UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA UNIVERSITARIA DE INGENIERIA TECNICA DE TELECOMUNICACION

TRABAJO FIN DE CARRERA

TITULO: ESTUDIO DEL ESTANDAR PROFIBUS. IMPLEMENTACION DE UN PROGRAMA DE APLICACION

ESPECIALIDAD: TELEFONIA Y TRANSMISION DE DATOS

AUTOR: MARIA ARANZAZU ALDAY VERDU

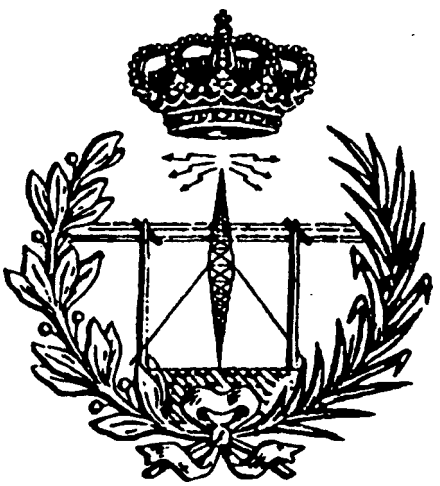
TUTOR: DOMINGO MARRERO MARRERO

MES/AÑO: JUNIO-95

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA UNIVERSITARIA DE INGENIERIA TECNICA DE TELECOMUNICACION



TRABAJO FIN DE CARRERA

TITULO: ESTUDIO DEL ESTANDAR PROFIBUS. IMPLEMENTACION DE UN PROGRAMA DE APLICACION

Calificación : **Fecha :**

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

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

I N D I C E

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

INTRODUCCION GENERAL

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

1 INTRODUCCION GENERAL

La idea original de este proyecto surgió por parte de la CEE (con subvención por medio del Ministerio de Industria) para verificar cómo equipos heterogéneos de diferentes fabricantes pueden intercambiar información sin pérdida de significado y evaluar la Tecnología de Buses de Campo Normalizados y promocionar la inclusión de buses estandarizados en los equipos de las compañías.

De este modo, siete empresas formaron el consorcio SIGMABUS: DISEL, ELIOP, INGELECTRIC-TEAM, ISOLUX-WAT, ROBOTIKER, SAC y SAINCO.

Así, me ofrecí para realizar dicho desarrollo en la empresa ISOLUX-WAT para así poder realizar a la vez mi proyecto de fin de carrera con una idea bastante interesante. Me puse en contacto con ellos y les pareció una buena idea.

Tras ello surgió la idea de presentar este desarrollo como proyecto de fin de carrera de la especialidad cursada, y para ello contacté con Domingo Marrero para que lo tutorizara, proponiéndole una buena idea, pues se permitirá conocer una arquitectura de comunicación en entorno industrial nueva y con un desarrollo e implementación propia.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

INTRODUCCION AL PROFIBUS

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

2 INTRODUCCION

2.1 OBJETIVOS

El objetivo fundamental que se plantea en el presente documento, consiste en conseguir que equipos de diferentes fabricantes españoles puedan comunicarse entre si, pudiendo coexistir de esta forma en las mismas instalaciones, y adicionalmente comunicarse con un amplio rango de dispositivos pertenecientes a fabricantes internacionales. Por otra parte supone la utilización de sistemas de comunicaciones de altas prestaciones, basados en estándares avanzados, los Buses de Campo de tipo PROFIBUS.

El resultado fundamental de este proyecto ha sido el desarrollo del software de una de las empresas perteneciente a dicho consorcio, ISOLUX-WAT, para su equipo W-90. Los desarrollos de este proyecto tienen el carácter de producto y han sido incorporados a las soluciones técnicas y catálogos comerciales.

Para el desarrollo y validación de adaptadores PROFIBUS, se han contemplado los siguientes trabajos a lo largo del proyecto:

- Especificación de requisitos funcionales de los adaptadores PROFIBUS, para fijar la funcionalidad y las características físicas de los desarrollos a realizar.

- Definición de un marco de referencia en cuanto a la interpretación de los estándares PROFIBUS.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

- Desarrollo del Software de aplicación necesario. (El Hardware ya estaba diseñado por la empresa para incorporar dicho equipo al bus de campo).

- Comprobación de la interoperabilidad entre un equipo W-90 de la empresa ISOLUX-WAT y un PC.

2.2 DESCRIPCION TECNICA DEL PROYECTO

2.2.1 PERFIL PROFIBUS

Los protocolos PROFIBUS para comunicación a través de Bus de Campo, disponen de una arquitectura modular de tres capas (Fig.2.1), correspondientes a los niveles 1, 2 y 7 (Físico, Enlace y Aplicación) del modelo de referencia OSI. Todos ellos están regulados y registrados en una única norma, la norma alemana DIN 19245.

Esta norma se divide actualmente en dos partes, ya que está en fase de desarrollo la parte tercera. Estas dos partes son las siguientes:

PARTE 1.- Define el bus de campo: Técnica de transmisión de datos, métodos de acceso al medio y protocolos de transmisión, interfase de servicio a la capa de aplicación, manejo.

PARTE 2.- Define la interfase de usuario FMS para comunicaciones universales: Modelo de comunicación, servicios para la aplicación de usuario, especificación de protocolo, codificación, interfase de capa de enlace de datos, manejo.

PARTE 3.- Define la interfase de usuario PROFIBUS-DP para rápidos intercambios de datos cíclico (periféricos descentralizados, monomaestro).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

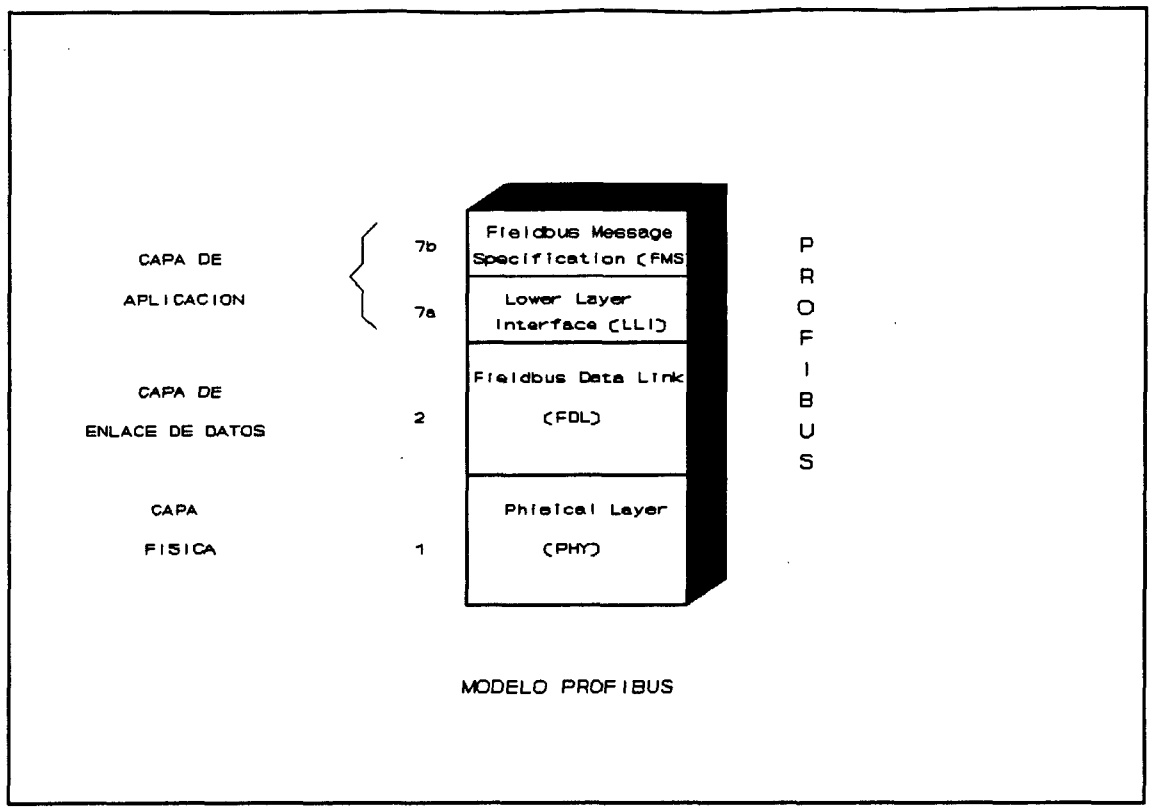


FIGURA 2.1

Las características de los diferentes protocolos presentes en la especificación son las siguientes:

NIVEL FISICO:

*Topología: Bus

*Rango de direcciones: 127 estaciones en un mismo segmento, de las cuales 32 pueden ser activas(poseedoras de testigo, maestras).

*Método de transmisión: Half-Duplex y transmisión asíncrona con UART (8 bits, 1 bit de paridad, 1 bit de stop) según norma DIN 66022/66203. Codificación NRZ.

*Medio de transmisión: Par trenzado.

*Longitud del cable: de 0,2m. a 1,2Km.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

*Velocidad de transmisión: 9,6, 19,2, 187,5 y 500 Kbps.

*Conector: DIN 41652 (ISO 4902) de 9 pines.

NIVEL DE ENLACE:

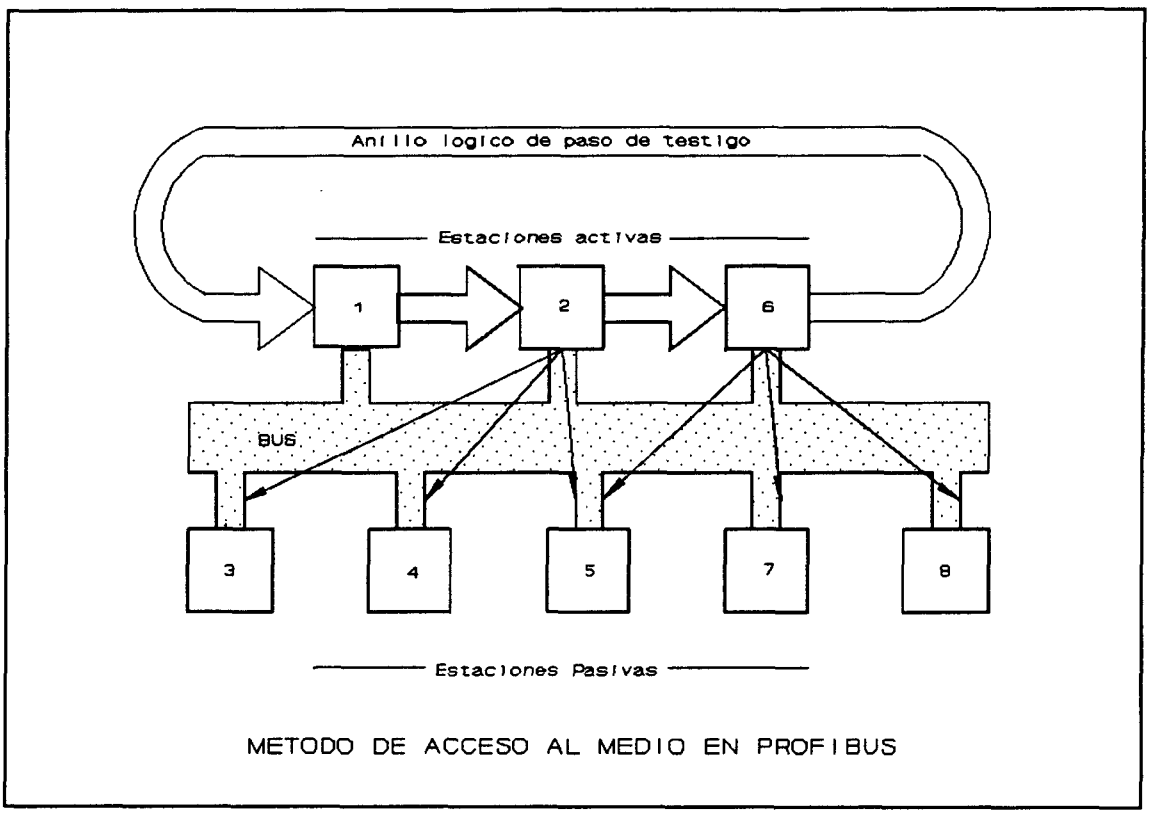


FIGURA 2.2

El protocolo de nivel de enlace de la especificación PROFIBUS, recibe el nombre de FDL (Fieldbus Data Link) y comprende normas para el control del enlace y para el control del acceso al bus.

En lo que se refiere al control del acceso al medio, la norma aplica un método híbrido entre la técnica de paso del testigo y "polling" o maestro-esclavo. Dicho de otro modo, supone que el testigo pasa periódicamente por todos los maestros conectados al bus, de forma que el maestro en posesión del testigo (estación activa) pueda interrogar a sus

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

estaciones esclavo (estaciones pasivas), permitiéndoles el acceso al bus mediante mensajes de respuesta a las peticiones por él emitidas.(Fig.2.2)

En cuanto al control del enlace o de la transmisión, el protocolo FDL proporciona los siguientes servicios al nivel superior (aplicación):

* Envío de datos sin reconocimiento.
* Envío de los datos con reconocimiento.
* Petición de datos con respuesta.
* Envío y petición de datos.
* Petición cíclica de datos con respuesta.
* Envío cíclico y petición de datos.

NIVEL DE APLICACION:

Las operaciones del nivel de aplicación, están definidas por los protocolos FMS (Fieldbus Messaging Specification) en lo que se refiere a la interacción con el usuario, y LLI (Low Level Interface) para intercambio de datos con el nivel FDL.

El subnivel FMS puede considerarse un subconjunto de la norma ISO 9506 MMS (Manufacturing Messaging Specification). Al igual que MMS, asume una modelización de los datos en base a objetos. Los tipos de servicios que ofrece al usuario son los siguientes:

* Gestión de conexión.
* Intercambio de variables.
* Gestión de eventos.
* Gestión e invocación de programas.
* Volcado bidireccional de dominios.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* Gestión de estados.

El subnivel LLI, se encarga básicamente de convertir los servicios FMS a servicios LLI.

2.2.2 CONTEXTO PROFIBUS.

PROFIBUS (PROcess FIeld BUS) fue un proyecto financiado por el Ministerio Federal de Investigación y Tecnología de Alemania en el que participaron 18 empresas para especificar, desarrollar y probar un nuevo bus de campo digital para instrumentos y dispositivos de control en el nivel más bajo de la jerarquía de automatización. El trabajo realizado entre 1987 y 1991 se concretó en un estándar alemán, concretamente DIN 19245, donde se especifica el protocolo PROFIBUS completo.

Los usuarios y fabricantes de productos PROFIBUS, se reúnen en el PROFIBUS User Group, donde más de 100 empresas participan en tareas de investigación, desarrollo y divulgación del estándar. La experiencia acumulada por estas empresas y los productos disponibles, posicionan a PROFIBUS como una de las opciones principales a nivel mundial para cubrir las necesidades entre instrumentación inteligente.

2.2.3 TERMINAL REMOTA PROGRAMABLE W-90 DE ISOLUX-WAT.

Esta terminal es usada para la telemedida y telecontrol de procesos de forma distribuida, con posibilidad de control lógico y analógico programable. Consta de un módulo o rack de control donde se alojan las tarjetas de la CPU, memoria, comunicaciones, controladora del bus de entradas/salidas, fuente de alimentación, modem, etc. Estas tarjetas se unen por medio de un back panel según un bus paralelo estándar. Este rack, por medio de las tarjetas alojadas en él, realiza las tareas principales de la terminal W-90 en cuanto control y proceso de la información, comunicaciones y protocolos, bases de datos,

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

autochequeo y diagnóstico, registro cronológico,etc.

Por medio de la tarjeta de acceso a la red de datos (ARD) se unen al módulo de control todas las tarjetas de interfase de entradas o salidas que se necesiten. Este bus de E/S, es un bus paralelo, específicamente diseñado, de alta inmunidad, tolerante a los fallos, e implementado en un cable plano o panel posterior, que recorre las tarjetas de E/S de la terminal W-90 hasta un máximo de 32 por cada controladora del bus de E/S. Las tarjetas de interfase de E/S se alojan en grupos de 16 en sus correspondientes racks y cada una de ellas controla 32 entradas o salidas, a excepción de las analógicas. Las tarjetas de interfase de E/S son inteligentes, de forma que pueden preprocesar los datos y comprobar toda la información, antes de actuar sobre las salidas, o de enviarla al módulo de control. También disponen de capacidad de autodiagnóstico on-line, comunicación redundante por el bus de E/S, eliminación de rebotes y filtrado en las entradas digitales, con la posibilidad de registrador cronológico, estados con memoria y contadores.

La tarjeta de acceso a la red de datos de la terminal W-90 presenta las siguientes características:

- 2 Microprocesadores 80C186.
- Memoria: 256 Kb de EPROM y 512 Kb de memoria RAM, no volátiles.
- Circuito supervisor de tensión y de las CPUs.
- LEDs para la indicación de estados internos.
- 6 Puertas serie de comunicaciones, 3 de RS232C y otras 3 de RS485.
- Banco de puentes para configuraciones.
- Posibilidad de conectarlo adaptada para PROFIBUS y FIP.
- Alimentación de +5v-0,5Amp.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

LA COMUNICACION EN LA TECNOLOGIA DE LA AUTOMATIZACION

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

3 LA COMUNICACION EN LA TECNOLOGIA DE LA AUTOMATIZACION

3.1 INTRODUCCION

La tendencia hacia la descentralización de la automatización está influenciando fuertemente el desarrollo técnico de los noventa. La microelectrónica moderna ofrece desarrollos de sistemas con posibilidades casi ilimitadas en la realización de productos innovadores.

El cambio de ocupación en los equipos periféricos de proceso ha producido la necesidad de una nueva comunicación. Antiguamente, la transferencia de señales ocurría cuando el proceso era acoplado a un ordenador, pero ahora los componentes inteligentes de un proceso requieren una comunicabilidad en sus propias acciones de funcionamiento. Cuando se introdujo la interfase digital, hizo casi inevitable el paso hacia la red local. El tipo de red en la base del nivel de automatización inmediatamente cercano al proceso técnico, ha sido recientemente denominado **fieldbus**, o lo que es lo mismo, bus de campo.

Parecida a la tendencia en el desarrollo de redes de ordenadores incompatibles, existe un gran número de incompatibilidades en el área de campo, haciendo componentes de distintos fabricantes no integrables en un sistema común. Esto impide la rápida expansión y la económica utilización de equipos de campo inteligentes y produce una tendencia hacia los sistemas abiertos. Por lo tanto, la industria ha hecho un esfuerzo para desarrollar un estándar en el campo de las comunicaciones entre ordenadores para fabricantes y automatización de edificios en los últimos años. La meta es la comunicación abierta, de tal forma que los componentes de automatización de diferentes vendedores puedan ser conectados juntos en una red común sin problemas de incompatibilidad.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

En 1987, el Ministerio de la Alemania Federal para la investigación y tecnología solicitó el proyecto de colaboración 'Field Bus'. Trece compañías y cinco institutos trabajaron juntos para desarrollar un sistema de bus de campo abierto de nombre PROFIBUS (PROcess Field BUS), basado en el modelo de referencia de ISO/OSI. La meta oficial del proyecto era propagar rápidamente el PROFIBUS como el estándar de bus de campo. El éxito importante se produjo en 1991, cuando PROFIBUS fue estandarizado en el estándar alemán, DIN-19245. Para empezar, un estándar de bus de campo completamente especificado es disponible en cualquier parte del mundo por cualquiera sin necesidad de permiso alguno.

El grupo de investigación de la tecnología del microprocesador en el 'Forschungszentrum Informatik' (FZI, traducido como centro de investigación de la ciencia de ordenadores) en Karlsruhe (Alemania), soportó la responsabilidad de una parte esencial del proyecto - el protocolo de usuario y la interfase de usuario. El grupo estuvo también implicado en la especificación de su propia interfase de usuario. Fue especialmente importante el concepto de desarrollo y realización de los servicios de usuario y el protocolo de usuario de PROFIBUS.

A principios de 1988, el grupo de investigación fue capaz de ofrecer a sus colaboradores un controlador PROFIBUS para AT compatibles con PCs. La placa está basada en el chip microprocesador V25, y comprende la totalidad de las funciones de un maestro sirviendo además como hardware de referencia. Después, el grupo desarrolló controladores para varios sistemas de control programables, realizó conexiones con los microcontroladores de la familia 8086, 8051 y 68000, así como la implementación de el protocolo PROFIBUS para el microcontrolador 8051.

El grupo de investigación también desarrolló pruebas y funciones de métodos de evaluación para los componentes del PROFIBUS. Se desarrolló, entre otras cosas, una red de medida de funciones, así que las fábricas de componentes podían comprobar sus funcionamientos bajo condiciones normales. Se creó Una experimentación PROFIBUS y

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

laboratorio de demostración para este propósito, en la que casi todos los componentes PROFIBUS en el mercado se integraron dentro de una red piloto.

Este capítulo es una introducción a buses de proceso y campo. En la tecnología de la comunicación, un bus es una línea común para transmisión de información entre todas las estaciones. Los buses de proceso y de campo transmiten datos para control de procesos de producción en sistemas de producción automatizados. Es análogo a una red telefónica uniendo diferentes equipos de automatización.

3.2 AREAS DE APLICACION PARA BUSES DE PROCESO Y CAMPO

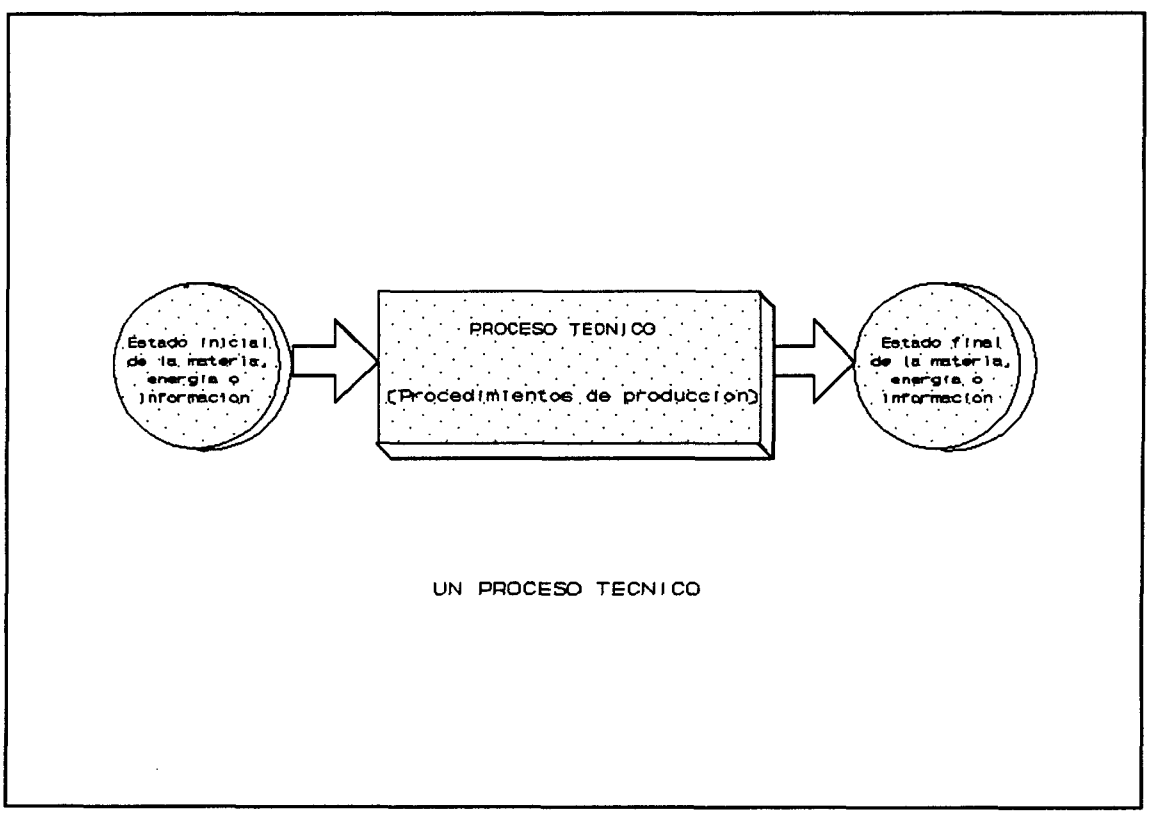


FIGURA 3.1

Las industrias requieren producto, material y flujo de información. El cambio en detalles de producción, por ejemplo, la información concerniente del proceso de producción

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

de un área determinada (con respecto al cronograma y topología), se realiza sobre el bus de proceso y campo. Dadas las distintas filosofías de usuario (para control, servicios, operaciones, mantenimiento, ...), los requisitos para el tratamiento de producto en los diferentes procesos y áreas de aplicación, se hace indispensable un sistema de comunicación adaptable. Esta flexibilidad del sistema puede conseguirse realizando diferentes opciones o subconjuntos de funciones.

La determinación precisa de especificaciones de comunicación facilita enormemente la adaptación de un sistema de comunicación a diferentes áreas de aplicación. Los términos **proceso** y **proceso técnico** pueden ser definidos como sigue (figura 3.1):

* Un proceso es un conjunto de sucesos de sistemas interactuando, los cuales transforman, transfieren o almacenan materia, energía o información.

* Las variables físicas de un proceso técnico pueden ser adquiridas e influenciadas por medios técnicos.

Existen tres procesos diferentes tratados en la tecnología de la automatización (figura 3.2):

* Los **procesos técnicos** sirven al primer propósito de una empresa, por ejemplo, fabricación de productos, producción de energía, comprobación de partes fabricadas y almacenamiento de productos manufacturados.

* Los **procesos de aplicación** forman sistemas de adquisición, procesamiento, control y salida de información. Normalmente son controladores de proceso y componentes microelectrónicos programables, por ejemplo los controladores programables.

* Los **procesos de comunicación** en la ciencia de los ordenadores son procedimientos dinámicos los cuales intercambian detalles entre parejas de comunicación, a través del procesamiento de programas de comunicación. El conjunto de procesos de comunicación que interactúan en un enlace común (cable, fibra óptica,...) se denomina sistema de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

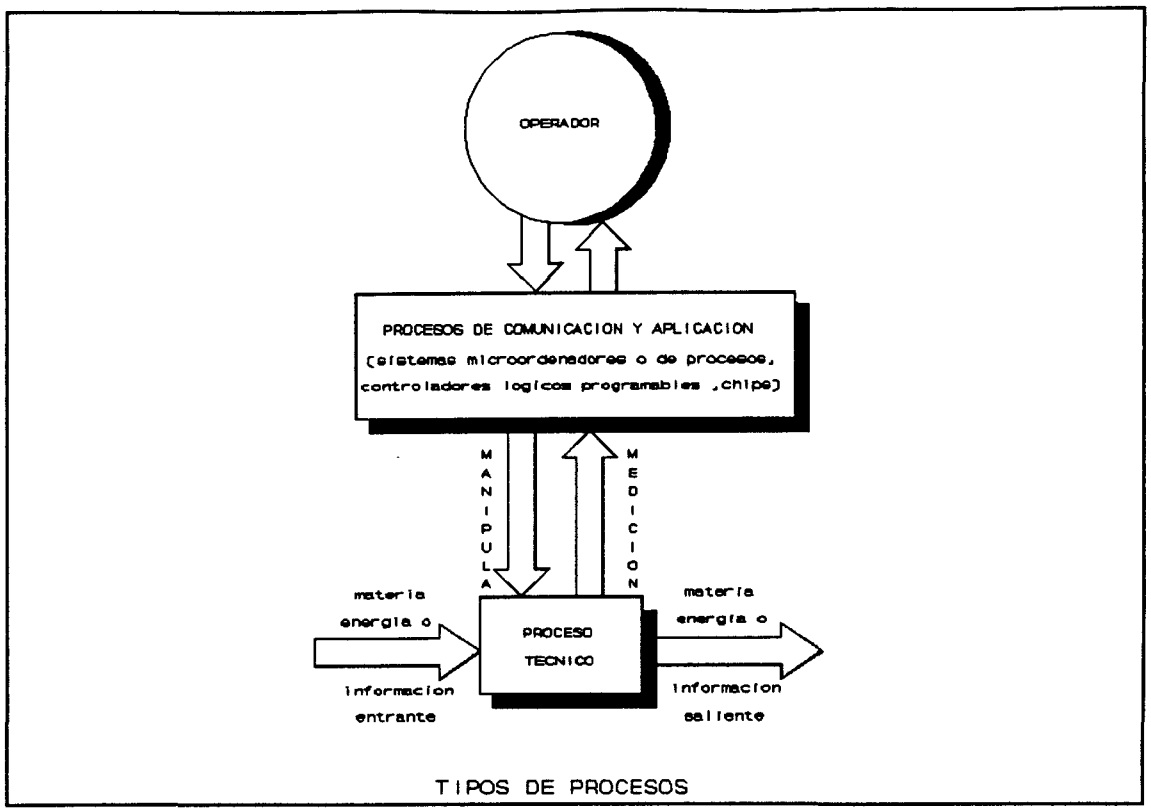


FIGURA 3.2

comunicación.

Los procesos técnicos pueden ser clasificados considerando los procedimientos que prevalecen:

* **Procesos de flujo** (procesos continuos), por ejemplo la producción de energía en plantas eléctricas, trabajos del acero, procesos caloríficos, procesos químicos.

* **Procesos secuenciales** (procesos batch), por ejemplo los procesos de carga, comienzo y finalización de procesos, procesos de producción, procesos de testeo.

* **Procesos de tipo discreto** (producción de partes discretas), por ejemplo procesos de almacenaje, procesos de transporte, procesos de circulación.

Los procesos técnicos se pueden además distinguir por la manera en que los elementos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

procesados soportan cambios. Esto incluye:

* Procesos de alteración.
* Procesos de moldeo.

Los procesos de alteración se caracterizan por la alteración química del material, por ejemplo la producción de acero o la producción de productos químicos intermedios. Los procesos de moldeo comprenden el trocear o cortar el acero en la ingeniería de la producción, el moldeo en un proceso de ensamblado o la fabricación.

Otras áreas incluyen:

* Material en bruto y producción de energía.
* Dirección de servicios de construcción (automatización de construcciones).

A continuación se presenta otras áreas de aplicación y se explica las características más importantes de los buses de proceso y campo.

3.2.1 INGENIERIA DE PRODUCCION

Una empresa es mucho más que fabricar. Hoy en día el área de fabricación debe ser flexible para adaptarla al área de ventas. Esto significa que en toda fase de producción, el flujo libre de información entre las estaciones debe ser asegurado incluso cuando cambien los requisitos de los productos. Estos lazos de control requieren un desarrollo del área de producción. Esto concierne tanto al flujo de producto y control de calidad, como a los departamentos de compras y de ventas.

Para manejar el flujo de datos en el nivel físico, el sistema de comunicación **fieldbus** (bus de campo), debe proporcionar las estructuras lógicas de comunicación. Para la

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

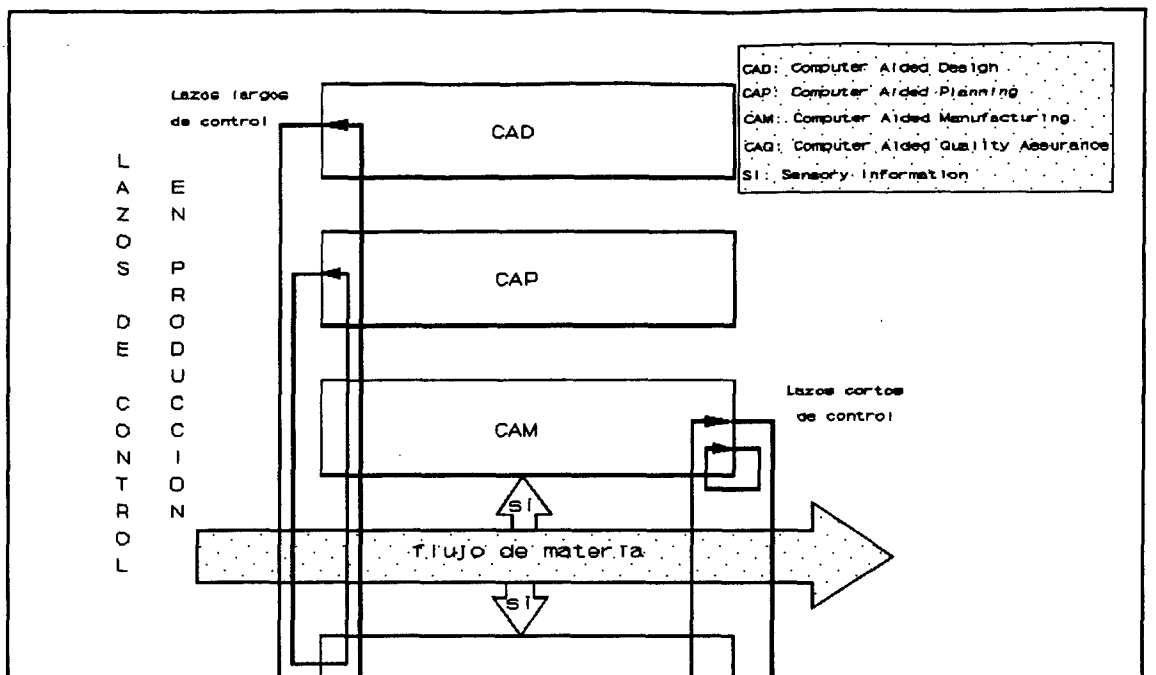


FIGURA 3.3

aplicación, éstas son las relaciones de comunicación entre los procesos, los objetos direccionados (datos) y los servicios usados. Los servicios son operaciones de objetos de comunicación de una clase definida. Un valor medido, por ejemplo, es un objeto que pertenece a la clasificación de variables. Leer y escribir son servicios aplicables a esta clase de objetos.

Además, algunas de estas áreas de aplicación necesitan límites fijados del tiempo de transmisión del mensaje y protección de acceso a datos no autorizados o servicios no permitidos.

Las relaciones de comunicación necesitan componentes que cubran varias áreas de aplicación en el nivel de campo. Estas relaciones están, por ejemplo, condicionadas por la secuencia de fases de producción en una compañía de fabricación en la cual probablemente

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

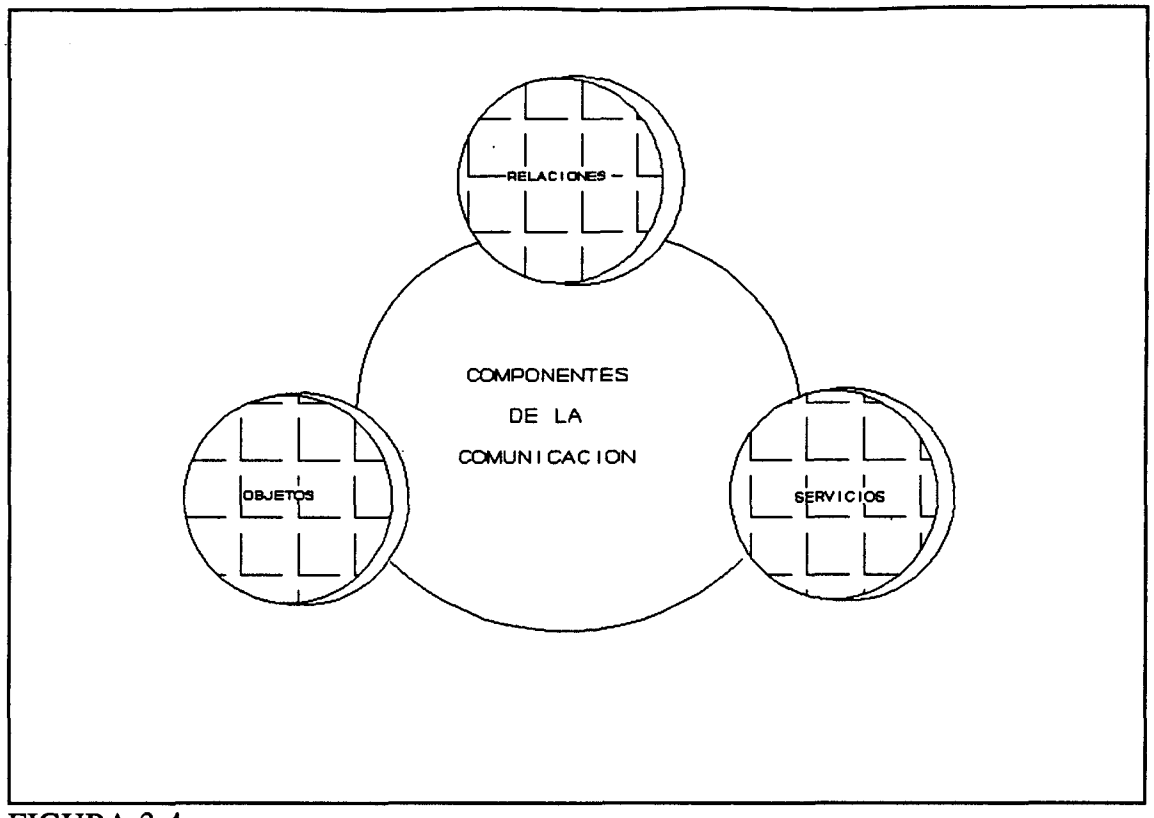


FIGURA 3.4

prevalece la producción de distintas partes. Requisitos totalmente distintos serán los de sistemas para procesos continuos, por ejemplo, la continua producción de una fábrica.

Para fabricar un producto, el proceso de fabricación de producción necesita una buena lista disponible de materiales en bruto y suministradores, piezas de trabajo, herramientas y productos medio acabados. Estos, por otro lado, requieren ventas, almacenaje, montaje intermedio y transporte. Tales requerimientos determinan el perfil de comunicación del bus de proceso y campo. A través de las relaciones de comunicación se realizan las diferentes necesidades de información y coordinación existentes entre estas fases de producción.

Para la implementación de un bus de campo óptimo, es importante conocer qué tipos y qué tasa de flujo de mensajes son posibles. En el área de equipos de campo, los cuales ejecutan tareas directamente concernientes a los procesos (por ejemplo controles de motores),

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

es de primera importancia la capacidad de tiempo real en un bus de campo (medida en ms). Otras características, sin embargo (por ejemplo, la transferencia eficiente de una gran cantidad de datos o de sobreescritura de parámetros durante la operación), son de menor importancia.

En el campo de las aplicaciones de moldeo, la tarea de carga en línea (on-line) y fuera de línea (off-line) de programas es de vital importancia. En este caso es esencial conocer si la información relevante del proceso necesita ser transmitida en tiempo real.

Normalmente, los aparatos soportados por los buses de campo son baratos y pueden ser controlados por controladores programables. Generalmente, la cantidad de datos transferida es pequeña (< 1 Kbyte) y cubre una distancia máxima de varios cientos de metros. En la figura 3.5., las tres columnas de la derecha muestran las características comparativas de los mensajes de campos típicos. Los mensajes que no son en tiempo crítico, mostrados en las tres columnas de la izquierda, tienen necesidades enteramente diferentes.

3.2.2 INGENIERIA DE PROCESO

Las tecnologías de procesos para productos intermedios, como por ejemplo en refinerías, plantas de producción químicas y plantas de acero, son procesos mixtos. Los resultados de control no suelen ser disponibles inmediatamente después de ejecutarse. Las intervenciones para la optimización de las propiedades del producto se realizan por medios físicos, como por ejemplo modificación de la composición del material en bruto, temperatura y presión.

Las constantes de tiempo de tales procesos son considerablemente grandes (sobre 100 más). Esto no significa sin embargo que el tiempo no juegue un papel importante. Especialmente en estos procesos, la calidad de una intervención de control se determina por el cumplimiento de los intervalos de tiempos predefinidos. Aquí el intervalo de tiempo de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

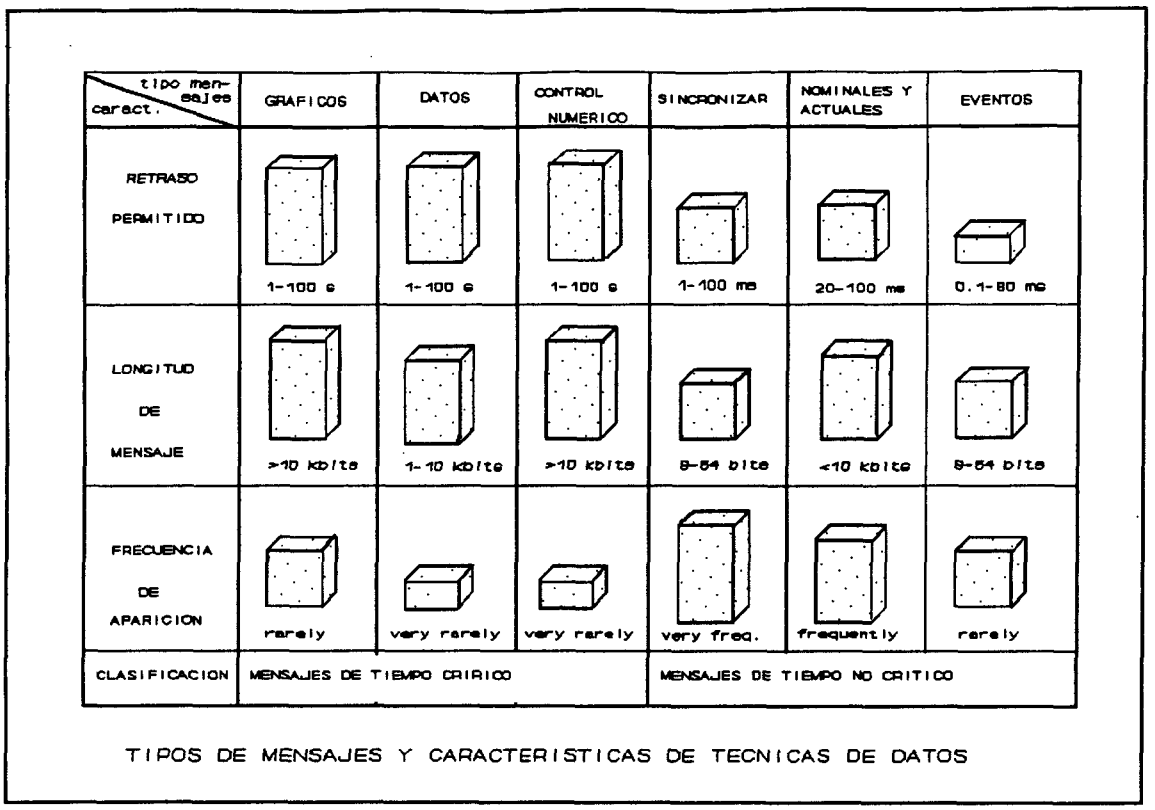


FIGURA 3.5

muestra de una variable (por ejemplo el escrutinio regular de un valor de medida) es decisivo, mientras que no es importante en el caso de ingeniería de producción.

El flujo de información en aplicaciones de proceso abarca áreas mayores que en las de producción. De este modo, se debe esperar un desbordamiento de mensajes provenientes de unidades autónomas diferentes de una compañía por el canal de transmisión.

Otra característica importante es la duración de un proceso, el cual determina para un alcance considerable las opciones de servicio y extensiones funcionales del equipo técnico. Los procesos continuos son rara vez interrumpidos, algunas veces solamente una vez al mes o al año, así que la máquina tiene que ser servida durante la operación. En ingeniería de proceso, los procesos secuenciales están entre procesos discretos y continuos (por ejemplo, la fabricación de productos farmacéuticos). El equipo manejaría variables tales como

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

pequeños tiempos de vida de los productos, interrupción frecuente de los procesos de producción así como frecuentes cambios en los productos. Aquí el procesamiento de la información es similar a los procesos discretos.

Los datos transferidos por un bus de campo y procesados por equipos de campo, sus asignaciones funcionales y sus transmisiones en el bus en diferentes fases operacionales, dependen de las características arriba mencionadas de tecnologías de procesamiento y procesos de producción.

3.3 INTERCONEXION DE SISTEMAS DE COMUNICACION INDUSTRIAL

Ya que el flujo de información en una compañía puede ser muy complejo, se estructura normalmente en distintos niveles jerárquicos. Se asignan niveles de comunicación los cuales sitúan diferentes necesidades en la tecnología de la comunicación. En el nivel de planificación, por ejemplo, se transmiten los datos de ingeniería preparados a los lugares de trabajo. Aquí un gran volumen de datos que no son de tiempo crítico se transmiten normalmente por MAP (Manufacturing Automation Protocol). En este nivel, se dispone de las rutas de la información por la infraestructura, las cuales pueden ser utilizadas de diferentes formas (por ejemplo, bases de datos de usuario y buzones). Estas pueden ser integradas por las estaciones de trabajo en paquetes de trabajo que pueden ser usados , por ejemplo, en sistemas de planificación de producción (PPS). Lo más lejano va desde el nivel de planificación bajando al nivel de campo.

