posición de la Lista de Espera indicada por
el
                ' campo identificador numérico de trama
                ListaEspera(ColaA1.Posicion(ColaA1.PrimeroCola).trama(1) And 31).Ocupada =
False
            Case 64
                ' Si la trama es de rechazo se llama al procedimiento encargado de procesarla
                Call GestionRechazo(ColaA1.Posicion(ColaA1.PrimeroCola).trama())
            Case 128
```

```
                    ' Si la trama es de alrma se llama al procedimiento encargado de procesarla
                    Call GestionAlarma(ColaA1.Posicion(ColaA1.PrimeroCola).trama())
                Case 192
                    ' Si la trama es de petición de configuración se anula la configuración vigente y
                    ' se muestra un mensaje de alarma por pantalla
                    Call frmPantalla.AbandonarGestion_Click
                    MsgBox " Han surgido problemas en la configuración." & Chr(10) & "        Vuelva a
configurar el sistema", vbExclamation
            End Select
            ' Se actualiza la cola circular correspondiente
            ColaA1.ColaLlena = False
            ColaA1.PrimeroCola = (ColaA1.PrimeroCola + 1) And 31
            If ColaA1.PrimeroCola = ColaA1.UltimoCola Then ColaA1.ColaVacia = True
            Exit Sub
        End If
        If ColaA0.ColaVacia = False Then
        ' Se procede de igual modo con las tramas recibidas sin prioridad
            Call calculocrc(ColaA0.Posicion(ColaA0.PrimeroCola).trama())
            If ((ColaA0.Posicion(ColaA0.PrimeroCola).trama(1) And 64) = 64) Then
                Call AsentirRechazar(ColaA0.Posicion(ColaA0.PrimeroCola).trama())
            End If
            auxiliar =
ColaA0.Posicion(ColaA0.PrimeroCola).trama(ColaA0.Posicion(ColaA0.PrimeroCola).Longitud + 3)
            auxiliar = auxiliar +
ColaA0.Posicion(ColaA0.PrimeroCola).trama(ColaA0.Posicion(ColaA0.PrimeroCola).Longitud + 4)
            If (auxiliar <> 0) Then
                ColaA0.ColaLlena = False
                ColaA0.PrimeroCola = (ColaA0.PrimeroCola + 1) And 31
                If ColaA0.PrimeroCola = ColaA0.UltimoCola Then ColaA0.ColaVacia = True
                Exit Sub
            End If
            Select Case (ColaA0.Posicion(ColaA0.PrimeroCola).trama(2) And 224)
                Case 0
                    Call GestionInformacion(ColaA0.Posicion(ColaA0.PrimeroCola).trama())
                Case 32
                    ListaEspera(ColaA0.Posicion(ColaA0.PrimeroCola).trama(1) And 31).Ocupada =
False
                Case 64
                    Call GestionRechazo(ColaA0.Posicion(ColaA0.PrimeroCola).trama())
                Case 128
```

```
                Call GestionAlarma(ColaA0.Posicion(ColaA0.PrimeroCola).trama())
            Case 192
                Call frmPantalla.AbandonarGestion_Click
                MsgBox " Han surgido problemas en la configuración." & Chr(10) & "        Vuelva a
configurar el sistema", vbExclamation
        End Select
        ColaA0.ColaLlena = False
        ColaA0.PrimeroCola = (ColaA0.PrimeroCola + 1) And 31
        If ColaA0.PrimeroCola = ColaA0.UltimoCola Then ColaA0.ColaVacia = True
    End If
End Sub


' Este procedimiento se encarga de procesar las tramas de rechazo recibidas, retransmitiendo
' las tramas correspondientes
Public Sub GestionRechazo(ByRef trama() As Byte)
    Dim j As Byte
    Dim ID As Byte
    ' Se toma de la trama de rechazo el identificador de la trama que hay que retransmitir
    ID = trama(1) And 31
    If (ListaEspera(ID).trama(1) And 32) = 32 Then
        ' Si la trama a retransmitir es prioritaria, se almacena en ColaB1, de transmisión
        ' con prioridad
        Do While ColaB1.ColaLlena = True
            ' Espera activa hasta que haya un hueco en la cola
            DoEvents
        Loop
        For j = 0 To ((ListaEspera(ID).trama(2) And 31) + 4)
            ' Se copia en la posición correspondiente de la cola de transmisión la trama
            ' a ser retransmitida byte a byte
            ColaB1.Posicion(ColaB1.UltimoCola).trama(j) = ListaEspera(ID).trama(j)
        Next j
        ' Se actualiza la cola circular correspondiente
        ColaB1.ColaVacia = False
        ColaB1.Posicion(ColaB1.UltimoCola).Longitud = ListaEspera(ID).trama(2) And 31
        ColaB1.UltimoCola = (ColaB1.UltimoCola + 1) And 31
        If ColaB1.UltimoCola = ColaB1.PrimeroCola Then ColaB1.ColaLlena = True
        ' Se le da al timer de vencimiento de la trama un valor basura mientras no sea
        ' físicamente retransmitida, para evitar que vuelva a ser introducida en cola
        ' de transmisión sin ser necesario
```

```
            ListaEspera(ID).TimerTrama = Time() + #12:00:59 AM#
        Else
        ' Si la trama a ser retransmitida es no prioritaria, se procede de igual modo pero con
        ' relación a ColaB0 de transmisión sin prioridad
            Do While ColaB0.ColaLlena = True
                DoEvents
            Loop
            For j = 0 To ((ListaEspera(ID).trama(2) And 31) + 4)
                ColaB0.Posicion(ColaB0.UltimoCola).trama(j) = ListaEspera(ID).trama(j)
            Next j
            ColaB0.ColaVacia = False
            ColaB0.Posicion(ColaB0.UltimoCola).Longitud = ListaEspera(ID).trama(2) And 31
            ColaB0.UltimoCola = (ColaB0.UltimoCola + 1) And 31
            If ColaB0.UltimoCola = ColaB0.PrimeroCola Then ColaB0.ColaLlena = True
            ListaEspera(ID).TimerTrama = Time() + #12:00:59 AM#
        End If
    End Sub

    ' Este procedimiento se encarga de generar el rechazo o  asentimiento de las tramas recibidas
    ' que lo requieren, según se haya recibido con o sin errores respectivamente
    Public Sub AsentirRechazar(ByRef trama() As Byte)
        Dim Ack(4) As Byte
        Dim j As Byte
        Dim crc As Long
        Dim aux As Currency
        ' Dirección origen = PC
        ' Dirección destino = Nodo que emitió la trama a asentir/rechazar
        Ack(0) = trama(0) \ 16
        ' El Ack tiene el mismo identificador numérico y prioridad que la trama que se asiente
        ' o rechaza, pero no requiere asentimiento, y lleva el Flag de trama perdida a 1
        Ack(1) = (trama(1) And 63) + 128
        j = trama(2) And 31
        aux = trama(j + 3)
        aux = aux + trama(j + 4)
        ' Según el sindrome calculado para la trama, se asiente o rechaza
        If aux = 0 Then
            ' Si el síndrome es 0, implica que no hubo errores durante la recepción, así
            ' que se genera asentimiento
            Ack(2) = 32
        Else
```

```
        ' Si el síndrome es distinto de 0, implica que hubo errores durante la recepción, así
        ' que se genera rechazo
        Ack(2) = 64
      End If
      ' Se calcula el CRC de la trama de asentimiento/rechazo generada
      Call calculocrc(Ack)
      If (Ack(1) And 32) = 32 Then
        ' Si la trama es prioritaria, se introduce en cola de transmisión con prioridad
        Do While ColaB1.ColaLlena = True
          ' Espera activa hasta que haya un hueco en la cola
          DoEvents
        Loop
        For j = 0 To 4
          ColaB1.Posicion(ColaB1.UltimoCola).trama(j) = Ack(j)
        Next j
        ' Se registra la longitud del campo de datos, que es nula
        ColaB1.Posicion(ColaB1.UltimoCola).Longitud = 0
        ' Se actualiza la cola circular correspondiente
        ColaB1.ColaVacia = False
        ColaB1.UltimoCola = (ColaB1.UltimoCola + 1) And 31
        If ColaB1.UltimoCola = ColaB1.PrimeroCola Then ColaB1.ColaLlena = True
      Else
        ' Se procede de igual modo con tramas no prioriarias, introduciéndolas en este caso
        ' en cola de trasnmisión sin prioridad
        Do While ColaB0.ColaLlena = True
          DoEvents
        Loop
        For j = 0 To 4
          ColaB0.Posicion(ColaB0.UltimoCola).trama(j) = Ack(j)
        Next j
        ColaB0.Posicion(ColaB0.UltimoCola).Longitud = 0
        ColaB0.ColaVacia = False
        ColaB0.UltimoCola = (ColaB0.UltimoCola + 1) And 31
        If ColaB0.UltimoCola = ColaB0.PrimeroCola Then ColaB0.ColaLlena = True
      End If
      DoEvents
    End Sub


    ' Este procedimiento se encarga de gestionar las tramas de información
```

```
Public Sub GestionInformacion(ByRef trama() As Byte)
    Dim i As Byte
    Dim numero As Currency
    Dim contador As Byte
    Dim nodo As Byte
    Dim direccion As String
    Dim dato As String
    nodo = trama(0) \ 16
    ' Mientras queden datos por analizar, se continúa iterando
    Do While i < (trama(2) And 31)
        direccion = ""
        dato = ""
        DoEvents
        Select Case trama(i + 3)
        ' Según el código de instrucción se analiza el "paquete" de aplicación de distinto
        ' modo
            Case 128
                ' Se trata de una respuesta a una petición de lectura de dato
                ' Se obtiene la dirección del dato solicitado
                numero = trama(i + 4)
                numero = numero * 256
                numero = numero + trama(i + 5)
                ' Se expresa dicha dirección como una cadena de caracteres de
                ' 4 cifras en hexadecimal
                direccion = Conversion(numero)
                ' El dato solicitado se expresa como una cadena de caracteres de
                ' 2 cifras en hexadecimal
                dato = Conversion(trama(i + 6))
                dato = Mid(dato, 3, 2)
                ' Se actualiza el contador de datos, haciendo que apunte al primer
                ' byte de la siguiente instrucción
                i = i + 4
                ' Se genera texto de respuesta con la información del dato
                Respuestas.Texto(Respuestas.siguiente) = "Origen: nodo " & nodo & ". Posición $" &
direccion & ". Dato $" & dato & Chr(13) & Chr(10)
                Respuestas.siguiente = (Respuestas.siguiente + 1) And 31
                contador = contador + 1
            Case 10
                ' Se trata de un mensaje de programa copiado en RAM con éxito
                ' Se genera texto de respuesta correspondiente
```

```
                Respuestas.Texto(Respuestas.siguiente) = "Origen: nodo " & nodo & ". Instalación de
Programa            local en Ram completada con éxito." & Chr(13) & Chr(10)
                Respuestas.siguiente = (Respuestas.siguiente + 1) And 31
                ' Se actualiza el contador de datos, haciendo que apunte al primer
                ' byte de la siguiente instrucción
                 i = i + 1
                contador = contador + 1
              End Select
            Loop
            Unload frmmensajes
            ' Se informa al usuario del número de nuevos mensajes que ha recibido, y se muestra
            ' la pantalla de mensajes
            frmmensajes.Label4 = "Tiene " & contador & " mensaje(s) de respuesta nuevo(s) "
            frmmensajes.Show
          End Sub

          ' Este procedimiento se encarga de la gestión de las tramas de alarma recibidas
          Public Sub GestionAlarma(ByRef trama() As Byte)
            ' Aqui viene el código de gestión de alarmas que depende del tipo de alarmas que se definan
            Dim i As Byte
            Dim caracteres As Byte
            Dim mensaje As String
            Dim nodo As Byte
            caracteres = trama(2) And 31
            nodo = trama(0) \ 16
            mensaje = ""
            For i = 0 To (caracteres - 1)
              ' Se lee el mensaje de alarma carácter a carácter
              mensaje = mensaje & Chr(trama(i + 3))
            Next i
            ' Se almacena el mensaje de alarma en la cola de alramas correspondiente
            Alarmas.Texto(Alarmas.siguiente) = "Origen: nodo " & nodo & ". Mensaje: " & Chr(13) &
Chr(10)
            ' Se actualiza la cola de mensajes de alarma
            Alarmas.Texto(Alarmas.siguiente) = Alarmas.Texto(Alarmas.siguiente) & mensaje & Chr(13)
& Chr(10)
            Alarmas.siguiente = (Alarmas.siguiente + 1) And 31
            Unload frmmensajes
            ' Se muestra la pantalla de mensajes indicando que se ha recibido un nuevo mensaje
            ' de alarma
```

```
    frmmensajes.Label4 = "Tiene 1 mensaje de alarma nuevo "
    frmmensajes.Show
End Sub



' Este procedimiento se encarga de insertar una trama generada por el PC y que requiere
' asentimiento, en la Lista de Espera
Public Sub InsertarTrama(ByRef trama() As Byte, contador As Byte)
    Dim i As Byte, j As Byte
    Dim insertada As Boolean
   trama(2) = trama(2) + contador
    Do While insertada = False
      ' Se recorre la Lista de Espera hasta encontrar un hueco libre en ella
      i = 0
      Do While (insertada = False) And (i < 32)
        ' Se recorre la Lista de espera una y otra vez en espera activa hasta que
        ' aparece una posición libre en esta
        If ListaEspera(i).Ocupada = False Then
          ' Cuando se encuentra un hueco, se ocupa con la trama a insertar, y
          ' se actualizan los campos de dicha posición
          insertada = True
          ListaEspera(i).Ocupada = True
          ListaEspera(i).Longitud = contador
          ListaEspera(i).TimerTrama = Time() + #12:29:59 AM#
          ListaEspera(i).Transmisiones = 0
          ' Se da a la trama el identificador numérico de trama que se corresponde
          ' con el índice de la posición de la Lista de Espera que ocupa
          trama(1) = trama(1) Or i
          ' Se calcula el CRC de la trama, ya que todos sus campos están ya determinados
          Call calculocrc(trama())
          For j = 0 To 31
            ' Se copia la trama en su posición en la Lista de Espera byte a byte
            ListaEspera(i).trama(j) = trama(j)
          Next j
          ' A continuación se introduce la trama en Cola de Transmisión, con o sin
          ' prioridad, según corresponda
          If (trama(1) And 32) = 32 Then
            Do While ColaB1.ColaLlena = True
              ' Espera activa hasta que se produzca un hueco en la cola
```

```
                    DoEvents
                Loop
                ColaB1.Posicion(ColaB1.UltimoCola).Longitud = contador
                For j = 0 To 31
                  ' Se copia la trama en la posición que corresponde byte a byte
                  ColaB1.Posicion(ColaB1.UltimoCola).trama(j) = trama(j)
                Next j
                ' Se actualiza la cola circular correspondiente
                ColaB1.ColaVacia = False
                ColaB1.UltimoCola = (ColaB1.UltimoCola + 1) And 31
                If ColaB1.UltimoCola = ColaB1.PrimeroCola Then ColaB1.ColaLlena = True
              Else
                ' Se procede de igual modo con las tramas sin prioridad con respecto
                ' a ColaB0
                Do While ColaB0.ColaLlena = True
                  DoEvents
                Loop
                ColaB0.Posicion(ColaB0.UltimoCola).Longitud = contador
                For j = 0 To 31
                  ColaB0.Posicion(ColaB0.UltimoCola).trama(j) = trama(j)
                Next j
                ColaB0.ColaVacia = False
                ColaB0.UltimoCola = (ColaB0.UltimoCola + 1) And 31
                If ColaB0.UltimoCola = ColaB0.PrimeroCola Then ColaB0.ColaLlena = True
              End If
            End If
            ' Se incrementa el índice de Lista de Espera, para chequear la siguiente posición
            i = i + 1
          Loop
          DoEvents
        Loop
      End Sub
```

Formulario Pantalla

```
' Este formulario es el cuadro de diálogo principal. En él se distinguen
' tres partes fundamentales: presentación de la aplicación, espera de
' configuración y gestión del sistema. Son excluyentes, de modo que
' sólo una de ellas está activa en cada momento.

Option Explicit


' Este procedimiento restaura el diálogo de presentación de la aplicación
Public Sub AbandonarGestion_Click()
   MSComm1.PortOpen = False     ' Se cierra el puerto serie
   Call Form_Load               ' Se restaura el diálogo de presentación
End Sub

' Este procedimiento cancela el proceso de configuración del sistema
Private Sub AbortarEspera_Click()
   Call AbandonarGestion_Click
   ' Se muestra un mensaje por pantalla
   MsgBox " Ha cancelado el proceso de configuracion. ", vbInformation
End Sub

' Este procedimiento inicia el proceso de configuración del sistema
Private Sub AceptarConfiguracion_Click()
   ' Se reconfigura el formulario mostrando un diálogo de espera mientras se completa la
configuración
   Dim Desecho As Variant
   FrameEspera.Visible = True
   AbortarEspera.Enabled = True
   FrameConfiguracion.Enabled = False
   LabelConfiguracion.Enabled = False
   FramePuerto.Enabled = False
   COM1.Enabled = False
   COM2.Enabled = False
   SalirConfiguracion.Enabled = False
   AceptarConfiguracion.Enabled = False
   TimerEspera.Enabled = True
```

```
        ' Se configura el Puerto Serie seleccionado y se envía un Reset a los nodos
        ' Se abre el puerto de comunicaciones
        MSComm1.PortOpen = True
        ' Se envía el Reset a través de la línea DTR a nivel bajo
        MSComm1.RTSEnable = True
        ' Se establece comunicación de datos binarios
        MSComm1.InputMode = comInputModeBinary
        ' Se establece comunicación a 9600 bauios, sin paridad, con 8 bits de datos y 1 bit de stop
        MSComm1.Settings = "9600,N,8,1"
        ' Se establece la lectura de todo el buffer de entrada al acceder a Mscomm1.Input
        MSComm1.InputLen = 0
        ' Se usa una variable auxiliar para poder vaciar el buffer de entrada de datos basura
        Desecho = MSComm1.Input
        ' Se establece a 3 el número de bytes umbral de recepción
        MSComm1.InputLen = 3
        ' Se establece que se active el evento OnComm por comEvReceive cada vez que hay al menos
tres
        ' datos en el buffer de recepción
        MSComm1.RThreshold = 3
        ' Se establece que se active el evento OnComm por comEvSend cada vez que el buffer de
transmisión
        ' se quede vacío
        MSComm1.SThreshold = 1
        ' Se devuelve el reset de los nodos a situación inactiva
        MSComm1.RTSEnable = False
        ' Se llama al procedimiento que permite enviar a los nodos una trama de identificación
        Call Modulo.EnvioInicializacion
    End Sub


    ' Este procedimiento permite seleccionar el Puerto Serie 1 para establecer las comunicaciones
con los nodos
    Private Sub COM1_Click()
      MSComm1.CommPort = 1
    End Sub


    ' Este procedimiento permite seleccionar el Puerto Serie 2 para establecer las comunicaciones
con los nodos
```

```
Private Sub COM2_Click()
   MSComm1.CommPort = 2
End Sub


' Este procedimiento es el que se dispara cada vez que arranca la aplicación
Private Sub Form_Load()
   Dim i As Byte
   ' Inicialización de variables globales
   Sistema.Numeronodos = 15
   Sistema.Configurado = False
   ColaA0.ColaVacia = True
   ColaA0.ColaLlena = False
   ColaA0.PrimeroCola = 0
   ColaA0.UltimoCola = 0
   ColaA1.ColaVacia = True
   ColaA1.ColaLlena = False
   ColaA1.PrimeroCola = 0
   ColaA1.UltimoCola = 0
   ColaB1.ColaVacia = True
   ColaB1.ColaLlena = False
   ColaB1.PrimeroCola = 0
   ColaB1.UltimoCola = 0
   ColaB0.ColaVacia = True
   ColaB0.ColaLlena = False
   ColaB0.PrimeroCola = 0
   ColaB0.UltimoCola = 0
   For i = 0 To 31
      ListaEspera(i).Ocupada = False
   Next i
   configurando = False
   RecibiendoCabecera = True
   ErrorTramaRx = False
   ' Se configura el formulario, mostrando el diálogo para poder configurar el sistema
   Unload frminformacion
   Unload frmlecesc
   Unload frmmodificar
   FrameEspera.Visible = False
   TimerEspera.Enabled = False
   AbortarEspera.Enabled = False
```