Para la vigilancia y control de procesos técnicos, las interfases apropiadas deben ser disponibles por el usuario y el equipo periférico. Para esto, se deben reflejar los requisitos técnicos en un informe cuando se planifique la interconexión de sistemas. Las consideraciones históricas (por ejemplo, los más antiguos interfases estandarizados) también juegan un importante papel. Estos tienen un tiempo de vida largo, particularmente en algunos sectores tradicionales de la industria, como la industria textil y la automatización de los

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

edificios. La existencia de cableados y conexiones que representan unas inversiones de muchos años, se usarían, por lo menos por un período de tiempo transitorio. El coste para los componentes conectables (sensores, actuadores, controladores lógicos programables(PLC),...) va desde diez a varios miles de dólares. Estos requieren bajos precios para la conexión a la infraestructura de comunicación, lo que aumentaría un poco el precio original de los componentes.

Una arquitectura de red de información consistente conteniendo distintos niveles, es un prerequisito indispensable para la producción de ordenadores integrados, control y vigilancia de procesos. Para una compañía tener una poderosa y consistente red de información, debe introducir los tipos de red adaptadas en los diferentes niveles de información para la comunicación con altos y bajos tipos de redes por equipos enlazados apropiadamente.

3.3.1 LOS NIVELES JERARQUICOS DE REDES DE INFORMACION

El creciente grado de automatización en ingenierías de proceso y producción, demanda el establecimiento de estructuras de comunicación jerárquicas. Los centros de interés en fabricación integrada de ordenadores (CIM), son los conceptos de red que implican sistemas de comunicación que envuelven todos los niveles de automatización.

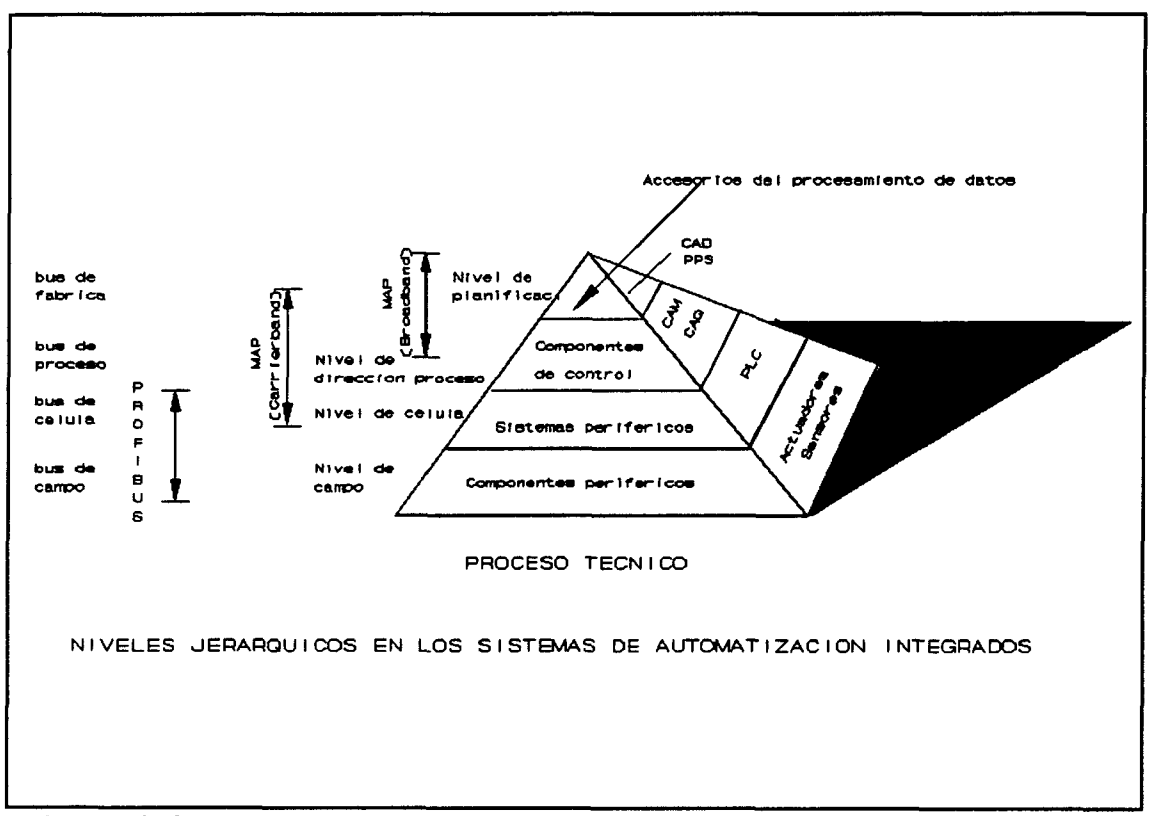
Estos sistemas de comunicación necesitan estructuras para las conexiones de establecimiento entre los departamentos de planificación / automatización, los departamentos de márquetin / técnico y también para capacitar la conexión a redes de comunicación públicas.

La variedad de equipos de diferentes vendedores para todas las clases de aplicación de campos necesita tecnología estandarizada. Los esfuerzos de la estandarización, no obstante, deben recogerse en un informe con los diferentes medios de transmisión, las

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

necesidades para la transferencia (volumen de datos, tiempos de respuesta) y la diversidad de sistemas, controles y estaciones de trabajo para interconectarlas (estaciones de ordenadores descentralizados, mainframes).

La figura 3.6., muestra la estructura de la producción del ordenador integrado. Todo nivel jerárquico tiene necesidades en el medio de transmisión físico (cable, fibra óptica, radio), en el método de transferencia (por ejemplo, protegido o no, banda base o ancho de banda) y en actuación. Condiciones de menor importancia son los parámetros como costo de la instalación, límites físicos y legales, lo ya invertido y por último,pero no menos importantes que las anteriores, las tasas de precio y actuación.



NIVELES JERARQUICOS EN LOS SISTEMAS DE AUTOMATIZACION INTEGRADOS

FIGURA 3.6

Existen muchos tipos de redes para toda clase de condiciones y necesidades, las cuales son, sin embargo, normalmente estándares específicos de vendedores.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

El nivel de campo

El nivel más bajo de la jerarquía de automatización es el nivel de campo el cual incluye equipos de campo, tales como sensores y actuadores. La tarea de estos equipos es transferir datos entre el producto manufacturado y los procesos técnicos. Los datos de fabricación del producto se obtienen por medición (por ejemplo, la identificación del producto y la calidad del mismo). Los valores medidos son las bases para la intervención en procesos técnicos, o comprobaciones. Por medio de la medición y la comprobación, las herramientas de producción implicadas pueden ser óptimamente vigiladas y controladas.

El tipo de medición y comprobación es diferente dependiendo del proceso técnico (continuo, procesos de parte discreta). Pueden ser, binario o analógico. Los valores medidos se pueden disponer por un corto período de tiempo (dinámico) o por un largo período de tiempo (casi-dinámico). Estos valores pueden servir como entrada a varias funciones como:

* representación
* almacenaje
* documentación
* comprobación
* conversión

El nivel de control (no representado en la figura 3.6) se puede situar justamente encima del nivel de campo, que es el caso de las así llamadas funciones indirectas, por ejemplo, regulación y comprobación. Aquí influye el procesamiento directamente en el nivel de campo con señales de control y se diseñan pequeños lazos de control. Los datos de optimización, diagnóstico y vigilancia fluyen en lazos grandes de control y se dirigen a las funciones de capa del nivel de control, pero no tienen regulación directa o influencia funcional en el nivel de campo. Esta información que fluye a las funciones del nivel de control, proporcionando nueva calidad de producción, es para el usuario el aspecto más

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

importante de un sistema de comunicación. Para realizar tal sistema de comunicación, es de gran importancia un consistente concepto de las estructuras de comunicación en todos los niveles, por ejemplo, la representación de los objetos de aplicación (valores medidos, parámetros de control) y la prestación y calidad de los servicios de comunicación (lectura, escritura, notificaciones, ...).

El bus de campo se usa para la transferencia de información en el interior así como también fuera del lazo de control. Por esto, la representación y el transporte de objetos se debe adaptar a las necesidades del bus de proceso y campo. Por la representación del objeto nosotros queremos decir la representación binaria exacta de un objeto de aplicación, por ejemplo, un valor medido, estos objetos se transmiten sobre el bus de forma exacta. Debido a requisitos de regulación los cuales tienen que ser estrictamente conservados en un proceso técnico (tiempo real), la aplicación en el interior del lazo de control requiere funciones de transporte cíclico que transmitan información fuente a intervalos regulares. Consideremos el ejemplo de un sistema de obtención de medición de datos. Esto significa que la información obtenida a intervalos regulares por sensores debe ser continuamente leída mediante "polling".

Estos valores medidos que llegan periódicamente se transmitirían al destinatario a intervalos de tiempo por un llamado servicio de transporte cíclico (por ejemplo, un controlador programable) para un mayor procesamiento. La representación de datos debe ser lo más corta posible para reducir el tiempo de transferencia de mensaje en el bus (tiempo de respuesta limitado).

Los mensajes en el bus se pueden acortar cuando las partes cíclicas no se transmiten todo el tiempo pero se confirman de antemano. En el ejemplo 3.3, debe ser información sobre el tipo de dato (por ejemplo un entero de 16bits). La información en esta forma no se necesita directamente en el nivel de control, ya que normalmente estos datos (dato en bruto en el ejemplo) todavía necesita preparación. Este método se llama transferencia de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

información orientada a señal. Las implementaciones futuras, sin embargo, implicarán un flujo de información orientada a aplicación (por ejemplo control de una máquina) para transferencia de información para y desde el nivel de control. Esto significa que para propósitos de control no es necesario transmitir cada valor sencillo al nivel de control. Sería suficiente transmitir un valor medio en intervalos de un minuto. Los valores fuera del ancho, por otro lado, tendrían alta prioridad. Esta transferencia de información orientada al trabajo necesita un equipo unido a la red de comunicación superior, la cual normalmente será una red local (LAN). Este equipo de unión sólo convierte la representación de la información.

El procesamiento de funciones periódicas se debe ejecutar por subrutinas las cuales se distribuyen a varios equipos en el bus (aplicación distribuida) así como por subrutinas las cuales se concentran en un equipo de campo (concentrado, aplicación centralizada). La tendencia más correcta es la del último caso. Además, la carga del bus puede ser reducida y aumentada la transferencia de datos al nivel de control. Si se necesita una actualización cíclica de datos en el nivel de control para funciones de control, no es tiempo crítico largo. Para otros propósitos, se dispone de una transferencia de mensaje acíclico para demanda de las parejas en comunicación.

El nivel de célula

El término célula viene de la estructura de producción en la fabricación flexible en la cual las tareas independientes son realizadas en facilidades de subproducción autónomas. Puede ser una fabricación o una célula de montaje. Debido a una configuración flexible de las células dependiendo de las necesidades específicas del producto, el sistema puede adaptarse fácilmente al producto manejado y a los cambios de cantidad.

En procesos continuos, las células corresponden a unidades operacionales. Sin embargo, tienen más conexiones cruzadas (de tomas constantes) debido al flujo de producto continuo. Los procesos de grupo aproximan más el modelo porque permiten producción

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

flexible de productos de alta calidad, incluso en pequeñas cantidades.

En el nivel de célula, el flujo de información consiste principalmente en carga de programas, parámetros y datos. En procesos con pequeños tiempos de máquina desocupados y reajustes, se hace durante el proceso de producción. En pequeños controladores se puede necesitar cargar subrutinas durante un ciclo de fabricación. Esto determina las necesidades temporales y no el carácter de la transferencia de información. Es acíclico y, dependiendo del tiempo de ciclo de máquinas o suma total, se debe realizar en un intervalo de tiempo dado. Este flujo de información es principalmente unidireccional. Ocasionalmente, sin embargo, puede ser acompañado por un flujo adicional de información del equipo de campo directamente conectado a este bus. Más aún, las sincronizaciones de máquina y manejos de suceso deben requerir tiempos de respuesta cortos en el bus. Estos requerimientos estrictos en tiempo real no son compatibles con la transferencia de tiempo excesivo de programas de aplicación, haciendo necesaria una segmentación adaptable de mensaje.

El nivel de control de proceso

El nivel de control de proceso, también llamado director supervisor de planta, consiste en células combinadas en grupos que el técnico de control procesa. Las células son subredes diseñadas con una aplicación orientada a la funcionalidad.

El proceso técnico es una subunidad de una planta de producción que puede consistir en varios procesos técnicos similares o no (por ejemplo, bloques de plantas de energía). Un proceso técnico puede ser dividido en a pie de obra y obras exteriores. Obras exteriores, por ejemplo, pueden ser los suministradores de energía de varios procesos técnicos.

La descripción del flujo de información se limitará a un proceso técnico y se usará el ejemplo de intercambio de datos con el operador de proceso. El término operador de proceso proviene de la ingeniería de procesos donde el operador (humano) era el agente

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

central que aseguraba la función propia de la planta por medio del control y la intervención donde se necesitara en el proceso técnico. En la actualidad, esas intervenciones se limitan a la fijación de los objetivos de producción, al arranque y desconexión de las máquinas y a las actividades de emergencia. Los siguientes ejemplos ilustran cuánto influencian las aproximaciones tradicionales en los diferentes sectores de la automatización moderna todavía vigentes en la tecnología de la comunicación digital moderna.

Como en tiempos pasados, cuando el procesamiento automático no era todavía conocido, las señales de las tecnologías de procesamiento se transmitían directamente por medio de equipos funcionales y de automatización al nivel de control, aunque las señales no se mostraban en grandes paneles de representación sino en las pantallas de representación. Por lo tanto, la información que llegaba debía ser digitalizada. El propósito de tal representación es limitar la información amplia a la información de trabajo y condensarla. Los diagnósticos y optimización de sistemas se influenciarán considerablemente por el flujo de información en el nivel de control de proceso.

Tal flujo de información en el nivel de control puede realizarse a través de una transferencia de datos acíclica, la cual se emplea actual y normalmente en redes MAP. En el nivel de control, la elección entre transferencia cíclica y/o acíclica será menos importante que el tipo de preparación de datos. Se está haciendo más y más dominante una aproximación orientada al objeto para información en la comunicación de MAP/MMS (Manufacturing Automation Protocol/Manufacturing Message Specification) y en estándares correspondientes (acuerdos adicionales en estándares para áreas específicas de aplicación). Esta aproximación solamente permite la interpretación del dato de aplicación cuando varios propósitos de aplicación y sectores de aplicación están interactuando.

Otras funciones del nivel de control de proceso

Además de la función principal del nivel de control, por ejemplo control de procesos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

técnicos, existen fases de instalación, arranque y desconectado de máquinas, así como también intervenciones de emergencia. Aquí, solamente se considerará la información necesaria para la instalación (por ejemplo para el arranque inicial de una máquina).

Existen dos categorías para equipos de campo:

* Equipos de campo con características especiales diseñados para aplicaciones específicas.

Normalmente un equipo así (por ejemplo un drive controlador de la velocidad) se instala solamente una vez o para la duración de la tarea. Una intervención de servicio es solamente necesaria en caso de romperse una máquina. El equipo de control debe contener toda la información de servicio necesaria para hacer otra programación o herramienta de configuración.

* Equipos de campo que obtienen su funcionalidad solamente después de su integración dentro del proceso.

Estos equipos (por ejemplo convertidores de temperatura) se producen en masa y deben ser cargados con los parámetros específicos cuando se intercambien durante el servicio. Hasta ahora, los datos necesarios para la carga eran disponibles en listas de parámetros, pero actualmente las ventajas de la comunicación digital ya facilitan trabajar en esta área. El prerequisito es un equipo de campo de base de datos localizado en el nivel de control que proporciona información para el servicio de un equipo de control o de un terminal manual (Hand Held Terminal, HHT).

Para integrar varias áreas de tecnología de automatización, las estructuras de información se hacen más y más entrelazadas. No es solamente necesario proporcionar formas diferentes de comunicación para las distintas funciones (por ejemplo, manejo,

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

mantenimiento y arranque), sino también hacer posible comunicación entre niveles.

Se puede comprobar en la figura anterior, que PROFIBUS no usa el nivel de planificación, así que no entraremos a explicar dicho nivel.

3.3.2 FUNCIONALIDAD DEL EQUIPO

Los equipos usados en la tecnología de la automatización se diseñan normalmente para una función particular (por ejemplo, medición o control). Adicionalmente deben adaptarse a sus medios específicos para el manejo:

* En habitaciones cerradas (sequedad, humedad).

* Al aire libre.

* En entornos con alto riesgo de explosión.

Para intervenciones de manejo y servicios, son esenciales las funciones siguientes:

* Funciones de manejo.

* Conectabilidad de equipos de servicio.

Con respecto al flujo de información de y para un equipo, son importantes los siguientes criterios:

* Puntos de acceso. Incluyen parámetros de control ajustables los cuales son llamados objetos. Los parámetros de servicio como puntos de acceso son esenciales. Por ejemplo, los temporizadores pueden leer los sistemas de comunicación y proporcionar información sobre

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

los datos de la próxima intervención).

* Volumen de datos de entrada. Para intervenciones de servicio tales como arranques y desconexiones de instalaciones industriales, es importante conocer el volumen de datos transferidos a los equipos. El volumen de datos, el tamaño del buffer del equipo receptor y la clase de servicios disponibles de transmisión (servicio de carga hacia abajo conveniente o servicio simple de escritura) influencian la duración de tales intervenciones.

* Condiciones de acceso. Normalmente, los parámetros de referencia serán solamente accedidos durante pequeños tiempos de máquina no críticos. Se debe ordenar una autorización de acceso para acceder a los parámetros que controlan los procesos técnicos o por los que se podría obtener la información confidencial sobre el proceso.

Las funciones deben ser localizadas en un equipo o distribuidas en varios. Los datos de ciertos grupos de equipos (multiplexadores, entradas/salidas de campo, equipos de subsistema de automatización) se deben transmitir sobre el bus a un concentrador de señales para preprocesamiento. Además de las conexiones de bus, existen también conexiones analógicas y digitales sencillas. Los datos son disponibles para el procesamiento a través de representación digital apropiada y transferencia a la unidad de procesamiento correspondiente.

Equipos de campo

Los equipos de campo se consideran equipos conectados directamente al proceso físico (equipos de interfase de procesos), que incluye:

* Elementos de captación (sensores).

* Elementos de control final (actuadores).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* Lectores de código de barras.

* Sistemas de medida (por ejemplo, robots de medida, equipos de reconocimiento de objetos).

Los interfases de estos equipos varían desde los interfases analógicos no estandarizados a los digitales estandarizados con transferencia de datos para funciones, funciones de aplicación y servicio. Existen varios tipos de funciones de aplicación que tienen que ser adaptadas a la interfase del proceso. La funcionalidad del equipo y el canal de transmisión se determina por necesidades de campo como:

* Protección de explosión

* Longitud del medio, energía por el cable

* Calidad del cable

* Necesidades de los ordenadores host

* Filosofías de usuarios

Por esta razón existen numerosas necesidades en el tráfico de datos y la representación de objetos.

Equipos de control de procesos

La adaptación funcional de equipos de control de procesos a las tareas de automatización es de gran importancia. Tales equipos tienen las siguentes funciones:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* Control de proceso

* Configuración y parametrización (remota)

* Diagnóstico de red

* Comunicación con redes de alto nivel

Estas son funciones necesarias en el área de proceso. Los equipos son alimentados individualmente y normalmente instalados en cuartos específicos o cubiertas para protección contra factores ambientales agresivos.

3.4 COMUNICACION ABIERTA

El principio básico de **comunicación abierta** es permitir la comunicación entre sistemas de aplicación de vendedores específicos que no son compatibles normalmente con cualquier otro. Esto requiere un gran trabajo de estandarización que concierne solamente a la comunicación, por ejemplo, la forma en que los mensajes son intercambiados.

Esta heterogeneidad de los sistemas beneficia al usuario, que no está así limitado a un único tipo de sistemas para sus distintas aplicaciones. Tiene la ventaja de poder adaptarse a condiciones de aplicación y al presupuesto disponible, pero por otro lado, tal heterogeidad dificulta considerablemente la interconexión de equipos de distintos fabricantes.

El significado actual (semántica) de la transferencia de datos depende del tipo de aplicación. Se representa en equipos por procesos de aplicación que usan el sistema de bus para comunicación.

La comunicación abierta entre un proceso de aplicación y otro necesita la definición

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

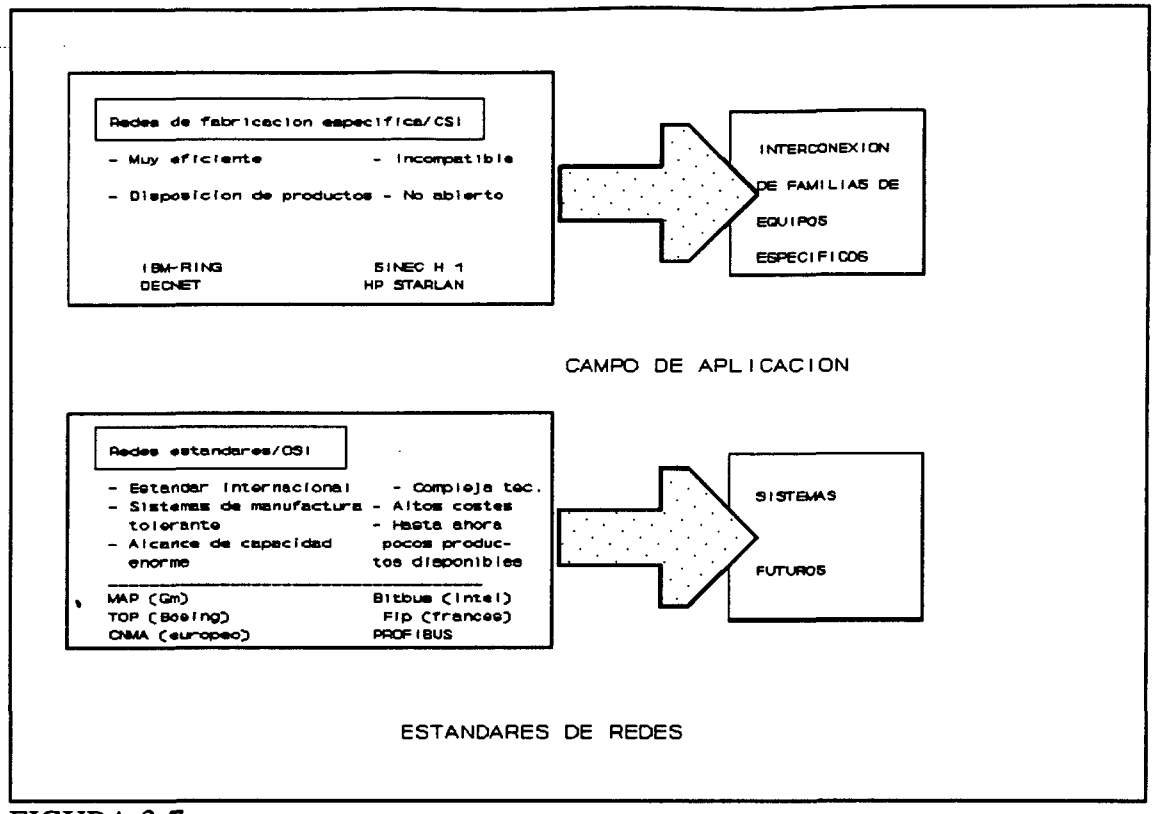


FIGURA 3.7

de tareas de aplicación. Por ejemplo, un equipo tiene acceso a los objetos locales de otro equipo por medio de ciertos servicios.

Los sistemas de comunicación existentes (figura 3.7) pueden ser clasificados en dos arquitecturas. La arquitectura CSI (Closed Systems Interconnection) es una red local en la que todos los componentes vienen y se diseñan por el mismo vendedor. El usuario se obliga a comprar soluciones completas de un vendedor y a adherirse al mismo vendedor para todas las extensiones siguientes del sistema para garantizar la compatibilidad. Si uno quiere conectar equipos de diferentes vendedores, las soluciones especiales, que son caras y complicadas, deben ser adaptadas al resto del sistema. Es decir, los grandes fabricantes dieron soluciones para la interconexión de sus equipos, pero no con los equipos de otros fabricantes.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Por esto es por lo que los componentes de la arquitectura OSI (Open Systems Interconnection) intentan desarrollar soluciones independientes del vendedor. Por definición una especificación común será capaz de integrar equipos de diferentes vendedores en un sistema multivendedores. Estos sistemas son diseñados para una aplicación específica (por ejemplo, para tecnología de control de producción) y se componen de equipos de diferentes vendedores. La interconexión de redes hace a su vez más difícil el problema, ya que puede haber diferentes redes con distintos servicios de transmisión, que requieran diferentes interfases.

3.4.1 EL MODELO ISO/OSI

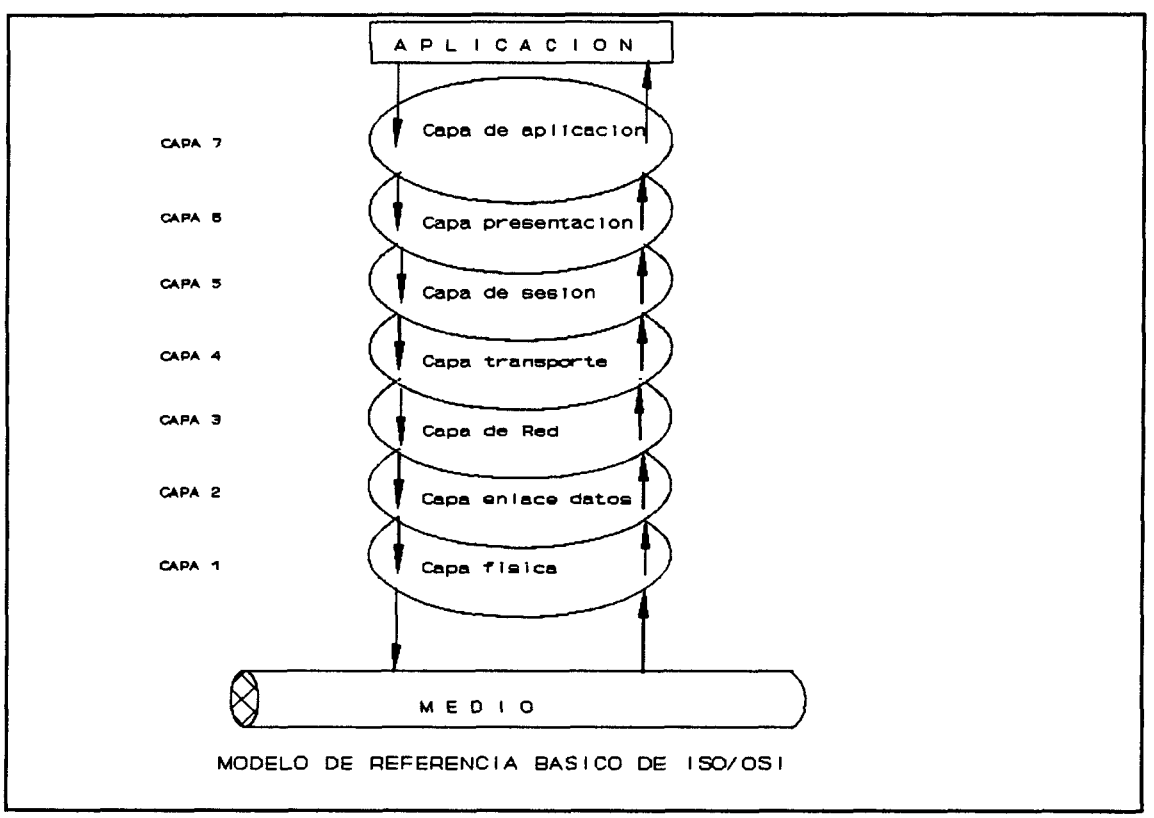


FIGURA 3.8

Los sistemas de comunicación se diseñan según la organización internacional para la estandarización (ISO) /modelo de referencia OSI que estructura la funcionalidad de la

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

comunicación en siete capas. Cada capa se proyecta sobre la totalidad del sistema de comunicación y determina un protocolo específico de cada capa que está en concordancia con los equipos del sistema. Los servicios se proporcionan en la interfase con el siguiente nivel superior. Con estos servicios el usuario tiene un sistema de comunicación transparente que permite simplificar y hacer más eficiente la transferencia de datos entre estaciones. Un sistema de comunicación transparente significa que los mensajes transmitidos sobre este sistema no son distorsionados ni modificados.

El proceso de comunicación se divide en jerarquías de siete capas atravesadas una a una por el mensaje de arriba a abajo, o viceversa, durante la comunicación. Toda capa proporciona servicios a la capa inmediatamente superior y devuelve los servicios usados de la capa inmediatamente inferior para ejecutar sus tareas. Así, en una estación, cada capa intercambia información solamente con sus capas adyacentes (la superior y la inferior).

Vamos a entrar a la descripción de las diferentes capas:

Capa 1: Capa física. La capa física determina los parámetros eléctricos, mecánicos, funcionales y de proceso de la conexión física. Su función es la del mantenimiento de la interfase física (por ejemplo, la transferencia de un conjunto de bits en "bruto", stream). En esta capa se determina la forma física de las señales lógicas, qué "pins" se usan para cada función,...

Las funciones de esta capa son:

- Activación y desactivación de conexiones físicas.

- Transmisión de unidades de datos del nivel físico.

- Gestión del nivel físico.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Un bien conocido ejemplo de especificación de la capa física es la interfase V.24. PROFIBUS usa la interfase RS 485 (RS: Recommended Standard).

Capa 2: Capa de enlace de datos. La capa de enlace de datos proporciona significado funcional y de procedimiento a la estabilización, mantenimiento e interrupción del enlace de datos entre varias estaciones. El conjunto de bits o stream se completa con información adicional. Para transferir datos, se forman tramas que, aparte del mensaje actual, contiene datos del transmisor y receptor así como información de control.

Las funciones son:

- Establecimiento y liberación de las conexiones de enlace de datos
- Mecanismo de correspondencia de las unidades de datos
- Delimitación y sincronismo
- Control de secuencia
- Detección de errores
- Recuperación de errores
- Control de flujo
- Intercambio de identificadores y parámetros
- Control de interconexión del circuito de datos
- Gestión del nivel de enlace

Además, el protocolo de la capa de enlace de datos debe detectar la pérdida de paquetes de datos y reaccionar según lo sucedido.

Ejemplos son: HDLC, LAP-B y ADCCP .

Capa 3: Capa de red. La capa de red hace el encaminamiento (routing), la selección de un camino a través de una red de nodos en los cuales tiene lugar la transferencia de datos. Estos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

caminos no deben nunca sobrecargarse. Por lo tanto se debe encontrar el camino más eficiente.

Otra importante tarea de la capa de red es controlar el flujo de datos en la red. En las subredes los datos son almacenados en bufferes al final de un subcamino, y solamente después de que el mensaje en su totalidad ha sido recibido, pueden entrar al camino de la siguiente subred. Estos buffers, sin embargo, no deben sobrecargarse. Tales problemas surgen en redes de área local (LANs) cuando varios de ellos se conectan a gateways.

Las principales funciones son:

- Encaminamiento y retransmisión
- Conexiones de red
- Multiplexación de las conexiones de red
- Segmentación y bloqueo
- Detección y recuperación de errores
- Secuenciamiento
- Control de flujo
- Transferencia de datos expeditos
- Reinicio (Reset)
- Selección del servicio
- Gestión del nivel de red

Un ejemplo típico del protocolo de capa de red es el CCITT X.25 (Datex-P) que se usa en España y en servicios del correo alemán para intercambio de paquetes.

Capa 4: Capa de transporte. La capa de transporte ofrece un servicio de transporte seguro a la capa inmediatamente superior para proteger detalles de la transferencia de datos de ésta. Establece uno o más canales lógicos que dirige por los servicios de las tres capas inferiores.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

La capa de transporte ejecuta el control de error desde el final del sistema al final del sistema. Si es necesario, divide el mensaje en pequeños paquetes y recibe las solicitudes de otras estaciones cuando los paquetes son erróneos o perdidos.

Esta capa reordena los paquetes cuando llegan desordenados y automáticamente recupera de errores. La capa de transporte completa la parte de orientación el transporte de la comunicación. Los protocolos desde las capas uno a la cuarta son llamados protocolos orientados a red, mientras que de la cinco a la séptima se llaman protocolos orientados a aplicación (niveles altos).

Las funciones generales son:

- Ofrece los servicios de control final-final
- Hace que la red tenga la calidad solicitada por los niveles superiores
- Recuperación de errores final-final
- Multiplexación
- Control de flujo
- Gestión del nivel de transporte

Ejemplos pueden ser : DOD y TCP.

Capa 5: Capa de sesión. La capa de sesión es la más baja de las orientadas a aplicación. Una sesión puede ser usada por ejemplo para entroncar en un sistema externo o para establecer la transferencia entre sistemas externos. La capa de sesión trabaja muy cerca del sistema de operación del host y ofrece los siguientes servicios:

- Iniciación y liberalización de la sesión.
- Control de operaciones durante la sesión.
- Control de flujo de datos.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

- Control de diálogo.
- Gestión del nivel de sesión.

Capa 6: Capa de presentación. La tarea de la capa de presentación es hacer disponible los servicios de la capa de aplicación, que interpreta el significado de los datos intercambiados. Los datos transmitidos se modifican de forma que los procesos de aplicación envueltos en la comunicación puedan entenderlos.

La capa de presentación es particularmente importante para la protección de datos de accesos no autorizados. Los procedimientos adecuados de codificación y descodificación se pueden implementar en esta capa.

Sus funciones:

- Solicitud del establecimiento de la sesión
- Negociación y renegociación de la sintaxis de presentación
- Formateado y transformación de datos
- Transformaciones de propósito general
- Solicitud de terminación de la sesión
- Direccionamiento y multiplexación
- Gestión del nivel de presentación

Capa 7: Capa de aplicación. Como la capa más superior del modelo OSI de referencia, la capa de aplicación no proporciona servicio explícito, sino servicios específicos de aplicación al usuario. Muchas veces el usuario (por ejemplo, un proceso de aplicación) se representa por un ejemplo de la capa de aplicación.

Esto significa que la capa de aplicación no ejecuta sus propios servicios. Se ejecutan por un proceso de aplicación correspondiente que se encuentra fuera de la capa de aplicación.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

La tarea de la capa de aplicación es meramente proporcionar protocolos específicos de aplicación. Tales ejemplos en la tecnología de la comunicación son llamados ejemplos de aplicación. Denotan unidades lógicas (como programas de aplicación o usuarios) que comunican cada una con la otra. Así, tenemos diferentes clases de proceso de aplicación:

- De gestión del sistema
- De gestión de aplicación
- De usuario

Servicios que proporciona:

- Transferencia de ficheros (FTAM y FTP)
- Terminal Virtual (VTP)
- Transferencia y manipulación de tareas

Funciones generales del modelo de referencia OSI. Hablando con generalidad, el modelo de referencia OSI muestra cómo puede ser conseguida la comunicación entre sistemas físicamente conectados de diferentes vendedores.

Asumimos un caso general de comunicación entre sistemas con las siguientes características:

* El enlace físico entre sistemas puede incluir componentes de diferentes redes y calidades.

* Los errores en transferencia de datos como deficiencias en la red (tasas de error por ejemplo) se pueden corregir.

* La información se puede transmitir de forma orientada a diálogo.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* La comunicación es completamente independiente de la representación de la información inherente al sistema.

* Los esquemas de protección de datos son posibles.

Un bus de campo para aplicaciones de automatización específicas no necesita todas estas características, sólo las importantes.

3.4.2 CONFORMIDAD DE LOS PROTOCOLOS ABIERTOS

Los protocolos abiertos son prerequisitos para la comunicación entre equipos de diferentes vendedores. La base para los protocolos "abiertos" es la publicación de la especificación de servicio y protocolo para que cualquier vendedor pueda implementar su propio equipo de comunicación. Si la especificación se conserva estrictamente, la comunicación libre de error entre sistemas abiertos sería teóricamente posible. En la práctica, no obstante, las especificaciones de protocolo son incompletas o ambiguas, y permiten diferentes interpretaciones, resultando distintas implementaciones.

Si una implementación cumple con una especificación dada, se habla de conformidad. Esta conformidad es la calidad de un producto dado que debe ser probado en un test de conformidad por una institución autorizada. La prueba de conformidad es el único camino para evitar implementaciones incompatibles de estándares de protocolos, ya que el certificado de una prueba de conformidad es un sello de calidad y contribuye a asegurar la calidad de los componentes de automatización.

El concepto de prueba. Los objetivos de la prueba del test de conformidad son implementaciones de una o varias capas de protocolo OSI adyacentes. Un objetivo del test se llama test de implementación (IUT, implementation under test), el sistema en el que IUT se integra test de sistema (SUT).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Para el test de conformidad, la IUT solamente puede ser accedida por sus interfases a la capa inmediatamente superior o inferior. Mientras los modelos de control se pasan a otra interfase al final, en la interfase opuesta, las reacciones correspondientes se observan y se registran. Estas funciones de prueba de operación y observación de las capas inmediatamente superiores e inferiores, se llaman prueba descendente y prueba ascendente respectivamente.

La calidad de la prueba de conformidad depende de las varias calidades de casos de pruebas seleccionadas y el grado obtenido de cobertura de la especificación de protocolo probado. Ya que la consideración de todos los probables casos no es posible en la práctica, los test exitosos dependen de la elección de las grandes posibilidades. Por lo tanto un test sólo puede probar la existencia de ciertas facultades pero nunca puede probar que un componente está libre de error.

3.4.3 INTEROPERABILIDAD DE PROTOCOLOS ABIERTOS

La prueba de conformidad solamente se encarga de comprobar una implementación seleccionada según la especificación. La prueba de interoperabilidad comprueba la interoperabilidad de varias implementaciones, ya probadas en conformidad, para encontrar incompatibilidades, que son debidas a una elección incorrecta de parámetros de configuración en el sistema de configuración, o al papel jugado por la semántica del servicio en interoperabilidad. Como la prueba de semántica no es objetivo de la prueba de conformidad, la prueba de interoperabilidad es, especialmente para usuarios en comunicación, un chequeo adicional.

La interoperabilidad se prueba por un sistema de referencia, llamado sistema multi-vendedor. La implementación bajo prueba debe ser capaz de comunicar con los equipos del sistema multi-vendedor.

3.4.4 INTERCAMBIABILIDAD DE EQUIPOS DE PROTOCOLO ABIERTO

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

La intercambiabilidad es una nueva palabra clave en la tecnología de la automatización que designa la intercambiabilidad de equipos de diferentes vendedores. Un prerequisito, no obstante, es que los equipos tengan la misma función de aplicación y características. La intercambiabilidad es una característica que necesita la clasificación de objetos en grupos que pueden ser reemplazados por otro con respecto a ciertas clases de características. Para conseguirlo, deben definirse los perfiles ofreciendo al usuario criterios para la elección de los equipos.

3.4.5 ESTANDARES, ESTANDARES COMERCIALES, ESPECIFICACIONES Y PERFILES

Vamos a definir algunos términos que se usan frecuentemente en PROFIBUS.

Especificación. Una descripción formal que un sistema (de comunicación) se supone que hace se llama especificación. Si esta especificación es obligatoria para un grupo de usuarios (por ejemplo industria, ciudad), recibe el nombre de estándar, o especificación estándar. Por lo tanto un estándar es una definición obligatoria de características para un producto específico. Un estándar comercial es una especificación establecida sobre un grupo de usuarios a través del uso difundido y observado por todos los miembros de este grupo.

El objetivo principal del proyecto bus de campo (Fieldbus, fundado por el Ministerio de Alemania Federal de la investigación y tecnología) fue crear una especificación para la capa de aplicación de PROFIBUS, lo que más tarde llegó a ser el estándar alemán.

En el desarrollo de la especificación, se distinguen los niveles siguientes:

* Modelo de arquitectura. Por la descripción de sistemas abiertos, se definen los tipos de objeto y se determinan las relaciones entre los diferentes componentes. Además, en este nivel se sientan las direcciones generales para la comunicación entre estos tipos de objetos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

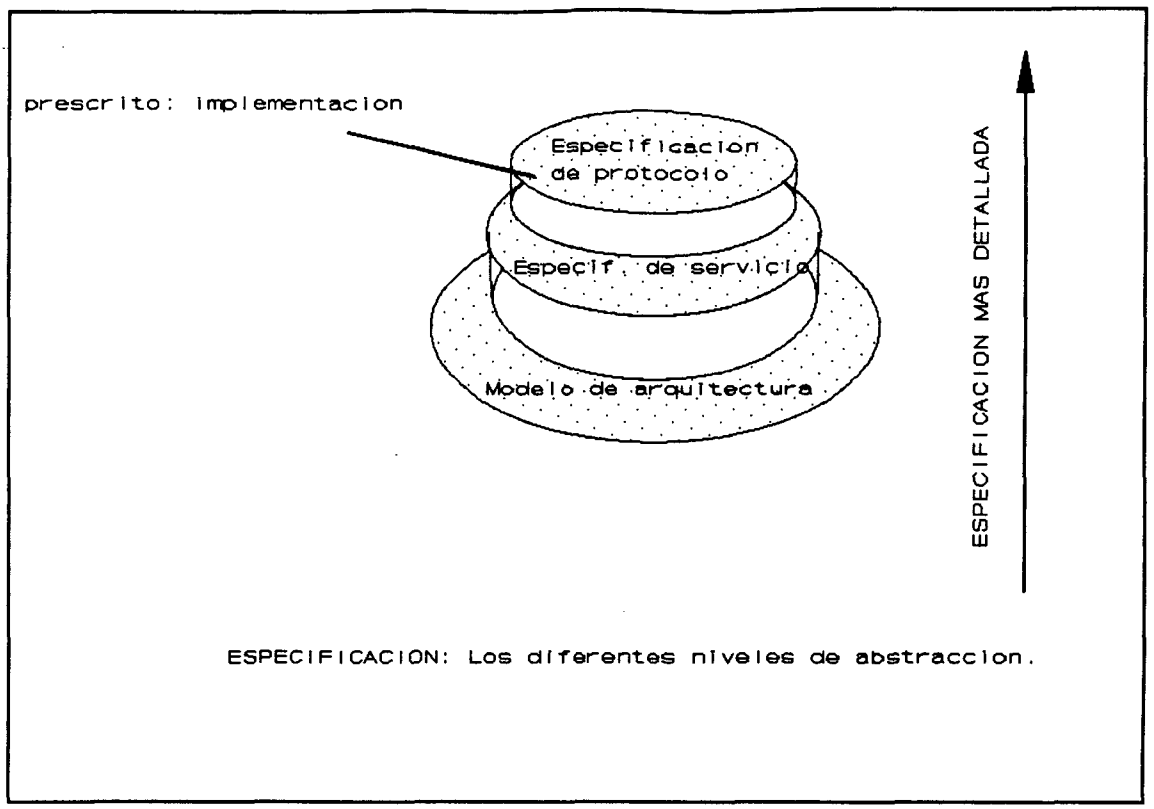


FIGURA 3.9

y se determina la estructura del modelo de capa lógico.

* Especificación de servicio. Define los servicios a ser proporcionados por cada capa y la interfase abstracta para el uso de estos servicios.

* Especificación de protocolo. Define los mecanismos en una capa, por la que se proporciona un servicio pedido.

Las implementaciones hardware y software no se ven afectadas por estos convenios. En la figura se muestran los diferentes niveles de abstracción de la especificación.

Objetivo de la especificación estándar de PROFIBUS. Una especificación estándar es una representación lógica abstracta de modelos de comunicación, sus servicios y protocolos.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Solamente especifica las condiciones de interfase física, la sintaxis, las semánticas y aspecto del protocolo de los mensajes para que puedan ser accedidos por las estaciones conectadas al interfase de bus. La especificación estándar es obligatoria para todos los vendedores y usuarios de PROFIBUS. PROFIBUS, no obstante, es solamente un estándar básico que permite, o incluso requiere, la selección de parámetros para optimizar las comunicaciones. En PROFIBUS, por ejemplo, se pueden implementar los diferentes rangos de baudios, pero en curso solamente tiene sentido un valor de transmisión común en un segmento de bus específico.