```
    FrameGestion.Enabled = False
    Label1Gestion.Enabled = False
    Label2Gestion.Enabled = False
    Label3Gestion.Enabled = False
    Label4Gestion.Enabled = False
    InformacionGestion.Enabled = False
    LecEscGestion.Enabled = False
    ModificarGestion.Enabled = False
    AbandonarGestion.Enabled = False
    TimerAsentimientos.Enabled = False
    TimerTransmisiones.Enabled = False
    TimerProcesado.Enabled = False
    FrameConfiguracion.Enabled = True
    FramePuerto.Enabled = True
    LabelConfiguracion.Enabled = True
    COM1.Enabled = True
    COM2.Enabled = True
    COM2.Value = True
    MSComm1.CommPort = 2
    SalirConfiguracion.Enabled = True
    AceptarConfiguracion.Enabled = True
End Sub


' Este procedimiento permite mostrar el formulario de informacion
Private Sub InformacionGestion_Click()
    frminformacion.Show
End Sub


' Este procedimiento permite mostrar el formulario de lectura/escritura
Private Sub LecEscGestion_Click()
    frmlecesc.Show
End Sub


' Este procedimiento permite mostrar el formulario de modificación-copia de programa
Private Sub ModificarGestion_Click()
    frmmodificar.Show
End Sub
```

```
' Este procedimiento se dispara cada vez que se produce un error o un evento en las
comunicaciones
Public Sub MSComm1_OnComm()
   Static Cabecera(2) As Byte
   Static NumeroBytes As Byte
   Dim i As Byte
   Dim VarAuxiliar As Variant
  ' Se inicializan las variables estáticas en caso de reconfiguración
  ' del sistema
   Select Case MSComm1.CommEvent
     ' Gestión de los errores en la comunicación
     Case comEventFrame
     ' Se ha producido un error en la recepción de la trama
       ' Se registra dicho error
       ErrorTramaRx = True
      ' Si el error se produce en el byte que contiene
      ' la longitud del campo de datos, el error es fatal,
      ' y se debe reconfigurar el sistema
      If RecibiendoCabecera = True Then
        If MSComm1.InBufferCount = 2 Then
          ' Se muestra un mensaje de alarma por pantalla
          MsgBox " Se ha producido un error en la comunicación. " & Chr(10) & " Vuelva a
reconfigurar el sistema. ", vbExclamation
           Call AbandonarGestion_Click
           Exit Sub
        End If
      End If
    ' Gestión de los eventos en la comunicación
    Case comEvReceive
    ' Bytes recibidos por puerto serie. Cada trama se recibe
    ' en dos tiempos: 3 bytes de cabecera y (n+2) bytes de
    ' datos y crc
ETIQUETA:   If RecibiendoCabecera = False Then
          ' Si se esperan n+2 datos y crc, entonces:
          ' Se leen los bytes del buffer de entrada
          VarAuxiliar = MSComm1.Input
          ' Si se produjo algún error en la recepción de la trama,
          ' esta se desecha
```

```
        If ErrorTramaRx = True Then
            ErrorTramaRx = False
            Exit Sub
        End If
        ' Se integran la cabecera y los datos-crc en una
        ' sola trama
        For i = 0 To NumeroBytes + 2
            ' Los 3 primeros bytes de la trama son los que componen la cabecera
            If i <= 2 Then TramaRecibida(i) = Cabecera(i)
            If i >= 3 Then TramaRecibida(i) = VarAuxiliar(i - 3)
        Next i
        If Sistema.Configurado = True Then
            ' Si el sistema está configurado, se llama al procedimiento encargado de
            ' encolar las tramas recibidas adecuadamente
            Call RecepcionTrama(TramaRecibida())
        Else
            ' Las tramas recibidas durante el proceso de configuración reciben un
            ' tratamiento particular en el procedimiento al que se llama
            Call ConfiguracionenCurso(TramaRecibida())
        End If
        ' La proxima recepción será de una cabecera
        RecibiendoCabecera = True
        ' Se habilita el disparo de evento en recepción cuando se hayan recibido
        ' los 3 bytes de la cabecera
        MSComm1.RThreshold = 3
        ' Si ya hay suficientes datos en el buffer de entrada, se vuelven a leer
        ' datos
        If MSComm1.InBufferCount >= MSComm1.RThreshold Then
            GoTo ETIQUETA
        End If
    Else
        ' Si lo que se espera es una cabecera:
        ' Se lee el buffer de entrada
        MSComm1.InputLen = 3
        VarAuxiliar = MSComm1.Input
        ' Se escribe la cabecera en forma de matriz-bytes
        For i = 0 To 2
            Cabecera(i) = VarAuxiliar(i)
        Next i
        ' Se prepara el procedimiento para recibir n+2
```

```
                    ' bytes de datos y crc
                    NumeroBytes = (Cabecera(2) And 31) + 2
                    MSComm1.RThreshold = NumeroBytes
                    If NumeroBytes > 29 Then
                       ' Si el valor del campo longitud del campo de datos es mayor que 29,
                       ' se ha producido un error que hace que el sistema pierda el sincronismo
                        ' de trama, por lo que obliga a reconfigurar el sistema
                        ' Se da un mensaje de alarma por pantalla
                        MsgBox "Se ha producido un error en la recepción" & Chr(13) & Chr(10) & " de
tramas en el PC. Reconfigure el sistema.", vbExclamation
                        Call frmPantalla.AbandonarGestion_Click
                    End If
                    ' La próxima vez que se lea el buffer de recepción, se leerá el campo de datos
                    ' y el CRC de la trama cuya cabecera se ha recibido
                    MSComm1.InputLen = NumeroBytes
                    ' Se redimensionan las matrices para que puedan guardar los datos que se
                    ' recibirán en el futuro
                    ReDim Datos(NumeroBytes - 1)
                    ReDim trama(NumeroBytes + 2)
                    ' Se producirá evento de recepción cuando se haya recibido completamente el
                    ' campo de datos y el CRC de la trama cuya cabecera se acaba de recibir
                    MSComm1.RThreshold = NumeroBytes
                    RecibiendoCabecera = False
                    ' Si ya hay suficientes datos en el buffer de entrada, se vuelven a leer
                    ' datos
                    If MSComm1.InBufferCount >= MSComm1.RThreshold Then
                       GoTo ETIQUETA
                    End If
                End If
            Case comEvSend
            ' Se ha transmitido el contenido del buffer de salida con
            ' éxito, luego una nueva transmisión es posible
                ListoParaEnviar = True
            End Select
        End Sub


        'El procedimiento siguiente permite al usuario abandonar la aplicación
        Private Sub SalirConfiguracion_Click()
            End
```

```
End Sub


' Este procedimiento se ejecuta periódicamente. Se encarga de chequear si el timer
' de vencimiento de las tramas en espera de asentimiento, ha vencido ya, en cuyo
' caso reintroduce tal trama en cola de transmisión para que sea retransmitida
Private Sub TimerAsentimientos_Timer()
   Dim i As Byte
   Dim j As Byte
   For i = 0 To 31
      ' Se recorre toda la Lista de Espera posición por posición
      If ListaEspera(i).Ocupada = True Then
         ' Cuando se detecta una posición ocupada por una trama, se chequea
         ' su timer
         If ListaEspera(i).TimerTrama < Time() Then
            ' Si el timer ha vencido, se reintroduce la trama en cola de transmisión
            If (ListaEspera(i).trama(1) And 32) = 32 Then
               ' Si la trama es prioritaria, se introduce en Cola B1, de tranmisión
               ' con prioridad
               Do While ColaB1.ColaLlena = True
                  ' Espera activa hasta que se produzca un hueco en la cola
                  DoEvents
               Loop
               For j = 0 To ((ListaEspera(i).trama(2) And 31) + 4)
                  ' Se copia la trama en la cola de transmisión byte a byte
                  ColaB1.Posicion(ColaB1.UltimoCola).trama(j) = ListaEspera(i).trama(j)
               Next j
               ' Se actualiza la cola circular
               ColaB1.ColaVacia = False
               ColaB1.Posicion(ColaB1.UltimoCola).Longitud = ListaEspera(i).trama(2) And 31
               ColaB1.UltimoCola = (ColaB1.UltimoCola + 1) And 31
               If ColaB1.UltimoCola = ColaB1.PrimeroCola Then ColaB1.ColaLlena = True
              ' Se da un valor basura al timer de la trama para evitar que vuelva a ser
              ' reintroducida en cola de transmisión hasta que sea necesario
               ListaEspera(i).TimerTrama = ListaEspera(i).TimerTrama + #12:00:59 AM#
            Else
               ' Se procede de forma análoga cuando la trama es no prioritaria, pero
               ' con relación a ColaB0
               Do While ColaB0.ColaLlena = True
                  DoEvents
```

```
                    Loop
                    For j = 0 To ((ListaEspera(i).trama(2) And 31) + 4)
                       ColaB0.Posicion(ColaB0.UltimoCola).trama(j) = ListaEspera(i).trama(j)
                    Next j
                    ColaB0.ColaVacia = False
                    ColaB0.Posicion(ColaB0.UltimoCola).Longitud = ListaEspera(i).trama(2) And 31
                    ColaB0.UltimoCola = (ColaB0.UltimoCola + 1) And 31
                    If ColaB0.UltimoCola = ColaB0.PrimeroCola Then ColaB0.ColaLlena = True
                    ListaEspera(i).TimerTrama = ListaEspera(i).TimerTrama + #12:00:59 AM#
                End If
              End If
            End If
          Next i
        End Sub


        ' Este procedimiento se dispara si pasado un tiempo, la  configuración del sistema no se ha
completado
        ' con lo que se cosidera que dicho intento de configuración es fallido
        Private Sub TimerEspera_Timer()
          Call AbandonarGestion_Click
          ' Se da un mensaje de alarma por pantalla
          MsgBox " No se ha logrado configurar el Sistema. " & Chr(10) & "          Revise las
conexiones. ", vbExclamation
        End Sub


        ' Este procedimiento se ejecuta periódicamente. Se encarga de disparar el procedimiento
        ' encargado del procesado de tramas
        Private Sub TimerProcesado_Timer()
          Call ProcesarTramas
        End Sub


        ' Este procedimiento se encarga de chequear si hay alguna trama en cola de transmisión
        ' y en ese caso dispara la transmisión de la primera de ellas
        Private Sub TimerTransmisiones_Timer()
          Dim TramaTX() As Byte
          Dim i As Byte
          Dim Dimension As Byte
```

```
Dim SalidaTrama As Variant
If ListoParaEnviar = True Then
   ' Si el sistema puede iniciar una nueva transmisión, se chequean las colas de
   ' transmisión para ver si hay alguna trama en espera de ser transmitidas
   ' En primer lugar se chequea la cola de transmisión con prioridad, y después
   ' la cola de transmisión sin prioridad
   If ColaB1.ColaVacia = False Then
      Dimension = (ColaB1.Posicion(ColaB1.PrimeroCola).Longitud) + 4
      ' Se redimensiona la matriz de txon conforme al tamaño de la trama que
      ' va a ser transmitida
      ReDim TramaTX(Dimension) As Byte
      For i = 0 To Dimension
         ' Se configura la matriz de txon con los datos de la trama
         TramaTX(i) = ColaB1.Posicion(ColaB1.PrimeroCola).trama(i)
      Next i
      ' Se transmite físicamente la trama
      SalidaTrama = TramaTX()
      MSComm1.Output = SalidaTrama
      ' Se inhabilita el comienzo de una nueva transmisión hasta que no haya
      ' finalizado la transmisión que acaba de iniciarse
      ListoParaEnviar = False
      ' Se actualiza la cola circular de transmisión
      ColaB1.ColaLlena = False
      ColaB1.PrimeroCola = ((ColaB1.PrimeroCola + 1) And 31)
      If ColaB1.PrimeroCola = ColaB1.UltimoCola Then ColaB1.ColaVacia = True
      ' Si la trama tiene como dirección origen el PC se registran los timers
      ' de vencimiento cuando la trama requiere asentimiento
      If TramaTX(0) < 16 Then
         ' Si la trama requiere asentimiento, se registra el valor de vencimiento
         ' del timer de la trama
         If (TramaTX(1) And 64) = 64 Then
            ListaEspera(TramaTX(1) And 31).TimerTrama = Time() + #12:00:05 AM#
            ' Se incrementa en 1 el número de transmisiones que ha sufido la
            ' trama
            ListaEspera(TramaTX(1) And 31).Transmisiones = ListaEspera(TramaTX(1) And
31).Transmisiones + 1
            ' Cuando la trama ha sufrido 5 transmisiones, implica que hay algún
            ' error en el sistema, con lo que se aborta la configuración
            If ListaEspera(TramaTX(1) And 31).Transmisiones >= 5 Then
               Call AbandonarGestion_Click
```

```
                        ' Se muestra un mensaje de alarma por pantalla
                        MsgBox " Cierto nodo no responde. Reconfigure el sistema", vbExclamation
                        Exit Sub
                    End If
                End If
            End If
        Else
            ' Se procede de manera análoga con las tramas no prioritarias en espera de
            ' ser transmitidas, con relación a ColaB0
            If ColaB0.ColaVacia = False Then
                Dimension = (ColaB0.Posicion(ColaB0.PrimeroCola).Longitud) + 4
                ReDim TramaTX(Dimension) As Byte
                For i = 0 To Dimension
                    TramaTX(i) = ColaB0.Posicion(ColaB0.PrimeroCola).trama(i)
                Next i
                SalidaTrama = TramaTX()
                MSComm1.Output = SalidaTrama
                ListoParaEnviar = False
                ColaB0.ColaLlena = False
                ColaB0.PrimeroCola = ((ColaB0.PrimeroCola + 1) And 31)
                If ColaB0.PrimeroCola = ColaB0.UltimoCola Then ColaB0.ColaVacia = True
                If TramaTX(0) < 16 Then
                    If (TramaTX(1) And 64) = 64 Then
                        ListaEspera(TramaTX(1) And 31).TimerTrama = Time() + #12:00:05 AM#
                        ListaEspera(TramaTX(1) And 31).Transmisiones = (ListaEspera(TramaTX(1)
And 31).Transmisiones) + 1
                    End If
                    If ListaEspera(TramaTX(1) And 31).Transmisiones >= 5 Then
                        Call AbandonarGestion_Click
                        MsgBox " Cierto nodo no responde. Reconfigure el sistema", vbExclamation
                        Exit Sub
                    End If
                End If
            End If
        End If
    End If
End Sub
```

Formulario Información

```
' Este formulario presenta el cuadro de diálogo que permite al usuario obtener
' información sobre el sistema acerca del número y la naturaleza de los nodos
' que lo componen

Option Explicit
' El texto de presentación representa la información que solicita el usuario
' como una cadena de caracteres
Dim TextoPresentacion As String


' Este procedimiento permite abandonar el formulario
Private Sub Command1_Click()
   Unload Me
End Sub


' Este procedimiento se encarga de representar en TextoPresentacion la información
' sobre la naturaleza del nodo seleccionado por el usuario
Private Sub DarInformacion(ByVal nodo As Byte)
   Dim i As Byte, j As Byte, inform3 As Byte
   Dim inform1 As Currency, inform2 As Currency
   ' Se almacena la información que se recibió del nodo seleccionado durante el proceso de
   ' configuración codificada numéricamente en variables auxiliares para su posterior
   ' tratamiento
   inform1 = Sistema.tiponodos(nodo).informacion(0) * (2 ^ 16) +
Sistema.tiponodos(nodo).informacion(1) * (2 ^ 8) + Sistema.tiponodos(nodo).informacion(2)
   inform1 = inform1 \ 16
   inform2 = Sistema.tiponodos(nodo).informacion(3) * (2 ^ 16) +
Sistema.tiponodos(nodo).informacion(4) * (2 ^ 8) + Sistema.tiponodos(nodo).informacion(5)
   inform2 = inform2 \ 16
   inform3 = Sistema.tiponodos(nodo).informacion(6)
   Text3 = TextoPresentacion & Chr(13) & Chr(10) & " E/S presentes en el nodo " & nodo & "
:" & Chr(13) & Chr(10) & Chr(13) & Chr(10)
   ' Se analiza cada bit de informacion, de modo que en función de si están a "1" ó "0",
   ' se representa un texto u otro indicativo de la naturaleza del pin asociado
   For i = 0 To 19
      If (inform1 And (2 ^ i)) = (2 ^ i) Then
         Select Case i
```

```
Case 0
   Text3 = Text3 & " PE7 activo."
   If (inform2 And (2 ^ i)) = (2 ^ i) Then
      Text3 = Text3 & " Entrada analógica." & Chr(13) & Chr(10)
   Else
      Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
   End If
Case 1
   Text3 = Text3 & " PE6 activo."
   If (inform2 And (2 ^ i)) = (2 ^ i) Then
      Text3 = Text3 & " Entrada analógica." & Chr(13) & Chr(10)
   Else
      Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
   End If
Case 2
   Text3 = Text3 & " PE5 activo."
   If (inform2 And (2 ^ i)) = (2 ^ i) Then
      Text3 = Text3 & " Entrada analógica." & Chr(13) & Chr(10)
   Else
      Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
   End If
Case 3
   Text3 = Text3 & " PE4 activo."
   If (inform2 And (2 ^ i)) = (2 ^ i) Then
      Text3 = Text3 & " Entrada analógica." & Chr(13) & Chr(10)
   Else
      Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
   End If
Case 4
   Text3 = Text3 & " PE3 activo."
   If (inform2 And (2 ^ i)) = (2 ^ i) Then
      Text3 = Text3 & " Entrada analógica." & Chr(13) & Chr(10)
   Else
      Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
   End If

Case 5
   Text3 = Text3 & " PE2 activo."
   If (inform2 And (2 ^ i)) = (2 ^ i) Then
      Text3 = Text3 & " Entrada analógica." & Chr(13) & Chr(10)
```

```
    Else
      Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
    End If
  Case 6
    Text3 = Text3 & " PE1 activo."
    If (inform2 And (2 ^ i)) = (2 ^ i) Then
      Text3 = Text3 & " Entrada analógica." & Chr(13) & Chr(10)
    Else
      Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
    End If
  Case 7
    Text3 = Text3 & " PE0 activo."
    If (inform2 And (2 ^ i)) = (2 ^ i) Then
      Text3 = Text3 & " Entrada analógica." & Chr(13) & Chr(10)
    Else
      Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
    End If
  Case 8
    Text3 = Text3 & " PD5 activo."
    If (inform2 And (2 ^ i)) = (2 ^ i) Then
      Text3 = Text3 & " Función: SPI - SS." & Chr(13) & Chr(10)
    Else
      If (inform3 And 128) = 128 Then
        Text3 = Text3 & " Salida de propósito general." & Chr(13) & Chr(10)
      Else
        Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
      End If
    End If
  Case 9
    Text3 = Text3 & " PD4 activo."
    If (inform2 And (2 ^ i)) = (2 ^ i) Then
      Text3 = Text3 & " Función: SPI - SCK." & Chr(13) & Chr(10)
    Else
      If (inform3 And 64) = 64 Then
        Text3 = Text3 & " Salida de propósito general." & Chr(13) & Chr(10)
      Else
        Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
      End If
    End If
  Case 10
```