El estándar de bus de campo PROFIBUS, incluye las siguientes especificaciones:

* Modelo de arquitectura. Incluye las capas de bus de campo y el equipo virtual de campo (VFD). Un VFD es la perspectiva abstracta de un equipo de campo real en el equipo pareja de campo.

* Especificación de servicio y protocolo. Aquí los servicios y protocolos se describen separadamente para cada capa.

* Tecnología de transmisión. Especifica las propiedades eléctricas, la tecnología de interfase y técnicas de transmisión.

Perfiles PROFIBUS. Para determinar la aplicación del estándar PROFIBUS para las industrias, equipos o áreas de aplicación, se deben considerar las interfases técnicas, protocolos y servicios para las capas OSI 1, 2 y 7 definidas en el estándar. La elección y determinación de características opcionales, que se prescriben en el estándar básico, dan como resultado los perfiles PROFIBUS. Para que un proceso de aplicación consiga interoperabilidad en un área de aplicación de una industria dada, los objetos y funciones dinámicas se deben definir para permitir la interoperabilidad entre equipos. La completa interoperabilidad de equipos similares de diferentes vendedores necesita más información y

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

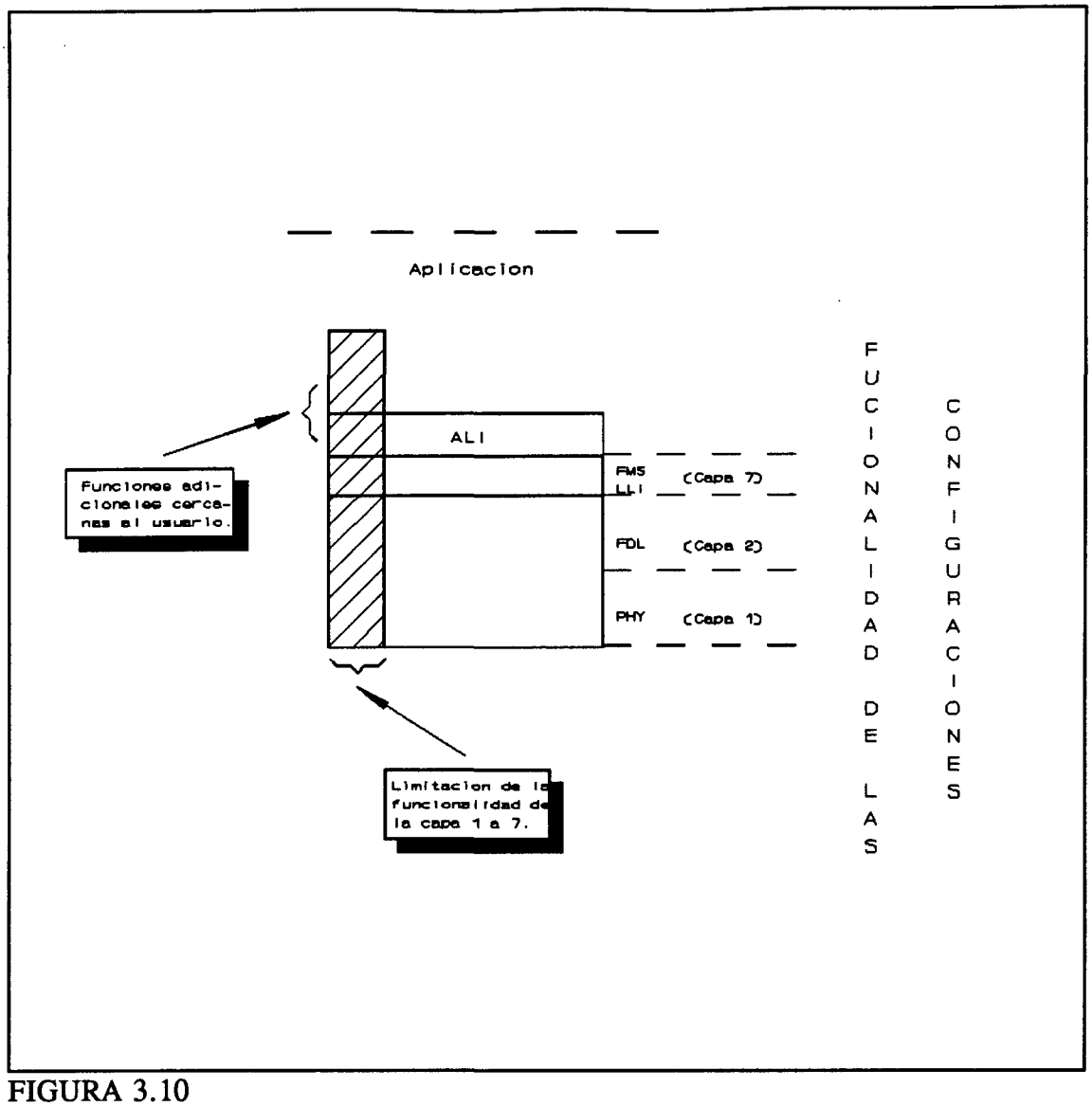


FIGURA 3.10

una definición de conducta dinámica de los equipos.

Solamente cuando se ha establecido un perfil es un estándar básico completo y sus aplicaciones llegan a ser claras. Por la clasificación de equipos similares en grupos, se obtiene una clase de datos, por los que los equipos puedan ser fácilmente comparados y echa la correcta elección.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

El establecimiento de un perfil es un procedimiento muy complejo que se puede llevar a cabo desde diferentes puntos de vista. Las industrias con practicas más antiguas, como automatización de obras o industria textil, pueden determinar perfiles amplios que pueden llegar a ser después las bases para sistemas de comunicación íntegros.

Por otro lado, los vendedores de componentes de automatización pueden ofrecer sus productos con equipamiento estándar PROFIBUS, lo que sería suficiente para la mayoría de las aplicaciones. La estandarización de los perfiles es una de las tareas principales en la tecnología de la automatización para ser efectuada en los 90.

La estandarización de los perfiles PROFIBUS es dirigida por El Grupo de Usuarios PROFIBUS bajo cooperación de los expertos de las industrias implicadas.

En las siguientes áreas de aplicación, los perfiles que se han publicado y trabajado en [PNO 91a] son:

* Tecnología de sensores y actuadores.

* Tecnología de energía.

* Controladores.

* Automatización de obras.

* Ingeniería de maquinaria textil.

Como un ejemplo, consideraremos un perfil de tecnología para sensores/actuadores. Ya que este campo es relativamente complejo, los perfiles se dividen en una lista general y en varias de datos de función específica. La página de datos general incluye todas las

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

especificaciones generales de las funciones de los equipos y la capacidad de comunicación. Las páginas de datos de función indican todas las funciones específicas de un equipo. El perfil de un equipo se arregla con las especificaciones incluidas en la hoja de datos generales así como esas en las respectivas hojas de datos de funciones específicas.

Ya existen las hojas de datos de funciones específicas para las siguientes funciones:

* Salidas analógicas

* Entradas analógicas

* Salidas binarias

* Entradas binarias

Cada hoja de datos de funciones específicas consta básicamente de dos partes: una descripción funcional desde el punto de vista del usuario y una descripción funcional desde el punto de vista de la comunicación.

Ambas descripciones tienen contenidos idénticos pero son propuestas para diferentes usuarios: el último usuario, y los implementadores y diseñadores de red, respectivamente.

No todos los sensores y actuadores deben soportar la transmisión de todos los datos. Antes bien, en los perfiles, los tipos de equipos se clasifican en cuatro grupos en concordancia con sus funciones. Los sensores del grupo más bajo (grupo 1) transmiten solamente el valor medido, no diferenciándose mucho de un equipo clásico con interfase de 20 Ma, excepto en la capacidad del bus. Los equipos de los grupos superiores proporcionan funciones de inteligencia distribuida que permiten transmisión de mensajes de error, datos de inicialización, valores límites y valores medidos en un formato conveniente (por ejemplo

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

coma flotante).

Los grupos de equipos son compatibles hacia arriba lo que significa que los equipos de un grupo superior deben incluir todas las características de los equipos del grupo inferior del mismo tipo. La capacidad de comunicación de los controladores con los equipos de campo se describe en el perfil de comunicación lo que es parte del conjunto de perfiles de los sensores-actuadores. Para un equipo pasivo (por ejemplo, un sensor o un actuador), las funciones del perfil de comunicación delimitan la funcionalidad máxima vista por las funciones contenidas en la página de datos de funciones específicas. Esto permite la comunicación con los controladores correspondientes.

Para un equipo activo (PC o PLC), por otro lado, el perfil de comunicación es la funcionalidad mínima. Los controladores que tienen esto son capaces de comunicar con cualquier sensor o actuador.

Un perfil de comunicación puede ser considerado una máscara, a través de la cual se obtiene un punto de vista de un controlador en las funciones de sensor o actuador.

Se espera que PROFIBUS llegue a ser incluso más amistoso para el usuario, encontrando una aceptación difundida en aplicaciones de sensores y actuadores cuando los correspondientes perfiles se publiquen.

Especificación del software. La realización de una especificación estándar se hace en dos pasos : primero, se fija la especificación del software que puede ser dividido en un diseño y una especificación revisada. En segundo lugar, se implementa el software para un sistema real. Por diseño, la especificación del software para un aplicación necesita el rango de características ofrecidas por el estándar al que se restringe y un software de comunicación orientado a aplicación que puede ser optimizado. Por ejemplo, en la implementación de un equipo, pueden omitirse los grupos de servicio totalmente si no son necesarios en la

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

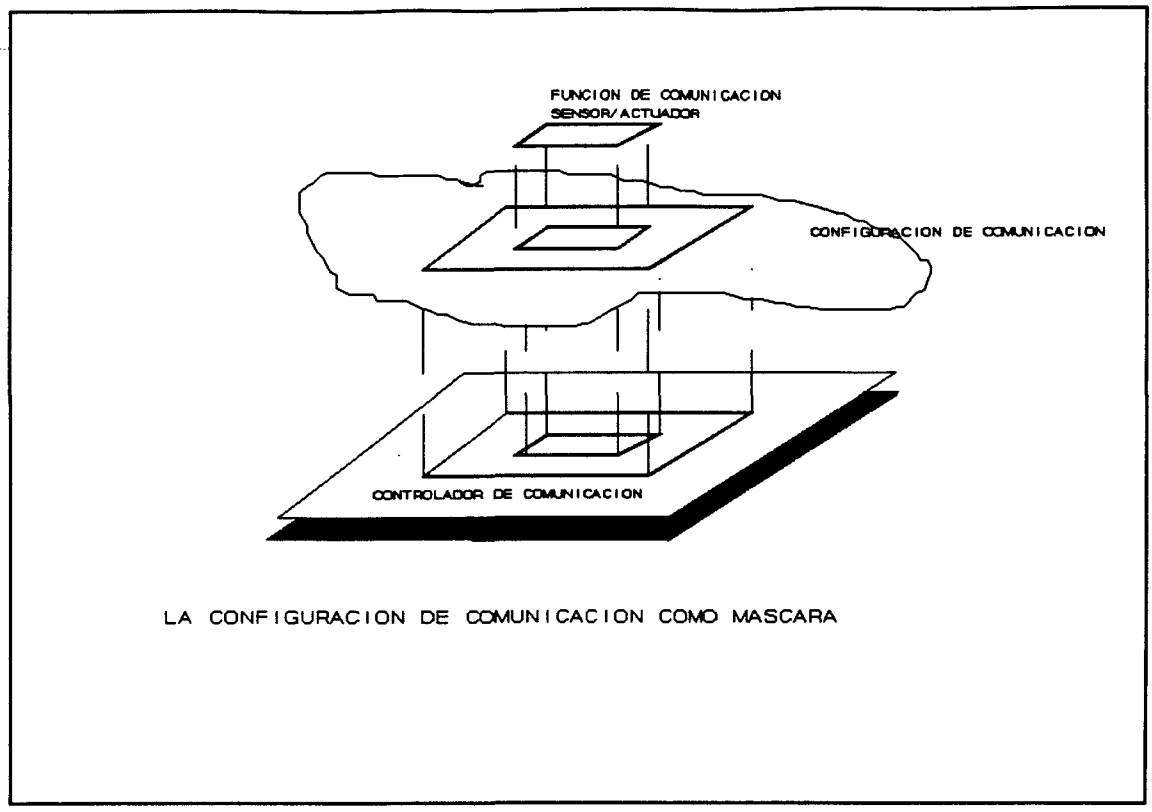


FIGURA 3.11

aplicación planeada. La implementación se realiza normalmente para una familia de procesadores específicos en un lenguaje de programación específico.

3.5 SOLUCIONES EXISTENTES DE RED

Las estaciones de trabajo y Pcs se deben conectar por conexiones ordenador a ordenador en el nivel de manejo, las cuales son soportadas por sistemas de comunicación que eficientemente transfieren un gran volumen de datos (por ejemplo, los contenidos de las bases de datos). Estas transferencias se basan normalmente en la difundida red Ethernet y sus altos protocolos, o en redes de vendedores específicos como DECnet (DEC= Digital Equipment Corporation) o SNA (System Network Architecture por IBM). El aumento de integración de soluciones de PC da como resultado un número en aumento de redes de PC. Las soluciones para la integración de los ordenadores heterogéneos con servicio de archivo integrado han

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

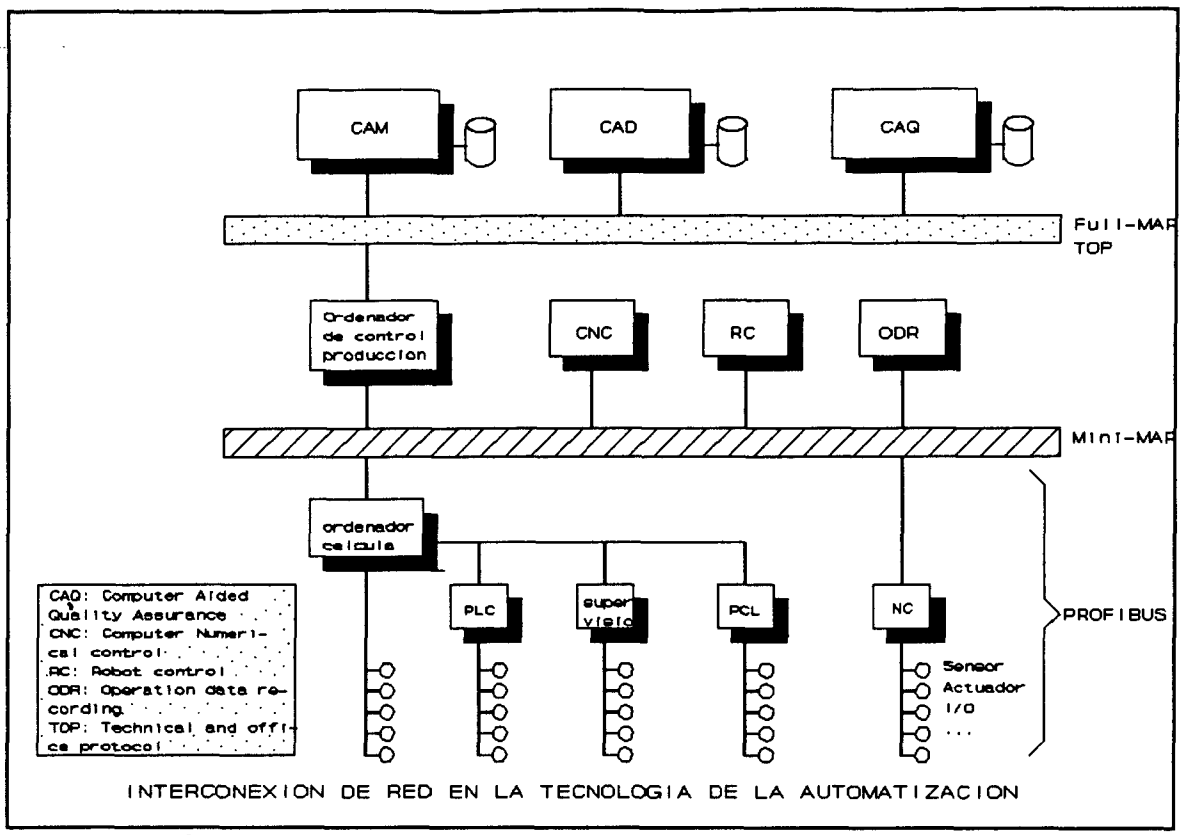


FIGURA 3.12

sido ya desarrolladas, tales como Novell o LocalTalk (usada por Apple Macintosh). Estas soluciones tienen una característica en común: son diseñadas para ofrecer comunicación. Debido a la normalmente falta de interfases comunes, son limitadas las posibilidades de transferencia de datos en el área de la automatización industrial.

Estas circunstancias permitieron mayor manufacturas cerca de los 80 para desarrollar los dos protocolos estándar TOP (Technical and Office Protocol) para la integración de la red en oficinas y MAP para la automatización de fábricas. MAP cubre los niveles intermedios de información sin operaciones de tiempo crítico.

MAP es un estándar de comunicación difundido para el medio de producción basado en estándares de ISO. Además de los elementos de servicio de aplicación común (CASE), MAP ofrece servicios de comunicación específicos para controles de proceso, a los que se

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

les llama elementos de servicio de aplicación específica (SASE). SASE incluye transferencia de ficheros, acceso y manejo (FTAM), además de otras cosas. Más aún, con la estandarización de MMS (Manufacturing Message Specification), MAP proporciona servicios específicos y protocolos para equipos de automatización, tales como controladores de lógica programable, equipos de control numérico y robots.

Las características de MMS son:

* Significado único y consistente (semántica) de objetos (por ejemplo, sucesos, variables, programas,...)

* Independencia de equipos

* Disponibles aproximadamente 80 servicios para el usuario

* Gran flexibilidad conseguida por la posibilidad de especificaciones adicionales definidas para cierta clase de equipos (Estándares compañeros)

* Definición consistente de sintaxis por significado de las reglas de codificación ASN.1 para mensajes (ASN=Abstract Syntax Notation)

* Normalmente transferencia de mensajes orientados a conexión, lo cual significa que antes de ser un mensaje transmitido a una estación par, se debe establecer una conexión (canal lógico) con la entidad par.

* Después del establecimiento de una conexión, el protocolo MMS trabaja en un modelo orientado a la transacción. La transferencia de ficheros por partición de segmentos se llama transacción.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Las industrias de procesamiento demandan tiempos de respuesta más rápidos y soluciones menos caras para el nivel inferior de automatización de procesos que las de MAP. Estas demandas se pueden satisfacer reduciendo la funcionalidad y conveniencia del protocolo, lo que quiere decir que no es realizable la extensión completa del modelo de capas de ISO. Una de estas arquitecturas es la Mini-MAP cuya capa de aplicación se basa en la capa de enlace de datos(FIGURA 3.13). PROFIBUS toma una aproximación similar, que se explicará más adelante.

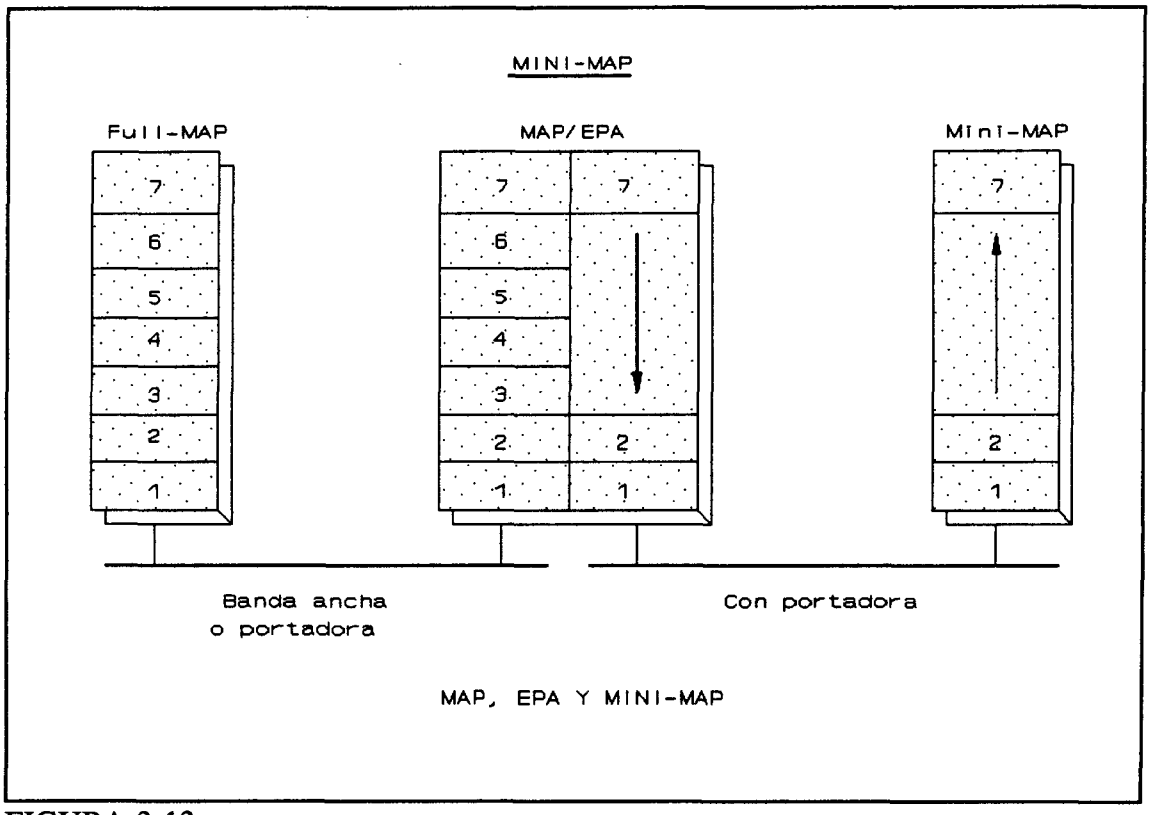


FIGURA 3.13

La principal diferencia entre Mini-MAP y Full-MAP no yace solamente en las distintas funcionalidades y realización de las capas OSI, sino también en la conexión al medio. Mini-MAP es disponible solamente en transmisiones de bandas de portadoras y proporciona una funcionalidad limitada a comparación de Full-MAP. Además, permite solamente una longitud de mensaje limitado y no proporciona control de flujo. Un puente

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

(gateway) MAP/EPA (EPA = Enhanced Performance Architecture) conecta Mini-MAP y Full-MAP.

3.6 MANEJO DE RED

El manejo de red incluye el conjunto de todas las actividades que notan o influencian el uso de recursos de sistema. En sistemas abiertos, los recursos son esencialmente capacidad de almacenamiento, prestación del procesador, y recursos[1] como las conexiones. Las subtareas de manejo incluyen planificación (adopción de objetivos generales, análisis de problemas y realización de decisión), control y vigilancia (definición de subobjetivos, asignación de actividades, control de objetivos que han sido alcanzados).

Las metas del manejo de red son:

* Soporte de la planificación e instalación de una red.
* Seguridad de que las operaciones son de confianza.
* Simplificación y automatización de la depuración.
* "sintonización" de software de comunicación.
* Apuntar las modificaciones estructurales necesarias y sus implicaciones.
* Extensión o modificación del hardware o software de red.

Otros objetivos pueden ser:

* La disposición de una aproximación simple, de ocultar todas las características de la red y la información interna de el usuario que no sea una aplicación relevante.

* La capacidad de adaptación autónoma para cambiar condiciones de operabilidad.

[1] Nos centramos primeramente en el sistema de comunicación; así, por recursos queremos decir recursos relevantes para la comunicación.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Además, existe el término administración que denota un subconjunto de manejo de red. Administración se refiere a las actividades de período corto o medio (tales como recuperación de error o actual facturación), mientras manejo de red es el período más amplio.

Objetivos del manejo de red en comunicación industrial. Para comunicación industrial, los objetivos arriba mencionados siguen siendo verdaderos. Pero se debe destacar aquí, que las prioridades para alguna de estas tareas en el área de bus de campo son diferentes de, por ejemplo, la integración de red de tramas.

Los requisitos principales para el manejo de red en el área de bus de campo (y por lo tanto también para PROFIBUS) son:

* Planificación de Ayuda de una red. Las redes para comunicación industrial necesitan ser planeadas en detalle. Particularmente para aplicaciones complejas, se necesitarán herramientas de planificación y diseño de sistema apropiado.

* Seguridad en los fallos. En la producción industrial, un sistema de comunicación falla provocando que sea muy caro los tiempos de baja de la máquina. Pero además por causa de los estrictos requisitos de seguridad, un fallo no puede ser tolerado. Proporcionar funciones de control apropiadas es la tarea del manejo de red, lo que incluye el control de la operación, detección de operaciones de error y uso de pruebas y equipos de diagnósticos.

Estandarización en el manejo de red. La estandarización del manejo de red es un direccionamiento tópico por todos los cuerpos pricipales de estandarización (ISO, ECMA, IEEE). El objeto de todos los esfuerzos de estandarización es crear las condiciones necesarias para un manejo de la red abierta, y la funcionalidad del manejo de red debe ser estructurado apropiadamente (por ejemplo, la arquitectura ISO).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

EL manejo de la red PROFIBUS. El estándar PROFIBUS también juega un importante papel con respecto al manejo de la red abierta. Este estándar especifica, el "manejo" funcional de la capa 7 de un equipo PROFIBUS para participar en el manejo de una red abierta.

3.7 COMUNICACION INDUSTRIAL CON BUSES DE CAMPO

En los niveles más bajos de automatización de un sistema de comunicación, que incluyen equipos terminales como sensores y actuadores, y en la interfase con controladores programables, las soluciones MAP/TOP son demasiado caras y/o no alcanzan el tiempo corto de respuesta requerido (no exceder de unos pocos ms), dependiendo de la aplicación.

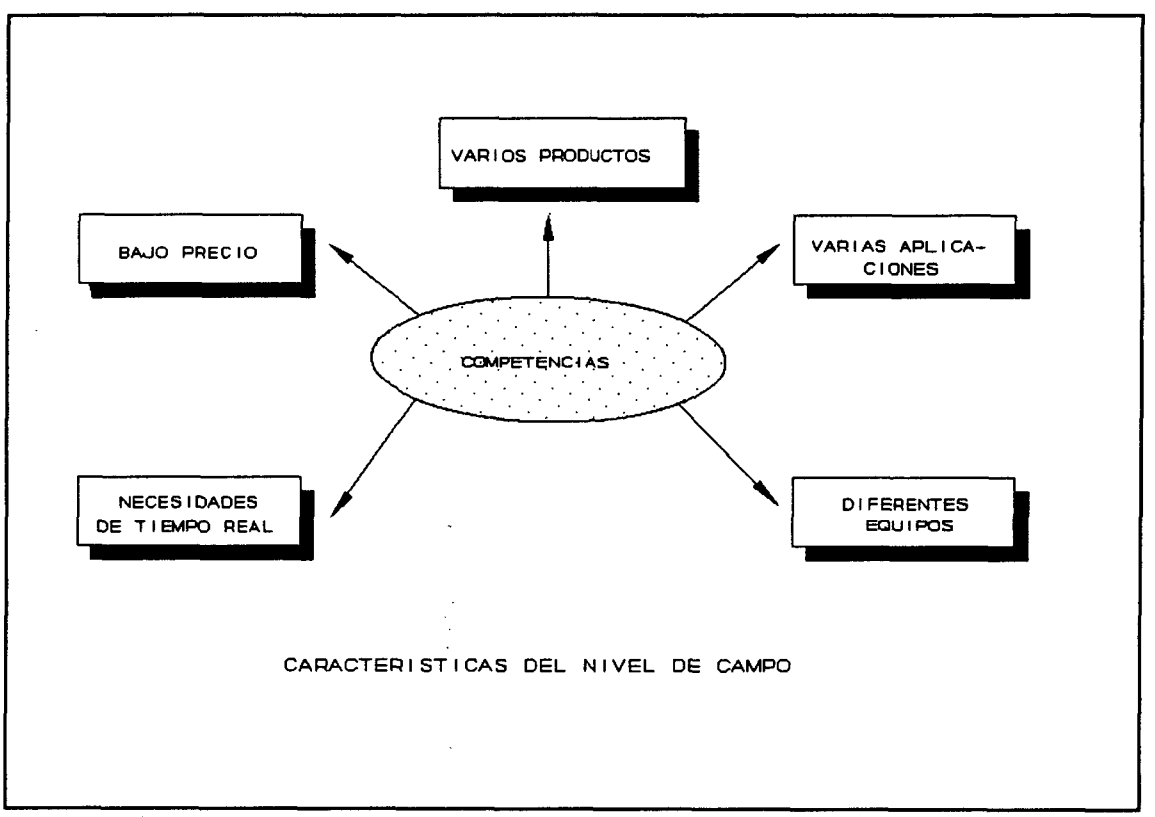


FIGURA 3.14

La planificación, instalación y mantenimiento de la red en el nivel de campo debe no

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

ser cara y técnicamente posible. Un nuevo estándar de bus de campo debe ofrecer un buen coste/tasa de prestación para la aplicación. Los requisitos de aplicación para un estándar de campo incluyen:

* sensores y actuadores cada vez más inteligentes, las soluciones frecuentemente hasta ahora (como la conexión sobre entradas/salidas digitales de V.24 o interfase de 20mA) no son suficientes. Se requiere un interfase de bus de campo digital.

* debe ser posible la conexión de equipos de medida y actualizadores con largas líneas y bajas tasas de datos (instalaciones de campo).

* debe ser posible la conexión directa de pequeños controladores programables con pequeñas líneas y gran cantidad de datos.

* normalmente, deben ser posibles los tiempos muy pequeños de respuesta y mensajes cortos.

* se espera que la seguridad de los datos sea la mayor posible.

* si es necesario se debe usar los equipos con seguridad intrínseca.

* la conexión de componentes "relativamente mudos", como oposición a más procesadores inteligentes, requiere lo específico, más medidas caras.

* los componentes interconectados no pueden ser frecuentemente controlados localmente. Son tan simples que no tienen una conexión de terminal o están localizados en áreas inaccesibles (por ejemplo, en una caja bajo el agua).

* un gran número de componentes interconectados (por ejemplo, sensores) se producen en masa y solamente se individualizan con la integración en la red.

* las demandas del usuario para interfases abiertos llega a ser cada vez más altas.

Servicio y control de un bus de campo. Estas redes se instalan y se sirven en las áreas de proceso y campo, que tienen ciertos requerimientos de aplicación, como:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* por razones de coste, la mayoría de las operaciones de manejo de red deben ser ejecutadas por una estación dedicada.

* el manejo de red debe ser posible sin ninguna herramienta de configuración.

* la operación remota y diagnósticos remotos facilitan el trabajo de servicio del personal. Si es necesaria la intervención en obra, el error se diagnostica remotamente y las partes de repuesto apropiadas se traen directamente al sitio.

* los buses de campo se integran normalmente en redes de alto nivel (backbone, de soporte). Esto requiere un concepto de manejo consistente.

* el manejo de red sería hecho como mínimo hasta cierto punto por el personal de servicio existente, lo que evita la necesidad de emplear expertos especiales en comunicación de datos.

* los mecanismos deben ser instalados para control continuo de componentes "que no hacen nada" en operaciones normales (por ejemplo, equipos para chequeo de condiciones fuera de límite).

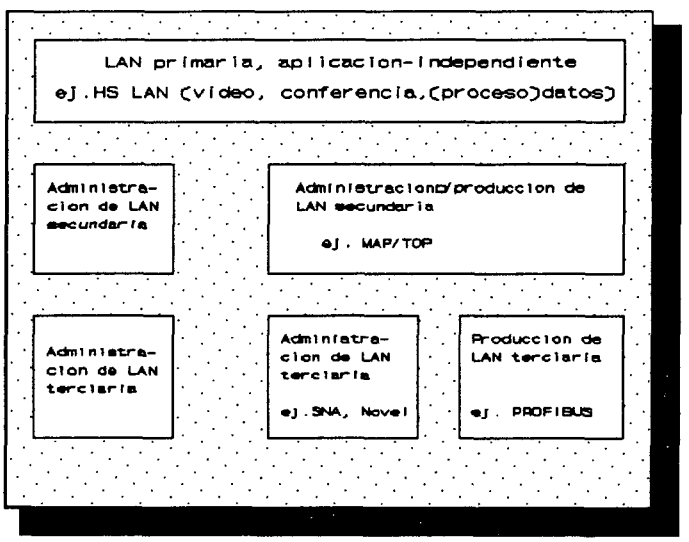
* un sistema de control distribuido tiene redundancia distribuida, los fallos de cortos tiempos de los componentes pueden ser tolerados en algunos casos, pero sería posible una recuperación automática de los componentes.

A veces, estos requisitos no concuerdan con las condiciones reales, necesitando un compromiso aceptable económicamente. PROFIBUS ofrece uno con la especificación RS485 en el nivel físico para áreas sin seguridad intrínseca para unir distancias superiores a 1200m con un rango de transmisión de 93,75Kbit/s y de 200m con 500Kbit/s, incluso en una línea simple de pares entrelazados.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

3.7.1 VISTA GENERAL: LA TECNOLOGIA LAN PARA BUSES DE CAMPO

Los buses de proceso y campo pertenecen al gran número de redes locales que comunican sus estaciones a una distancia relativamente corta una de la otra. En general las LANs se instalan como redes en casa (edificios o plantas), en fábricas o en suelo privado de grandes instituciones. Estas redes conectan todo tipo de equipos, como terminales, estaciones de trabajo, procesadores,... En plantas de fábricas, pueden ser equipos de procesos relevantes (como sensores, actuadores, transmisores, controladores programables o robots,...)



EL CONCEPTO DE JERARQUIA DE LAN EN UNA EMPRESA

FIGURA 3.15

Las técnicas apropiadas LAN se han desarrollado para numerosas aplicaciones, como se comentará más adelante. La totalidad de la acción de LANs no se cubrirá. Será enfatizada el área con buses de proceso y campo dentro de LANs.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

LANs puede ser clasificada aproximadamente dentro de estos tipos:

* LANs primarias. Son LANs de altas prestaciones. Soportan una acción amplia de servicios y son usadas, en general, por áreas de aplicación no específicas. Entre otras cosas, pueden soportar transmisión de voz y de imagen. Las LANs primarias se basan en tecnología de banda ancha haciendo los costes relativamente altos, que sólo se ven justificados con un gran número de estaciones participantes en la utilización óptima de los servicios de transmisión de voz e imagen. El medio de transmisión comprende cable coaxial y fibra óptica. La velocidad de la información permanece en alcances de 100Mbit/s. Estas redes son conocidas bajo el nombre High-Speed LANs (HSLAN). En la tecnología de la automatización pueden usar redes auxiliares (backbone) y soportar nuevos servicios.

* LANs secundarias. También son de altas prestaciones. Suelen superar distancias de 2,5Km. Aunque los costes de inversión son inferiores a pesar de las técnicas de banda ancha utilizadas, no son adecuadas para el soporte de servicios de transmisión de imagen. El medio de transmisión típico es el cable coaxial con velocidades de transmisión que alcanzan velocidades superiores a 50Mbit/s. Un ejemplo típico es la red Ethernet con una tasa de transmisión de 10Mbit/s.

* LANs terciarias. Son conocidas como las LANs de bajo coste. No son caras y se usan especialmente en aplicaciones -o sistemas- de redes de PC específicas y redes de área de proceso y campo. Líneas trenzadas y protegidas y cables coaxiales no caros en tecnología de banda base hacen que la instalación de tales redes sea barata y fácilmente manejables. Las tasas de transmisión van desde 9,6Kbit/s a 2,5Mbit/s. La distancia de transmisión puede superar los 2,5Km, dependiendo de la tasa de transmisión.

En lo siguiente, hablaremos solamente de aspectos de la LANs terciarias. En la figura 3.16, se muestran los requisitos típicos de usuario para las capas 1 y 2 de un bus de campo.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

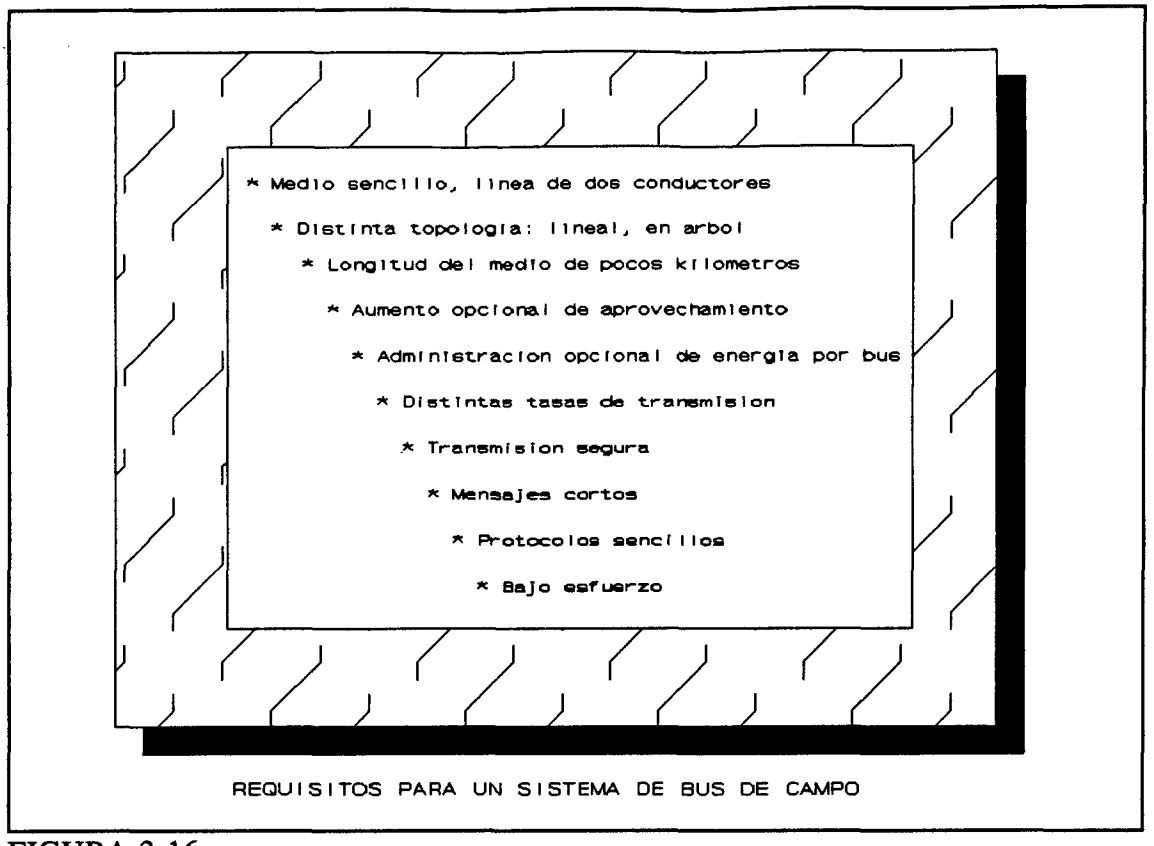


FIGURA 3.16

Topologías de red. Además de la estructura de bus, las redes pueden tener topología de anillo o de estrella. En el comienzo de la tecnología de la automatización, el tipo estrella fue la estructura más frecuente.

Estas redes de tipo estrella necesitan cableado complejo, ya que todas las estaciones se conectan a una unidad central. Alcanza el más alto grado de automatización, el cableado más complejo y menos económico. Así, se prefieren las soluciones de bus en los nuevos sistemas de comunicación.

Buses serie y paralelo. La información digital en un sistema de ordenadores se procesa normalmente en unidades de 8, 16 o 32 bits. En los sistemas de ordenadores modulares, los buses de datos paralelos proporcionan comunicación de alta velocidad entre los diferentes componentes del procesamiento. Normalmente, se usan líneas de buses de datos de 32 y

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

líneas de direccionamiento de 24 ó 32. Los direccionamientos y los datos pueden ser transmitidos en operaciones multiplexadas sobre las mismas líneas. Con buses paralelo, todas las operaciones de acceso al medio y a la transmisión se controlan normalmente por líneas separadas, así que durante la transferencia de datos se puede determinar el siguiente maestro de bus. Los buses paralelo comprenden el VMEbus, Multibus II, NuBus, EISA and Futurebus+.

Los conceptos modernos de buses paralelos van más allá de las funciones tradicionales de la capa 2 del modelo OSI, y define además, por ejemplo, formatos de mensaje, medidas para configuración del sistema e inicialización.

La transferencia de datos entre controladores y ordenadores en el área de campo necesita salvar distancias superiores a los 2Km. En este caso, un bus serie es la solución más económica. Las señales de bit paralelo se convierten en de bit serie por los componentes integrados del controlador de bus de campo. Generalmente hablando, los sistemas de bus se pueden clasificar según las siguientes características:

* Métodos de transmisión del mensaje (multiplexación por división en el tiempo o en frecuencia).

* Protocolo de transmisión síncrono o asíncrono.

* Acceso al bus controlado o al azar.

* Control de bus centralizado o descentralizado.

Técnicas de transmisión. Para buses serie, ambas técnicas son posibles, multiplexación por división en el tiempo (TDM) y multiplexación por división de frecuencia (FDM). En la figura 3.17, se muestra la clasificación de la transmisión y los métodos de acceso para los

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

sistemas de bus serie.

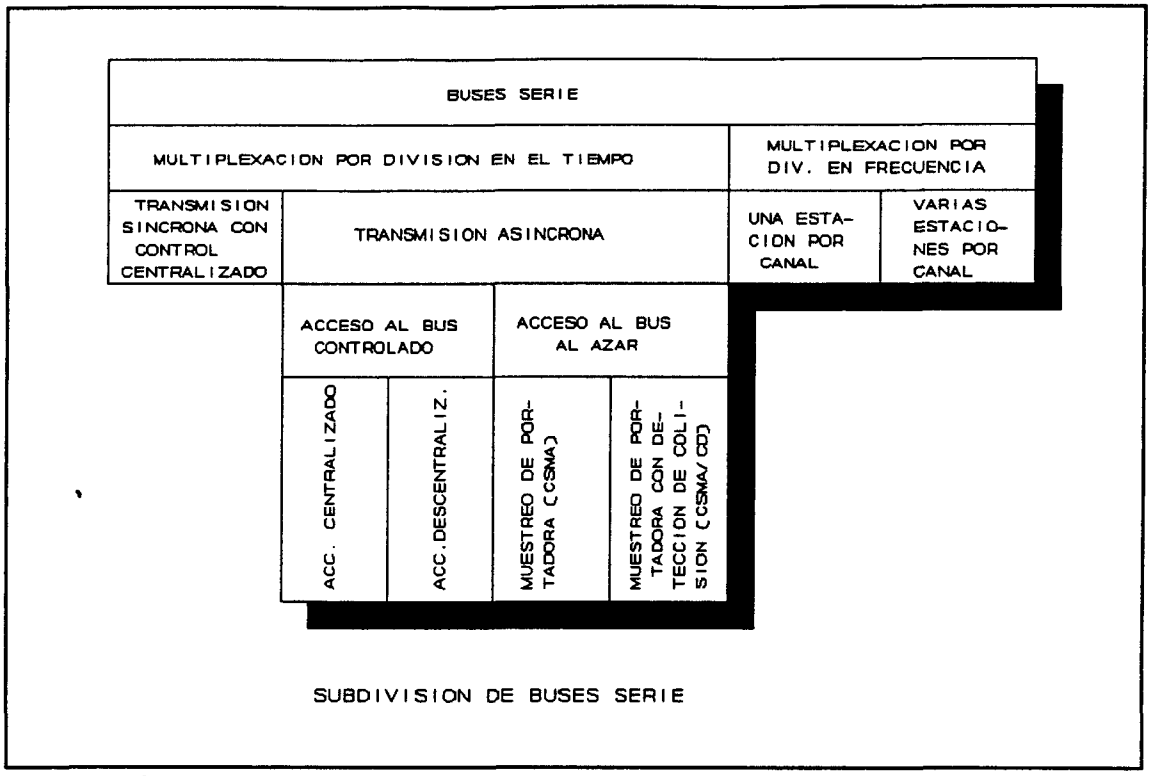


FIGURA 3.17

En multiplexación por división en el tiempo, las distintas estaciones de bus solamente pueden transferir datos en el bus una después de otra, haciéndose necesario reglas para el acceso al bus, que se explicarán en su momento. La multiplexación por división en el tiempo es un método difundido y que puede ser fácilmente implementado. Así que, para buses de proceso y campo, este tipo de acceso al bus es el más frecuente.