```
        Text3 = Text3 & " PD3 activo."
        If (inform2 And (2 ^ i)) = (2 ^ i) Then
          Text3 = Text3 & " Función: SPI - MOSI." & Chr(13) & Chr(10)
        Else
          If (inform3 And 32) = 32 Then
            Text3 = Text3 & " Salida de propósito general." & Chr(13) & Chr(10)
          Else
            Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
          End If
        End If
      Case 11
        Text3 = Text3 & " PD2 activo."
        If (inform2 And (2 ^ i)) = (2 ^ i) Then
          Text3 = Text3 & " Función: SPI - MISO." & Chr(13) & Chr(10)
        Else
          If (inform3 And 16) = 16 Then
            Text3 = Text3 & " Salida de propósito general." & Chr(13) & Chr(10)
          Else
            Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
          End If
        End If
      Case 12
        Text3 = Text3 & " PA7 activo."
        If (inform2 And (2 ^ i)) = (2 ^ i) Then
          Text3 = Text3 & " Pulse Acumulator." & Chr(13) & Chr(10)
        Else
          If (inform3 And 8) = 8 Then
            Text3 = Text3 & " Salida de propósito general." & Chr(13) & Chr(10)
          Else
            Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
          End If
        End If
      Case 13
        Text3 = Text3 & " PA6 activo."
        If (inform2 And (2 ^ i)) = (2 ^ i) Then
          Text3 = Text3 & " Output Capture." & Chr(13) & Chr(10)
        Else
          Text3 = Text3 & " Salida de propósito general." & Chr(13) & Chr(10)
        End If
      Case 14
```

```
        Text3 = Text3 & " PA5 activo."
        If (inform2 And (2 ^ i)) = (2 ^ i) Then
          Text3 = Text3 & " Output Capture." & Chr(13) & Chr(10)
        Else
          Text3 = Text3 & " Salida de propósito general." & Chr(13) & Chr(10)
        End If
      Case 15
        Text3 = Text3 & " PA4 activo."
        If (inform2 And (2 ^ i)) = (2 ^ i) Then
          Text3 = Text3 & " Output Capture." & Chr(13) & Chr(10)
        Else
          Text3 = Text3 & " Salida de propósito general." & Chr(13) & Chr(10)
        End If
      Case 16
        Text3 = Text3 & " PA3 activo."
        If (inform2 And (2 ^ i)) = (2 ^ i) Then
          Text3 = Text3 & " Output Capture." & Chr(13) & Chr(10)
        Else
          Text3 = Text3 & " Salida de propósito general." & Chr(13) & Chr(10)
        End If
      Case 17
        Text3 = Text3 & " PA2 activo."
        If (inform2 And (2 ^ i)) = (2 ^ i) Then
          Text3 = Text3 & " Input Capture." & Chr(13) & Chr(10)
        Else
          Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
        End If
      Case 18
        Text3 = Text3 & " PA1 activo."
        If (inform2 And (2 ^ i)) = (2 ^ i) Then
          Text3 = Text3 & " Input Capture." & Chr(13) & Chr(10)
        Else
          Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
        End If
      Case 19
        Text3 = Text3 & " PA0 activo."
        If (inform2 And (2 ^ i)) = (2 ^ i) Then
          Text3 = Text3 & " Input Capture." & Chr(13) & Chr(10)
        Else
          Text3 = Text3 & " Entrada de propósito general." & Chr(13) & Chr(10)
```

```
                End If
            End Select
        End If
    Next i
End Sub


' Este procedimiento permite mostrar el formulario que muestra los mensajes recibidos
' desde los nodos
Private Sub Command3_Click()
    frmmensajes.Show
End Sub


' Este procedimiento se encarga de comprobar el correcto formato de los datos a
' través de los cuales el usuario solicita información sobre un nodo, y en caso
' de que estos sean correctos, dispara la presentación de la información
Private Sub Detalles_Click()
    Dim nodo As Variant
    Dim Numnodo As Byte
    ' Se da formato numérico al número de nodo introducido por el usuario como
    ' un carácter
    nodo = CVar(Text2)
    ' Para cada posible nodo, se comprueba si este ha sido seleccionado
    For Numnodo = 1 To Sistema.Numeronodos
        If nodo = Numnodo Then
            ' Si el nodo seleccionado pertenece al sistema, se llama al procedimiento
            ' encargado de representar la información sobre este
            Call DarInformacion(Numnodo)
            Exit Sub
        End If
    Next Numnodo
    ' Si el nodo seleccionado no pertenece al sistema, se informa al usuario de tal
    ' circunstancia para que seleccione otro nodo
    Text3 = TextoPresentacion & " Introduzca nº de nodo como número de uno o dos dígitos entre
1 y " & Sistema.Numeronodos
End Sub


' Este procedimiento inicializa el formulario
```

```
Private Sub Form_Load()
   Text2 = "1"
   ' Se presenta como información de presentación el número de nodos que componen
   ' el sistema
   TextoPresentacion = " El Sistema lo compone(n) " & Sistema.Numeronodos & " nodo(s)." &
Chr(13) & Chr(10)
   Text3 = TextoPresentacion
End Sub
```

Formulario LecEsc

```
' Este formulario presenta al usuario un cuadro de diálogo a través del cual
' este puede leer y escribir datos en la memoria local de los nodos

Option Explicit
Dim TramaLecEsc(31) As Byte
Dim contador As Byte
Dim Lecturas As Byte


' Este procedimiento habilita el cuadro de diálogo que permite al usuario escribir
' y leer datos en memoria local del nodo seleccionado, y comprueba que este pertenezca
' al sistema
Private Sub Command1_Click()
   Dim nodo As Variant
   Dim Numnodo As Byte
   ' Se da formato numérico al número de nodo, introducido como cadena de caracteres
   ' por el usuario
   nodo = CVar(Text1)
   ' Se recorre todo el listado de nodos del sistema
   For Numnodo = 1 To Sistema.Numeronodos
      ' Si el nodo seleccionado coincide con el identificador de uno de los nodos
      ' del sistema, se habilita el menú de lectura/escritura
      ' el nodo seleccionado
      If nodo = Numnodo Then
         Command1.Enabled = False
         Label3.Enabled = True
         Label4.Enabled = True
```

```
                Label4.Enabled = True
                Label5.Enabled = True
                Frame1.Enabled = True
                Frame2.Enabled = True
                Text1.Locked = True
                Text2 = ""
                Text3.Enabled = True
                Text4.Enabled = True
                Option1.Enabled = True
                Option2.Enabled = True
                Command2.Enabled = True
                Command4.Enabled = True
                Command5.Enabled = True
                ' Direccion origen = PC ; Dirección destino = Nodo seleccionado
                TramaLecEsc(0) = Numnodo
                ' TP = True ; ASEN = True ; PRIOR = True
                TramaLecEsc(1) = 224
                ' Si se seleccionó NO PRIORIDAD, entonces PRIOR = False
                If Option4.Value = True Then TramaLecEsc(1) = 192
                TramaLecEsc(2) = 0
                ' Se inicializan los contadores de configuración de trama
                contador = 0
                Lecturas = 0
                Exit Sub
              End If
            Next Numnodo
            ' Si el nodo seleccionado no coincide con el identificador de ninguno de los
            ' nodos del sistema, no se habilita el menú de lectura/escritura, y se informa
            ' de ello al usuario
            Text2 = " Introduzca nº de nodo como un número de dos dígitos entre 1 y " &
Sistema.Numeronodos
        End Sub


        ' Este procedimiento permite introducir un nuevo "paquete de aplicación" correspondiente
        ' a instrucciones de lectura o escritura
        Private Sub Command2_Click()
          Dim i As Byte
          Dim Posicion As Currency
          Dim dato As Currency
```

```
Dim Texto As String
Text3 = Mid(Text3, 1, 4)
Text4 = Mid(Text4, 1, 2)
' Se da formato numérico a la dirección del dato a leer o escribir introducida
' como una cadena de caracteres
Posicion = Valor(Text3, 4)
If Posicion = 65536 Then
   ' Si el formato es incorrecto, el valor devuelto por la función Valor toma
   ' el valor basura indicado, de modo que se informa de tal situación al
   ' usuario, y se sale del procedimiento sin intriducir el "paquete de
   ' aplicación" en la trama
   Text3 = "Formato incorrecto"
   Exit Sub
End If
If Option2 = True Then
   ' Si se ha seleccionado operación de escritura se da formato numérico
   ' al dato introducido como una cadena de caracteres
   dato = Valor(Text4, 2)
   If Posicion > 32767 Then
      ' Si la dirección del dato está fuera de RAM, no es posible realizar
      ' la escritura, por lo que se abandona el procedimiento sin introducir
      ' el "paquete de aplicación" en la trama y se informa de ello al usuario
      Text3 = "Fuera de RAM"
      Exit Sub
   End If
   If dato = 65536 Then
      ' Si el formato del dato es incorrecto, el valor devuelto por la función
      ' Valor toma el valor basura indicado, de modo que se informa de tal
      ' situación al usuario, y se sale del procedimiento sin intriducir el
      ' "paquete de aplicación" en la trama
      Text4 = "Formato incorrecto"
      Exit Sub
   End If
End If
' Si no hay suficiente espacio en la trama para un nuevo "paquete de aplicación"
' se informa de ello al usuario y se abandona el procedimiento
If (27 - contador) < 3 Then
   Text2 = Text2 & "Trama completa. Pulse Enviar.         "
   Exit Sub
End If
```