En la transmisión analógica de señales de voz y vídeo, un buen método es la multiplexación por división en frecuencia. Se basa en la división del alcance de frecuencia del medio en bandas de frecuencia. Sobre los canales físicos ya establecidos, varias tramas de datos se transfieren simultáneamente en los canales lógicos. La desventaja principal de esta técnica es que la transferencia de datos sólo puede ser unidireccional, lo cual significa que toda estación ha de tener canal de transmisión y canal de recepción.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

La técnica de transmisión de PROFIBUS es la multiplexación por división en el tiempo.

Métodos de acceso al medio. El acceso al medio se controla según las reglas de que toda estación comprueba que está desocupado el bus para la transmisión y recepción de mensajes. Los buses de división en el tiempo pueden ser clasificados aproximadamente en las siguientes categorías:

* Acceso controlado. Puede ser dividido en acceso centralizado o descentralizado. En el árbitro centralizado, una estación maestra distribuye el acceso al bus a las diferentes estaciones. Tal sistema usa el método del "polling", es decir, interrogación cíclica de las estaciones en las distintas variaciones o tienen líneas cortas auxiliares que conectan las estaciones. En el área de bus de campo, se usa principalmente los sistemas maestro/esclavos para hacer el poolling a las estaciones. En estos sistemas existe solamente una estación activa (maestra). Los nuevos desarrollos usan "the Flying Master Principle" en el método de acceso al medio centralizado. La estación maestra (por ejemplo la estación de control) da la autorización de acceso a las otras estaciones una después de otra, las cuales tienen temporalmente funcionalidad de maestras (de ahí el nombre de maestras flotantes).

Un método de acceso descentralizado es el acceso múltiple por división en el tiempo (TDMA). Un intervalo de tiempo fijo se reserva para cada estación del bus en el que puede acceder al bus. Este método es apenas usado en el área de bus de campo porque las tramas y los bits deben estar sincronizados para hacer posible reconocer el comienzo de un corte de tiempo. Entre los métodos de acceso descentralizados, el método del paso del testigo es el más conocido. PROFIBUS lo usa con un soporte de método de maestro/esclavo, que se explicará.

* Acceso al azar. Quiere decir que desde el punto de vista del bus, el acceso al medio

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

viene determinado por la casualidad. Los métodos de CSMA y CSMA/CD (Carrier Sense Multiple Access/Collision Detection) son los mejores ejemplos conocidos de este método. Una estación que quiera transmitir un mensaje escucha primeramente el bus. Cuando no detecta ningún mensaje en transmisión empieza a transmitir. Si otra estación empieza a transmitir simultáneamente se produce una colisión en el bus. Este incidente se resuelve por diferentes algoritmos. Ethernet es una implementación de este método de acceso, pero no es apropiado para el área de bus de campo ya que, en situaciones críticas, el tiempo real requerido no puede ser satisfecho a pesar de lo casual del protocolo, especialmente cuando varias notificaciones de evento ocurren en el mismo momento.

* Acceso híbrido. En la práctica existen varias formas mezcladas de acceso en las que entraremos más adelante.

Especialmente en el área de bus de campo la discusión sobre el mejor método de acceso ya han alcanzado una ideología. Criterios como:

* Conducta en tiempo real

* Garantía de acceso en un cierto período de tiempo

* Control de prioridad de estaciones

* Acceso al bus en intervalos de tiempo equidistantes

* Distinto acercamiento al acceso al medio (maestro/esclavo)

son esenciales para la elección de un sistema específico para una tarea dada.

Medio de transmisión. Los sistemas de bus de campo son varios sistemas de bus. Las líneas

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

de bus en tales sistemas son casi exclusivamente realizables en cables con uno, dos o más hilos. Dependiendo del campo de aplicación, los aspectos económicos la existencia de estándares de compañías, los medios de transmisión siguientes se pueden usar en tecnología de bus de proceso y de bus de campo.

* Cable coaxial. . . . . (Ethernet)
* Par trenzado . . . . . (PROFIBUS)
* Fibras ópticas . . . . . (FDDI(backbone))
* Redes actuales alternativas. . . . . (En automatización de edificios)

Los pares trenzados son la solución menos cara para el área de procesos y campos, ya que son baratos y fácilmente instalados. La desventaja que tienen es una tasa de transmisión limitada (máximo de 10Mbit/s en metros y unos pocos Kbit/s en kilómetros) y susceptibilidad de interferencia, que puede ser reducida cuando se usen los cables protegidos.

3.8 EL MODELO DE COMUNICACION PROFIBUS

3.8.1 PROFIBUS EN EL ENTORNO OSI

La arquitectura PROFIBUS, que se modela según el modelo de referencia ISO/OSI, se divide en tres capas (figura 3.18):

* La capa de aplicación

* La capa de enlace de datos

* La capa física

La arquitectura PROFIBUS se divide en: capa 1 (PHY Layer), capa 2 (FDL layer)

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

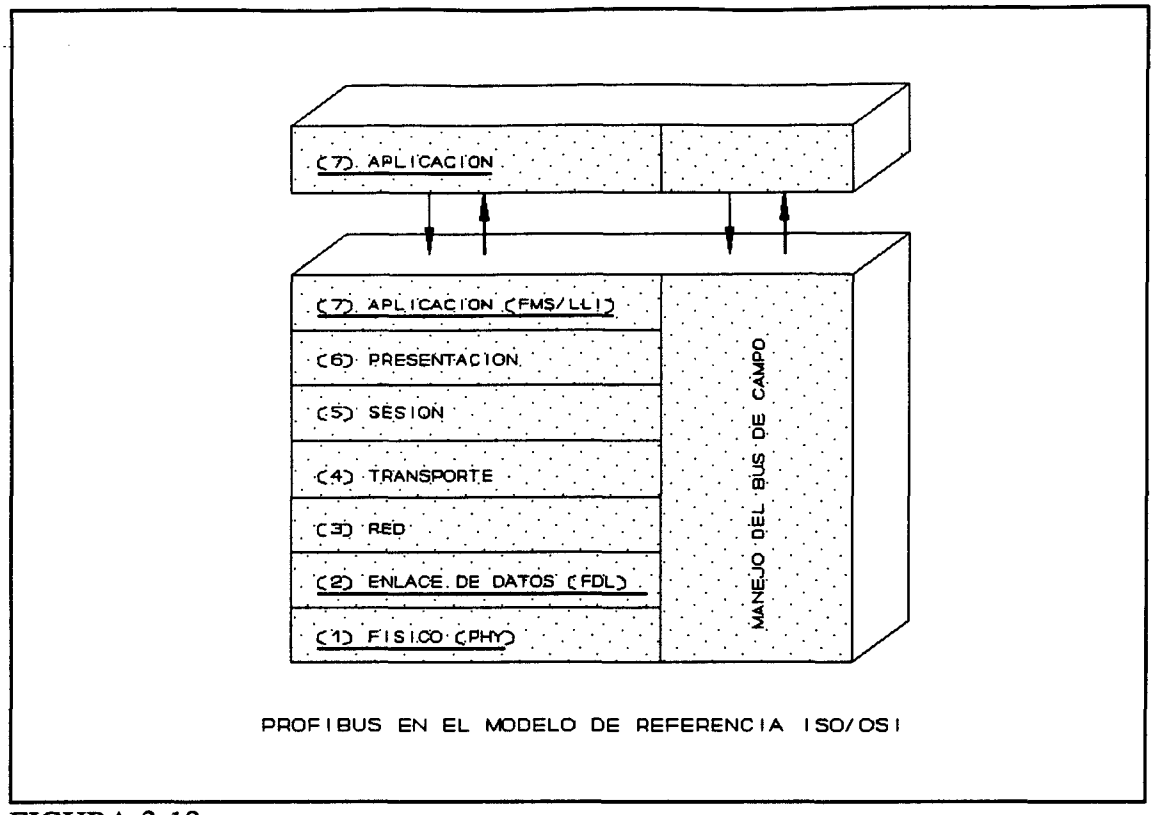


FIGURA 3.18

y capa 7 (compuesta por las subcapas FMS y LLI). La especificación de las dos capas bajas se definen en DIN 19245 parte 1 [DIN91a] y la capa de aplicación se especifica en la parte dos del estándar [DIN91b]. De la capa 3 a la seis se han omitido por eficiencia y alguna de sus funciones necesarias se han transferido a la capa de aplicación.

Los protocolos de bus de campo se deben restringir a solamente la funcionalidad que sea absolutamente necesaria, ya que los requisitos de tiempo real se deben satisfacer. Las características siguientes muestran cómo los protocolos PROFIBUS se han simplificado:

* La segmentación de los mensajes largos (>235 byte) no se soporta.

* No se soporta la agrupación de los mensajes cortos. La combinación de muchos mensajes cortos en un paquete de mensaje largo no está en concordancia con los requisitos de los mensajes cortos, y por lo tanto la especificación no proporciona dicha función.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* No se proporciona en el estándar el soporte de funciones de enrutamiento en la capa de red.

* Excepto para una configuración mínima obligatoria, los subconjuntos convencionales de servicios se pueden crear dependiendo de las necesidades de la aplicación. Este es un aspecto particularmente importante para pequeños sistemas (sensores, ...)

* Son opcionales otras funciones como proyectos de protección de password.

Aunque la eficiencia del protocolo de PROFIBUS es amplia, la simplificación ha reducido la funcionalidad del sistema. Ofrecer al usuario un acoplamiento funcionalmente conveniente y óptimo por las altas prestaciones de un bus de campo, es el objetivo de PROFIBUS, lo que no requiere solamente especificación de protocolos, servicios e implementaciones, sino también - y llega a ser bastante importante para los sistemas modernos- un diseño apropiado de la red y una configuración correcta de sus parámetros.

3.8.2 PROCESOS DE APLICACION DE PROFIBUS

Desde el punto de vista de la comunicación, el proceso de aplicación incluye todos los programas, recursos y tareas como sistemas operacionales, procesos de aplicación real y drivers de comunicación que no se pueden asignar a ninguna capa de comunicación.

Con el modelo de comunicación de bus de campo, es posible combinar virtualmente programas de aplicación distribuida en un proceso universal. Los procesos de aplicación individual se distribuyen en diferentes equipos. Varios procesos de aplicación se pueden localizar en un equipo, los cuales son representados por objetos de procesos (variables, programas, ...) y operaciones aplicables a estos (escribir o leer variables, carga, comienzo y término de programas). En la figura se muestra el significado lógico de la transferencia de datos para el proceso de aplicación. Los datos se transfieren transparentemente, lo que significa que, para los procesos de aplicación, no existe una diferencia visible en la transferencia de datos de proceso a proceso, lo que ocurre en un procesador sencillo o sobre

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

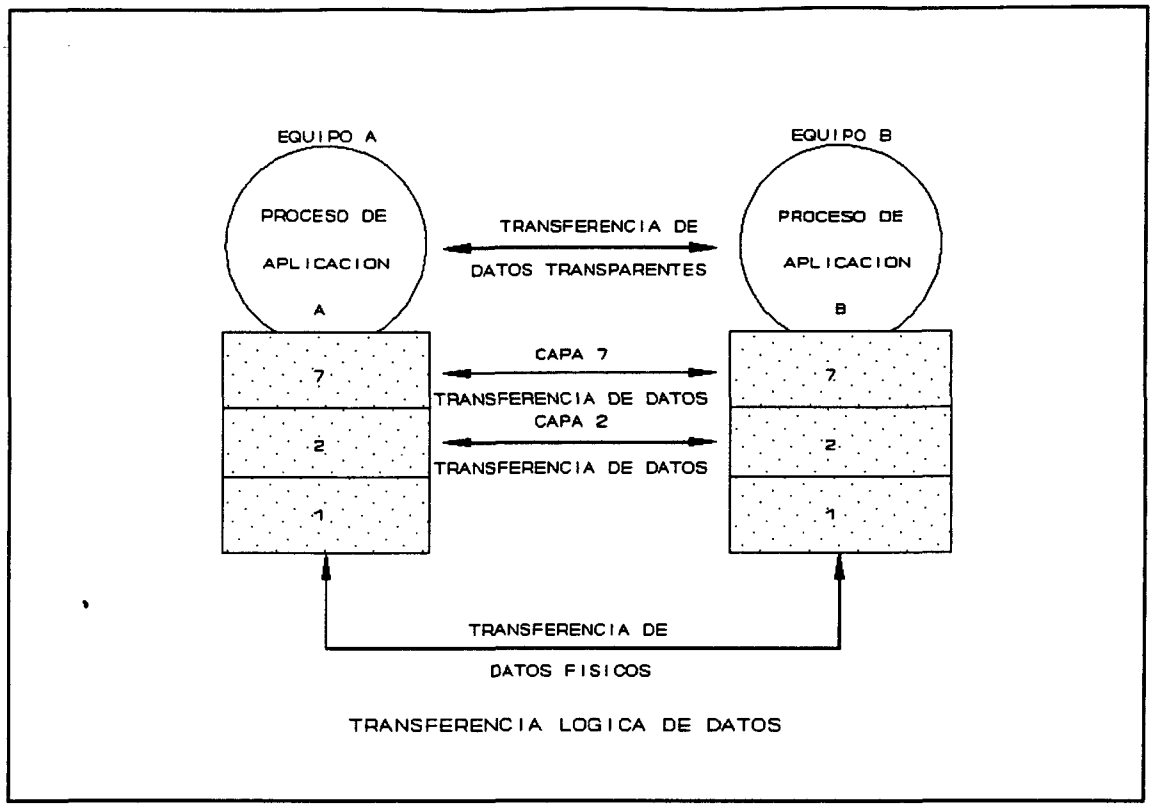


FIGURA 3.19

PROFIBUS. Todas las capas del modelo ISO/OSI se adhieren a este principio de lógica transferencia de datos. El protocolo de la capa N del equipo "A" comunica solamente con el mismo nivel de protocolo de la capa N del equipo B.

3.8.3 EQUIPOS DE CAMPO VIRTUALES Y LA INTERFASE DE CAPA DE APLICACION

El propósito de comunicación en los niveles de proceso y campo es transferir datos (como medida de valores, programas, eventos,...) entre dos estaciones en comunicación. En tecnología de la automatización, cada equipo tiene tales datos. Ya que los datos representan objetos del proceso de aplicación, en PROFIBUS son llamados objetos de proceso.

Para la comunicación entre un proceso de aplicación de un equipo y el de otro, los

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

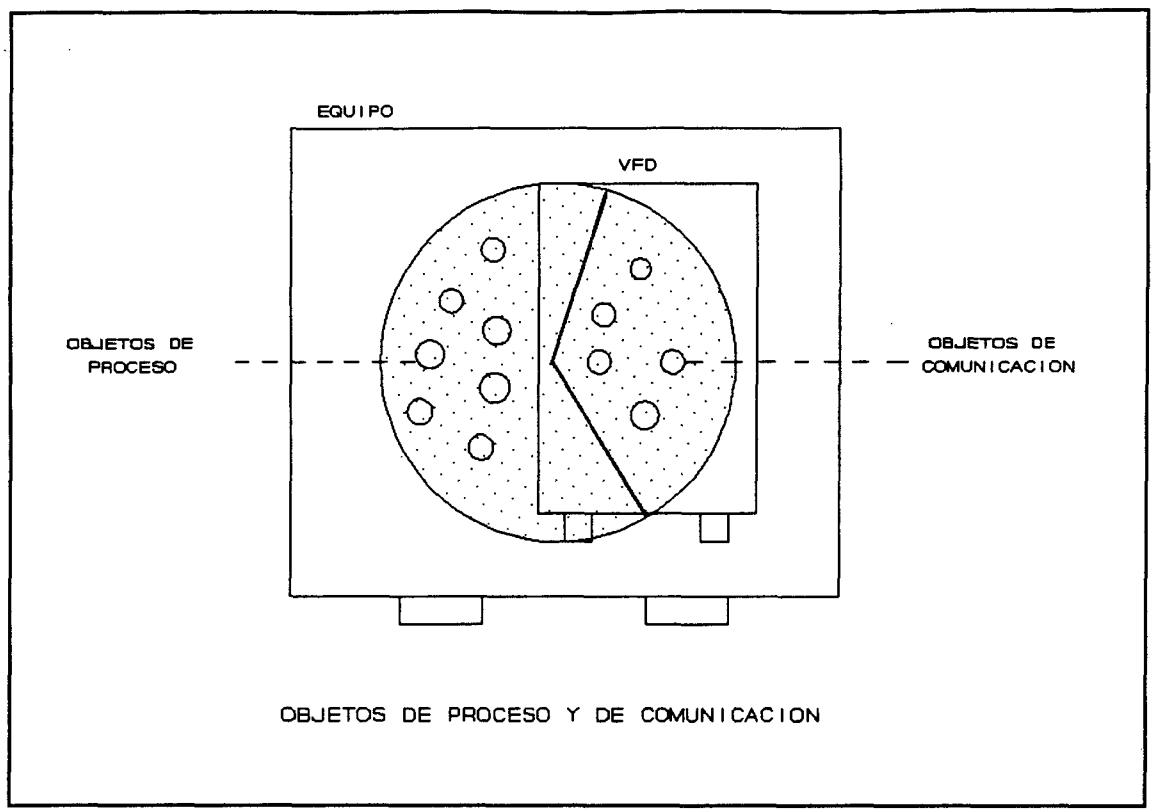


FIGURA 3.20

objetos de proceso transferidos se deben hacer conocer al sistema de comunicación PROFIBUS, lo que quiere decir que los objetos de proceso se deben listar como objetos de comunicación en un Diccionario Objeto (OD) (comparable a la guía de teléfonos). De este modo, un proceso de aplicación debe tener sus objetos visibles y disponibles para PROFIBUS antes de que sean direccionados y procesados por los servicios de comunicación. La comunicación de procesos de aplicación con el de otros en diferentes equipos, necesita para una comunicación más eficiente,más información que el conocimiento de los objetos de comunicación. Normalmente, las estaciones se sitúan a distancia una de otra, o no son accesibles durante la operación; así que deben ser definidas inequívocamente con sus rasgos en la red. Los datos como el nombre del vendedor, nombre del modelo y perfil necesitan leerse por el bus. Además, la información del estado de la interfase de comunicación del equipo y del equipo real (por ejemplo, indicación de datos de servicio) son aspectos muy importantes de PROFIBUS.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Un diccionario de objetos público proporciona a todas las estaciones conectadas al bus, características de equipo estandarizadas, servicios idénticos e interfases uniformes, fijando las bases para la comunicación abierta entre equipos de diferentes vendedores. En PROFIBUS, esta vista consistente de un equipo se denomina Equipo de Campo Virtual (VFD).

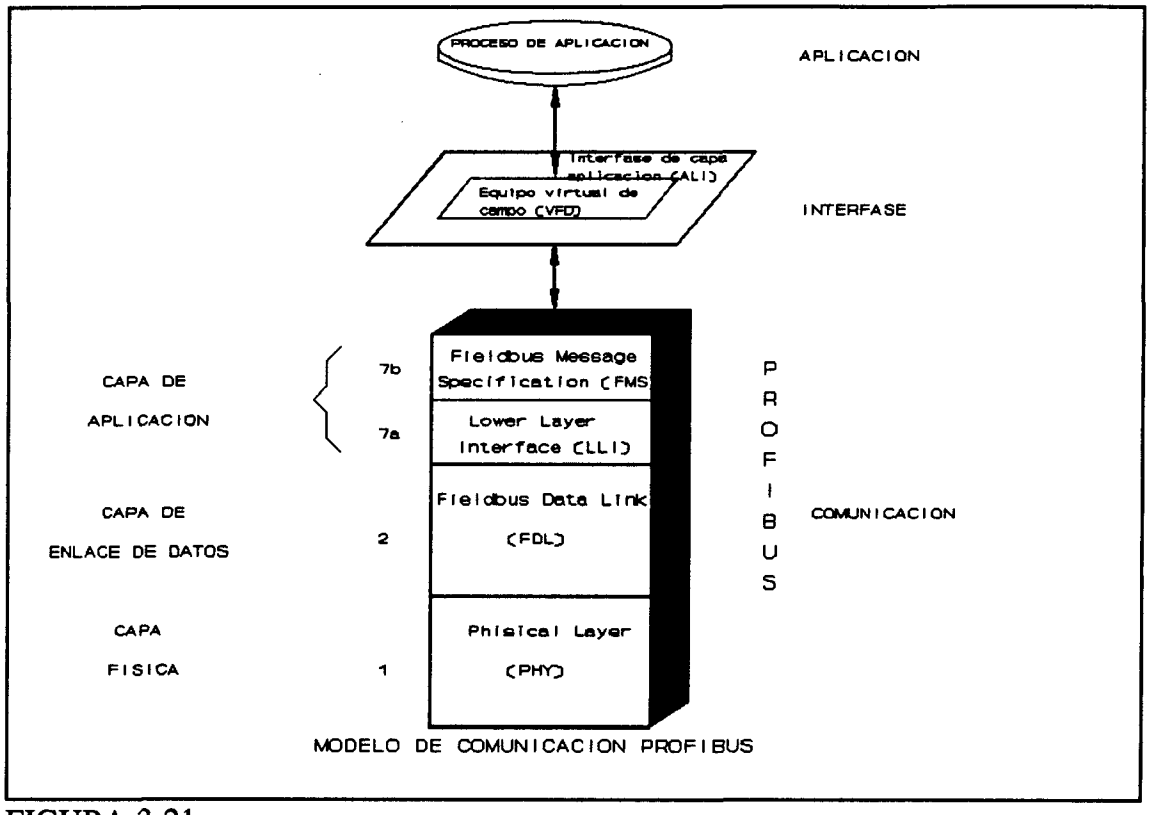


FIGURA 3.21

La especificación de PROFIBUS describe el efecto de los servicios PROFIBUS en los objetos de comunicación de un proceso de aplicación solamente para VFD. La adaptación de VFD a equipos de campo reales y viceversa, no está sujeto al estándar PROFIBUS. Entre la capa de aplicación de PROFIBUS y el proceso de aplicación real yace la denominada Interfase de capa de aplicación (ALI).

Los servicios de la capa de aplicación PROFIBUS pueden ser accedidos por una capa

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

intermedia que es un driver de PROFIBUS. Proporciona funciones de comunicaciones adicionales adaptadas al proceso de aplicación. Son por ejemplo, funciones de segmentación, algoritmos de conversión para la representación de mecanismos de control adicional. Además, la interfase de la capa de aplicación hace la adaptación del VFD a equipo de campo real.

Relaciones entre los procesos de aplicación. Las relaciones lógicas existen entre procesos de aplicación con el propósito específico de transferir datos. En el caso de PROFIBUS, todas las relaciones de comunicación se deben definir antes de que una transferencia de datos se empiece. Estas relaciones se listan en la capa 7 en la lista de relación de comunicación (CRL).

Según ISO 7498, ambos extremos de una relación de comunicación se llaman puntos finales de comunicación. El proceso de aplicación tiene acceso al sistema de comunicación PROFIBUS a través de estos puntos terminales. El proceso de aplicación los direcciona por referencias de comunicación (direcciones de los puntos terminales de comunicación).

Estas referencias de comunicación son específicas del equipo y tienen que ser determinadas por el operador de red. Pueden existir varias relaciones de comunicación entre dos procesos de aplicación a través únicamente de puntos terminales de comunicación asignados.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

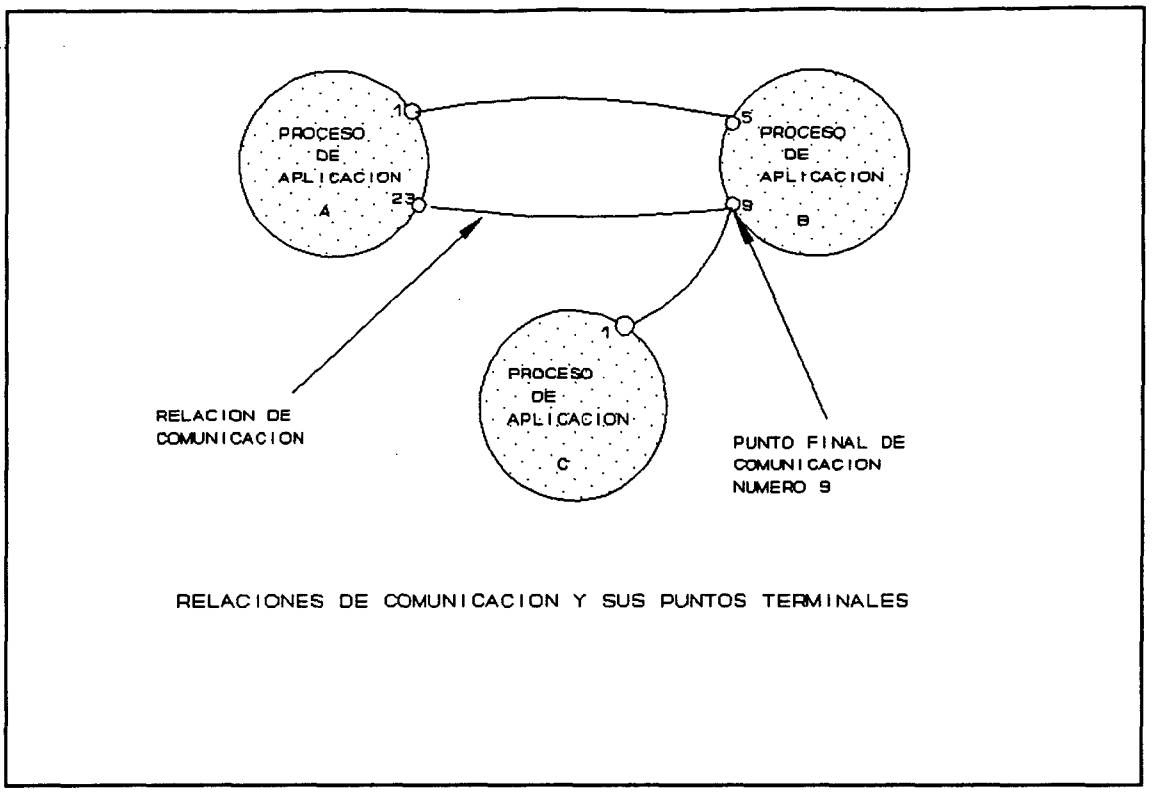


FIGURA 3.22

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

CAPA DE APLICACION EN PROFIBUS

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

4 CAPA DE APLICACION PROFIBUS

4.1 INTRODUCCION

Vamos a considerar los aspectos más importantes de la capa siete de la especificación de PROFIBUS.

La capa de aplicación (Capa 7) de PROFIBUS y la interfase de la capa de aplicación (ALI) se pueden dividir en los siguientes componentes:

Interfase de la capa de aplicación (Application Layer Interface, ALI). Es la interfase del proceso de aplicación individual a la interfase estandarizada de la capa de aplicación, y viceversa. La ALI junto con el Equipo de Campo Virtual abarcan el manejo del servicio y del objeto.

Especificación del mensaje de bus de campo (Fieldbus Message Specification, FMS). La subcapa superior de la capa siete tiene tres funciones:

La máquina de protocolo vigila las reglas en la tranferencia de las unidades de datos del protocolo (PDU) entre los diferentes equipos.

Generación de las unidades de datos de protocolo (PDU's) y codificación. Si un usuario llama a un servicio de la FMS, un "paquete" se genera para transmitir los datos. En la generación del paquete, se debe estructura la información sobre el equipo deseado (por ejemplo, un número de orden, tipo de servicio, parámetros,...) de una forma predeterminada, llamada sintaxis. La sintaxis usada en PROFIBUS se encuentra definida en el estándar ASN1.1 (Abstract Syntax Notation One 8824, CCITT X.208) del estándar ISO. Después de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

ser generado, el paquete de información se debe codificar. En los procesos de aplicación y comunicación, se usa un modo interno de representación para el procesamiento local. Para la transmisión sobre el bus de campo, sin embargo, esta representación binaria es demasiado ineficiente y no es apta para comunicación abierta. Por lo tanto, las reglas de codificación apropiadas se han diseñado por PROFIBUS satisfaciendo la necesidad de mensajes cortos en el bus.

Decodificación e interpretación de las unidades de datos de protocolo (PDU's). Los mensajes entrantes deben ser tratados como se describe más adelante, pero en secuencia inversa.

Interfase de la capa inferior (Lower Layer Interface, LLI). En FMS se usan exclusivamente las relaciones de comunicación entre los procesos de aplicación como recursos, pero pueden también usarse para manejo, control y planificación en capas adyacentes, para esta función se utiliza la interfase de la capa inferior, LLI.

Manejo de red (Fieldbus Management Application, FMA). Aquí se hacen disponibles las funciones de manejo para el usuario.

4.2 LA INTERFASE DE LA CAPA DE APLICACION

El principio básico de comunicación abierta es permitir comunicación entre diferentes sistemas de aplicación de vendedores específicos que normalmente son incompatibles. Para conseguir una comunicación abierta, son necesarios algunos esfuerzos de estandarización, los cuales solamente conciernen al lado de la comunicación, o a la forma en la que trabaja el sistema de transporte transparente. El significado concreto de la transferencia de datos depende de la aplicación.

Si un proceso de aplicación de un equipo quiere comunicar con otro equipo, se debe

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

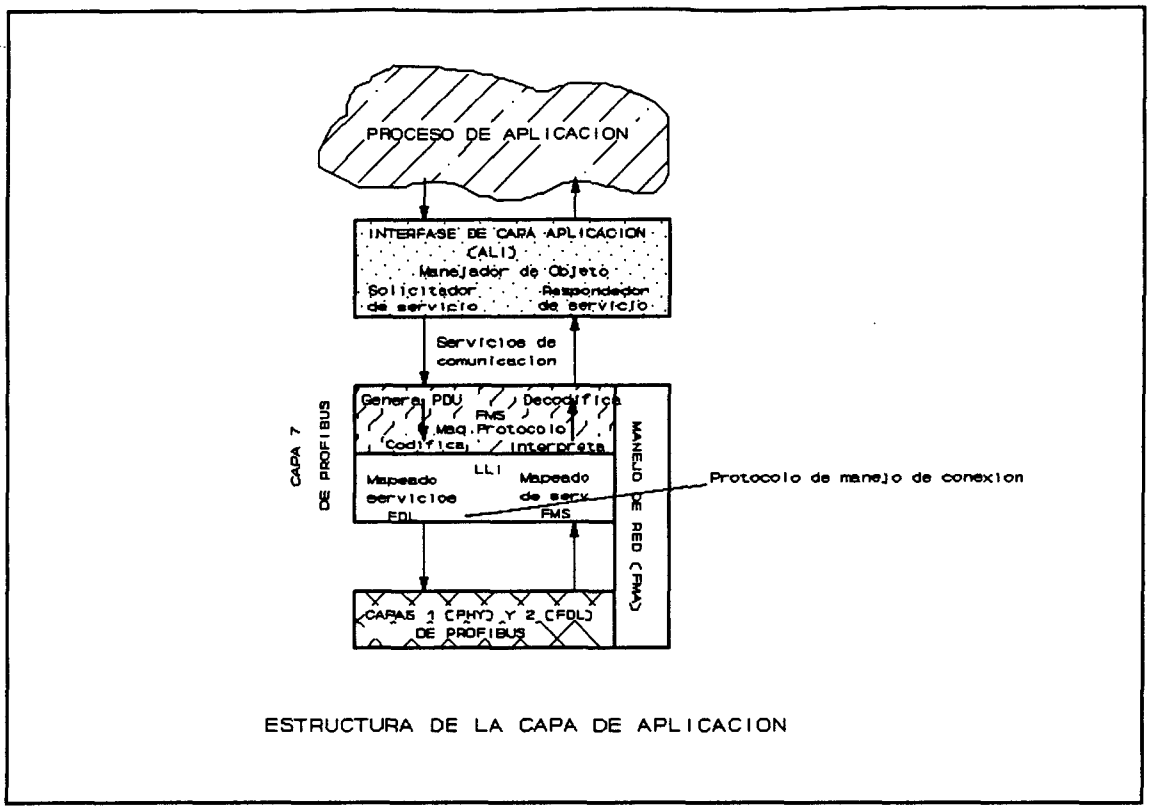


FIGURA 4.1.

permitir el acceso al otro proceso de aplicación a los objetos que existen localmente por ciertos servicios. Los servicios se refieren a operaciones con los objetos de una clase definida. Un valor medido, por ejemplo, es un objeto que pertenece a la clase de variables. Los servicios permitidos en esta clase son leer y escribir.

De este modo, la tarea de la Interfase de la capa de aplicación (ALI) es planificar los objetos locales en existencia (objetos de proceso) dentro de los conocidos por PROFIBUS (objetos de comunicación) y transmitir las necesidades de servicio. La Capa de Aplicación asume el papel de el solicitador de servicio (cliente). El papel de proporcionante del servicio (servidor) se puede describir mediante un Equipo de Campo Virtual (VFD) y desde el punto de vista del sistema de comunicación. Este VFD se define por un modelo de conducta dentro del FMS.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

4.3 ELEMENTOS DE COMUNICACION DE LA CAPA 7

Desde el punto de vista del usuario de funciones de aplicación, una comunicación normalmente tiene lugar entre dos estaciones. Una estación tiene la función de cliente (solicitador del servicio) y otra estación la función de servidor (proporcionante del servicio). Un cliente debe, por ejemplo, ser un controlador lógico programable, un servidor, un sensor. Un sensor operará siempre como servidor solamente, mientras otros equipos como SPS pueden asumir funciones de clientes dependiendo de la tarea actual.

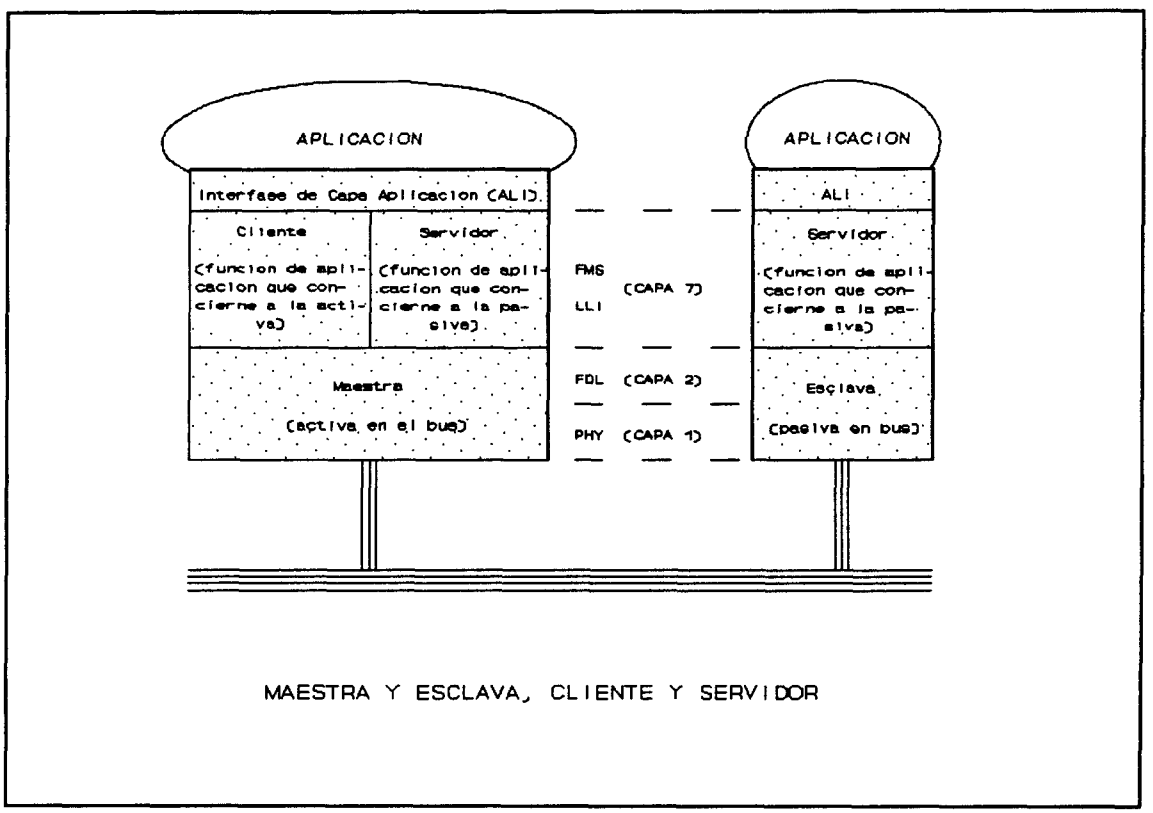


FIGURA 4.2.

Cuando las estaciones en comunicación se ven desde el punto de vista de transferencia de mensaje, se llaman maestra y esclava (Ver figura 4.2). Con respecto a la comunicación, una maestra es una estación activa y una esclava es una estación pasiva. El medio de acceso en PROFIBUS se distribuye mediante la trama de testigo, que se pasa de maestra a maestra.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Como oposición a la maestra y esclava, el cliente y servidor tienen Funciones de Capa de Aplicación. Normalmente en PROFIBUS un cliente debe tener funcionalidad de maestra, mientras que para funciones de servidor, es suficiente una funcionalidad de esclava.

Para comunicación entre cliente y servidor, la capa 7 de PROFIBUS proporciona básicamente los siguientes tres elementos de comunicación:

* Objetos de comunicación
* Servicios de comunicación
* Relaciones de comunicación

Los objetos de comunicación, pueden ser datos medidos, partes de programas o puesta de parámetros que son intercambiados entre estaciones. Estos objetos son disponibles para las respectivas estaciones por medio de los **servicios de comunicación.** Los servicios más comunes son leer y escribir.

Una **relación de comunicación** establece la conexión lógica entre las respectivas estaciones. La condición para el establecimiento de la relación de comunicación lógica es la existencia de una conexión física: ambas estaciones deben estar conectadas en la red (en analogía al teléfono, una conexión debe existir entre dos personas aunque no se comuniquen en ese momento). También, al menos una de las dos estaciones debe ser cliente, ya que una comunicación directa no puede ser establecida entre dos servidores. La mayoría de las relaciones de comunicación son entre parejas de entidades (relaciones cliente-servidor), aunque PROFIBUS también soporta multicast y broadcast.

Los datos de diseño de todas las relaciones de comunicación de una estación se listan en la Lista de Relación de Comunicación (CRL). Es normal dar esta lista en forma de tabla. Cada línea en la tabla contiene los parámetros de una referencia de comunicación (CR), y cada referencia de comunicación denota una relación específica entre dos procesos de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

aplicación.

Una tabla similar también existe para los objetos de comunicación de cada estación. Esta tabla se llama Diccionario Objeto (OD). Cada línea contiene las entradas para un objeto y tiene a la cabeza un índice (de objeto). Ambas entradas y los índices se pueden leer en las estaciones remotas.

La Lista de Relación de Comunicación y los Diccionarios Objeto de los equipos no tienen que ser en forma tabular; deben también ser almacenados como una estructura de datos o de cualquier otra forma. El estándar determina solamente la estructura de sus transmisiones sobre el bus. Además, deben ser listados en hojas de datos. Para sensores, las entradas en el CRL y OD se fijarán normalmente por el vendedor. Algunos equipos (por ejemplo, controladores) pueden ser parametrizados o programados por el usuario durante el diseño. En redes amplias se puede soportar esta tarea por herramientas de configuración.

4.4 LOS OBJETOS PROFIBUS

La comunicación en un sistema PROFIBUS se basa en objetos, sobre los que los datos pueden ser transferidos. Los objetos tienen atributos tales como tipo de datos, derechos de acceso, llaves de acceso,... Para cada tipo de objeto, se ha definido un conjunto de operaciones que se les puede aplicar.

Tomemos el ejemplo de comienzo de un programa, 'comienzo' es la operación a ser ejecutada en el objeto 'programa'. Este ejemplo ilustra también los problemas de comunicación abierta entre equipos de diferentes vendedores. Aunque es la misma operación, puede ser ejecutada de forma distinta por cada equipo.

La interconexión de un equipo real a un sistema de comunicación abierto requiere un diseño estrictamente determinado de las manipulaciones reales de objetos de comunicación

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

para los correspondientes servicios PROFIBUS. El sistema de comunicación, que se modela según el modelo ISO/OSI, es siempre un sistema de transporte de mensajes transparente, lo que significa que el contenido del programa no es importante para el sistema de comunicación. Es un mensaje para un objeto del tipo Programa. Estos objetos pertenecen solamente a los procesos de aplicación.

Los contenidos de un objeto transferido entre procesos diferentes de aplicación deben hacerse saber en sus estructuras al sistema de comunicación. Esto se hace en PROFIBUS con todos los objetos de proceso introduciéndolos en el propio Diccionario de Objetos para hacerlos disponibles a otros procesos de aplicación. Se puede considerar un Diccionario Objeto como un directorio de teléfonos de todos los objetos disponibles. Al introducir un objeto de proceso en un Diccionario Objeto, se convierte en un objeto de comunicación, y se hace conocido para todos los procesos remotos de aplicación.

4.4.1 TIPOS DE OBJETO ESTATICOS Y DINAMICOS

En PROFIBUS, son distinguidos en un proceso de aplicación objetos estáticos y dinámicos, diferenciándose solamente en el tiempo de definición. Los objetos de comunicación estáticos se determinan durante la fase de comienzo de la red, ya que sus estructuras no cambian durante la operación.

Los objetos de comunicación estáticos:

-> variables (variables simples, arrays y registros)
-> dominio
-> evento

Los objetos de comunicación dinámicos:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

-> Lista de variables

-> Invocación de programas

Estos objetos no pueden modificar su estructura ni borrarse durante la operación. Los objetos de comunicación dinámicos, en cambio, se pueden borrar o modificar su estructura del Diccionario Objeto durante la operación.

Los objetos de tipo variable simple son unidades invisibles, tales como valores de medida, la hora o el estado de un equipo. Estas unidades tienen un tipo de datos prescrito, cuyos componentes simples, si existen, no se pueden direccionar individualmente.

Un array es un conjunto de elementos de variables simples del mismo tipo, cuyos componentes sencillos se pueden direccionar individualmente.

Un registro es un conjunto de elementos de variables simples de distintos tipos, cuyos componentes pueden ser individualmente direccionados.

El objeto dominio es un área de memoria lógicamente conectada de una longitud máxima predeterminada que pueden contener tanto datos como programas. El tipo de datos de un dominio es siempre un string de octetos (octeto es un término general para ocho bits, un byte).

El objeto evento contiene un mensaje importante que se manda normalmente con alta prioridad a otras estaciones.

Una lista de variables puede ser creada dinámicamente durante la operación del sistema de comunicación. Constituye una lista de las descripciones de los objetos variable, que han sido definidos durante la fase de inicialización. Las listas variables pueden, por ejemplo, ser usadas para la generación de diagramas de flujo de proceso.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

En el objeto invocación de programa, los dominios se combinan en una unidad que contiene un programa ejecutable, parámetros apropiados y bloques de datos. Una invocación de programa se define durante la fase planeada o generada durante la operación de sistema de comunicación.

4.4.2 TIPOS DE DATOS

La estructura de variable simple se define por el tipo de datos. Este tipo de datos debe determinarse por el estándar PROFIBUS o combinado con tipos de datos (estándares) en una estructura de datos. Por consiguiente, es posible definir un propio tipo de datos. PROFIBUS contempla los siguientes tipos de datos predefinidos:

* booleano
* entero (8, 16, 32 bits)
* sin signo o unsigned (8, 16, 32 bits)
* punto flotante
* string de octetos (codificación binaria)
* string visible (ISO 646)
* fecha (fecha de calendario e indicación de hora)
* hora del día (milisegundos contados desde media noche)
* diferencia horaria (indicación de hora en milisegundos)
* string de bits (en unidades de ocho bits)

Los tipos de objeto Invocación de programa, evento, Lista de variable y Dominio son tipos de datos prescritos. Variables array y registros, son combinaciones de objetos de datos de Variable Simple.

4.4.3 OBJETOS DESCRITOS EXPLICITA E IMPLICITAMENTE

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

PROFIBUS distingue entre objetos explícitos e implícitos: los objetos explícitos son descritos en el Diccionario Objeto del Equipo de Campo Virtual (VFD), donde estas descripciones pueden también ser recogidas mediante los servicios PROFIBUS, mientras que la descripción de objetos implícitos solamente puede ser encontrados en el estándar.

Los objetos explícitos son objetos de procesos de aplicación tales como variables simples, eventos o invocación de programas. Ya que estos objetos se usan como objetos de comunicación, necesitan una descripción explícita en el Diccionario Objeto. Los objetos implícitos, por otro lado, son objetos únicos del VFD. Las descripciones de los correspondientes objetos pueden por lo tanto ser determinados -implícitamente- en la especificación de VFD. Un objeto implícito, por ejemplo, puede incluir tipos de datos, descripciones de objetos e información de estado.

4.4.4 ATRIBUTOS DE OBJETO

Los objetos explícitos tienen atributos que son descritos formalmente en la especificación PROFIBUS, lo que es la base para el acceso estandarizado a objetos en un sistema de comunicación abierto.

Como ejemplo, vamos a dar la descripción de el objeto dominio, con atributo de clave definiendo una clave de acceso para un objeto:

OBJECT:	DOMAIN
KEY ATTRIBUTE:	INDEX
KEY ATTRIBUTE:	DOMAIN NAME
ATTRIBUTE:	MAX OCTETS
ATTRIBUTE:	PASSWORD
ATTRIBUTE:	ACCESS GROUPS
ATTRIBUTE:	ACCESS RIGHTS

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

ATTRIBUTE: LOCAL ADDRESS
ATTRIBUTE: DOMAIN STATE
ATTRIBUTE: UPLOAD STATE
ATTRIBUTE: COUNTER
ATTRIBUTE: EXTENSION

INDEX: Dirección lógica del objeto de dominio
DOMAIN NAME: Nombre del objeto dominio
MAX OCTETS: Número máximo permitido de octetos para el dominio
PASSWORD: Contiene la clave para el mecanismo de protección de acceso
ACCESS GROUPS: Asignación del objeto a diferentes grupos de acceso.
ACCESS RIGHTS: Los datos concernientes a los tipos permitidos de acceso, como lectura para todos, escritura para el grupo de acceso número 3.
LOCAL ADDRESS: La indicación específica de sistema del área de memoria de existencia real
DOMAIN STATE: Contiene el estado del dominio
UPLOAD STATE: Estado de la máquina de estado de carga hacia arriba (por ejemplo, no existente, cargando, cargado)
EXTENSION: Contiene los datos específicos de perfiles (y aplicación)

La descripción de objeto arriba mencionada muestra que los objetos orientados a aplicación en PROFIBUS (por ej. dominio) no consisten solamente en tipos de datos string de octetos. Para salvaguardar la consistencia de los datos, los objetos tienen una máquina de estado de objeto (Figura 4.3). Por esta razón no es posible definir y empezar un área de memoria durante la fase de carga como una invocación de programa, por ej. un programa ejecutable.

4.5 ACCESO A LOS OBJETOS VIA PROFIBUS

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

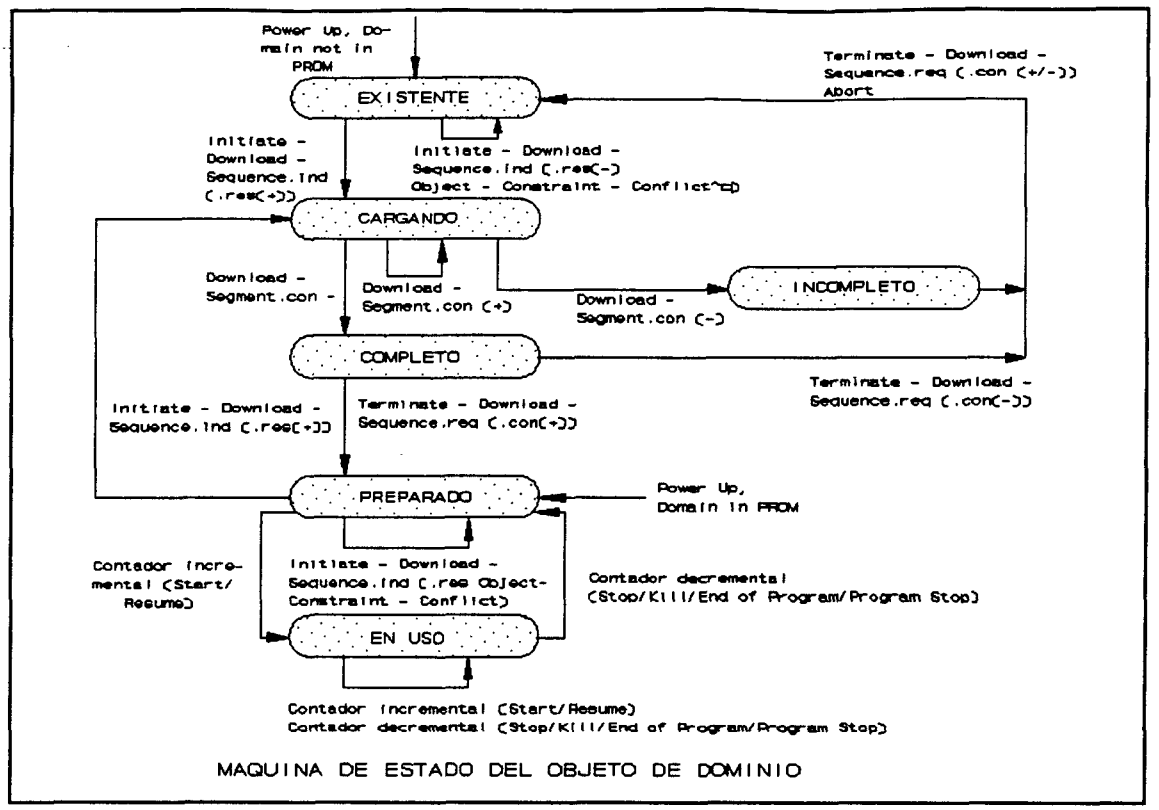


FIGURA 4.3.

4.5.1 MECANISMO DE ACCESO AL OBJETO (DIRECCIONAMIENTO)

PROFIBUS conoce cuatro tipos diferentes de direccionamiento. La aplicación de este direccionamiento de objeto, es restringida según el tipo de objeto y servicio, p.e., no todos los servicios permiten direccionamiento físico de un objeto de usuario. PROFIBUS soporta los siguientes cuatro tipos:

* **Direccionamiento lógico (Index).** El direccionamiento lógico proporciona acceso a los objetos visibles para el sistema de comunicación de campo por cortas direcciones especiales, denominadas índices en PROFIBUS, lo que permite la transmisión de mensajes rápidos de sólo unos pocos octetos de longitud sobre el bus. Estos índices se listan en los Diccionarios Objeto de los respectivos equipos de campo. El Diccionario Objeto contiene las descripciones de todos los objetos de un equipo accesibles por el bus. La descripción de un

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

objeto incluye el índice, nombre, tipo de dato y algunas veces la longitud del objeto. Las descripciones de objeto se pueden transferir entre estaciones en comunicación. Para transferir los datos de usuario, el índice se debe indicar en el solicitador del servicio.

* **Direccionamiento físico.** Un direccionamiento físico puede ser usado sólo con el 'objeto de acceso físico', un objeto de datos que puede ser direccionado con una dirección física (privada). Este tipo de direccionamiento sólo se usa en caso especiales (como diagnosis de errores ocurridos), ya que va contra el concepto de comunicación abierta.

* **Direccionamiento implícito.** Un direccionamiento implícito en PROFIBUS es una referencia de comunicación, que se usa para direccionar VFD's y procesos de aplicación. Junto con la referencia de comunicación, la invocación ID (número de orden) asigna implícitamente respuestas que llegan a la estación solicitadora a las peticiones de servicio no confirmadas aún.

* **Direccionamiento con nombres.** Los objetos de comunicación pueden ser direccionados también por un nombre o en su lugar por la dirección lógica. Estos nombres de objeto tienen una longitud máxima de 32 caracteres y se determinan normalmente por la aplicación, por medio de los perfiles y convenios específicos para industria.

4.5.2 DICCIONARIO OBJETO

PROFIBUS trabaja de forma orientada al objeto. Antes de que la comunicación sobre el bus sea posible, los objetos de comunicación y sus significados o estructura deben conocerse en el sistema de comunicación. Esta es la misión del Diccionario Objeto.

Las descripciones en el Diccionario Objeto solamente son válidas para transferencia sobre el bus. El usuario no está limitado a ellas en la configuración del sistema. La descripción de objeto se define en la estación donde existe realmente el objeto (OD fuente).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

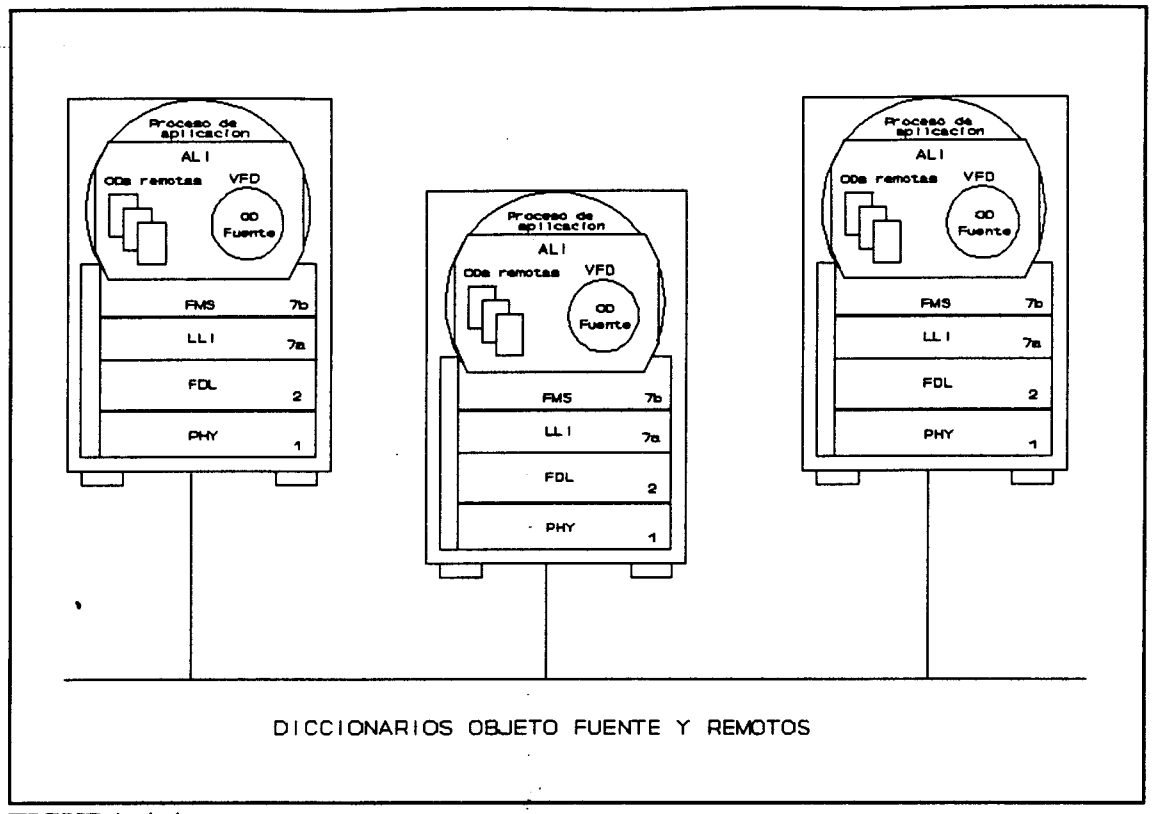


FIGURA 4.4.

Las otras estaciones tienen una copia completa o parcial de las descripciones de objeto (OD remota) de sus parejas de comunicación. De ese modo, cada estación mantiene una OD fuente para los objetos de comunicación local, y una o más ODs remotas como se muestra en la figura 4.4. Existen servicios de direccionamiento especializados para recuperar y modificar un OD fuente sobre el bus.

El diccionario Objeto puede contener hasta seis subdiccionarios. Cada subdiccionario objeto contiene descripciones de una cierta clase de objetos de comunicación. La descripción del OD contiene el contenido de los subdirectorios objetos que sigue. Cada descripción de objeto tiene una dirección única o un nombre o índice sobre el cual puede ser leído del diccionario de la propia estación. Este índice es al mismo tiempo la dirección corta del objeto.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

PROFIBUS tiene los siguientes subdiccionarios:

* **Cabecera OD.** La cabecera OD contiene la información estructural y directora del OD. Se almacena en el OD en primera posición bajo el índice 0 y puede ser leída por la pareja de comunicación.

* **Lista estática de tipos.** Consta de dos subdiccionarios (Figura 4.5), que contienen los Tipos de Datos y las Descripciones de la Estructura de Tipos de Datos conocidos por el sistema de comunicación. Han sido definidos durante el diseño de sistema y fase de inicialización.

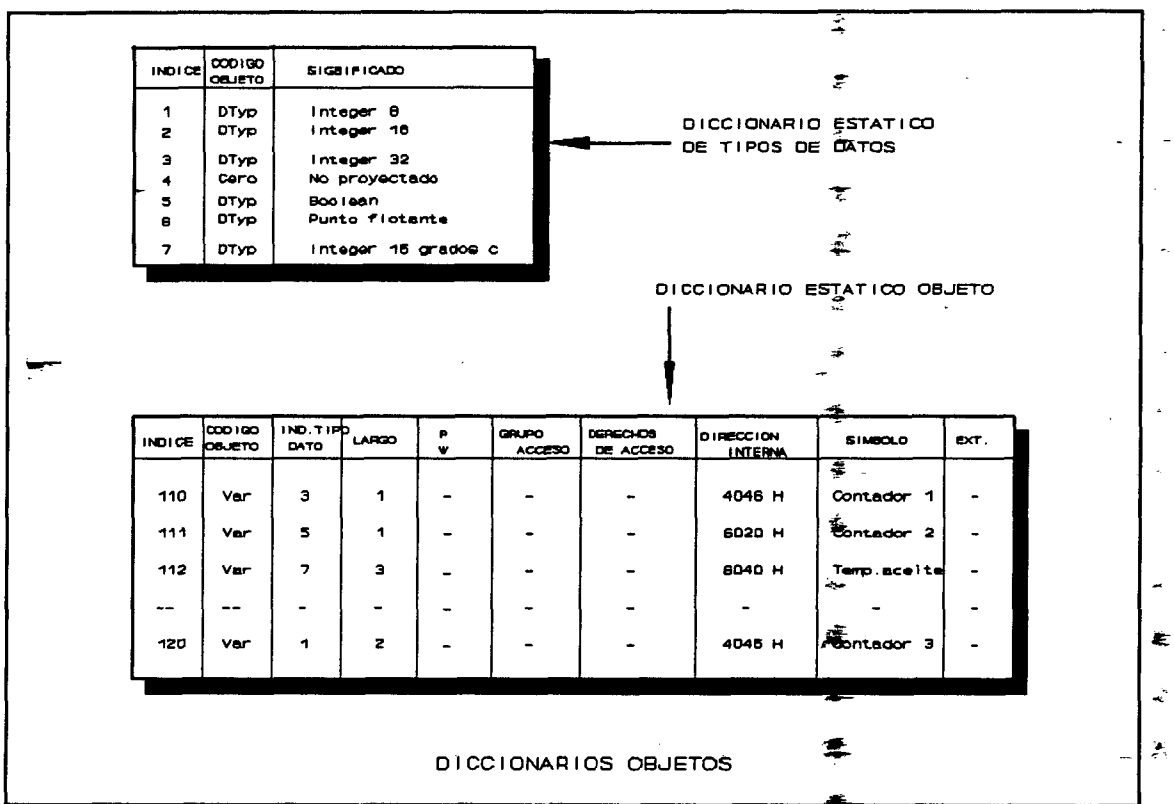


FIGURA 4.5.

* **Diccionario Objeto Estático.** El Diccionario Objeto Estático (Figura 4.5) contiene la descripción de objetos de Variables Simples, Arrays, Registros, Dominios y Eventos. Se

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

asigna para cada índice una descripción de objeto. Además del índice, al objeto se le puede dar un nombre que sería único si se usa en la red íntegra. La descripción de objeto del Diccionario Objeto se define normalmente durante las fases de diseño e inicialización.

* **Lista Dinámica de Listas de Variables (opcional).** Contiene la descripción de objeto de las Listas de Variable. Son clases de objetos de datos formadas de un grupo de Variables Simples, Arrays y Registros definidas en el Diccionario Objeto Estático. El grupo puede tener un nombre e índice, y puede generarse dinámicamente y borrado durante la operación. La descripción de la lista de variable consta de muchos índices definidos en el Diccionario Objeto Estático.

* **Lista dinámica de invocaciones de programas (opcional).** Contiene las descripciones de objeto de los objetos de Invocación de Programas. La descripción de objetos de programa básicamente consta de índices de dominios que han sido definidos en el Diccionario de Objetos Estático. El diccionario de invocación de programa dinámico puede ser dividido en dos grupos:

. Invocaciones de Programas Predefinidos (algunas veces almacenado en PROM)

. Invocaciones de Programas dinámicos (definidos durante la operación)

Creación y borrado de descripciones de objeto de comunicación.

En diccionarios estáticos, las descripciones de los objetos de comunicación pueden ser modificados solamente durante las fases de diseño o de inicialización, ya que de otra forma tendría que ser restaurada la perspectiva de los objetos de comunicación. En Diccionarios Objetos de invocación de programas y variables, no obstante, pueden ser definidos o borrados dinámicamente los nuevos objetos de comunicación durante la operación de procesos de aplicación distribuida, si es requerido por la funcionalidad del equipo y la aplicación. Estos es posible en listas dinámicas, ya que la definición de Listas de Variable

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

e Invocaciones de Programa consta de definiciones de objetos de comunicación de listas estáticas, y la consistencia se mantiene por un mecanismo de enganche de protocolo.

El solicitador del servicio (cliente) y el proveedor del servicios (servidor) deben intercambiar las descripciones de objeto entradas o borradas antes de que puedan ser transferidos por el Diccionario Objeto los datos afectados.

4.6 RELACIONES DE COMUNICACION

4.6.1 LISTA DE RELACION DE COMUNICACION (CRL)

Desde la situación del usuario, la comunicación con los procesos de aplicación de las otras estaciones se produce sobre canales lógicos. Estos canales lógicos se definen durante la fase de diseño y entrados en la Lista de Relaciones de Comunicación (CRL). Cada estación tiene una lista tal para sus relaciones de comunicación. La determinación de las relaciones de comunicación es independiente de sus aplicaciones. Son finalmente cargadas en el software de la capa siete por el director de red.

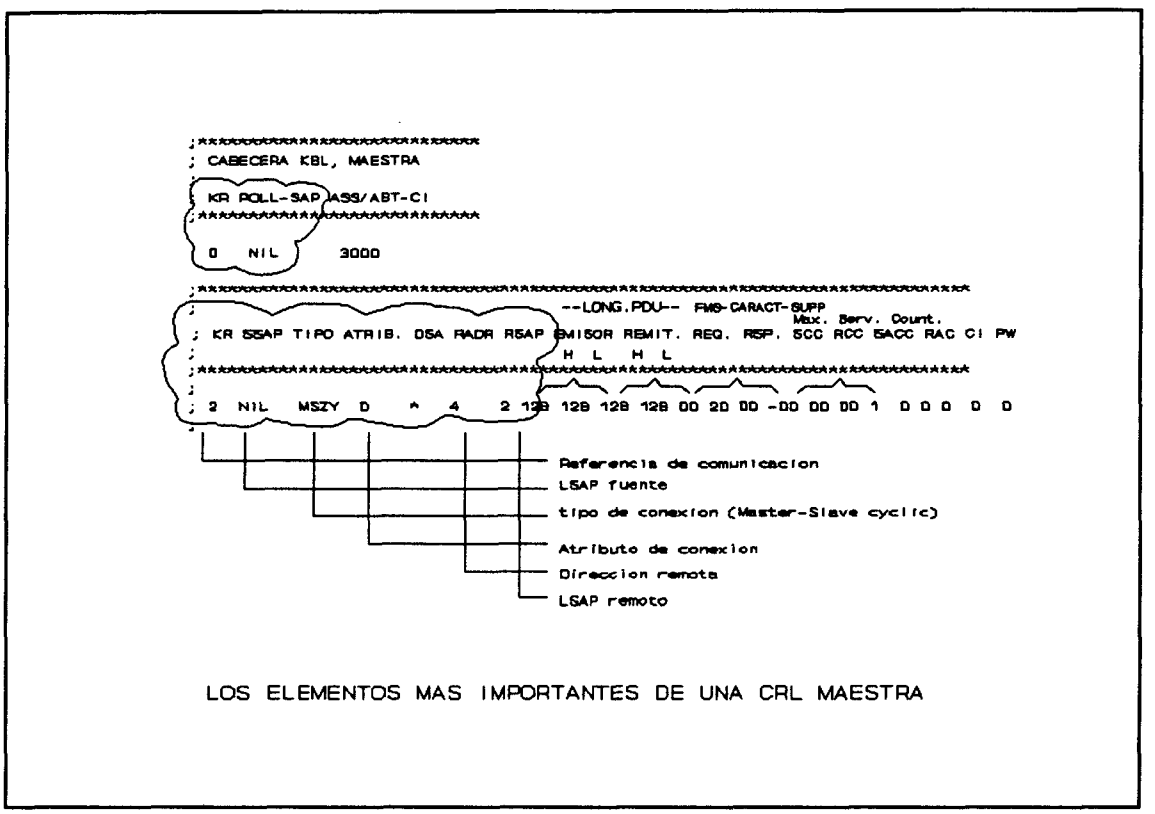
La Lista de Relación de Comunicación consta de dos partes, la FMS-CRL y la LLI-CRL, correspondientes a la información necesitada en las dos subcapas. Para los usuarios finales, por consiguiente, esta distinción no es importante; solamente deben ser determinadas por ellos algunas de las entradas especificadas en el estándar.

Una estación puede tener un máximo de 63 relaciones de comunicación. Más aún, tiene la capacidad de transmitir y recibir mensajes de difusión (broadcast) para lo que se usa una relación de comunicación específica. Además de estas relaciones de comunicación, que empiezan con la referencia de comunicación (CR) 1, existe una estrada bajo el índice CR 0, por ejemplo, la cabecera, que contiene información general así como la longitud de la tabla,... La referencia de comunicación es un código para la relación de comunicación que

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

solamente se usa en la estación. Antes de que la petición de servicio se transmita al bus, el programa de interfase de PROFIBUS vigila que la estación se direccione por esta referencia de comunicación específica en el CRL.

Se muestra en la figura 4.6. un ejemplo de las entradas en un CRL de una estación maestra. Se destacan las entradas más importantes para el usuario final:



LOS ELEMENTOS MAS IMPORTANTES DE UNA CRL MAESTRA

FIGURA 4.6.

* SSAP (el LSAP de la estación): Este es el número del punto de acceso al servicio local de la estación para la respectiva relación. El significado de este punto de acceso, también referidos como Punto de Acceso al Servicio de Enlace (LSAP), se explica más adelante.

* tipo de relación: Se explica en el punto 4.6.2.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* atributo de relación: Se explica en el punto 4.6.3.

* RADR (dirección remota): Dirección de estación de la otra estación.

* RSAP (LSAP remoto): Número de punto de acceso de la otra estación para la respectiva relación.

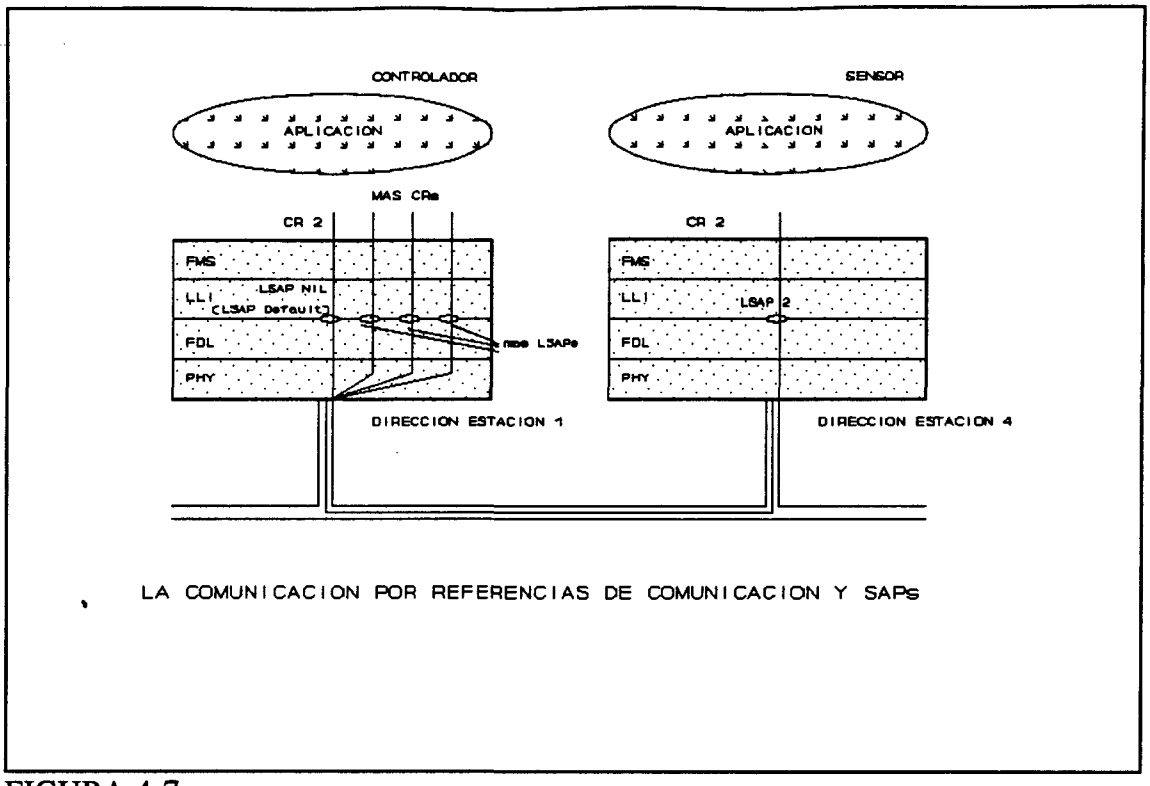
Otras entradas en el CRL también conciernen a la longitud de la trama (longitud de la PDU) y los servicios permitidos en esta relación de comunicación (características de FMS soportadas). Los otros parámetros no se explicarán ya que son irrelevantes para el primer contacto con PROFIBUS.

Para entender el significado de las entradas LSAP local y LSAP remota, consideraremos principalmente cómo una trama pasa desde la aplicación a través de las diferentes capas de la interfase PROFIBUS hasta el bus (figura 4.7.). El punto en el que un mensaje pasa a la interfase entre la capa 2 y 7 de PROFIBUS se llama punto de acceso al servicio de enlace (LSAP).

Una trama entrante del bus, que se direcciona para una estación, se debe pasar por un SAP específico, y la referencia de comunicación correspondiente a la respectiva aplicación. Si se compara la estación PROFIBUS con un edificio de apartamentos, que se identifican por el número de casa (=dirección de la estación), los diferentes SAPs son las puertas individuales de los apartamentos dentro del edificio.

Esta asignación de las LSAPs a las relaciones de comunicación es arbitraria, cada SAP, no obstante, se puede asignar solamente a una relación en un período de tiempo determinado. Los números permitidos para los SAPs yacen entre 0 y 63 con las restricciones siguientes:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



LA COMUNICACION POR REFERENCIAS DE COMUNICACION Y SAPs

FIGURA 4.7.

* El SAP número 1 es solamente para servicios de direccionamiento.
* El SAP número 63 se reserva para broadcast.

Además, existe un SAP implícito no direccionable. Este es una SAP que no se direcciona explícitamente, y de este modo permite tramas muy pequeñas. Todas las tramas que no tienen un número de SAP explícito para transmitir o recibir se asignan automáticamente a este SAP implícito.

Para las esclavas con 'conexiones abiertas', no se necesita la indicación de la dirección de la otra estación y el número de SAP. En conexiones abiertas se puede direccionar por cualquier número el SAP de la otra estación. A diferencia de una dirección específica o número de SAP de la estación, 'todas' se entran en el CRL.

El CRL, como se muestra en la figura 4.6., se puede configurar como un editor

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

normal de PC. Puede ser leído por cualquier controlador común de PC PROFIBUS. En otros sistemas PROFIBUS, como PLCs, el CRL puede programarse de una forma diferente. Por ejemplo, el CRL puede existir preconfigurado en el sistema que opera o como un bloque de función para la interfase.

4.6.2 SIN CONEXION Y ORIENTADO A CONEXION

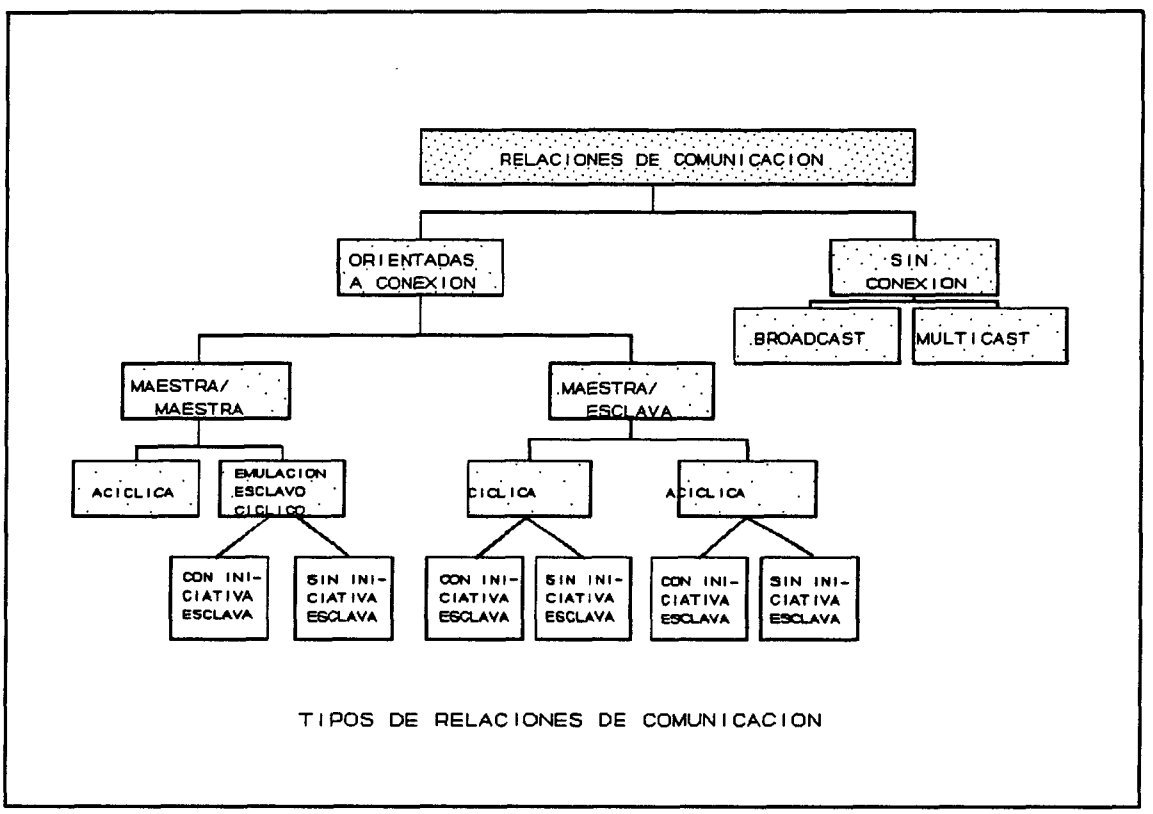


FIGURA 4.8.

PROFIBUS generalmente distingue entre relaciones de comunicación orientadas a conexión y las no orientadas a conexión (figura 4.8). Ya que en una relación no orientada a conexión no existe un canal lógico de vuelta, solamente se dispone de servicios no confirmados para multicast y broadcast para el usuario PROFIBUS.

Los servicios confirmados tienen que ser transmitidos en relaciones de comunicación

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

orientados a conexión, ya que, por razones de direccionamiento la respuesta de vuelta requiere un canal de regreso. Como oposición a la comunicación sin conexión, la conexión debe primeramente inicializarse en una fase de inicialización, seguido de una fase de transferencia de datos y, finalmente, la conexión debe ser liberada.

TABLA 4.1.

NOMBRE	TIPOS DE CONEXION
MSAC	MASTER-SLAVE ACYCLIC
MMAC	MASTER-MASTER ACYCLIC
MSAC_S1	MASTER-SLAVE ACYCLIC WITH SLAVE INITIATE
MSCY	MASTER-SLAVE CYCLIC
MSCY_S1	MASTER-SLAVE CYCLIC WITH SLAVE INITIATE
MULT	MULTICAST
BRCT	BROADCAST

NOMBRES PARA LOS TIPOS DE CONEXION PROFIBUS

La inicialización requiere mutuo acuerdo y transferencia de parámetros de cualidades, como el tamaño de los buffers de emisión y recepción y la clase de servicios soportados. El aumento de redundancia del protocolo, unido a la inicialización de una conexión, se justifica por el flujo de control y los mecanismos de seguridad de datos que se proporcionan después. Con PROFIBUS, el usuario puede contar con un gran número de tipos de relación de comunicación con diferentes características. La figura 4.8. relaciona las opciones de servicio soportadas. La tabla 4.1. lista los nombres abreviados para estos tipos de conexiones.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

4.6.3 CONEXIONES ABIERTAS Y DEFINIDAS

Además de las características arriba mencionadas, las relaciones de comunicación orientadas a conexión pueden ser clasificadas en relaciones definidas y abiertas. En la Lista de Relaciones de Comunicación, a su acceso se la llama atributo de conexión; para conexiones abiertas se usa una 'O', y para conexiones definidas una 'D'.

En conexiones definidas, la pareja de comunicación se determina en las fases de diseño e inicialización. En el establecimiento de la conexión, fases de transferencia de datos y la liberación de la conexión, las direcciones de la capa dos ya están acabadas. Así que la protección de acceso ofrecida por la capa 2 se puede ya ser proporcionada durante la fase de establecimiento de la conexión.

En conexiones abiertas, las direcciones de la capa 2 están enteras en la lista durante la fase de establecimiento de la conexión. Después una conexión abierta se comporta como una conexión definida.

4.6.4 TRANSFERENCIA DE DATOS CICLICA Y ACICLICA

Las relaciones maestro/esclavo implican principalmente equipos simples en el lado de la esclava los cuales se comportan como un servidor respecto a la funcionalidad de la capa 7, excepto para las Notificaciones de Eventos. En el lado de la maestra, una representación de proceso requiere transferencia de datos cíclica rápida con los sensores. Frecuentemente, los actuadores tienen que ser puestos al día cíclicamente para que la maestra detecte un fallo en la relación de comunicación (fallo de seguridad, control de redundancia,...). Los equipos sencillos de entrada/salida, lectores de código de barras y controladores sencillos tienen que ser disponibles para comunicar acíclicamente cuando el proceso de aplicación de la maestra está operando de modo espontáneo (cargando parámetros), o cuando la transferencia de datos tiene lugar en unos intervalos de tiempo grandes (cada 24 horas, inicialización de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

procesos,...). Para asegurar transporte de datos eficiente, PROFIBUS soporta ambos tráficos de datos, cíclico y acíclico.

Visto desde la interfase de la capa 7, el usuario no puede distinguir entre tráfico de datos cíclico y acíclico. Los requisitos de servicio de la capa 7/FMS se diseñan en relaciones de comunicación predefinidas para tráfico de datos cíclico o acíclico en la LLI y FDL, y se convierten en concordancia.

Para transferencia de datos cíclica, en la que la esclava es preguntado (polling) a intervalos cíclicos, la maestra tiene un SAP sencillo para cada par de estaciones, el Poll SAP. El número de SAP del Poll SAP debe determinarse cuando la maestra se configure. En la figura 4.6., el (NIL)SAP implícito se eligió como poll SAP, cuyo número es introducido en la cabecera. El NIL-SAP se usa para conexiones cíclicas.

4.6.5 REALIZACION DE TRANSFERENCIA DE DATOS CICLICOS EN LA LLI

Una petición de servicio se maneja con datos provenientes de la memoria imagen de datos que se dispone localmente en la LLI de la conexión. Una transferencia de datos cíclica a la capa 2 pone al día esta memoria de datos. Ofrece acceso rápido, por ejemplo, para medida de valores, por el uso de las características de respuesta inmediata del protocolo FDL, que usa servicios FDL, CSRD (emisión y recepción cíclica de datos) y SRD (emiten y reciben datos). Estos servicios cíclicos de la capa 2 se basan en un establecimiento de la Lista de Poll por el usuario. La maestra hace un polling a las esclavas en concordancia con el orden prescrito en la lista, y los valores actualizados se transfieren a la memoria imagen de datos de la maestra. Desde allí, el solicitador del servicio (cliente) puede obtener los valores independientemente de la actual transferencia de datos sobre el bus (figura 4.9).

Esta clase de transferencia de datos necesita solamente una inicialización de la transferencia de datos cíclicos por el usuario vía un servicio de lectura o escritura en la Capa

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

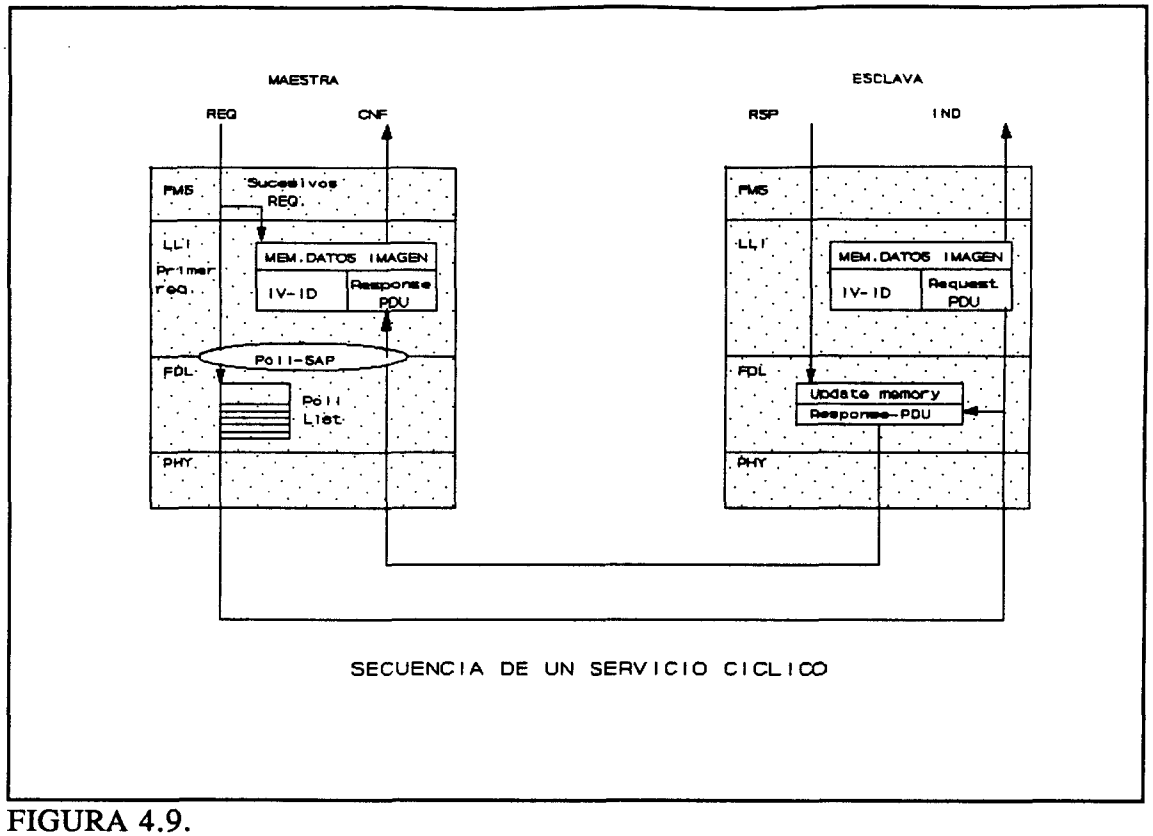


FIGURA 4.9.

de Aplicación. Esto activa el mecanismo de polling de la capa 2/FDL, y la transferencia de mensaje cíclico se realiza autonómicamente en la capa 2/FDL.

La transferencia de datos de la capa 2 es asíncrona para el solicitador del servicio.

4.6.6 INICIATIVA DE ESCLAVA

Los sistemas de bus de campo tradicional se caracterizan por una relación de comunicación maestro/esclavo. Una maestra es una estación de bus activa, lo que significa que puede usar el testigo exclusivamente. Una esclava es una estación de bus pasiva que solamente transfiere mensajes sobre el bus después de una petición de una maestra.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

La capa 7/FMS de PROFIBUS solamente puede distinguir entre estaciones de comunicación, no entre estaciones maestra y esclava con diferentes derechos de acceso al bus. En los procesos de aplicación, no obstante, los eventos importantes de usuario (p.e. alarmas) se transmiten a las estaciones por los servicios de la capa 7 que normalmente necesitan un derecho de acceso al bus. Esto significa que la estación debe ser una maestra que tenga un mecanismo de dirección del testigo.

Para este caso especial, PROFIBUS proporciona la opción del esclavo con iniciativa. Ofrece las ventajas de un esclavo, por ejemplo menor esfuerzo necesitado de implementación que la maestra, ya que no tiene dirección de testigo. La esclava con iniciativa necesita funcionalidad adicional: cuando la maestra le hace a la esclava un polling puede autonómicamente invocar servicios. Los servicios solamente permitidos son los servicios no confirmados de la capa 7/FMS.

4.7 SERVICIOS DE LA CAPA 7

Desde el punto de vista de procesos de aplicación, el sistema de comunicación proporciona servicios de la misma forma en mucho que un driver. Estos servicios consisten en proporcionar un número de funciones al usuario por lo que las tareas específicas pueden ser perfectas estrictamente. En PROFIBUS, esto permite comunicación transparente y abierta entre procesos de aplicación.

4.7.1 SERVICIOS CONFIRMADOS Y NO CONFIRMADOS

PROFIBUS distingue entre servicios confirmados y no confirmados. En los servicios no confirmados, un telegrama se transmite en una dirección, mientras en servicios confirmados se devuelve una confirmación a la transmisora. Por ejemplo, el servicio de Lectura pertenece a los servicios confirmados; el servidor proporciona datos sólo después de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

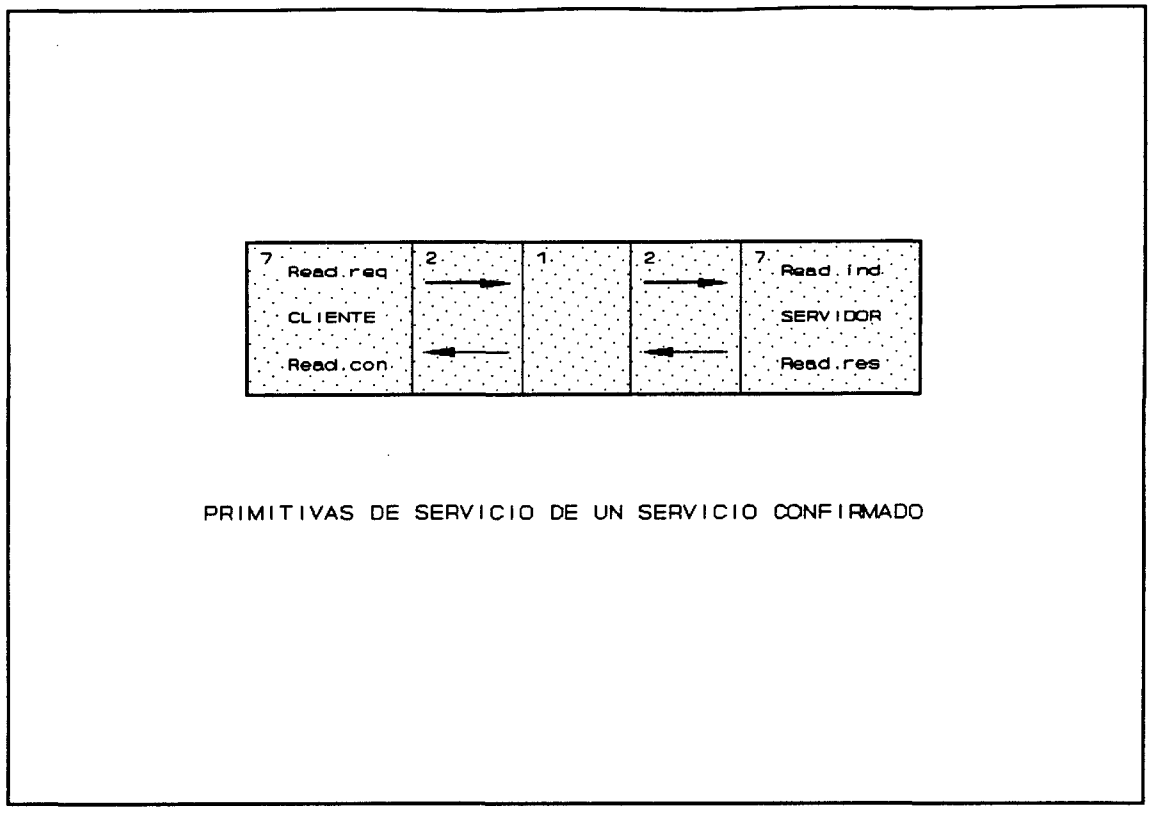


FIGURA 4.10.

una petición del cliente. Tal servicio de confirmación consta de dos partes, **las primitivas de servicio.** El ejemplo del servicio Read en la figura 4.10 ilustra los términos técnicos. La respuesta o confirmación contiene los datos de usuario.