```
If Option2 = False Then
   ' Se ha seleccionado operación de lectura
   If Lecturas = 6 Then
      ' Si ya se han introducido 6 "paquetes de aplicación" de instrucciones
      ' de lectura, no es posible introducir una nueva, por lo que se informa
      ' de ello al usuario y se abandona la aplicación
      Text2 = Text2 & " No es posible nueva lectura          "
      Exit Sub
   End If
   ' Se añade la selección hecha por el usuario al listado de instrucciones que
   ' se enviarán al nodo
   Text2 = Text2 & "Lectura: posición $" & Text3 & "                  "
   ' Se introduce en la trama el "paquete de aplicación" correspondiente
   TramaLecEsc(contador + 3) = 1
   TramaLecEsc(contador + 4) = Posicion \ (2 ^ 8)
   TramaLecEsc(contador + 5) = Posicion And 255
   ' Se incrementa el contador de bytes de datos de la trama en un número igual
   ' a la longitud del "paquete de aplicación" introducido
   contador = contador + 3
   ' Se incrementa en 1 el número de operaciones de lectura seleccionadas
   Lecturas = Lecturas + 1
Else
   ' Se ha seleccionado operación de escritura
   If (27 - contador) < 4 Then
      ' No hay espacio en la trama para un nuevo "paquete de aplicación"
      ' correspondiente a una operación de escritura, así que se abandona
      ' el procedimiento y se informa de la situación al usuario
      Text2 = Text2 & "Imposible escribir más datos.    "
      Exit Sub
   End If
   ' Se añade la selección hecha por el usuario al listado de instrucciones que
   ' se enviarán al nodo
   Text2 = Text2 & "Esc : dato $" & Text4& "; posición $" & Text3 & "    "
   ' Se introduce en la trama el "paquete de aplicación" correspondiente
   TramaLecEsc(contador + 3) = 2
   TramaLecEsc(contador + 4) = Posicion \ (2 ^ 8)
   TramaLecEsc(contador + 5) = Posicion And 255
   TramaLecEsc(contador + 6) = dato
   ' Se incrementa el contador de bytes de datos de la trama en un número igual
   ' a la longitud del "paquete de aplicación" introducido
```

```
        contador = contador + 4
    End If
End Sub


' Al ejecutarse este procedimiento se oculta el formulario
Private Sub Command3_Click()
    Unload Me
End Sub


' Este procedimiento inhabilita el menú de lectura/escritura, y devuelve al usuario
' al cuadro de diálogo principal del formulario
Private Sub Command4_Click()
    Command1.Enabled = True
    Label3.Enabled = False
    Label4.Enabled = False
    Label4.Enabled = False
    Label5.Enabled = False
    Frame1.Enabled = False
    Frame2.Enabled = False
    Text1.Locked = False
    Text3.Enabled = False
    Text4.Enabled = False
    Option1.Enabled = False
    Option2.Enabled = False
    Command2.Enabled = False
    Command4.Enabled = False
    Command5.Visible = True
    Command5.Enabled = False
End Sub


' Este procedimiento completa la trama cuyo campo de datos ha quedado configurado
' anteriormente, y llama al procedimiento encargado de gestionar su transmisión
Private Sub Command5_Click()
    ' Si no se ha introducido ningún "paquete de aplicación", no se transmite nada
    If contador = 0 Then Exit Sub
    ' Se prepara el CRC para que pueda ser calculado
    TramaLecEsc((contador And 31) + 3) = 0
```

```
    TramaLecEsc((contador And 31) + 4) = 0
    ' Se llama al procedimiento encargado de gestionar la transmisión de la trama
    Call InsertarTrama(TramaLecEsc(), contador)
    ' Se inhabilita el botón que dispara este procedimiento
    Command5.Visible = False
End Sub


' Este procedimiento se ejecuta cuando se carga el formulario y habilita el cuadro
' de diálogo principal del formulario, inhabilitando en resto
Private Sub Form_Load()
    Label3.Enabled = False
    Label4.Enabled = False
    Label4.Enabled = False
    Label5.Enabled = False
    Frame1.Enabled = False
    Frame2.Enabled = False
    Text3.Enabled = False
    Text4.Enabled = False
    Option1.Enabled = False
    Option2.Enabled = False
    Command2.Enabled = False
    Command4.Enabled = False
    Command5.Enabled = False
End Sub
```

Formulario Modificar

```
' Este formulario presenta al usuario un cuadro de diálogo que le permite
' modificar las condiciones de ejecución en los nodos, así como cargar programas
' en memoria (RAM) local desde el PC
Option Explicit
Dim NumnodoMod As Byte
Dim NumnodoCop As Byte


' Al ejecutarse este procedimiento, el formulario queda oculto
Private Sub Command1_Click()
    Unload Me
```

```
End Sub


' Este procedimiento activa el cuadro de diálogo que permite al usuario modificar
' las condiciones de ejecución del programa local en el nodo previamente seleccionado
' por el usuario. Además comprueba que este pertenezca al sistema, y en caso de
' que no lo sea, informa al usuario de dicha situación
Private Sub Command2_Click()
   Dim nodo As Variant
   Dim i As Byte
   ' Se da formato numérico al número de nodo, introducido como cadena de caracteres
   ' por el usuario
   nodo = CVar(Text1)
   ' Se recorre todo el listado de nodos del sistema
   For i = 1 To Sistema.Numeronodos
      ' Si el nodo seleccionado coincide con el identificador de uno de los nodos
      ' del sistema, se habilita el menú de modificación del programa local en
      ' el nodo seleccionado
      If nodo = i Then
         Text1.Locked = True
         Command2.Enabled = False
         Command5.Enabled = False
         Command3.Enabled = True
         Command4.Enabled = True
         Label4.Enabled = True
         Check1.Enabled = True
         Check2.Enabled = True
         Check3.Enabled = True
         Check4.Enabled = True
         Frame1.Enabled = True
         NumnodoMod = i
         Exit Sub
      End If
   Next i
   ' Si el nodo seleccionado no coincide con el identificador de ninguno de los
   ' nodos del sistema, no se habilita el menú de modificación, y se informa de
   ' ello al usuario
   Label5 = " Introduzca nº de nodo como un número de dos dígitos entre 1 y " &
Sistema.Numeronodos
End Sub
```

```
' Este procedimiento configura una trama con información referente a las modificaciones
' que el usuario desea que se lleven a cabo en la ejecución del programa local en el
' nodo seleccionado. Posteriormente, se llama al procedimiento encargado de gestionar
' su transmisión
Private Sub Command3_Click()
   Dim ContadorMod As Byte
   Dim TramaMod(31) As Byte
   ' Se configura la cabecera de la trama
   ' Dirección origen = PC ; Dirección destino = nodo seleccionado
   TramaMod(0) = NumnodoMod
   ' TP = True ; ASEN = True ; PRIOR = True ; ID numérico de trama = 0 (provisionalmente)
   TramaMod(1) = 224
   ' Tipo de trama = trama de información ; Long. campo de datos = 0 (provisionalmente)
   TramaMod(2) = 0
   If Check1.Value = Checked Then
      ' Si el usuario cliqueó en la opción de "Suspender ejecución de programa local
      ' en el nodo seleccionado", se introduce "paquete de aplicación" correspondiente
      ' a dicha instrucción en el campo de datos
      TramaMod(ContadorMod + 3) = 32
      ' Se incrementa el contador de bytes de datos en un número igual a la longitud
      ' del "paquete de aplicación" de la instrucción
      ContadorMod = ContadorMod + 1
   End If
   If Check2.Value = Checked Then
      ' Si el usuario cliqueó en la opción de "Activar ejecución de programa local
      ' en el nodo seleccionado", se introduce "paquete de aplicación" correspondiente
      ' a dicha instrucción en el campo de datos
      TramaMod(ContadorMod + 3) = 64
      ' Se incrementa el contador de bytes de datos en un número igual a la longitud
      ' del "paquete de aplicación" de la instrucción
      ContadorMod = ContadorMod + 1
   End If
   If Check3.Value = Checked Then
      ' Si el usuario cliqueó en la opción de "Habilitar ejecución de programa local
      ' almacenado en memoria EPROM", se introduce "paquete de aplicación" correspondiente
      ' a dicha instrucción en el campo de datos
      TramaMod(ContadorMod + 3) = 16
      ' Se incrementa el contador de bytes de datos en un número igual a la longitud
```

```
        ' del "paquete de aplicación" de la instrucción
        ContadorMod = ContadorMod + 1
    End If
    If Check4.Value = Checked Then
        ' Si el usuario cliqueó en la opción de "Habilitar ejecución de programa local
        ' almacenado en memoria RAM", se introduce "paquete de aplicación" correspondiente
        ' a dicha instrucción en el campo de datos
        TramaMod(ContadorMod + 3) = 4
        ' Se incrementa el contador de bytes de datos en un número igual a la longitud
        ' del "paquete de aplicación" de la instrucción
        ContadorMod = ContadorMod + 1
    End If
    ' Si el usuario no había seleccionado ninguna opción no se envía la trama
    If ContadorMod = 0 Then Exit Sub
    ' En caso contrario se llama al procedimiento encargado de gestionar la transmisión
    ' de la trama configurada
    Call InsertarTrama(TramaMod(), ContadorMod)
    ' Se oculta el botón que que al ser cliqueado llama a este procedimiento
    Command3.Visible = False
End Sub


' Este procedimiento inhabilita el cuadro de diálogo que permite modificar las
' condiciones de ejecución del programa local en el nodo seleccionado, habilitando
' nuevamente el cuadro de diálogo principal del formulario
Private Sub Command4_Click()
    Label5 = ""
    Text1.Locked = False
    Command2.Enabled = True
    Command5.Enabled = True
    Command3.Visible = True
    Command3.Enabled = False
    Command4.Enabled = False
    Label4.Enabled = False
    Check1.Enabled = False
    Check1.Value = Unchecked
    Check2.Enabled = False
    Check2.Value = Unchecked
    Check3.Enabled = False
    Check3.Value = Unchecked
```

```
    Check4.Enabled = False
    Check4.Value = Unchecked
    Frame1.Enabled = False
End Sub


' Este procedimiento activa el cuadro de diálogo que permite al usuario cargar un
' programa en memoria RAM local del nodo previamente seleccionado por el usuario.
' Además comprueba que este pertenezca al sistema, y en caso de que no lo sea,
' informa al usuario de dicha situación
Private Sub Command5_Click()
    Dim nodo As Variant
    Dim i As Byte
    ' Se da formato numérico al número de nodo, introducido como cadena de caracteres
    ' por el usuario
    nodo = CVar(Text1)
    ' Se recorre todo el listado de nodos del sistema
    For i = 1 To Sistema.Numeronodos
        ' Si el nodo seleccionado coincide con el identificador de uno de los nodos
        ' del sistema, se habilita el menú copia programa en memoria local RAM del
        ' nodo seleccionado
        If nodo = i Then
            Text1.Locked = True
            Command2.Enabled = False
            Command5.Enabled = False
            Frame2.Enabled = True
            Drive1.Enabled = True
            File1.Enabled = True
            Dir1.Enabled = True
            Label7.Enabled = True
            Label8.Enabled = True
            Command6.Enabled = True
            Command7.Enabled = True
            Text2.Enabled = True
            NumnodoCop = i
            Exit Sub
        End If
    Next i
    ' Si el nodo seleccionado no coincide con el identificador de ninguno de los
    ' nodos del sistema, no se habilita el menú de copia, y se informa de ello
```