En los servicios de Write, un paquete de datos se transmite junto con la petición de servicio mientras que la respuesta solamente devuelve la confirmación de que el servicio ha sido ejecutado con éxito.

Los servicios sin confirmación, que incluyen InformationReport y EventNotification, componen el otro grupo (Figura 4.11). En estos servicios, los datos de usuario se proporcionan directamente con el servicio sin petición o confirmación. El transmisor del servicio es el servidor, el destino el cliente. Si un controlador usa un servicio sin confirmación para mandar datos a un sensor (p.e. para sincronización), el controlador es el

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

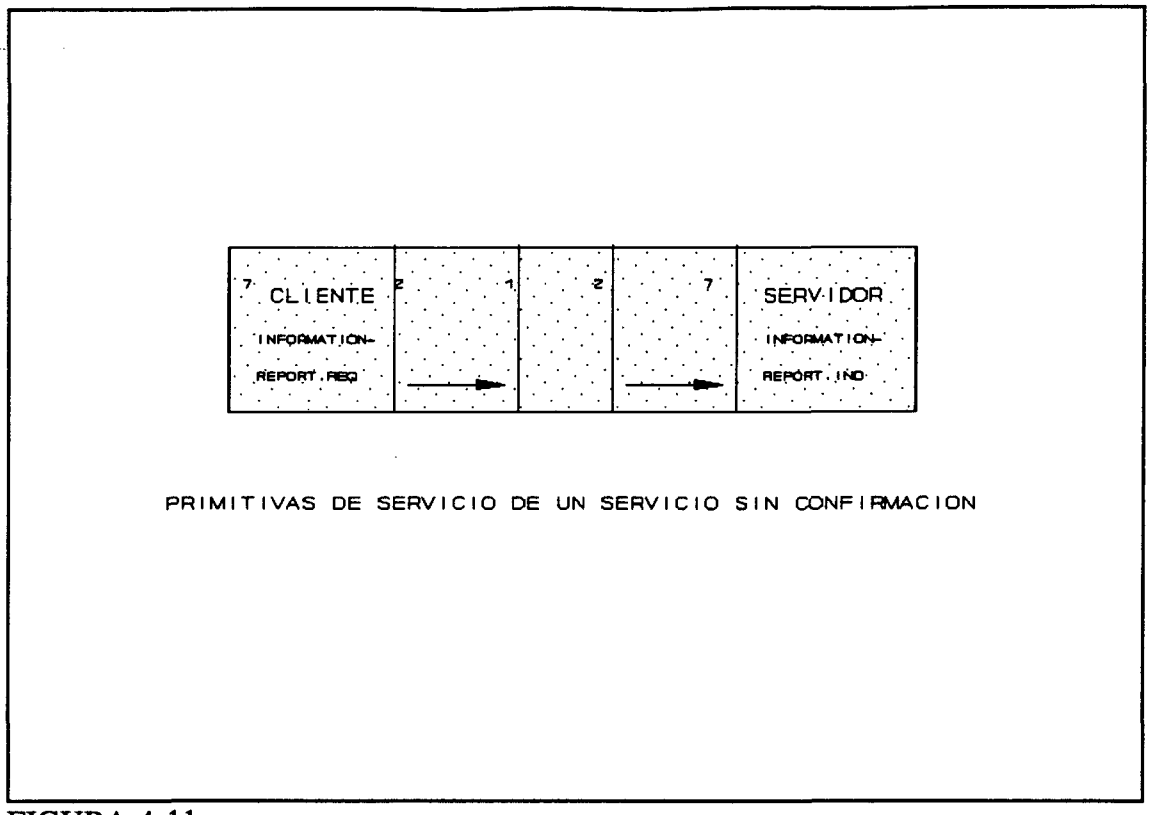


FIGURA 4.11.

servidor y el sensor el cliente. Los servicios sin confirmación pueden ser usados también para mensajes broadcast y multicast, en los que un transmisor direcciona todas o varias estaciones al mismo tiempo.

Un servicio sin confirmación se usa cuando un sensor necesita transmitir espontáneamente notificaciones de eventos. Se transmite con el servicio EventNotification. Este servicio se debe configurar en una conexión definida o abierta para lo que se usa una conexión con iniciativa de esclavo. Cuando se le hace un polling a una estación, en vez de transmitir ya los datos de usuario, se transmite al servicio requerido con la notificación de evento junto con la respuesta. El servicio requerido se almacena en buffers y se procesa cuando la estación reciba un polling la próxima vez.

Los servicios más importantes se muestran en la tabla 4.2. La forma en que los

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

TABLA 4.2.

SERVICIOS OBLIGATORIOS
- INITIATE: ESTABLECIMIENTO DE LA CONEXION
- ABORT: ABORTO DE LA CONEXION
- REJECT: RECHAZO DE UN SERVICIO INCORRECTO
- STATE: LECTURA DEL ESTADO DEL EQUIPO/USUARIO
- IDENTIFY: LECTURA DEL NUMERO DE IDENTIFICACION DE LA ESTACION
- GETOD: LECTURA DEL OD
SERVICIOS ADICIONALES SOPORTADOS
- READ: LEER VALOR
- WRITE: ESCRIBIR VALOR
- PHYSREAD: LECTURA SOLO CON DIRECCIONES DE EQUIPOS ESPECIFICOS
- PHYSWRITE: ESCRITURA SOLO CON DIRECCIONES DE EQUIPOS ESPECIFICOS
- INFORMATIONREPORT: TRANSMISION Y RECEPCION DE TRAMAS
- DOWNLOAD SERVICES: TRANSMISION DE BLOQUES DE DATOS
ADEMAS, SE PUEDE TRANSMITIR Y CONFIRMAR NOTIFICACIONES DE EVENTOS
- EVENTNOTIFICATION
- ACKNOWLEDGEEVENTNOTIFICATION
- ALTEREVENTMONITORING

servicios se diseñan en los servicios de la capa LLI y la capa 2 son irrelevantes para el usuario final. Así que será suficiente un ejemplo de servicio InformationReport (figura 4.12).

Cuando se solicita el servicio InformationReport por el programa de aplicación del maestro, el índice del objeto y la referencia de comunicación deben ser también indicados. Cuando la trama pasa a través de las capas desde lo alto a la base, cada capa interpreta la trama llegante como su 'datos de usuario' y añade su propio código de identificación al paquete. Esto hace que la trama vaya aumentando, pero permite fácil identificación en el otro lado. Allí, la trama llegante es 'desempaquetada' capa a capa y asignada a la localización correspondiente.

Para el servicio de Read, el que es quizás es más usado, el procesamiento de trama en las capas de una estaciones un poco más complicado. Particularmente en la solución 8051,

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

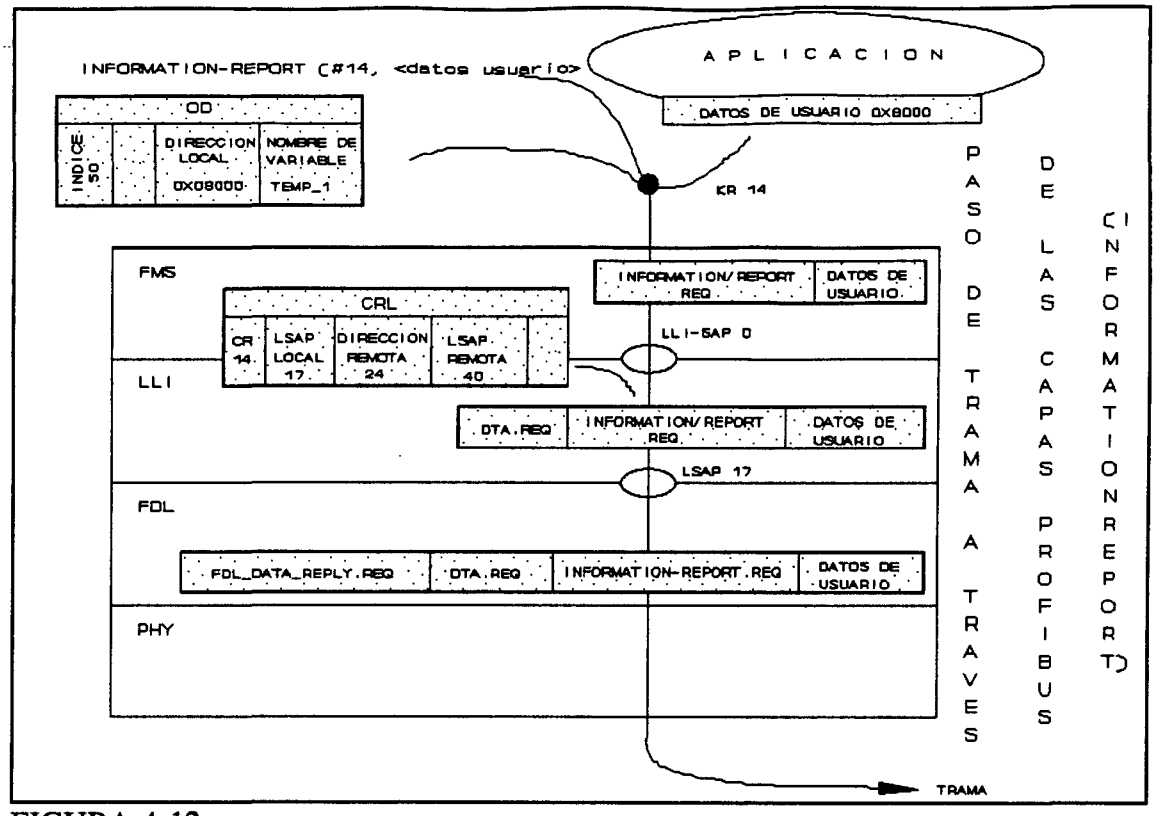


FIGURA 4.12.

que se explicará en el capítulo 4, las tramas no se procesan capa a capa , pero pasan directamente a la aplicación por razones de eficiencia. Allí la respuesta es generada inmediatamente y devuelta en el mismo ciclo. A esto se le denomina respuesta inmediata.

En el caso estándar, sin embargo, los datos de usuario se transmiten en el siguiente ciclo de trama o incluso más tarde. Esto por supuesto retrasa la respuesta.

Estas diferentes clases de servicios se basan en protocolos de comunicación para relaciones de comunicación de no conexiones y a las orientadas a conexiones. El protocolo para relaciones orientadas a conexiones necesita una conexión antes de empezar la comunicación. Después que la conexión ha sido establecida con éxito, tiene lugar la transferencia de datos. Para acabar la fase de transferencia de datos, se aborta la conexión mediante el servicio Abort. La figura 4.13 muestra un ejemplo de la ejecución de un servicio

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

confirmado en una conexión maestra-maestra.

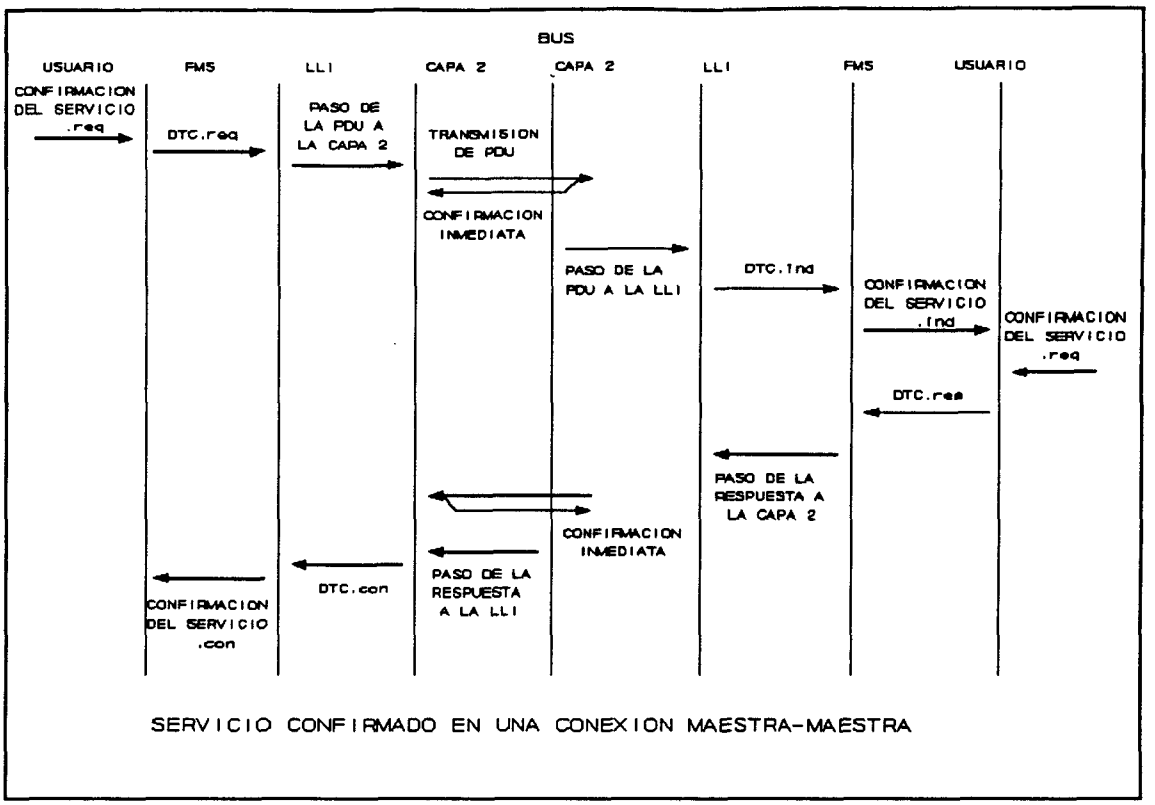


FIGURA 4.13.

4.7.2 GRUPOS DE SERVICIO

Los servicios de comunicación PROFIBUS se pueden clasificar en los siguientes tres grupos funcionales:

* servicios de aplicación
* servicios de administración
* servicios de manejo de red

El primer grupo incluye todos los servicios que dan acceso a los objetos de comunicación de un proceso de aplicación, que incluye los servicios Read para lectura o

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Write para escritura de variables o Stop para parar un programa.

Los servicios del segundo grupo son iniciados por el usuario también, pero solamente concierne a objetos o relaciones de comunicación de Equipos Virtuales de Campo. Un objeto, por ejemplo, es la descripción de un objeto de comunicación en un Diccionario Objeto. Un servicio que se puede aplicar a este objeto es GetOD; lee una entrada en el Diccionario Objeto.

Para inicializar y abortar una conexión basada en una relación de comunicación se usan los servicios Initiate y Abort. Los servicios de manejo de red mantienen las funciones técnicas de una red. Existen, por ejemplo, servicios que determinan el tamaño de buffers de mensaje, capas de reset, cargan y modifican la lista de relaciones de comunicación posibles. El manejo de red cubre todas las capas PROFIBUS. La aplicación y servicios de administración se dividen en subgrupos.

Cada equipo PROFIBUS no soporta todos los grupos de servicio. Cuando se establece la conexión, las estaciones comunicantes informan a cada otra de la disponibilidad de los servicios soportados. La tabla 4.3 muestra los objetos asociados a los diferentes servicios.

T A B L A 4.3. ASIGNACION DE LOS OBJETOS A SERVICIOS

SERVICIOS	OBJETOS	
Initiate		
Abort		T
Reject		
Status	F	T
UnsolicitedStatus	F	
Identify	F	T

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

GetOD	O	H	T						Dt	Ds	
InitiatePutOD	O		T								
PutOD	O	H	T	D	P		V	VL	Dt	Ds	E
TerminatePutOD	O		T								
InitiateDownloadSequence	O	H	T	D							
DownloadSegment	O	H	T	D							
TerminateDownloadSequence	O	H	T	D							
RequestDomainDownload	O	H	T	D							
InitiateUploadSequence	O	H	T	D							
UploadSegment	O	H	T	D							
TerminateUploadSequence	O	H	T	D							
RequestDomainUpload	O	H	T	D							
CreateProgramInvocation	O	H	T	D	P						
DeleteProgramInvocation	O	H	T	D	P						
Start			T		P						
Stop			T		P						
Resume			T		P						
Reset			T		P						
Kill			T		P						
Read			T				V	VL			
Write			T				V	VL			
ReadWithType			T				V	VL			
WriteWithType			T				V	VL			
PhysRead			T			Y					
PhysWrite			T			Y					
InformationReport							V	VL			
InformationReportWithType							V	VL			
DefineVariableList	O	H	T					VL			
DeleteVariableList	O	H	T					VL			
EventNotification											E

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

EventNotificationWithType			E
AcknowledgeEventNotification		T	E
AlterEventConditionMonitoring	O	T	E

Donde:

F: VFD	Y: Objeto de acceso físico
O:Diccionario Objeto	H: Descripción Objeto-OD
V: Variable Simple, array,registro	VL: Lista variable
T: Objeto de transacción	Dt: Tipo de dato
D: Dominio	P: Invocación de programa
Ds: Descripción estructura tipo dato	E: Evento

4.7.3 VISTA GENERAL DE LOS SERVICIOS PROFIBUS

Servicios de Aplicación

Cada grupo de servicio de aplicación se basa en un modelo abstracto que se explicará brevemente, seguido por una lista de todos los servicios soportados. En el punto 4.8. se discutirán de detalle todos los servicios soportados por PROFIBUS.

PROFIBUS reconoce los siguientes grupos de servicio de aplicación:

* Acceso a variable
* Acceso a dominio
* Invocación de programa

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* Manejo de eventos

El grupo de servicio **Acceso a Variable** proporciona servicios que permiten acceder a variables simples, registros, arrays y listas de variables. Los objetos de comunicación del tipo de lista de variable pueden ser definidos o borrados dinámicamente usando estos servicios.

Los siguientes servicios pertenecen a este grupo:

* Read
* Write
* ReadWithType
* WriteWithType
* PhysRead
* PhysWrite
* InformationReport
* InformationReportWithType
* DefineVariableList
* DeleteVariableList

Como se muestra más adelante, el grupo de servicio de **Acceso a Dominio** incluye servicios para carga hacia abajo y hacia arriba de áreas de memoria conectadas lógicamente. Durante la carga de proceso, la aplicación debe dividir grandes cantidades de datos en segmentos.

Para asegurar consistencia de los dominios cuando los segmentos están cargados, las operaciones de carga son iniciadas siempre por el servicio InitiateDownloadSequence (InitiateUploadSequence), y terminadas por el servicio TerminateDownloadSequence (TerminateUploadSequence).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Los siguientes servicios pertenecen a este grupo:

* InitiateDownloadSequence
* DownloadSegment
* TerminateDownloadSequence
* RequestDomainDownload
* InitiateUploadSequence
* UploadSegment
* TerminateUploadSequence
* RequestDomainUpload

El modelo de **Invocación de Programa** proporciona los siguientes servicios, que incluyen la combinación de varios dominios en un programa operable, y empiezan, paran y borran un programa:

* CreateProgramInvocation
* DeleteProgramInvocation
* Start
* Stop
* Resume
* Reset
* Kill

El **Manejo de Eventos** proporciona servicios que transmiten mensajes importantes del tipo objeto evento desde una estación a otra. El proceso de aplicación decide si un evento se activa basado en condiciones predefinidas. Cuando se conoce una condición, el proceso de aplicación despierta el servicio de notificación de eventos. Por AlterEventConditionMonitoring, la transmisión del evento sobre el bus puede ser capacitado o no en la estación activadora. El servicio de AcknowledgeEventNotification confirma

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

explícitamente una notificación de evento. Los siguientes servicios pertenecen a este grupo:

* EventNotification
* EventNotificationWithType
* AcknowledgeEventNotification
* AlterEventConditionMonitoring

Servicios de Administración

Cada grupo de servicio de administración se basa en un modelo abstracto, explicado brevemente aquí y seguido por una lista de servicios soportados:

* Soporte VFD
* Manejo de OD
* Manejo de contexto

El **Equipo Virtual de Campo (VFD)** es un modelo abstracto que describe los datos y conducta de un sistema de automatización, supervisado por un usuario remoto PROFIBUS. El modelo se basa en el objeto VFD que incluye todos los objetos y descripciones de objeto de objetos de comunicación accesibles por el usuario vía servicios. El Diccionario Objeto, por ejemplo, es un objeto de VFD.

Normalmente, un equipo posee sólo un VFD, pero es posible realizar más que uno por equipo. Los VFDs se direccionan sobre relaciones de comunicación. El soporte VFD proporciona servicios de estado e identificación, por lo que se transmite la información de estado de equipo específica y el medio VFD aplicable a este relación de comunicación.

Este grupo incluye los siguientes servicios:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* Status
* UnsolicitedStatus
* Identify

El manejo de Diccionario Objeto proporciona servicios para la lectura y escritura de los Diccionarios Objeto Fuentes de las estaciones de comunicación de VFDs. Cuando las descripciones de objetos de objetos usados normalmente se modifican se modifican en un Diccionario Objeto fuente, las estaciones concernientes deben ser informadas. Esto puede ser hecho abortando las conexiones afectadas por esta modificación.

Además las descripciones de objeto normalmente no afectan la consistencia del Diccionario Objeto así que tales conexiones pueden ser mantenidas. El proceso de aplicación decide la permanencia, ya que un Diccionario Objeto con información de aplicación adicional no se almacena en el sistema de comunicación por si mismo, pero sí en la Interfase de Capa de Aplicación (ALI) como parte de la aplicación.

Los servicios de OD incluyen:

* GetOD
* InitiatePutOD
* PutOD
* TerminatePutOD

Los **servicios de manejo de contexto** inicializan las conexiones, libera o aborta conexiones ocupadas, y rechaza peticiones de servicio ilegales. El contexto necesario por esto incluye todos los acuerdos hechos por las estaciones en una relación de comunicación. Esta información se almacena en la descripción del Diccionario Objeto y en la Lista de Relación de Comunicación de la capa 7/FMS.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Los siguientes servicios pertenecen a este grupo:

* Initiate
* Abort
* Reject

4.7.4 SERVICIOS DE ASIGNACION A ESTACIONES MAESTRA Y ESCLAVA

El modelo de referencia ISO/OSI incluye básicamente las capas desde la 1 a la 4 orientadas a red y desde la 5 a la 7 orientadas a la aplicación. La capa 7 sería independiente del método de acceso al bus de la capa 2. Esto se cumple también para PROFIBUS. En la capa 7, no se puede distinguir si la estación es activa (maestra) o pasiva (esclava).

La diferencia entre ambas interfases es que ciertos servicios solamente tienen sentido con una maestra, ya que sólo la maestra tiene derecho de acceso al bus y puede ser activada como cliente. Mientras todos los servicios en la capa 7 pueden ser implementados como un esclavo, el diseño en la capa 2 es imposible. Si una petición para un servicio no ejecutable llega, este servicio debe ser rechazado.

La tabla 4.4 da una visión de los servicios que deben ser ejecutados por un esclavo, y por una maestra, y cuales son opcionales. Aquí, distinguimos entre capacidades de cliente y servidor. Además indicamos si el servicio es confirmado o no.

T A B L A 4.4. SERVICIOS ASIGNADOS A MAESTRA Y ESCLAVA

SERVICIOS	ESCLAVA		MAESTRA		CFD/
	CLI	SERV	CLI	SERV	UNC
Initiate	-	M	O	M	CFD

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Abort	M	M	M	M	UNC
Reject	M	M	M	M	UNC
Status	-	M	O	M	CFD
UnsolicitedStatus	O	O	O	O	UNC
Identify	-	M	O	M	CFD
GetOD(forma corta)	-	M	O	M	CFD
GetOD(forma larga)	-	O	O	O	CFD
InitiatePutOD	-	O	O	O	CFD
PutOD	-	O	O	O	CFD
TerminatePutOD	-	O	O	O	CFD
InitiateDownloadSeq.	-	-	O	O	CFD
DownloadSegment	-	-	O	O	CFD
TerminateDownloadSeq.	-	-	O	O	CFD
RequestDomainDownload	-	-	O	O	CFD
InitiateUploadSeq.	-	O	O	O	CFD
UploadSegment	-	O	O	O	CFD
TerminateUploadSeq.	-	O	O	O	CFD
RequestDomainUpload	-	-	O	O	CFD
CreateProgramInvocat.	-	O	O	O	CFD
DeleteProgramInvocation	-	O	O	O	CFD
Start	-	O	O	O	CFD
Stop	-	O	O	O	CFD
Resume	-	O	O	O	CFD
Reset	-	O	O	O	CFD
Kill	-	O	O	O	CFD
Read	-	O	O	O	CFD
Write	-	O	O	O	CFD
ReadWithType	-	O	O	O	CFD
WriteWithType	-	O	O	O	CFD

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

PhysRead	-	o	o	o	CFD
PhysWrite	-	o	o	o	CFD
InformationReport	o	o	o	o	UNC
InformationReportWithType	o	o	o	o	UNC
DefineVariableList	-	o	o	o	CFD
DeleteVariableList	-	o	o	o	CFD
EventNotification	o	o	o	o	UNC
EventNotificationWithType	o	o	o	o	UNC
AcknowledgeEventNotification	-	o	o	o	CFD
AlterEventConditionMonitoring	-	o	o	o	CFD

Donde:

Cli: Cliente (Pedidor del servicio) Serv: Servidor (respondedor del servicio)

cfd: Servicio confirmado unc: servicio sin confirmación

M: Servicio obligatorio O: Opcional

4.7.5 MECANISMOS DE PROTECCION DE ACCESO

En la tecnología de comunicación, la protección de acceso para los sistemas de automatización es necesaria en muchos caso, e incluso importante para la seguridad de una compañía y sus equipos. En PROFIBUS, desarrollamos un mecanismo de protección de acceso diseñado para la seguridad requerida por los usuarios. PROFIBUS conoce los siguientes mecanismos de protección de acceso:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* protección de acceso para objetos
* protección de acceso para puntos terminales de comunicación

Con PROFIBUS, el usuario puede proteger los objetos de comunicación de accesos indeseados. Por otro lado, el acceso a objetos importantes en el nivel de campo, como enchufes, actuadores y parametrizadores, es solamente posible al personal autorizado. Además, las aplicaciones de tiempo crítico no se sobrecargarían. Como un arreglo, se concibió una solución de control de acceso, que no previene actuaciones criminales, pero protege de daños indeseados. Esta protección de acceso es opcional y puede ser desconectada.

La protección de objetos de comunicación puede ser implementada en la fase de diseño de Diccionario Objeto. La protección de los objetos concierne a:

* permisos de acceso dados a estaciones autorizadas. PROFIBUS reconoce grupos de aplicación y claves (passwords).
* servicios permitidos. Por ejemplo solamente Read con un objeto del tipo variable simple, o solamente Start con invocación de programas.

En relaciones orientadas a conexión y las de sin conexión, un punto terminal de comunicación se puede proteger durante el diseño de la Lista de Relación de Comunicación. El acceso a estos puntos terminales es dado solamente a una pareja específica, por ejemplo, la estación de control. Este control ya es válido para la fase de inicialización.

La elección de la pareja durante la inicialización de conexión es posible solamente con puntos terminales de conexión abierta. El acceso a estos puntos terminales se protege sólo después de que la conexión ha sido establecida con éxito. Así, la conexión que no ha sido todavía establecida, y consecuentemente ocupada, puede ser usada por varias estaciones (no obstante, solamente uno a la vez) mientras esta conexión sigue un servicio de aborto libera

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

a las otras estaciones.

4.8 SERVICIOS FMS Y SUS PARAMETROS

En este punto, se listarán todos los servicios FMS con sus parámetros de petición y de respuesta. Dos parámetros han desaparecido:

* dirección de la pareja de comunicación (Referencia de Comunicación, CR)
* número requerido (Invocación-ID)

Ambos parámetros tienen independencia del servicio en común. No se pueden listar en una descripción de servicio porque los servicios direccionan solamente objetos, nunca parejas de comunicación o las peticiones transmitidas por ellos. Cuando una petición de servicio se transmite en la interfase hacia la capa 7, no obstante, deben proporcionarse estos parámetros independientes del servicio.

El gran número de servicios PROFIBUS proporciona al usuario de una interfase conveniente para muchas áreas de aplicación. No obstante, el alcance completo de los servicios PROFIBUS no se necesita en todos los equipos y áreas. El campo de acción de los servicios puede ser delimitado por un perfil de aplicación específica, por ejemplo.

Los perfiles deben al menos incluir los siguientes servicios para conseguir una comunicación adecuada:

* Initiate
* Abort
* Reject
* Status
* Identify

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* GetOD

Para un esclavo pequeño, esto significa que debe proporcionar todos los servicios arriba mencionados solamente como un servidor, excepto el servicio reject. Es especialmente importante para PROFIBUS, ya que el campo de acción de la capa 7 debe conservarse pequeño para la implementación hardware y razones de funcionalidad. Más aún, los servicios ofrecidos por PROFIBUS tienen los siguientes criterios:

* Los servicios ofrecidos se adaptan al área de aplicación del bus de campo.
* PROFIBUS garantiza la persistencia de MAP (por ejemplo, MMS) en sensatez.
* Se deben satisfacer los requerimientos estritos de tiempo
* La concepción de los servicios es orientada a objeto.

A través de los servicios FMS, un proceso de aplicación es capaz de usar la funcionalidad de un proceso de aplicación remoto. Los servicios FMS se transmiten al proceso remoto como mensajes, unidades de datos de protocolo (PDUs). El proceso de aplicación debe proporcionar los parámetros necesarios para el servicio.

4.8.1 SOPORTE VFD

El Equipo Virtual de Campo (VFD) es un modelo abstracto que describe los datos y conducta del sistema de automatización desde el punto de vista de la pareja de comunicación. EL modelo VFD se basa en el objeto VFD. Contiene todos los objetos y descripciones de objeto que un usuario puede direccionar a través de los servicios. Las descripciones de objeto se almacenan en el diccionario objeto (OD). Cada VFD tiene un OD.

Status

Por medio de este servicio, el equipo/usuario puede leer el estado del VFD.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

UnsolicitedStatus

El proceso de aplicación usa este servicio para transmisión espontánea de estado de equipo y usuario. Un esclavo con capacidad de procesamiento autónomo puede usarlo como un servicio sin confirmación. También es posible la transmisión broadcast y multicast.

Todos los servicios sin confirmación permiten especificación de nivel de prioridad.

Identify

Este servicio lee información para identificar a un VFD. El usuario determina el nombre y dirección del equipo cuando inicializa la relación de comunicación usando 'initiate' y no 'identify'.

4.8.2 DIRECCION DE OD

Los servicios OD conciernen a las descripciones de los objetos del dominio de objeto, invocación de programas, variable simple, array, registro, lista de variable, evento, diccionario objeto, tipo de dato y descripción de estructura de tipo de dato.

Las descripciones de objeto se pueden leer por el servicio GetOD y escribirse con el servicio PutOD. El número de descripciones de objeto que se pueden transmitir en un servicio GetOD o PutOD depende de su longitud y de la longitud máxima del paquete (longitud de PDU). Esta longitud máxima de PDU depende del tamaño de los buffers de recepción y envío configurados en la Lista de Relación de Comunicación.

Los objetos que se escriben en el Diccionario Objeto se deben inicializar con un servicio InitiatePutOD y terminados con un servicio TerminatePutOD.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

GetOD

Mediante este servicio, se pueden leer una o más descripciones de objeto.

Para leer una descripción de objeto sencilla, se debe especificar el índice o el nombre. Para varias o todas las lecturas de las descripciones de objeto, se debe especificar el índice del primer descriptor de objeto deseado (StartIndex). Para leer totalmente el OD, normalmente se necesita usar varias veces el servicio GetOD.

Servicios PutOd

Los servicios PutOD meten una o más descripciones de objeto en el Diccionario Objeto. Antes de modificar el OD, el usuario puede ejecutar algunas actividades como llevar el proceso a un estado de seguridad.

Para todas las descripciones en un Diccionario Objeto, el usuario debe mantener la secuencia siguiente de servicios: InitiatePutOD, uno o más servicios PutOD y TerminatePutOD.

Distinguimos entre:

* Add OD. Normalmente significa añadir nuevas descripciones de objeto en el OD. Aquí existe entradas que no se modifican. Las descripciones de Objeto que se entraron localmente por la estación se pueden borrar si los equipos correspondientes no existen mucho tiempo en el sistema de comunicación.

* ModifyOD. Esta operación destruye la consistente perspectiva de la estación de sus objetos. Puede existir inconsistencias entre el OD fuente y el OD Remoto. ModifyOD se puede dividir en cargas nuevas y recargas parciales. Mediante el servicio TerminatePutOD,

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

se introducen los objetos de comunicación nuevos.

PutOD

Por este servicio una o más descripciones de objeto se pueden escribir en un Diccionario Objeto.

La descripción de la lista de objeto: La lista de objeto describe los que se introducen en el OD.

Con este servicio PutOD, se pueden borrar las descripciones de objeto. En este caso se manda una descripción de objeto vacía con la siguiente estructura:

. Indice de la descripción de objeto a borrar
. Código Objeto: NULL

TerminatePutOD

Este servicio termina la carga del Diccionario Objeto. Finalmente se generan los objetos de comunicación por PROFIBUS. Si ocurriera un error en la generación de los objetos, la versión original del OD no se restaura.

4.8.3 DIRECCION DE CONTEXTO

El contexto incluye todos los acuerdos concernientes a la relación de comunicación. Los servicios de manejo de contexto inician y liberan las conexiones. Los servicios ilegales se pueden rechazar. Las descripciones de objeto de OD como la Lista de Relación de Comunicación, almacenan la información necesaria para estas operaciones.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Cada conexión tiene sus objetos de transacción almacenados en el VFD. Sus números están en concordancia con el número aceptable de servicios paralelos. Un objeto de transacción en un servidor almacena la petición de servicio entrante hasta la ejecución por el proceso de aplicación. Este objeto supervisa la propia ejecución en un uso correcto y ocasional de operaciones, como peticiones de lectura que han sido transmitidas por VFD para los equipos reales de campo (o específicamente, al proceso de aplicación de un equipo real de campo).

Initiate

Este servicio inicia una conexión entre dos estaciones. Esta relación se debe almacenar en la Lista de Relaciones de Comunicación antes de la ejecución del servicio 'initiate'. Durante la inicialización, la estaciones acuerdan la longitud de la PDU (longitud del mensaje) y avisan una a la otra de los servicios que pueden ser llamados en esta conexión. Además,transmiten los números de versión actuales de los OD.

La petición de servicio debe proporcionar los primeros seis parámetros del servicio 'initiate' en la interfase ALI/FMS. Los otros parámetros se generan por la FMS (Fieldbus Message Specification).

Abort

Este servicio libera una relación de comunicación existente entre dos estaciones. Pueden abortar la conexión tanto el servidor como el cliente.

Reject

Mediante este servicio, la capa FMS rechaza las PDUs ilegales. El servicio de rechazo se genera cuando la petición de servicio hecha por el cliente no se puede manejar

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

por el VFD o por el proceso de comunicación. Los mensajes ilegales, por ejemplo, contienen una petición de servicio no permitida o inejecutable. Un rechazo se genera también cuando un mensaje es demasiado largo y no cabe en el buffer correspondiente. La ejecución errónea del servicio de lectura debido a, por ejemplo, especificación de índice erróneo, solamente conduce a una respuesta negativa.

4.8.4 DIRECCION DE DOMINIO

Un dominio es un área de memoria conectada que puede contener programas o datos. El tamaño de esta área de memoria se determina en la descripción de objeto. El tipo de datos de un dominio es siempre un string de octetos. La ejecución de un servicio de carga hacia arriba o hacia abajo se permite por dominio pero no a la misma vez.

Servicios de carga hacia abajo (DOWNLOAD)

Este servicio transmite datos desde el cliente al servidor. El ejemplo mostrado en la figura 4.14, ilustra claramente que mientras se ejecuta estos

servicios, los papeles de cliente y servidor cambian. Quiere decir que la comunicación en el bus se debe lincar como una estación activa para ejecutar un 'download'. Los esclavos nunca pueden proporcionar estos servicios. Esta tarea está en mantenimiento con la carga hacia abajo de MMS, mientras en MAP/MMS no se distingue entre estaciones pasivas y activas.

InitiateDownloadSequence

Inicia la carga hacia abajo e informa al servidor el índice o nombre de dominio que para cargarse.

DownloadSegment

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

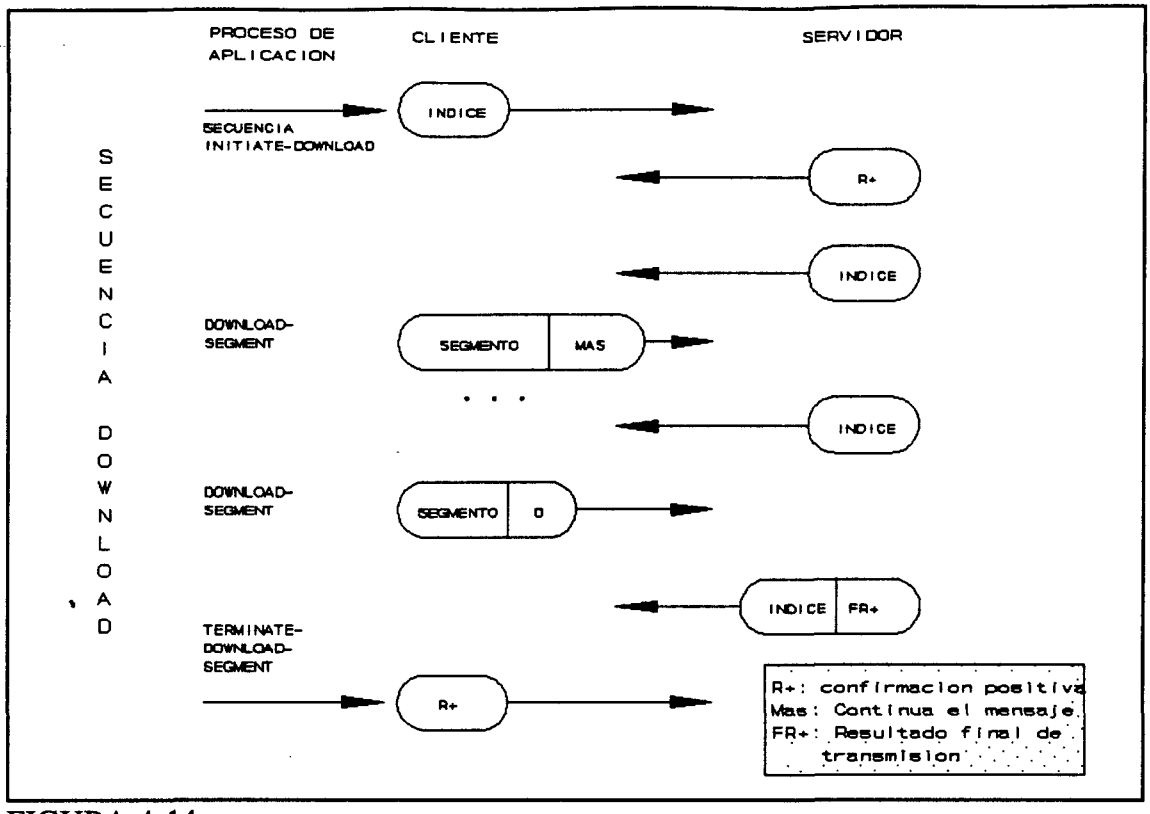


FIGURA 4.14.

Realiza la transferencia actual de datos. La PDU de respuesta contiene los datos pedidos.

TerminateDownloadSequence

Termina la secuencia de carga.

RequestDomainDownload

Mediante este servicio, el proceso de aplicación informa al proceso de aplicación remoto que transmite ciertos datos (en este caso son dominios que pueden contener, p.e., programas o parámetros). El proceso remoto transmite solamente la respuesta positiva a este servicio cuando la operación de carga ha sido ejecutada por una secuencia de carga hacia

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

abajo. La secuencia de carga debe ser iniciada por la estación remota.

Upload Services

Transmite datos desde el servidor al cliente.

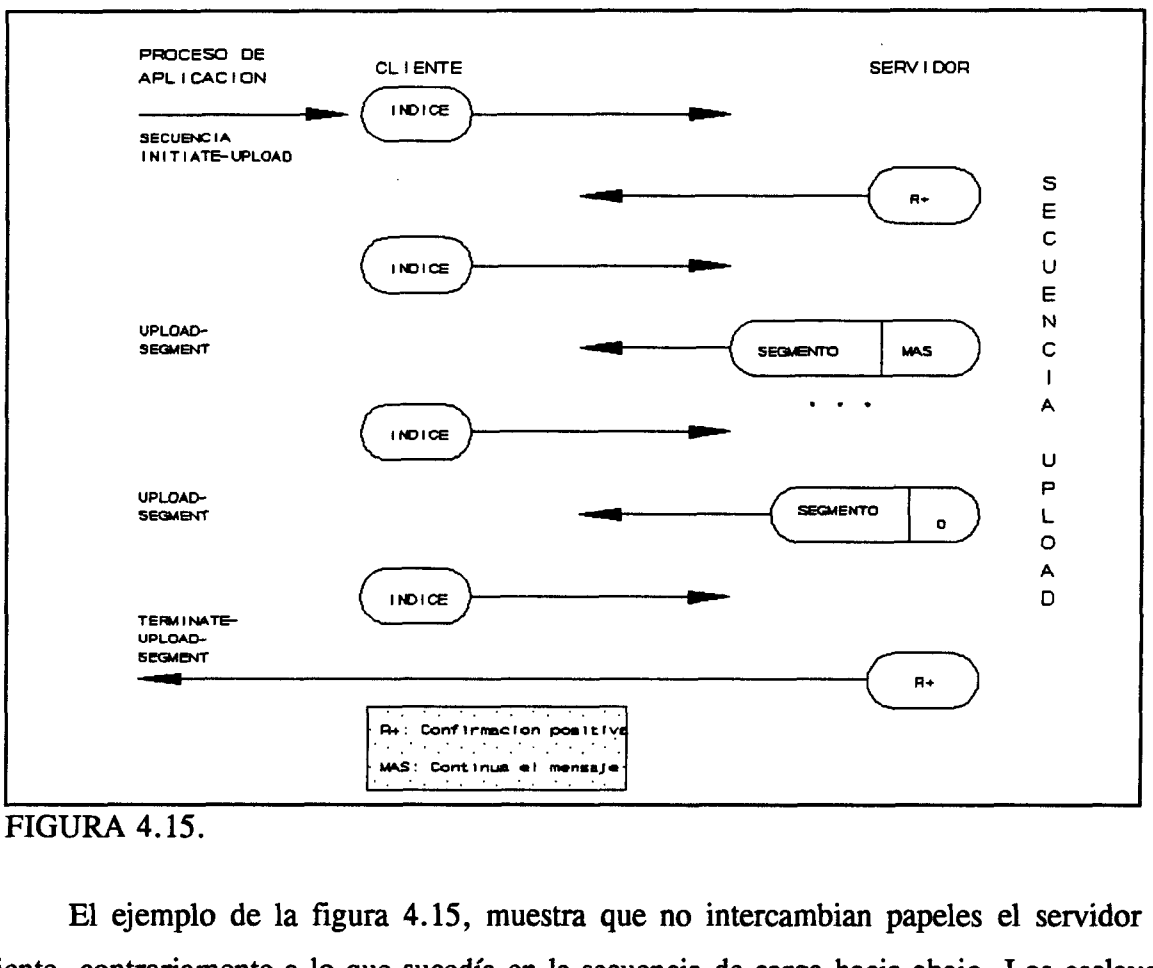


FIGURA 4.15.