```
        ' al usuario
        Label5 = " Introduzca nº de nodo como un número de dos dígitos entre 1 y " &
Sistema.Numeronodos
    End Sub


    ' Este procedimiento se encarga de configurar la(s) trama(s) necesaria(s) para
    ' cargar el programa seleccionado en memoria local RAM del nodo seleccionado
    Private Sub Command6_Click()
      Dim TramaCop(31) As Byte
      Dim Lineas As Byte
      Dim i As Integer
      Dim j As Integer
      Dim h As Byte
      Dim ContadorCop As Byte
      Dim Programa(31) As String
      Dim Longitudes(31) As Integer
      Dim Direcciones(31) As Currency
      Dim LineadeTexto As String
      Dim DireccionBase As Currency
      Dim TotalInstruc As Byte
      ' Comprueba si hay o no algún fichero seleccionado
      If File1.FileName = "" Then
        ' Si no lo hay, informa de tal situación al usuario y sale del procedimiento
        Label9.Caption = "¡ No hay archivo seleccionado!"
        Exit Sub
      Else
        ' Si lo hay, informa al usuario de que el proceso de copia ya se está
        ' llevando a cabo
        Label9.Caption = "   ¡Enviando Programa!"
      End If
      ' Se abre el archivo seleccionado, que contiene el código del programa a cargar
      ' en formato S19 de Motorola
      Open Text2 For Input As #1
      ' Se introduce en la línea de programa Programa(0) y la cadena de caracteres auxiliar
      ' LineadeTexto la primera línea del fichero
      ' El objeto de esto es tener un listado de líneas de programa de modo que cada una de
      ' ellas almacene todos los datos consecutivos y la dirección del primero de ellos
      Line Input #1, Programa(0)
      LineadeTexto = Programa(0)
```

```
If Mid(LineadeTexto, 1, 2) = "S0" Then
   ' Cuando la primera línea de fichero empieza por los caracteres "S", "0", se desecha
   ' dicha línea de fichero, se lee la siguiente y se almacena en la línea de programa
   ' Programa(0) y la cadena de caracteres auxiliar LineadeTexto
   Line Input #1, Programa(0)
   LineadeTexto = Programa(0)
End If
If Mid(LineadeTexto, 1, 2) <> "S1" Then
   ' Cuando las líneas de fichero no empiezan por los caracteres "S", "1", implica
   ' que el formato de la información contenida en el archivo es incompatible
   ' con el sistema, con lo cual se aborta el proceso de copia y se sale del
   ' procedimiento, informando de todo ello al usuario
   Label9.Caption = "¡ Incompatible con el sistema !"
   ' Se cierra el archivo previamente abierto
   Close #1
   Exit Sub
End If
' Se almacena en el campo Longitudes correspondiente a la línea de programa el valor
' del número de bytes de datos que contiene la línea de fichero leida
Longitudes(0) = Valor(Mid(LineadeTexto, 3, 2), 2) - 3
' Se almacena en el campo Direcciones correspondiente a la línea de programa el valor
' de la dirección a partir de la cual hay que insertar los datos de la línea de programa
Direcciones(0) = Valor(Mid(LineadeTexto, 5, 4), 4)
' Se recorre el fichero línea por línea mientras no haya un carácter de fin de fichero
Do While Not EOF(1)
      DoEvents
      ' Se lee la siguiente línea del fichero
      Line Input #1, LineadeTexto
      If Mid(LineadeTexto, 1, 2) = "S1" Then
         ' Si la línea comienza por "S", "1", contiene datos
         If Valor(Mid(LineadeTexto, 5, 4), 4) = Direcciones(Lineas) + Longitudes(Lineas)
Then
            ' Si la dirección de los datos de esa línea del fichero coincide con la dirección de
            ' la última línea de programa con un offset igual al número de datos que contiene
            ' dicha línea de programa (hasta el momento), eso implica que los datos de la nueva
            ' línea de fichero deben ser introducidos justo a continuación de los de la línea de
            ' programa por lo que se introducen en esta justo a continuación de los anteriores
            Programa(Lineas) = Mid(Programa(Lineas), 1, (Longitudes(Lineas) + 4) * 2) &
Mid(LineadeTexto, 9, (Valor(Mid(LineadeTexto, 3, 2), 2) - 3) * 2)
            ' La nueva longitud de esa línea de programa es igual a la que tenía antes más el
```

```
                    ' número de datos que contenía la línea de fichero
                    Longitudes(Lineas) = Longitudes(Lineas) + (Valor(Mid(LineadeTexto, 3, 2), 2) - 3)
                  Else
                    ' Si no se da las circunstancia anterior, hay que abrir una nueva línea de programa
                    Lineas = Lineas + 1
                    ' En dicha línea se guarda la dirección de comienzo y los datos de la línea de fichero
                    Programa(Lineas) = LineadeTexto
                    ' El campo Longitudes de la línea toma (por el momento) el valor del número de
datos
                    ' que contiene la línea de fichero leida
                    Longitudes(Lineas) = Valor(Mid(LineadeTexto, 3, 2), 2) - 3
                    ' El campo Direcciones, toma el valor de la dirección del primero de los datos, y lo
                    ' toma de la línea de fichero
                    Direcciones(Lineas) = Valor(Mid(LineadeTexto, 5, 4), 4)
                  End If
                Else
                  ' Si la línea de fichero leida no comienza por "S", "1", implica que es la última línea de
                  ' fichero, que no contiene datos de programa, por lo que se almacena en una nueva líne
de
                    ' programa sin más
                    Lineas = Lineas + 1
                    Programa(Lineas) = LineadeTexto
                End If
            Loop
            ' Una vez se han ordenado adecuadamente los datos contenidos en el fichero, este es cerrado
            Close #1
            ' Dirección origen = PC ; Dirección destino = Nodo seleccionado
            TramaCop(0) = NumnodoCop
            ' Se recorre el listado de líneas de programa una a una antes constituido
            For i = 0 To Lineas - 1
              DoEvents
              If (27 - ContadorCop) > (Longitudes(i) + 4) Then
              ' Si los datos de la línea de programa actual caben en la trama que está siendo configurada,
              ' se introduce un "paquete de aplicación" de instrucción de copia de programa en el campo
              ' de datos de dicha trama
                If TotalInstruc = 0 Then
                  ' Si es la primera instrucción de copia del programa a copiar, el código de instrucción
                  ' introducido es el de "Primera instrucción de copia en RAM"
                  TramaCop(3 + ContadorCop) = 11
                  ' Al llevar la primera instrucción de copia, esa trama es prioritaria, por lo que
```

```
                    ' PRIOR = True
                    TramaCop(1) = 32
                Else
                    ' Si no, el código de instrucción introducido es "Instrucción de copia en RAM"
                    TramaCop(3 + ContadorCop) = 8
                End If
                ' Longitud de campo de datos del "paquete de aplicación" = número de datos a copiar
                TramaCop(4 + ContadorCop) = Longitudes(i)
                ' Dirección de comienzo de copia = Dirección del primero de los datos
                TramaCop(5 + ContadorCop) = Direcciones(i) \ 256
                TramaCop(6 + ContadorCop) = Direcciones(i) Mod 256
                ' Se incrementa el número de instrucciones de copia del programa enviadas
                TotalInstruc = TotalInstruc + 1
                For j = 0 To Longitudes(i) - 1
                    ' Se introducen en el campo de datos del "paquete de aplicación" los datos uno a uno
                    TramaCop(7 + j + ContadorCop) = Valor(Mid(Programa(i), 9 + (2 * j), 2), 2)
                Next j
                ' Se actualiza la variable que representa el espacio de datos que queda aún libre en
                ' la trama que está siendo configurada
                ContadorCop = ContadorCop + 4 + Longitudes(i)
            Else
                ' Los datos de la línea de programa actual no caben en el espacio libre en el campo de
                ' datos de la trama que está siendo configurada
                If (27 - ContadorCop) > 9 Then
                    ' Si el espacio libre es superior a 9 bytes, se introducen nuevos datos en esta trama
                    If TotalInstruc = 0 Then
                        ' Si es la primera instrucción de copia del programa a copiar, el código de
                        ' instrucción introducido es el de "Primera instrucción de copia en RAM"
                        TramaCop(3 + ContadorCop) = 11
                        ' Al llevar la primera instrucción de copia, esa trama es prioritaria,
                        ' por lo que PRIOR = True
                        TramaCop(1) = 32
                    Else
                        ' Si no, el código de instrucción introducido es "Instrucción de copia en RAM"
                        TramaCop(3 + ContadorCop) = 8
                    End If
                    ' Dirección de comienzo de copia = Dirección del primero de los datos recién
introducidos
                    TramaCop(5 + ContadorCop) = Direcciones(i) \ 256
                    TramaCop(6 + ContadorCop) = Direcciones(i) Mod 256
```

```
            ' Se incrementa el número de instrucciones de copia del programa enviadas
            TotalInstruc = TotalInstruc + 1
            j = 0
            Do While (ContadorCop + 4 + j < 27) And (j < Longitudes(i))
               ' Se copian nuevos datos en el campo de datos hasta que la trama esté completa
               TramaCop(7 + j + ContadorCop) = Valor(Mid(Programa(i), 9 + (2 * j), 2), 2)
               j = j + 1
            Loop
            ' El campo longitud de datos del "paquete de aplicación" toma un valor igual
            ' al número de datos que se han podido introducir
            TramaCop(4 + ContadorCop) = j
            ' Se actualiza el espacio libre en la trama
            ContadorCop = ContadorCop + 4 + j
            ' Se eliminan de la línea de programa los datos ya introducidos, y se actualizan
            ' los campos Longitudes y Direcciones de dicha línea
            Longitudes(i) = Longitudes(i) - j
            Direcciones(i) = Direcciones(i) + j
            Programa(i) = "S1" & Mid(Programa(i), (2 * j) + 3)
         Else
         ' No hay espacio significativo para introducir nuevos datos
            ' TP = True ; ASEN = True
            TramaCop(1) = 192 + TramaCop(1)
            ' Tipo de Trama = Trama de Información
            TramaCop(2) = 0
            ' Se prepara el CRC para ser calculado
            TramaCop(3 + ContadorCop) = 0
            TramaCop(4 + ContadorCop) = 0
            ' Se llama al procedimiento encargado de gestionar la transmisión de la trama
            Call InsertarTrama(TramaCop(), ContadorCop)
            ' Se reinicializa el contador de datos de la trama y el primer byte de cabecera
            ' preparándolos para configurar una nueva trama
            ContadorCop = 0
            TramaCop(1) = 0
         End If
         ' Continuamos extrayendo datos de la misma línea de programa, con lo que hay que
         ' anular el incremento en el contador de líneas introducido por la instrucción
         ' Next i del bucle For
         i = i - 1
      End If
   Next i
```

```
' TP = True ; ASEN = True
TramaCop(1) = 192 + TramaCop(1)
' Tipo de Trama = Trama de Información
TramaCop(2) = 0
If ContadorCop < 26 Then
' Hay espacio suficiente en la trama para introducir un "paquete de aplicación" con
' la instrucción de "Fin de copia en RAM", que consta de 2 bytes
   ' Se introduce el código de instrucción correspondiente
   TramaCop(ContadorCop + 3) = 9
   ' Se introduce un byte con el valor del número de instrucciones de copia que han
   ' sido necesarias para cargar el programa
   TramaCop(ContadorCop + 4) = TotalInstruc
   ' Se prepara el CRC para ser calculado
   TramaCop(ContadorCop + 5) = 0
   TramaCop(ContadorCop + 6) = 0
   ' Se llama al procedimiento encargado de gestionar la transmisión de la trama
    Call InsertarTrama(TramaCop(), ContadorCop + 2)
Else
' No hay espacio suficiente en la trama para introducir un "paquete de aplicación"
' con la instrucción "Fin de copia en RAM", por lo que se activa la gestión de la
' transmisión de dicha trama, y se configura una nueva trama con el "paquete de
' aplicación" mencionado
   ' Se prepara el CRC para ser calculado
   TramaCop(ContadorCop + 3) = 0
   TramaCop(ContadorCop + 4) = 0
   ' Se llama al procedimiento encargado de gestionar la transmisión de la trama
    Call InsertarTrama(TramaCop(), ContadorCop)
   ' Para la nueva trama: TP = True ; ASEN = True
   TramaCop(1) = 192
   ' Tipo de trama = Trama de información
   TramaCop(2) = 0
   ' Código de instrucción = Fin de copia en RAM
   TramaCop(3) = 9
   ' Se introduce un byte con el valor del número de instrucciones de copia que han
   ' sido necesarias para cargar el programa
   TramaCop(4) = TotalInstruc
   ' Se prepara el CRC para ser calculado
   TramaCop(5) = 0
   TramaCop(6) = 0
   ' Se llama al procedimiento encargado de gestionar la transmisión de la trama
```

```
        Call InsertarTrama(TramaCop(), ContadorCop + 2)
    End If
    ' Se informa al usuario de que el envío del programa ha concluido
    Label9 = "¡Programa Enviado!"
    ' Se inhabilita el botón que activa este procedimiento
    Command6.Enabled = False
End Sub


' Este procedimiento inhabilita el cuadro de diálogo que permite al usuario cargar
' un programa en memoria RAM local del nodo seleccionado, devolviéndolo al cuadro
' de diálogo principal del formulario
Private Sub Command7_Click()
    Text2.Enabled = False
    Text1.Locked = False
    Frame2.Enabled = False
    Drive1.Enabled = False
    File1.Enabled = False
    Dir1.Enabled = False
    Label7.Enabled = False
    Label8.Enabled = False
    Command6.Enabled = False
    Command7.Enabled = False
    Command1.Enabled = True
    Command2.Enabled = True
    Command5.Enabled = True
    Label9 = ""
End Sub


' Este procedimiento determina la nueva ruta del directorio seleccionado por el usuario
' cuando este sufre algún cambio
Private Sub Dir1_Change()
    ' El texto muestra la nueva ruta
    Text2 = Dir1.Path
    ' Se establece como ruta de acceso al fichero la nueva ruta del directorio
    File1.Path = Dir1.Path
End Sub
```

```
' Este procedimiento determina la nueva unidad seleccionada por el usuario cuando
' esta sufre algún cambio
Private Sub Drive1_Change()
    ' El texto muestra la nueva unidad
    Text2 = Drive1.Drive
    ' Se establece como ruta de acceso al directorio la nueva unidad
    Dir1.Path = Drive1.Drive
    ' Se establece como ruta de acceso al fichero la nueva ruta del directorio
    File1.Path = Dir1.Path
End Sub


' Este procedimiento muestra cuales son el nombre y la ruta de acceso al fichero
' seleccionado
Private Sub File1_Click()
    If Len(File1.Path) = 3 Then
    ' El fichero se encuentra en el directorio raiz de alguna de las unidades
        ' El texto muestra la ruta de acceso y el nombre del fichero seleccionado
        ' con el formato adecuado
        Text2 = File1.Path & File1.FileName
    Else
    ' El fichero no se encuentra en el directorio raiz de alguna de las unidades
        ' El texto muestra la ruta de acceso y el nombre del fichero seleccionado
        ' con el formato adecuado, para lo cual es necesario introducir el carácter
        ' "\" entre la ruta de acceso al fichero, y el nombre del mismo
        Text2 = File1.Path & "\" & File1.FileName
    End If
End Sub


' Este procedimiento se ejecuta al cargarse el formulario, y habilita el cuadro
' de diálogo principal del mismo, deshabilitando el resto
Private Sub Form_Load()
    Label5 = ""
    Command2.Enabled = True
    Command5.Enabled = True
    Command3.Enabled = False
    Command4.Enabled = False
    Label4.Enabled = False
    Check1.Enabled = False
```

```
    Check2.Enabled = False
    Check3.Enabled = False
    Check4.Enabled = False
    Frame1.Enabled = False
    Frame2.Enabled = False
    Text2 = Dir1.Path
    Text2.Enabled = False
    Text2.Locked = True
    Drive1.Enabled = False
    File1.Enabled = False
    Dir1.Enabled = False
    Label7.Enabled = False
    Label8.Enabled = False
    Command6.Enabled = False
    Command7.Enabled = False
End Sub
```

Formulario Mensajes

```
' Este formulario muestra por pantalla los mensajes de alarma y respuesta recibidos
' por el PC. El formulario se muestra cuando se recibe algún mensaje nuevo, pero
' muestra no sólo los mensajes nuevos, sino que guarda un archivo histórico de los
' últimos 32 mensajes recibidos de cada tipo

Option Explicit

' Al ejecutarse este procedimiento el formulario queda oculto
Private Sub Command1_Click()
    Unload Me
End Sub

' Este procedimiento borra el archivo histórico de los mensajes de respuesta
Private Sub Command2_Click()
    Dim i As Byte
    ' Se recorre toda la lista circular de mensajes de respuesta
    For i = 0 To 31
        ' Se borra el contenido de cada posicion de la lista
        Respuestas.Texto(i) = ""
        ' El siguiente mensaje de respuesta que se reciba, quedará almacenado
```

```
        ' en la posición de la lista circular con índice 0
        Respuestas.siguiente = 0
    Next i
    ' El siguiente mensaje de respuesta que se reciba, quedará almacenado
    ' en la posición de la lista circular con índice 0
    Text1 = ""
End Sub

' Este procedimiento borra el archivo histórico de los mensajes de alarma
Private Sub Command3_Click()
    Dim i As Byte
    ' Se recorre toda la lista circular de mensajes de alarma
    For i = 0 To 31
        ' Se borra el contenido de cada posicion de la lista
        Alarmas.Texto(i) = ""
        ' El siguiente mensaje de respuesta que se reciba, quedará almacenado
        ' en la posición de la lista circular con índice 0
        Alarmas.siguiente = 0
    Next i
    ' El siguiente mensaje de alarma que se reciba, quedará almacenado
    ' en la posición de la lista circular con índice 0
    Text2 = ""
End Sub

' Este procedimiento se ejecuta cada vez que se carga el formulario y presenta
' por pantalla los 32 últimos mensajes de respuesta y alarma que ha recibido
' el PC
Private Sub Form_Load()
    Dim i As Integer
    ' El índice i apunta al último mensaje de respuesta recibido
    i = (Respuestas.siguiente - 1) And 31
    ' Se "limpia" la cadena de caracteres que almacenará los mensajes
    ' mensajes de respuesta
    Text1 = ""
    ' Se recorre la lista de mensajes de respuesta hacia atrás
    Do While i <> Respuestas.siguiente
        ' Se añaden los mensajes uno a uno, integrándolos en una sola cadena
        ' de caracteres
        Text1 = Text1 & Respuestas.Texto(i)
        i = (i - 1) And 31
```

```
    Loop
    ' Por último se añade un texto de fin de mensajes de respuesta
    Text1 = Text1 & Chr(13) & Chr(10) & " No hay más respuestas"
    ' El índice i apunta al último mensaje de alarma recibido
    i = (Alarmas.siguiente - 1) And 31
    ' Se "limpia" la cadena de caracteres que almacenará los mensajes
    ' mensajes de alarma
    Text2 = ""
    ' Se recorre la lista de mensajes de alarma hacia atrás
    Do While i <> Alarmas.siguiente
       ' Se añaden los mensajes uno a uno, integrándolos en una sola cadena
       ' de caracteres
       Text2 = Text2 & Alamas.Texto(i)
       i = (i - 1) And 31
    Loop
    ' Por último se añade un texto de fin de mensajes de alarma
    Text2 = Text2 & Chr(13) & Chr(10) & " No hay más alarmas"
End Sub
```