El ejemplo de la figura 4.15, muestra que no intercambian papeles el servidor y cliente, contrariamente a lo que sucedía en la secuencia de carga hacia abajo. Los esclavos proporcionan un servicio de carga hacia arriba como los servidores.

InitiateUploadSequence

El cliente puede iniciar la carga mediante este servicio.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

UploadSegment

Transfiere los datos desde el servidor al cliente.

TerminateUploadSequence

Termina la carga hacia arriba.

RequestDomainUpload

Por medio de este servicio, el proceso de aplicación informa al proceso de aplicación remoto que quiere transmitir ciertos datos (en este caso, los dominios que puedan, p.e., contener programas o parámetros). El proceso remoto transmite solamente la respuesta positiva a este servicio cuando la carga total ha sido ejecutada. Debe iniciar la carga la estación remota.

4.8.5 SERVICIOS DE INVOCACION DE PROGRAMAS

CreateProgramInvocation

Combina los dominios (p.e., programas, áreas de datos) que se han definido en el OD, en un programa y los define en la fase operacional en la invocación de programa dinámica-OD (DP-OD). Este programa puede ser entonces direccionado por una dirección o nombre lógico. Una invocación de programa creada por el servicio de invocación de programa recibe el atributo Deletable=true. El servidor automáticamente crea el atributo de 'Número de Dominios' en la descripción de objeto durante la ejecución del servicio; el atributo contiene el número de dominio. Asume que el primer dominio en la lista de índice contiene un programa ejecutable.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

El cliente crea una nueva invocacion de programa solamente cuando tiene derecho de acceso al dominio activo. De otra forma, devuelve un resultado negativo. Si la misma invocación de programa con el correspondiente derecho a acceso ya existe el servidor no crea un nuevo objeto pero pasa la dirección lógica (índice) del objeto existente al cliente.

DeleteProgramInvocation

Borra programas que fueron definidos en el DP-OD. Los objetos de invocación de programa pueden solamente ser borrados si tienen el código correspondiente (Deletable=true).

Start

Un programa se comienza con este servicio y corre desde el comienzo. Los contadores de los dominios activos se incrementan. Antes este servicio chequea si los dominios envueltos han entrado en el estado READY o IN-USE.

Stop

Para un programa pero no resetea al principio.

Resume

Conduce a un programa al estado RUNNING pero no resetea. Los contadores de los dominios activos se incrementan. Antes de esto, el servicio chequea si los dominios envueltos han entrado en el estado READY o IN-USE.

Reset

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Mediante este servicio un programa parado se puede resetear al comienzo con el atributo Reusable=true. La invocación de programa entra en el estado IDLE. Si el atributo devuelve Reusable=false, la invocación de programa entra en el estado UNRUNNABLE. Este servicio solamente puede ser ejecutado si precede un servicio de STOP.

Kill

Este servicio conduce una invocación de programa al estado UNRUNNABLE sin considerar el estado anterior. Si la invocación de programa estaba en los estados RUNNING o STOPPING, los contadores de los dominios activos se decrementan.

4.8.6 ACCESO A VARIABLE

Read

Usando este servicio, se pueden leer de la estación remota los valores de los objetos Variable Simple, array, registros y Lista de Variable. Los elementos sencillos de arrays y registros pueden ser accedidos por el subíndice.

Write

Este servicio escribe valores en los objetos de variable simple, array, registro y Lista de variable. Los elementos sencillos de arrays y registros pueden ser direccionados con el subíndice.

PhysRead

Se puede leer el valor de un objeto de acceso físico en la estación remota.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

PhysWrite

Asigna valores a los objetos de acceso físico.

InformationReport

Usando este servicio, se pueden transmitir los datos de los objetos variable simple, array, registro, y lista de variable. Después de la ejecución de este servicio, el cliente no recibe confirmación. Los elementos sencillos de arrays y registros pueden ser accedidos con el subíndice.

El servicio puede ser usado con broadcast y multicast, lo que permite mandar datos o varias o todas las estaciones en el bus.

DefineVariableList

Crea el objeto Lista de Variable en la estación par. El proceso de aplicación del cliente debe asegurar que los datos de la Lista de Variable se puedan transmitir en un mensaje sencillo (PDU).

DeleteVariableList

Puede ser usado para borrar un objeto de Lista de Variable existente en la estación par si el cliente tiene derecho de acceso a ese objeto. Solamente pueden ser borradas las borrables Listas de Variable.

ReadWithType

Usando este servicio, se puede leer los datos y la relatada descripción de tipo de dato

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

de los objetos variable simple, array, registro y lista de variable. Los elementos sencillos de los objetos array y registro pueden ser accedidos con el subíndice.

WriteWithType

Escribe datos en los objetos de variable simple, array, registro y lista de variable. Una descripción de tipo de dato se añade a los datos para ser escrita. Los elementos de los objetos array y registro se pueden acceder por el subíndice.

InformationReportWithType

Transmite datos y las descripciones de tipos de datos de objetos variable simple, array, registro o lista de variable. La estación de comunicación no confirma el servicio. Los elementos sencillos de los objetos array y registro se pueden acceder por el subíndice.[2]

4.8.7 MANEJO DE EVENTOS

EventNotification

Puede transmitir una notificación de evento. Es un servicio sin confirmación. El proceso de aplicación puede mandar una confirmación al generador de este servicio por el servicio AcknowleddgeEventNotification. Este servicio puede usarse en broadcast o multicast, permitiendo enviar al mismo tiempo los eventos a varias señales.

AcknowledgeEventNotification

Confirma una notificación de evento en el nivel de aplicación. Es el proceso de

[2]-Este servicio puede mapearse en la LLI (Lower Layer Interface) broadcast o multicast lo que permite valores para transmitirse a varias o a todas las estaciones del bus.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

aplicación que debe confirmar el evento.

AlterEventConditionMonitoring

Este servicio activa o desactiva el evento.

EventNotificationWithType

Usando este servicio, las notificaciones de eventos se pueden transmitir las que contiene datos y la relatada descripción de tipos de datos. Desde el punto de comunicaciones, es un servicio sin confirmación. Usando el servicio AcknoledgeEventNotification, el proceso de aplicación puede enviar una confirmación al generador de la notificación de evento.

4.9 DIRECCION DE RED

4.9.1 DIRECCION DE LA RED PROFIBUS

El estándar PROFIBUS [DIN 91a], [DIN 91b] juega un papel importante, incluso para el manejo de la red, para englobar la comunicación abierta. El estándar especifica qué funcionalidad de manejo de la capa 7 de una red PROFIBUS se debe proporcionar para ser capaz de participar en Manejo de Red Abierta.

Observaciones Generales

El objetivo de la especificación PROFIBUS es asegurar un conjunto de 'funciones básicas' para el manejo de red. Estas funciones básicas son componentes de funciones más complejas (por ejemplo, herramienta de configuración) que se pueden construir.

Para PROFIBUS, el procedimiento fue como sigue:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* Primero, se desarrolló un modelo conceptual para el manejo de PROFIBUS y las necesidades funcionales especificadas.

* Se desarrolló un modelo de arquitectura, que situó la función de manejo en el contexto del modelo general de PROFIBUS y se determinó las interfases en los diferentes niveles.

* Se especificaron las funciones, las cuales manipulan objetos de comunicación u objetos especificados por el manejo de red. Todos los objetos supervisados por el manejo de red se llaman 'objetos de manejo'.

* Se consideraron condiciones políticas técnicas y económicas.

Modelo

La capa 7 de la arquitectura de manejo de PROFIBUS se llama FMA7 (Field Management Layer 7). En la figura 4.16 se muestra la localización de las funciones de manejo en el modelo PROFIBUS. Se distingue entre manejo local y remota.

Localización de las funciones de manejo

Además de la interfase de aplicación, el FMA7 tiene interfases adicionales a las capas 'FMS', 'LLI' y FMA1/2, que manejan las capas 1 y 2. El estándar [DIN 91a] contiene las especificaciones para FMA1/2, y [DIN 91b] contiene la especificación para FMA7.

El manejo de red proporciona servicios de manejo que normalmente se clasifican en tres subfunciones: manejo de contexto, manejo de configuraciones y manejo de errores.

Las funciones en cada una de las subfunciones anteriores incluyen:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

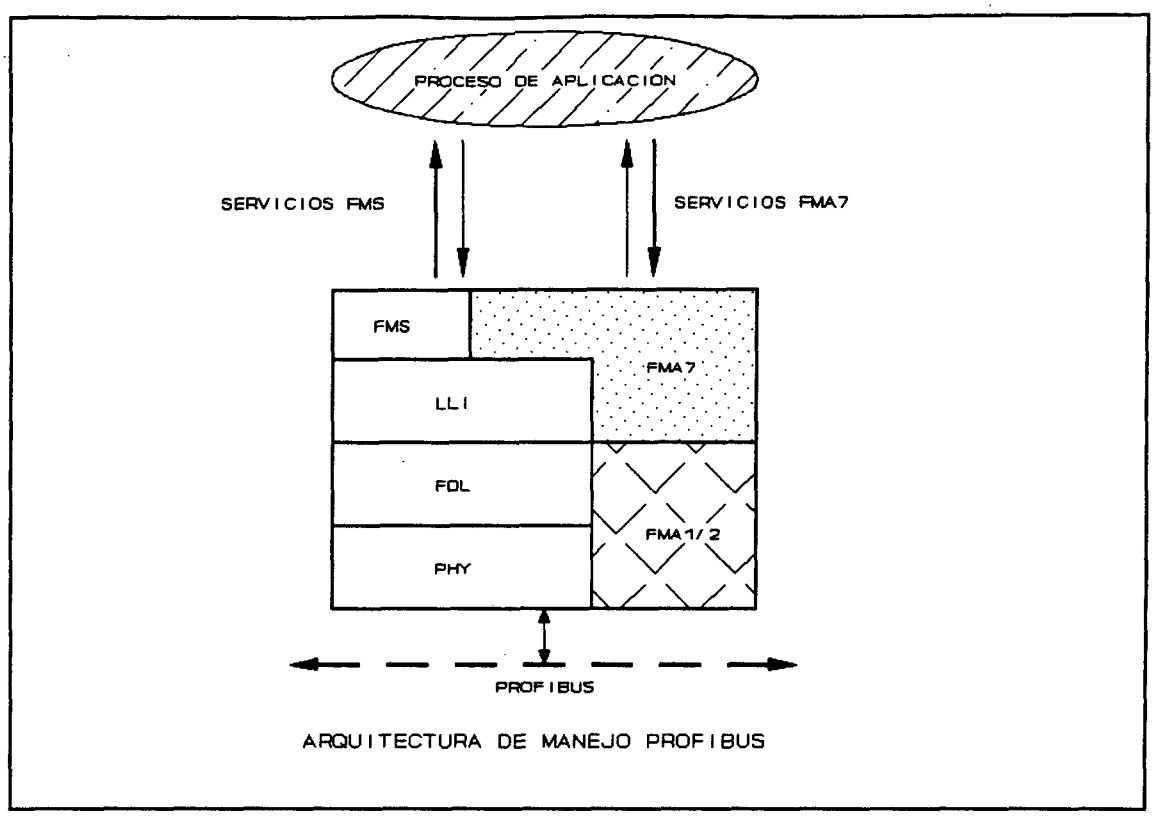


FIGURA 4.16.

* **Manejo de contexto.** La iniciación y liberación de conexiones de manejo importantes, sobre las que una estación del bus se puede observar remotamente o manipular por el manejo de red.

* **Manejo de configuración.** Todas las funciones conciernientes a la relación de una estación de bus con las otras estaciones. Estas incluyen carga y lectura de la Lista de Relaciones de Comunicación de una estación, variables de acceso, contadores y parámetros estadísticos de las capas 1 y 2, y adquisición e identificación de las estaciones del bus y sus componentes (como el código de vendedor, la versión del software,...)

* **Manejo de errores.** La vigilancia y evaluación de los errores, lo que, junto con los contadores estadísticos del manejo de configuración, permite diagnosticar los errores.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

El manejo de red PROFIBUS describe los objetos de manejo y los servicios específicos de objeto para manipularlos. Los objetos de manejo son un subconjunto de los objetos de comunicación especificados en el estándar PROFIBUS. Este estándar distingue entre manejo local y remoto.

Requisitos

En las especificaciones del manejo de red PROFIBUS, se consideran los siguientes requisitos:

* El manejo de red PROFIBUS debe estar en concordancia con los estándares existentes, p.e., el manejo de sistema en concordancia con [ISO].

* Ser posible implementar las estaciones PROFIBUS con un manejo mínimo. Es importante porque las estaciones 'sencillas' no necesitan ser equipadas con funciones de manejo solamente para razones de conformidad las cuales no son necesarias por la aplicación específica.

* El manejo de PROFIBUS debe adaptarse a su funcionalidad de requisitos de bus de campo (p.e., gran número de estaciones mudas, capacidad de tiempo real,...). Algunas funciones de manejo no son necesarias en el nivel de campo, como protección de acceso sofisticada, contador de servivio de usuario o compartición de carga dinámica. El principal foco de manejo es el diseño de red, encargado, mantenimiento y diagnosis de error.

* Otra condición es que el manejo de red PROFIBUS de la capa 7 se base en las funciones de manejo de la capa 1 y 2 (FMA1/2) especificadas en el estándar, y que se use la funcionalidad de transporte de la LLI (Lower Layer Interase).

4.9.2 SERVICIOS

© Del documento, los autores. Digitalización realizada por ULPGC. Bibliotecas Universitaria, 2006

El estándar PROFIBUS especifica un amplio conjunto de servicios, que se almacenan localmente en la estación. Es característica de los servicios de manejo local que no actividad en el bus sea causada cuando estén calladas. Cada invocación de un servicio de manejo por el proceso de aplicación en la interfase de aplicación de la FMA7 se transmite como una petición y confirmación de primitiva de servicio con la confirmación de primitiva del servicio correspondiente.

La notificación de evento del FMA7 al proceso de aplicación se transmite por un servicio de primitiva de indicación. Los servicios de manejo local proporcionados por la FMA7 en la interfase de aplicación permiten, por ejemplo, lo siguiente:

* La lectura, carga y sobreesritura de la Lista de Relación de Comunicación (CRL).

* Lectura y escritura de las variables de manejo local , como contadores estadísticos para diagnosis y depuración.

* Lectura de la identificación del equipo local.

* Preguntar sobre el estado de los Puntos de Acceso al Servicio Local (LSAP).

* Reseteo del FMA7.

* Notificación de evento del FMA7 al usuario

Los servicios de manejo local del FMA7 se planean en los servicios de manejo de FMA1/2, LLI y FMS.

Manejo remoto

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

El manejo remoto permite la manipulación de objetos de manejo en la estación remota por el uso importante de conexión de manejo, sobre las que las PDUs FMA7 se pueden transmitir entre clientes y servidores. Características:

* Todos los servicios remotos son opcionales, p.e. las estaciones que no soportan manejos de red sobre el bus se pueden conectar. esta característica es una concesión a equipos sencillos (sensores, actuadores). Será posible predeterminar todos los parámetros y almacenarlos en una memoria de solo lectura y capacidad de ser borrada (EPROM).

* El manejo remoto es usuario de LLI,p.e., usa la misma interfase de LLI como la capa 'normal' 7 (FMS).

* El concepto de manejo remoto permite el diseño completo y configuración de una estación PROFIBUS sobre el bus.

* Los servicios remotos se soportan sólo como un respondedor. Esto significa que herramientas especiales de configuración pueden iniciar los servicios de manejo de red como pedidores, y todas las otras estaciones sólo pueden responder como respondedoras.

* Los servicios de manejo remoto se proporcionan usando los servicios de manejo local (figura 4.17).

* El manejo remoto es orientado a conexión. Si una estación PROFIBUS soporta el manejo remoto (excepto por equipos de diagnosis o depuración), usa una conexión de manejo importante y es siempre servidor. Esta Conexión de Manejo Sobreentendida se determinó en la fase de diseño. Usando los servicios FMA7, es posible la ejecución no paralela.

* Diagnosis, diseño y herramientas de configuración pueden soportar varias conexiones de manejo y trabajan como clientes.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

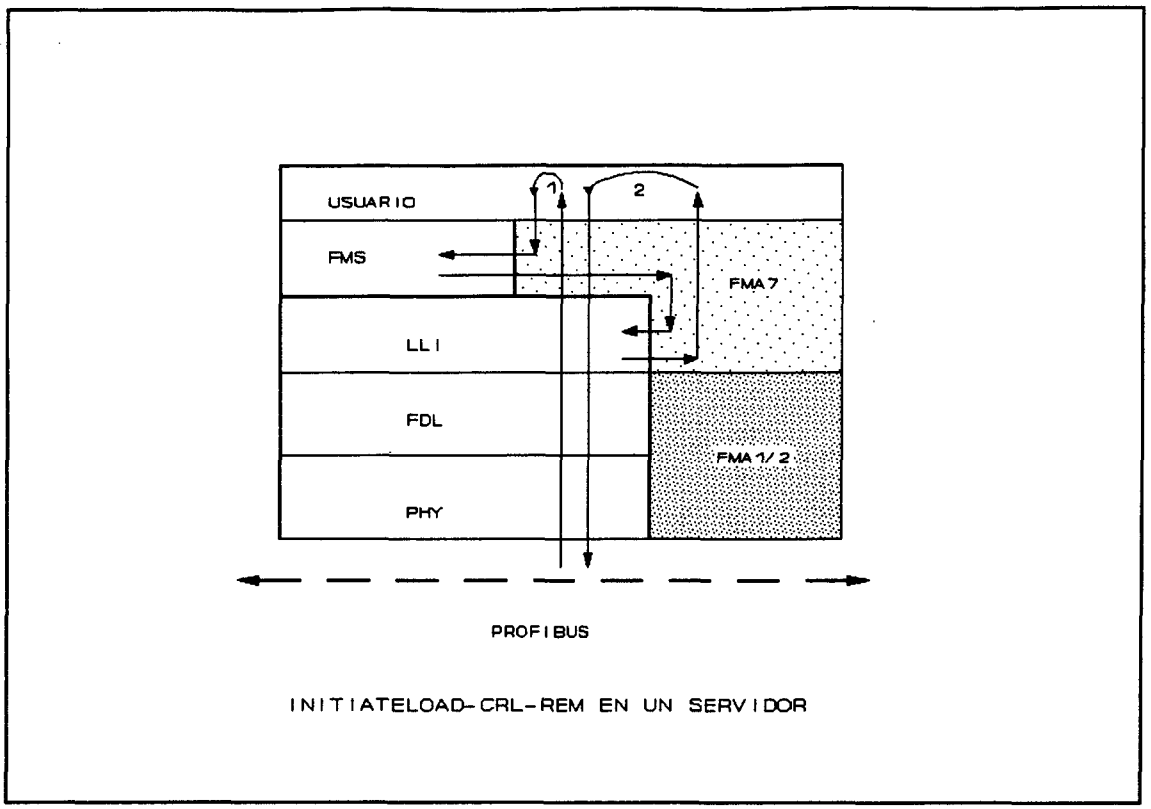


FIGURA 4.17.

En el manejo remoto, el procedimiento para acceder a una estación está predeterminado prácticamente. Cada estación PROFIBUS que soporta manejo remoto como un respondedor, debe ser capaz de soportar una conexión de manejo por defecto. Esta conexión está dentro de la Lista de Relación de Comunicación bajo la referencia de comunicación (CR) 1, y ocupar una entrada CRL. Si una estación es completamente diseñada por bus (diseño del CRL, parámetros del bus y OD), se debe predeterminar una conexión FMS adicional para este propósito en el CRL. Sólo se carga en el bus sobre conexiones FMS.

4.9.3 INTERFASES FMA7

La figura 4.16 muestra que, además de la interfase de usuario, el FMA7 tiene otras interfases a las capas adyacentes. A cada una de las interfases se le puede proporcionar

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

servicios locales, que el FMA7 usa para proporcionar los servicios locales solicitados en la interfase de usuario. El estándar describe cómo cada servicio FMA7 local se mapea en los servicios proporcionados a las interfases internas.

Se han especificado las siguientes interfases por el FMA7:

* **Interfase de FMS.** En esta interfase, se proporcionan servicios que activan, desactivan o resetean el FMS así como servicios para el procesamiento de parte de FMS del CRL. Por el servicio idéntico FMS, el FMA7 puede obtener la identificación FMS.

* **Interfase de LLI.** La interfase de la LLI a el FMA7 tiene la misma funcionalidad que la de FMS. Además, la LLI puede notificar un evento al FMA7 por el Servicio de Evento FMA7. El FMA7 ejecuta todos los servicios remotos sobre LSAP de Manejo de la LLI (Figuras 4.17 y 4.18).

* **Interfase de aplicación.** La funcionalidad de la interfase de aplicación es determinada y dividida en funciones opcionales adicionales y obligatorias. Como el nombre indica, es la interfase al proceso de aplicación, que puede proporcionar herramientas para el manejo de red usando la funcionalidad de la interfase de aplicación.

* **Interfase de FMA1/2.** Proporciona dos grupos de servicio: usando los servicios FMA1/2-Read_Value y FMA1/2-Set_Value, se pueden leer variables locales del FMA1/2 o escribir por la FMA7. El servicio idéntico de FMA1/2 puede solocitar una identificación.

4.9.4 MAPEO DE SERVICIOS FMA7

El FMA7 mapea todos los servicios proporcionados en la interfase de aplicación en los servicios locales y los ejecuta sobre los interfases de las capas adyacentes (FMS, LLI,FMA1/2). El estándar especifica cómo cada servicio FMA7 se debe mapear

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

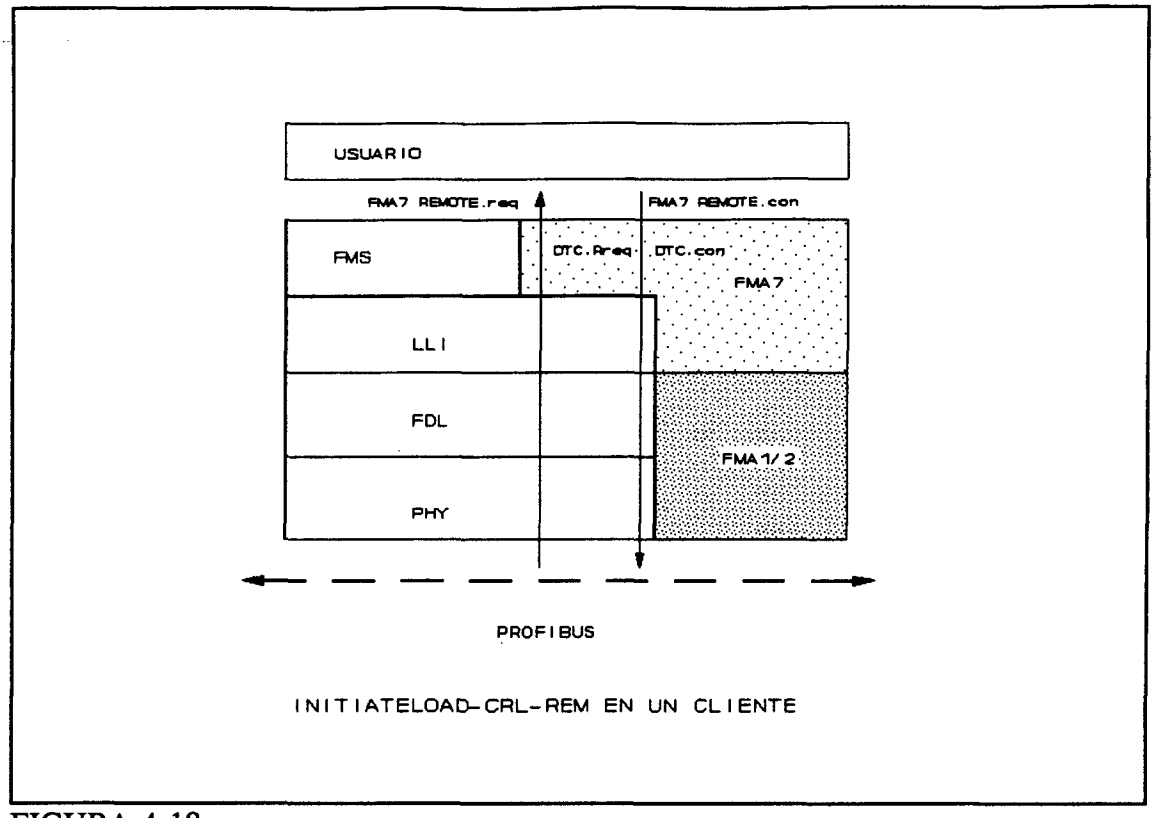


FIGURA 4.18.

internamente. Este mapeado se explicará para una ejecución de servicio en el servidor.

Ejemplo: Mapeando un servicio de manejo remoto en servicios de manejo local en un servidor.

Figura 4.17 muestra cómo un servidor responde a la llegada de una primitiva InitiateLoad-CRL-Rem. Request. Una llegada de PDU sobre el bus se procesa de la misma forma que una llegada de PDU de la comunicación normal de las capas inferiores (PHY y FDL). La LLI se da cuenta de que es una PDU de manejo, ya que la PDU llega en una conexión de manejo. Por lo tanto, la PDU no se transfiere a la FMS, pero sí a la FMA7, y desde la FMA7 al usuario de manejo. El usuario de manejo proporciona el servicio requerido y llama al servicio FMA7 local InitiateLoad-CRL-Loc. El FMA mapea esta petición de servicio local en los servicios de FMS y de LLI. Después de que se ha recibido la

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

confirmación, la FMA7 confirma la petición de servicio local del usuario. El usuario transmite una PDU InitiateLoad-CRL-Rem.con al cliente sobre el bus.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

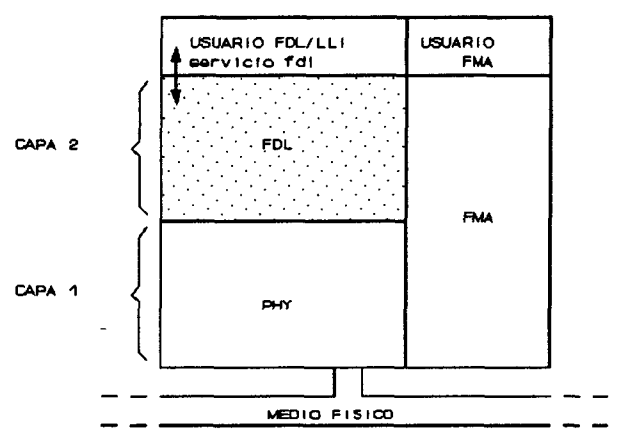
CAPAS ORIENTADAS AL TRANSPORTE

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

5 CAPAS ORIENTADAS AL TRANSPORTE

5.1 ENLACE DE DATOS DE BUS DE CAMPO

En la terminología PROFIBUS, la capa 2 se denomina Fieldbus Data Link (FDL). Funciona como máquina de estado y controla el acceso al medio. Adyacente a la FDL está el manejo del bus de campo de las capas 1 y 2 (FMA 1/2), lo que proporciona servicios de manejo a la Interfase de capa inferior adyacente (LLI) indirectamente sobre FMA7 para acceso interno a las capas 1 y 2. Se muestra su estructura en la figura 5.1.



ESTRUCTURA DE LA CAPA 2

FIGURA 5.1

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

5.1.1 SERVICIOS DE TRANSFERENCIA DE DATOS

Además de controlar el acceso al medio y el tiempo de rotación del testigo, la capa FDL debe proporcionar servicios de transferencia de datos usando protocolos de comunicación adecuados para el usuario FDL (normalmente la LLI). PROFIBUS ofrece tres servicios de transferencia de datos para mandar y pedir acíclicamente.

* SDA Send Data with Acknowledge

* SDN Send Data with no Acknowledge

* SRD Send and Request Data

SDA es un servicio básico por el que una estación activa puede mandar un mensaje a una pasiva e inmediatamente recibir la confirmación. Una estación pasiva solamente puede confirmar la recepción de datos, o reaccionar por los datos recibidos mediante transmisión de datos propios.

Los servicios SDN se usan principalmente para mensajes 'broadcast' o 'multicast' desde una estación activa (una estación que normalmente tiene derecho de acceso) a otras estaciones de bus, y de este modo permanecer sin confirmación. Todos los otros servicios se basan en una conexión recíproca entre un iniciador (estación que posee el testigo) y un respondedor (cualquier estación sin testigo), y necesitan ambos una confirmación o una respuesta. En publicaciones de la ciencia de ordenadores, esta conducta, importante para comportamiento en tiempo real del sistema de bus, se denomina respuesta inmediata.

SRD es un servicio por el que se transmiten los datos a una estación pasiva y al mismo tiempo desde donde se pide los datos. Los datos son entonces enviados inmediatamente con la respuesta. Como un caso especial de este servicio, una estación puede pedir datos mandando un 'mensaje vacío'.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Además de los servicios acíclicos arriba mencionados, las aplicaciones industriales suelen necesitar transmisión cíclica. El método apropiado de transmisión para equipos de campo sencillos, como sensores y multiplexores I/O, los cuales nunca se controlan ni tampoco (por razones de costo) tienen la funcionalidad de una estación de bus activa, es el polling controlado centralmente. PROFIBUS ofrece la posibilidad de almacenar una lista de POLL en un nivel cercano al hardware y, basado en un servicio SRD acíclico, de implementar un polling cíclico de estaciones pertenecientes a la lista. Además, localizando el procedimiento de polling en la capa de enlace, se acelera considerablemente las aplicaciones de tiempo real. El servicio correspondiente se denomina:

* CSRD Cyclic Send and Request Data with Reply.

5.1.2 ACCESO AL MEDIO

El Control de Acceso al Medio (MAC) debe satisfacer dos necesidades particulares de procesamiento y manufactura. En la comunicación entre equipos con capacidad similar y alguna inteligencia (como controladores programables), las estaciones deben darse un tiempo en un intervalo definido para ejecutar sus tareas de comunicación, pero la transferencia de datos en tiempo real debe ser posible en un nivel alto entre un equipo complejo y simple, equipos periféricos descentralizados. Para ambas necesidades, se prueban métodos de acceso al medio, los cuales se explican a continuación.

Métodos de acceso al medio

El método maestro/esclavo. En el método maestro/esclavo, una estación, la maestra, controla el acceso al medio distribuyendo explícitamente un permiso para transmitir datos durante una determinada cantidad de tiempo a otra estación, la esclava (figura 5.2). De este modo, se les puede asignar fácilmente diferentes prioridades a los nodos de red, dependiendo de las necesidades de la tarea.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

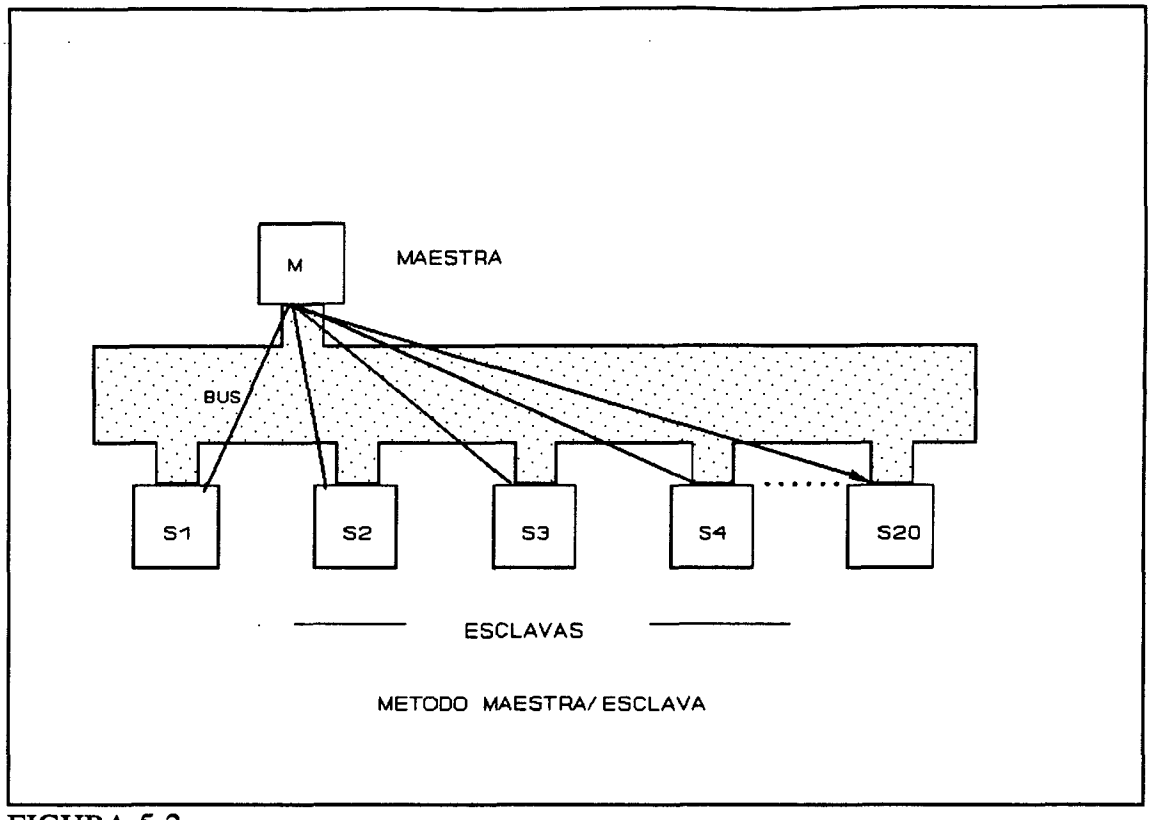


FIGURA 5.2

El método de paso de testigo. En LANs con varias estaciones equivalentes de distintas funcionalidades, se realiza una arbitración equilibrada por el método de paso de testigo. En este método, originalmente diseñado para topologías de anillo, una cierta secuencia de bit, llamada testigo, se pasa de una a otra estación en anillo lógico (figura 5.3). Usando una analogía, esta secuencia predefinida de bit se puede comparar al testigo de la carrera de relevos. Puede también ser pensado como permiso para escribir de transmisión. Si una estación quiere transmitir un mensaje, debe esperar a que le llegue el testigo y entonces tomarlo del medio. Después de la transmisión de datos, debe pasar el testigo a la siguiente estación maestra en un intervalo de tiempo predeterminado. Este método garantiza que en un Tiempo de Testigo máximo definible, cada maestra reciba el testigo.

Un inconveniente, no obstante, es el manejo de testigo caro y complejo que se necesita para los casos de error, como tramas de testigos perdidas. Esta función se debe

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

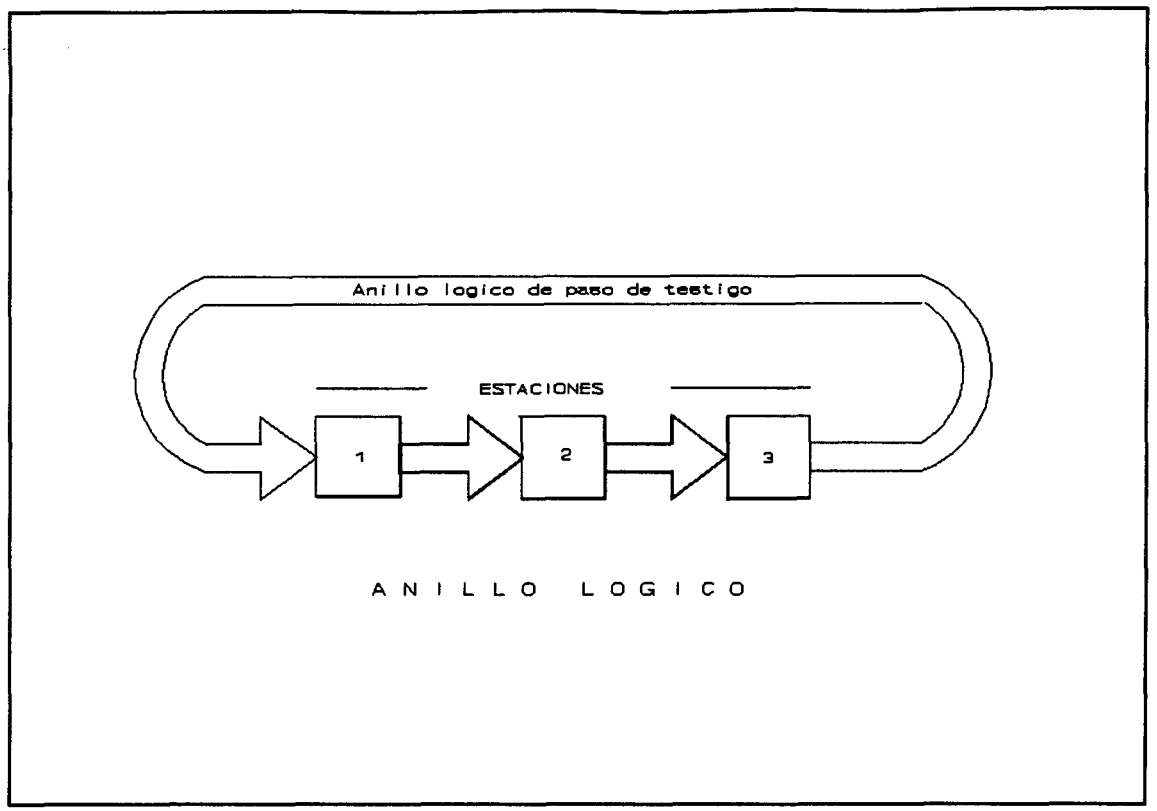


FIGURA 5.3

implementar en cada nodo de red. Además, parte de la capacidad de transmisión se usa solamente para manejo de acceso. Una ventaja es que los tiempos máximos de respuesta se pueden garantizar por los usuarios, debido a la naturaleza determinística de este método.

Descripción del concepto de PROFIBUS

La mayoría de los sistemas de automatización de campo consisten en una estación de control y un número de equipos de campo distribuidos. Esto generalmente implica una dirección centralizada, tráfico de datos cíclico, desde los equipos de campo a la estación de control central. Lo mejor para esta situación de comunicación es el método maestro/esclavo.

Por otro lado, sería también posible interconectar uniones de automatización de campo distribuidas en los niveles de función bajos y medios, como equipos de operación local,

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

control y programación. El método apropiado para este tipo de comunicación es el método de paso de testigo, ya que garantiza una distribución equilibrada del derecho de acceso al medio.

PROFIBUS usa un método de acceso al medio determinístico. Las estaciones activas acceden en concordancia al método de paso de testigo descentralizado. Para el tráfico entre equipos de campo pasivos y activos, se usa el método maestro/esclavo.

El método de acceso al medio híbrido satisface las necesidades de automatización de broad, especialmente con respecto a la ejecución, tiempo de respuesta, seguridad y esfuerzo.

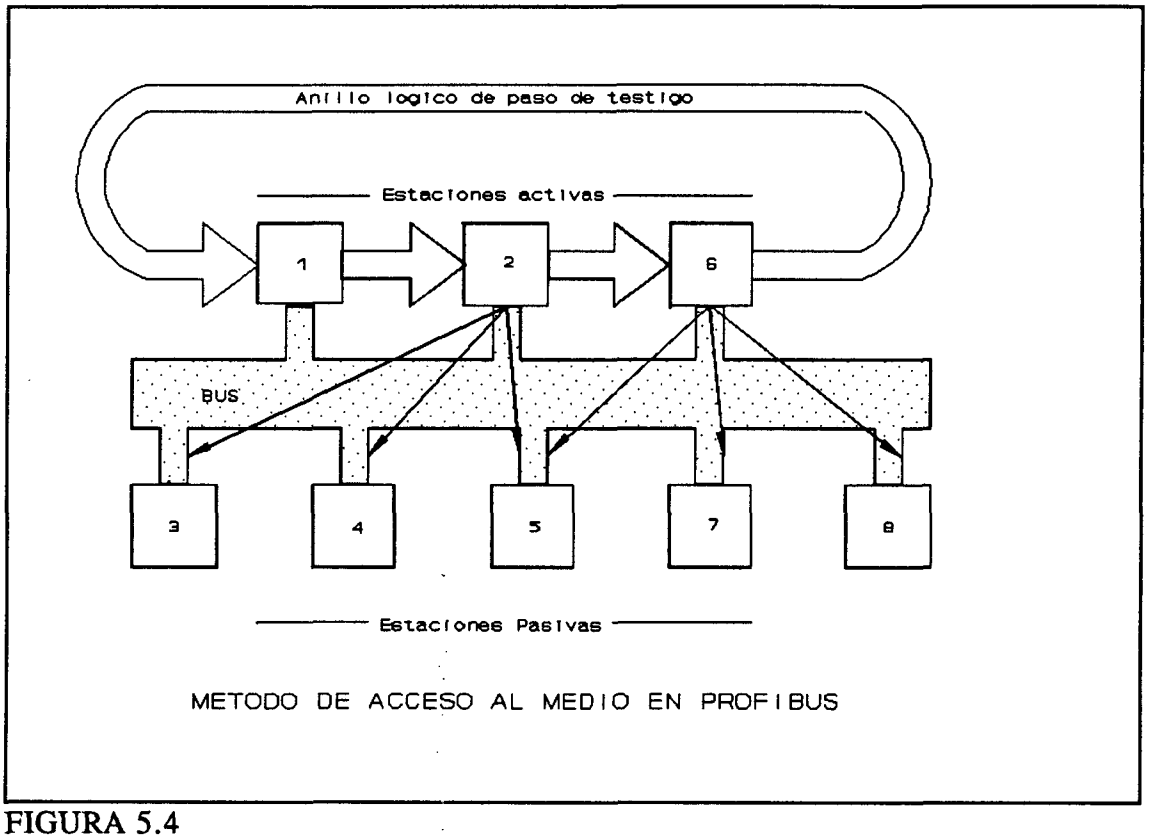


FIGURA 5.4

En la figura 5.4. se muestra una configuración PROFIBUS consistente en 3 estaciones activas y 5 pasivas. La secuencia de paso de testigo forma un anillo lógico. El testigo se pasa

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

de una estación activa a la siguiente, llevando una cierta secuencia, y la última estación finalmente pasa el testigo a la primera.

Estaciones añadidas y quitadas. Para la operación regular, la que también determina los tiempos de frecuencia del sistema, un número de condiciones se deben realizar. Cada estación debe conocer la dirección de las estaciones precedentes y sucesivas para conocer de qué estaciones se recibe el testigo y a qué estaciones se debe pasar. Después de la inicialización, cada estación calcula autónomamente estos parámetros y carga datos directamente durante la operación, si es necesario. Para este propósito, cada estación examina regularmente el espacio de dirección entre su propia dirección de red y su testigo sucesor, la apertura, para encontrar nuevas estaciones que están preparadas para entrar en el anillo. Por este método, es posible añadir nuevas estaciones a la red durante la operación o quitarlas sin interrumpir la comunicación del sistema.

Target Rotation Time T_{TR}. Para garantizar que cada estación es capaz de transmitir sus propios mensajes después de un cierto período máximo de tiempo, es necesario definir un intervalo de tiempo, en el que el testigo deba tener completamente atravesado el anillo lógico. Asignar tiempos fijos de transmisión a las estaciones activas no es una solución razonable para la mayoría de las aplicaciones, ya que el tiempo sería desperdiciado si una estación no tuviera nada que transmitir y el tiempo de transmisión no fuera suficiente para estaciones con largas colas de mensajes.

Prioridades de trama. Para capacitar la transmisión de importantes mensajes con alta prioridad, PROFIBUS usa dos niveles de prioridad. La transferencia normal de datos es llevada a cabo con baja prioridad. Los datos de sucesos de eventos especiales se pueden enviar con alta prioridad, y alcanzar su destino más rápido que los otros datos. Un ejemplo es el mandar datos de eventos a la estación de control central, lo que a la vuelta puede tomar medidas apropiadas -y con la misma alta prioridad- para responder al evento.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Con los servicios de la interfase de usuario FDL, el usuario puede elegir entre dos prioridades ('alta' y 'baja') que se transmiten con la petición de servicio en el byte FC (control de trama) de la trama. Para asegurar la transmisión de datos importantes, se puede empezar un ciclo de alta prioridad después de cada recepción del testigo, incluso si el no resta tiempo de permanencia del testigo. Después, no obstante, el testigo se debe pasar inmediatamente a la siguiente estación.

Si el tiempo de rotación del testigo T_{RR} es menor que el tiempo de rotación T_{TR}, se pueden transmitir más mensajes, empezando con mensajes de alta prioridad seguido por los de baja prioridad. Los servicios de baja prioridad tienen:

1. Procesamiento de la lista de Poll (servicios cíclicos)
2. Procesamiento de mensajes de baja prioridad (servicios acíclicos)
3. Grabación de estaciones activas (redibujo de la lista viva)
4. Puesta al día de los nodos (direcciones de la lista de nodos).

Esta secuencia no es fija, pero puede ser modificada bajo condiciones operacionales. El procesamiento de la lista de Poll, por ejemplo, se hace de forma segmentada, así que si no resta tiempo de permanencia del testigo en un número de ciclos de permanencia del testigo, se podría procesar servicios no acíclicos. Por este motivo el control FDL debe incluir procesamiento de todas las peticiones restantes de baja prioridad, incluyendo actualización de nodos, antes del comienzo de un ciclo nuevo de Poll. Se muestra una perspectiva sobre el procesamiento de ciclos de mensajes en la figura 5.5.

Estructura de trama. En PROFIBUS, los datos no se transmiten por caracteres. En cambio se transmiten por paquetes los cuales son cadenas ordenadas de caracteres de transmisor receptor universal asíncrono (UART).

Los servicios de transferencia de datos necesitan formatos de trama estandarizados.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

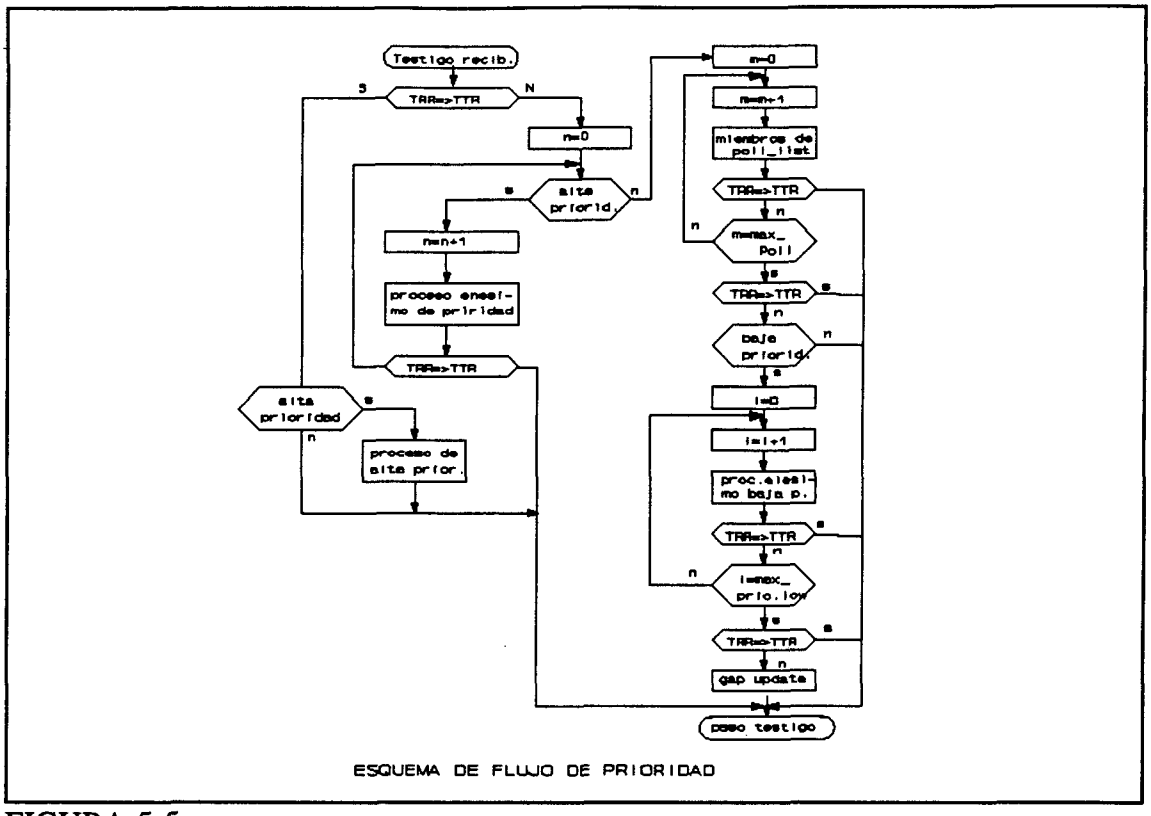


FIGURA 5.5

Junto con la codificación más eficiente, se debe asegurar la transmisión segura. Para este propósito, las tramas proporcionan información redundante adicional, lo que inevitablemente aumenta la cabecera de protocolo, lo que reduce la tasa de transmisión de red del protocolo.

La figura 5.6. muestra tres ejemplos de tramas posibles. Arriba vemos una trama sin campo de datos, así que es la trama más corta posible. Una trama siempre empieza con un delimitador de comienzo (SD) conteniendo el código del formato de trama. Después del SD viene una dirección de destino (DA) y una dirección fuente (SA) lo que identifica al receptor y al emisor. Sigue el control de trama (FC), por lo que el receptor reconoce el tipo de trama (llamada trama de confirmación o de respuesta). Además, contiene la prioridad de la trama y la información de control, lo que evita la pérdida de la trama, por ejemplo. Al final de la trama se encuentra una secuencia de chequeo de trama (FCS), que sirve para la seguridad de datos, seguido por el delimitador de final (ED). La estación llamada confirma la recepción

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

de trama por envío de una confirmación de trama o por una trama corta, lo que es solamente un carácter.

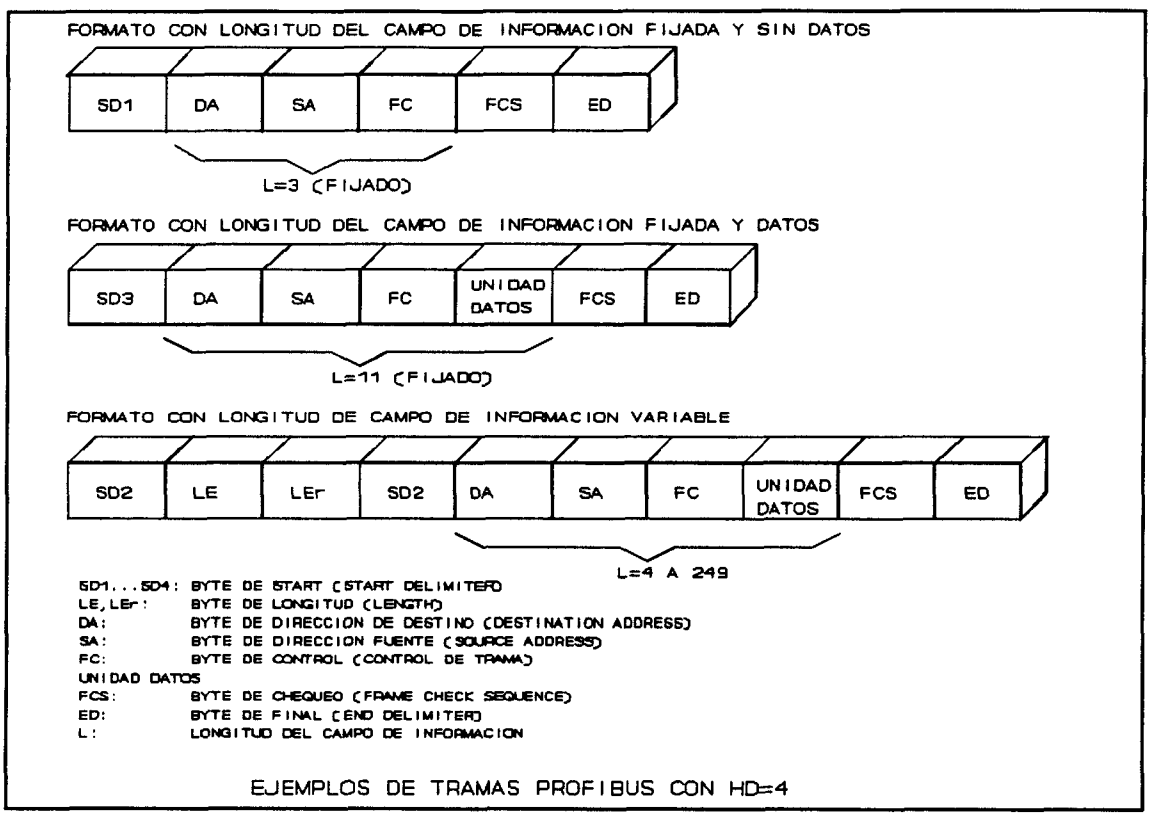


FIGURA 5.6

Los dos ejemplos superiores de la figura 5.6. se entienden fácilmente. El ejemplo inferior muestra la longitud de trama máxima en la capa 2 que consiste en 255 bytes, permitiendo 246 bytes de información.

Seguridad de datos. Aunque PROFIBUS proporciona buena protección contra los errores de transmisión, no se pueden prevenir totalmente. La razón de estos errores puede, por ejemplo, ser transmisores defectuosos, un medio pobremente protegido, reflexiones de señales o excesivas diferencias entre las frecuencias de reloj del receptor y transmisor.

Los componentes de la UART son capaces de detectar algunos de estos errores:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* Errores de trama: no se reconoció el bit de stop de un carácter.

* Errores de desbordamiento: sobreescritura de un carácter recibido por el siguiente antes de ser almacenado.

Para asegurar la transmisión libre de error, PROFIBUS proporciona seguridad con la distancia hamming 4 (HD=4). La distancia Hamming indica por cuántos dígitos binarios se diferencian dos caracteres válidos UART (la introducción de un bit de paridad incrementa 'artificialmente' la distancia Hamming en 1). En el caso de HD=4, se puede detectar y corregir un bit de error en un carácter, dos bits de error se pueden detectar pero no corregir. HD=4 puede realizarse si se transmite una secuencia de chequeo de trama en cada trama, lo que representa la paridad de la columna del campo de información del longitud L. Se forma comienza desde las sumas aritméticas sin carry de los caracteres transmitidos. Ya que los delimitadores de comienzo y final no se incluyen en la suma, se aseguran dichos caracteres contra otras de HD=5. No se proporciona la corrección de error. Después de la detección de un error, la trama se descarta y se repite la transmisión.

Errores. Para proporcionar una seguridad del sistema, PROFIBUS maneja los estados operacionales siguientes o de error:

* varios testigos en rotación.

* pérdida de testigo.

* error de paso de testigo.

* asignación múltiple de direcciones de estación.

* estaciones con receptores defectuosos.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* añado o eliminación de estaciones en operación.

* combinación arbitraria de estaciones activas y pasivas.

5.1.3 MAQUINA DE ESTADOS

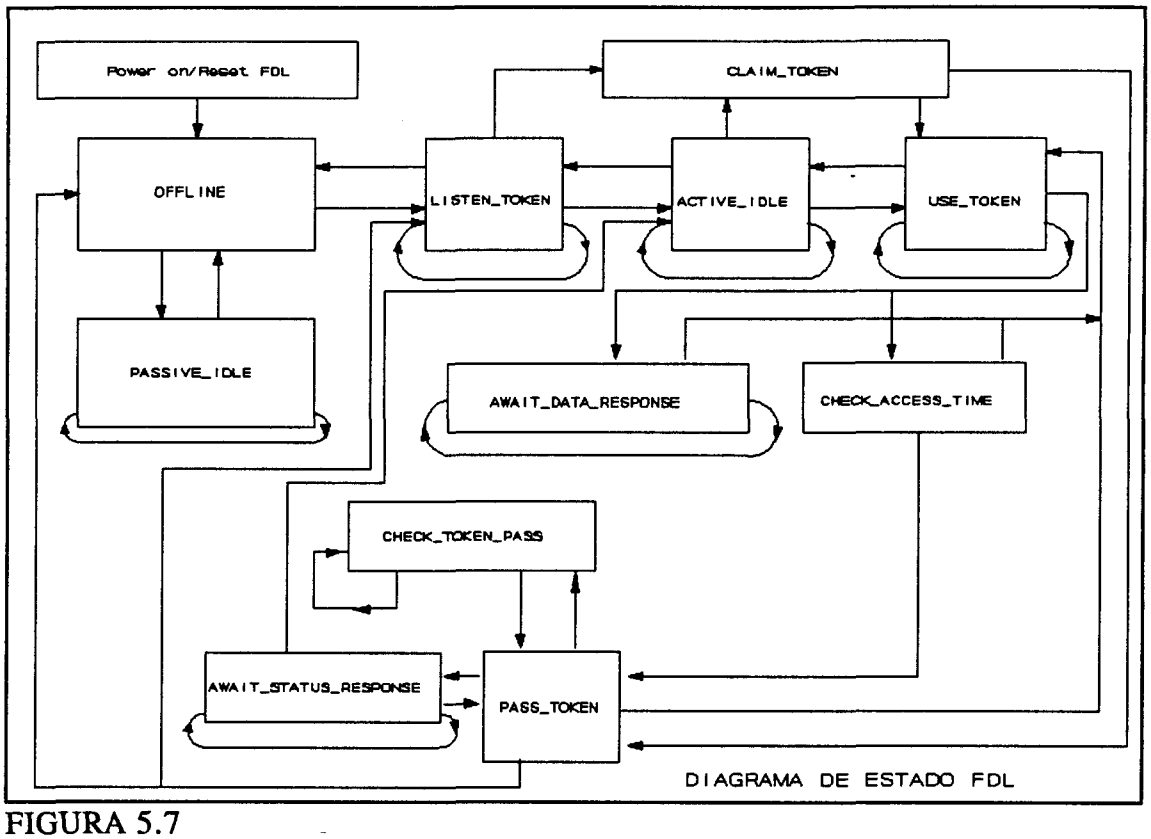


FIGURA 5.7

Se muestran en la figura 5.7. todos los estados necesarios en un controlador FDL para una correcta ejecución de comunicación.

Después del encendido del voltage de alimentación (Power ON), las estaciones de bus tanto activas como pasivas entran en el estado Offline y se ejecutan un testeo. Cargan los parámetros operacionales para la comunicación, y entonces se conecta al medio de transmisión.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Después, las estaciones pasivas entran en el estado Passive_Idle (figura 5.8). Listan la línea y confirman una trama direccionada para ellas mismas (excepto para broadcast). No pueden entrar en otros estados.

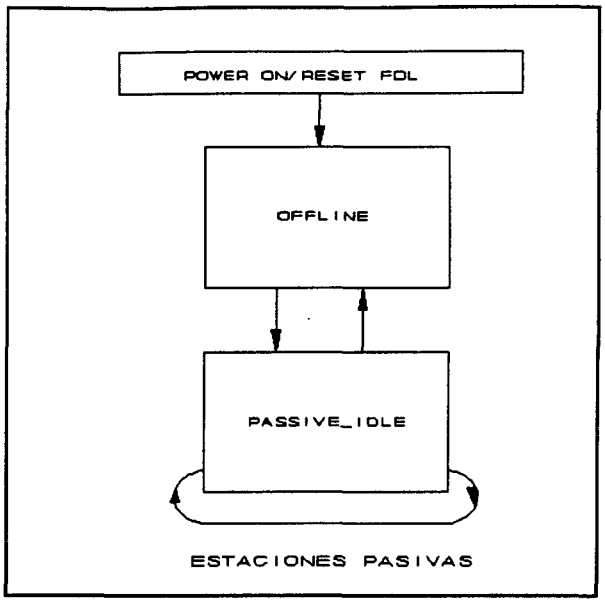


FIGURA 5.8

Después de Offline, las estaciones activas entran en el estado Listen_Token cuando se leen para entrar

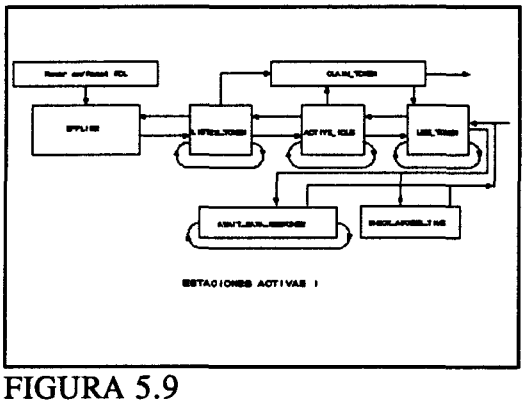


FIGURA 5.9

en el anillo lógico (figura 5.9). Un testigo es una secuencia de bit particular que se puede interpretar como el acceso para transmitir datos. En el estado Listen_Token, la estación escucha tramas en el bus y compila una lista de estaciones activas (LAS) por las direcciones de las tramas de testigo recibidas.

Después de la compilación de tal lista, la estación debe esperar hasta que se direccione la estación previa con un 'Estado de petición de FDL' y así invitada al anillo lógico. La nueva estación confirma con una 'lectura del anillo' y entra en el estado de Activa_Idle. Así entra oficialmente en el anillo.

Cuando la estación recibe una trama de testigo, entra en el estado Use_Token. Es el estado de una estación activa en el anillo lógico cuando tiene derecho de acceso. Todas las otras estaciones permanecen en es estado Active_Idle. La estación chequea el tiempo de

© Del documento, los autores. Digitalización realizada por ULPGC. Bibliotecas Universitaria, 2006

mantenimiento del testigo en el estado Check_Access_Time y ejecuta ciclos de mensaje por bus. Si una estación usa un servicios de respuesta, entra en el estado Await_Data_Response y aguarda la respuesta por un cierto tiempo.

Una estación puede ejecutar ciclos de mensajes hasta que se acabe el tiempo de mantenimiento del testigo. Entonces la estación entra en el estado PAss_Token en la que se pasa el testigo a la siguiente estación activa. Este paso de testigo se controla en el estado Check_Token_Pass. Si no existe sucesor conocido, se entra en el estado Await_Status_Response.

En este momento el FDL espera un cierto tiempo por una trama de confirmación. Si la estación activa no recibe nada o una trama corrupta, se vuelve a entrar en el estado Pass_Token. Si ocurre un error, la estación entra en el estado Listen_Token e informa a FMA1/2. Por otro lado, después de un procedimiento de paso de testigo normalmente libre de error, la estación asume el estado Active_Idle hasta que recibe el testigo la siguiente vez. La figura 5.10 muestra esta parte de la máquina de estado para estaciones activas.

Si una estación activa no recibe trama de testigo en un largo período de tiempo, entra en el estado Claim_Token e intenta inicializar (si no ha sido una estación activa por ahora) o reinicializar (si es una estación activa) el anillo lógico. En el último caso, la LAS permanece válida.

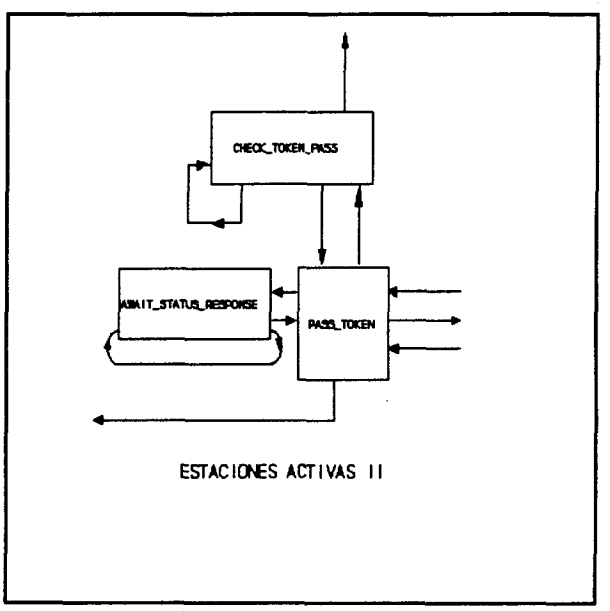


FIGURA 5.10

5.2 FMA1/2

El FMA1/2 (Fieldbus Management of the layers 1 and 2) proporciona funciones

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

para el manejo de las capas 1 y 2. Así, enlaza el usuario FMA1/2 (por ejemplo el FMA7) con las capas 1 y 2. La figura 5.11 muestra las interfases vía la cual el usuario FMA1/2 (=FMA7) accede a las dos capas. Por el uso de servicio.request FMA1/2, la FMA7 solicita que la FMA1/2 proporcione un servicio. El servicio.request FMA1/2 sólo es permitido para ser activo.

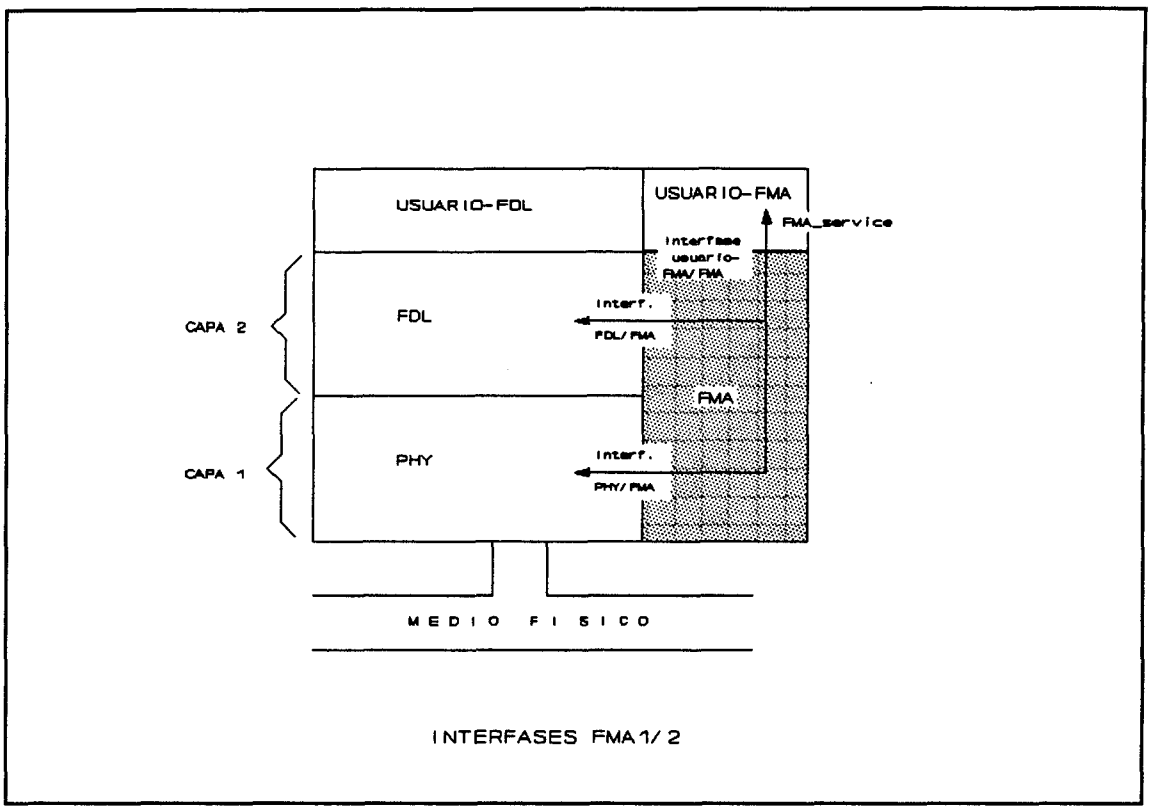


FIGURA 5.11

Antes de la ejecución de la petición, se chequean para compatibilidad los parámetros que se transmitieron en el bloque de petición de tarea y el parámetro de servicio relativo a la tarea. En FMA1/2, se procesan los servicios, adaptados y transferidos a la propia capa. Después del procesamiento de la tarea, la capa transfiere el servicio.confirm.primitive a la FMA1/2. La última genera un servicio FMA1/2 service.confirm para la FMA7. La primitiva de confirmación suele contener información de si el servicio se ejecutó correctamente. Algunos servicios permiten confirmación inmediata de la petición en la FMA1/2 antes de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

procesarse en las capas. El servicio FMA1/2 service.indication indica los errores y eventos en las capas y en el bus del usuario FMA1/2.

Los servicios se pueden clasificar en dos grupos: servicios locales que tratan de la estación en si misma y servicios que conciernen a otras estaciones en la red. En ambos grupos existen servicio obligatorios y optativos. Después de la ejecución, la mayoría de los servicios generan una respuesta informando al usuario por el FMA 1/2 sobre si el servicio se ejecutó o si produjo error.

5.2.1 SERVICIOS LOCALES

Reset. Por medio de este servicio, FMA7 puede resetear las capas 1 y 2. Después de que FMA1/2 recibe el reseteo, genera la señal de reseteo para ambas capas, la pasa por la interfase correspondiente y espera una confirmación de las capas.

Set Value. Por medio de este servicio opcional, se pueden asignar ciertos valores a variables en ambas capas. Los nombres de variable y valores deseados se pasan en una primitiva. El FMA1/2 genera un FDL y/o un PHY_SET_VALUE.request y lo transfiere a la(s) capa(s).

Read Value. Por este servicio opcional, se pueden leer las variables en ambas capas. Los nombres de variables se transfieren en primitivas. Entonces, el FMA1/2 genera un FDL y/o un PHY_READ_VALUE.request y lo transmite a las capas.

Event. Este servicio informa al usuario de ciertos eventos o errores en las capas 1 y 2. Cuando una indicación de error (PHY_EVENT.indication o FDL_FAULT.indication) ocurre en una de las capas, un FMA1/2.EVENT.indication se genera en el FMA1/2 después de la recepción de la señal y transmitida a la FMA7.

(R)SAP Activate FMA1/2. Mediante este servicio opcional, un usuario puede configurar y

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

activar un SAP. La comunicación se sitúa sobre estos SAPs (Puntos de Acceso al Servicio). Contienen las condiciones de mensajes permitidos y formatos de mensajes (dirección destino permitida, longitud de los datos,...). Mientras los mensajes llegan al SAP, se chequean según los requisitos prescritos para el SAP. Si es el caso, se transmiten los mensajes, sino se genera un mensaje de error. A primera vista, la comunicación sobre SAPs parece complicada, pero permite chequear mensajes y transmitir líneas y con esto evitar conexiones incorrectas. Para las funciones de respuesta de los servicios de contestación (SRD, CSRD), se debe establecer un RSAP por medio de RSAP Activate FMA1/2.

(R)SAP Desactivate FMA1/2. Este servicio es el complemento de (R)SAP Activate FMA1/2. Permite al usuario FMA1/2 desactivar un (R)SAP y también liberar todas las conexiones.

5.2.2 SERVICIOS DE SUPERPOSICION DE ESTACIONES

Ident FMA1/2. Mediante este servicio, el usuario del FMA1/2 puede obtener información de las versiones del hardware y software. Las estaciones activas pueden preguntar a las otras estaciones por su identidad, mientras las peticiones de las estaciones pasivas se limitan a la propia estación.

LSAP Status FMA1/2. Este servicio opcional puede ser usado por estaciones activas; permite obtener información sobre la configuración de un SAP para servicios FDL (Por ejemplo, SDA, SRD,...) de otra estación de la red.

Live List FMA1/2. Por este servicio opcional, se proporciona al usuario de una lista de todas las estaciones conectadas al bus. Esta lista se compila en la capa FDL, lo que genera un Request FDL Status con respuesta para cada dirección de estación posible. Por un polling de todas las direcciones posibles, el usuario obtiene una lista completa de las estaciones que no solamente incluyen estaciones activas sino también pasivas, distinto a la compilación de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

la lista viva con LAS (Lista de estaciones activas).

5.3 CAPA FISICA

La elección del medio de transmisión y de la interfase del bus física (figura 5.12), se

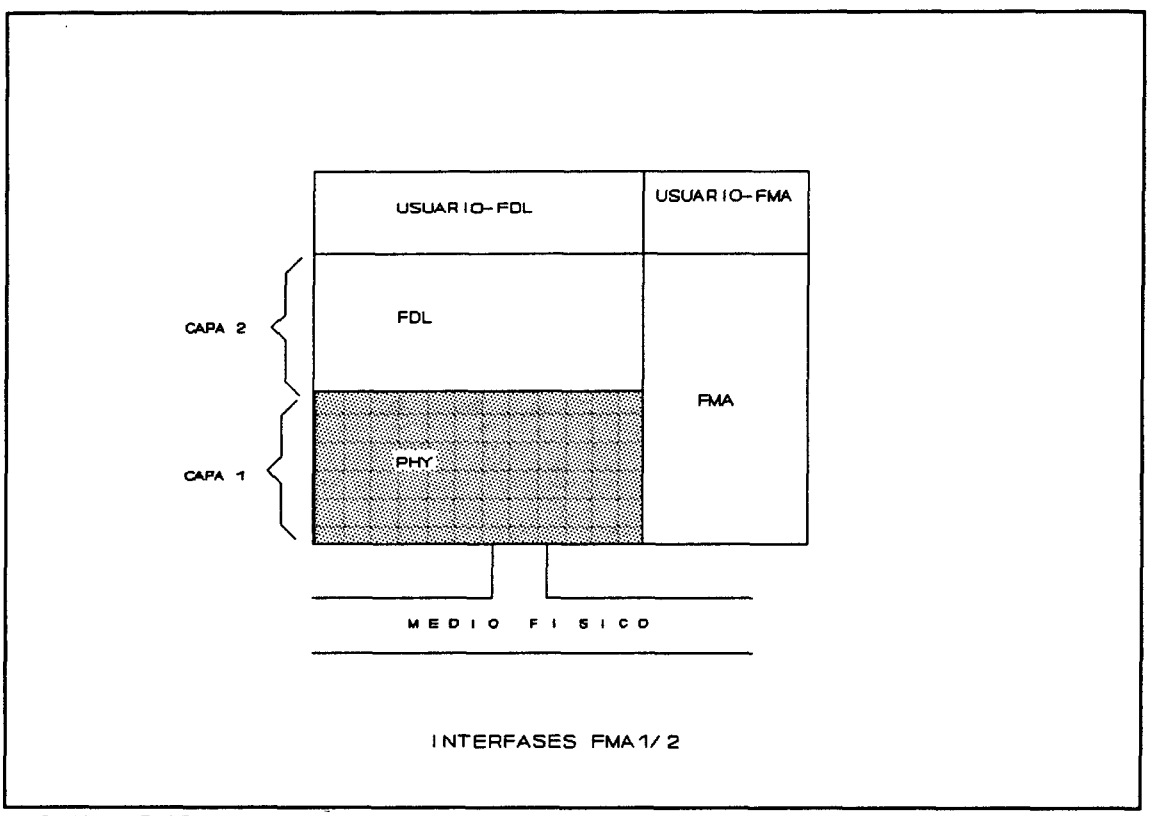


FIGURA 5.12

determina principalmente por el área de aplicación del sistema de campo. Además del grado requerido de posibilidad de transmisión, se debe considerar el coste para el propósito y la instalación del bus de línea. Por lo tanto, el estándar PROFIBUS proporciona la posibilidad de especificar diferentes clases de interfases físicas. La especificación PROFIBUS se basa principalmente en estándares ya existentes. Pero para satisfacer los requisitos de varias áreas de aplicación en la automatización de manufactura, la tecnología de proceso y automatización de edificios, se tiene que especificar características

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

específicas de campo (p.e., interfases, asignación de pin o terminaciones de bus). Estas especificaciones se basan en el estándar US, EIA RS-485 (ISO DP 8482).

5.3.1 TOPOLOGIA

PROFIBUS se basa en una topología de línea. El bus se puede estructurar como una línea con varios cables terminales de punto (longitud menor de 0,3m). Se está diseñando una versión para el alcance de velocidad baja, permite implementar una estructura en árbol. Esto permitiría una flexibilidad y al mismo tiempo una interconexión de red extensiva.

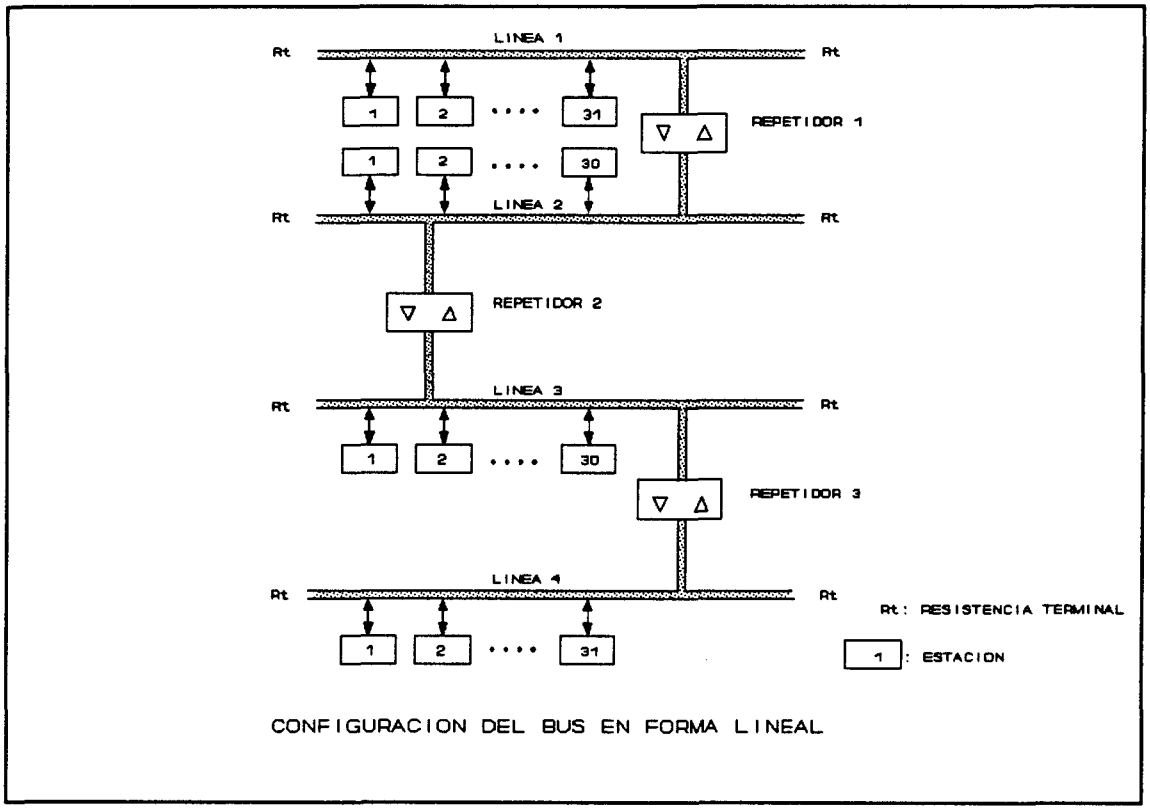


FIGURA 5.13

La longitud de la línea depende del alcance de transmisión. Para un alcance de 93,75 kbits/s, la longitud máxima de la línea no debe exceder de 1200m. Se deben conectar dos líneas por repetidores bidireccionales para que la longitud se pueda aumentar. Se permite un

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

máximo de tres repetidores entre dos estaciones con una longitud máxima de 4800m.

El número de estaciones está limitado a un máximo de 32 por línea. Cada repetidor conectado a una línea se considera como estación y por tanto reduce el número de estaciones permitidas en uno. Por esto es por lo que a una configuración de bus que consiste en cuatro líneas y tres repetidores, se le pueden conectar un máximo de 122 estaciones. Para la configuración del bus, se permiten preferiblemente 32 estaciones activas. Para aplicaciones de tiempo no crítico, no obstante, se permiten hasta 122 estaciones activas.

El problema se ilustra usando dos ejemplos de posibles configuraciones de bus. Para ambos ejemplos, se supone un alcance máximo de transmisión de 93,75 kbits/s, así que una línea puede tener una longitud de 1200m.

En la figura 5.13, se conectan cuatro líneas por tres repetidores para formar una fila. Si se suma las longitudes de cada línea (1200m), el resultado es la línea mencionada arriba de 4800m.

Se conectan dos repetidores a la línea 2: uno localizado entre la línea uno y dos , el otro entre la línea dos y tres. Cada estación se considera una estación, así que el número máximo de estaciones que se pueden conectar a esta línea, se reduce a 30. Cada terminal de línea tiene una resistencia R_t para evitar interferencias por reflexiones.

La figura 5.13 muestra que cuando se usan dos repetidores, se puede llegar a 3600m de longitud y 92 estaciones conectadas. Si se usa un repetidor, la longitud máxima de la línea se reduce a 2400m y 62 estaciones máximo.

En la configuración de forma de árbol (figura 5.14), se conectan varios repetidores a una línea. Se debe destacar, no obstante, que una conexión entre dos estaciones no puede tener más de tres repetidores. Así que la distancia máxima entre dos estaciones puede ser

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

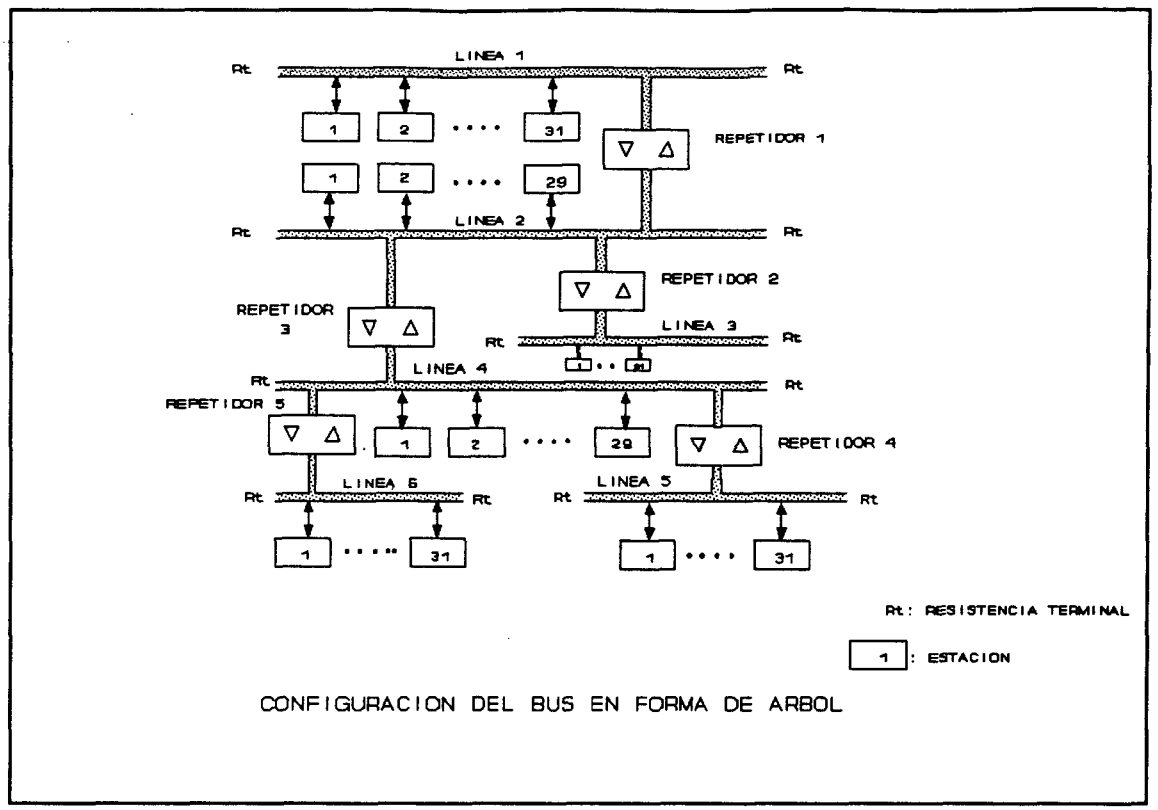


FIGURA 5.14

4800m.

5.3.2 TECNOLOGIA DE TRANSMISION

Los datos se transmiten según el estándar mencionado anteriormente. La versión más simple usa un par trenzado con protección, lo que permite operar en un medio ruidoso con grandes tasas de velocidad, bajo costo e instalación sencilla. Si no se espera interferencias electromagnéticas (EMI), se puede usar un cable sin protección.

Si se usan cables con líneas adicionales, se pueden transmitir señales para el control de dirección de repetidores, o se puede suministrar energía adicional por la conexión del bus, que está equipada con un conector SUB-D de 9 pines. Para mantener la inmunidad al ruido para control de dirección, se recomienda la tecnología RS-485.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

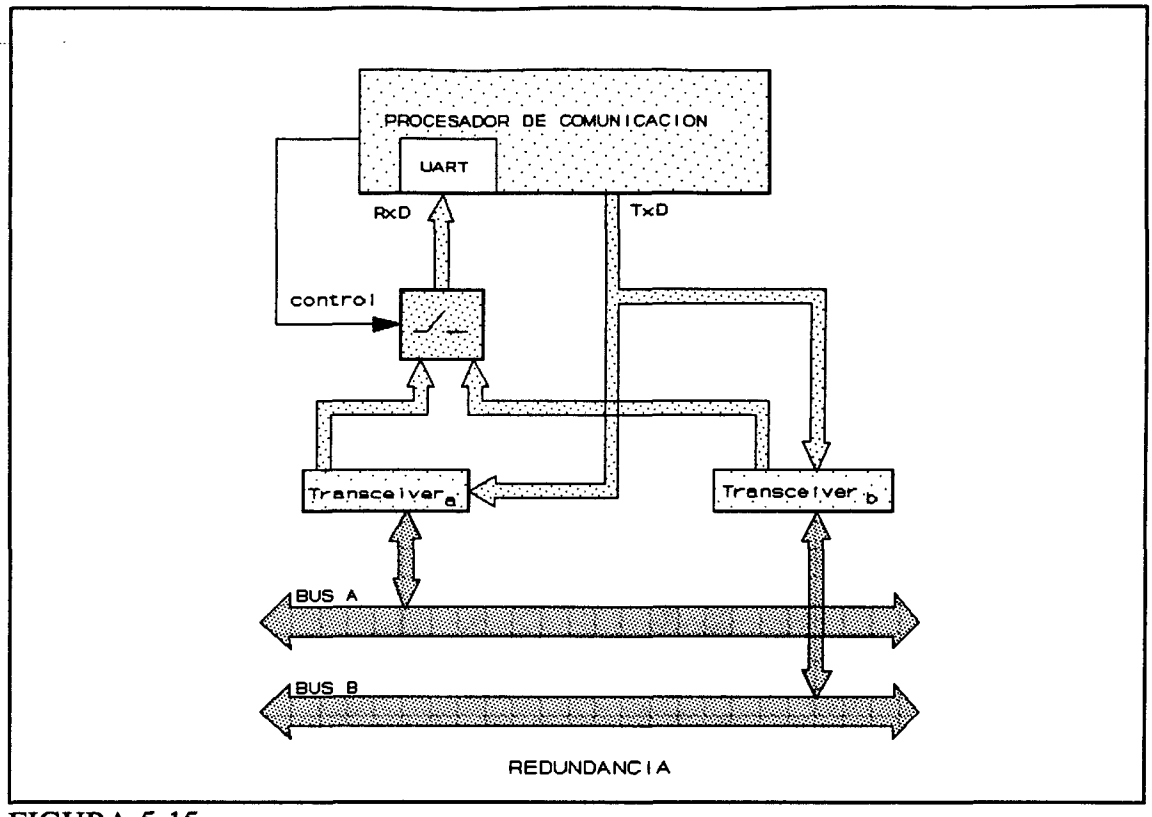


FIGURA 5.15

Como formato de carácter, se usan los caracteres de UART estándar. Este formato de carácter se explica en la sección 5.3.3. Para transmisión RS-485 así como para generación de caracteres sencillos, se dispone de importantes componentes integrados de distintos fabricantes, lo que facilita el desarrollo de los equipos de campo PROFIBUS y controlar y reducir los costos considerablemente.

La velocidad de transmisión de operación depende de la longitud del cable y puede alcanzar 500kbits/s. Se puede transmitir con un cable de longitud hasta 1200m; 9.6, 19.2 o 93.75 kbits/s, es posible hasta 600m; 187.5 kbit/s, y con una velocidad de 500kbit/s para 200m. Las longitudes de los cables son por causa de una atenuación máxima de 6db. Para un par de cable de un diámetro mínimo de 0,5mm^2, la atenuación de la señal está por debajo, así que es posible una longitud del cable doble.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Para aumentar la disponibilidad del sistema, se puede disponer de un segundo cable opcionalmente. No obstante, se instalaría a una cierta distancia del primer cable. Por razones de redundancia, cada estación del bus necesitaría dos transceivers separados, cada uno de los cuales se puede seleccionar cuando sea necesario (figura 5.15).

5.3.3 TECNICAS DE TRANSMISION

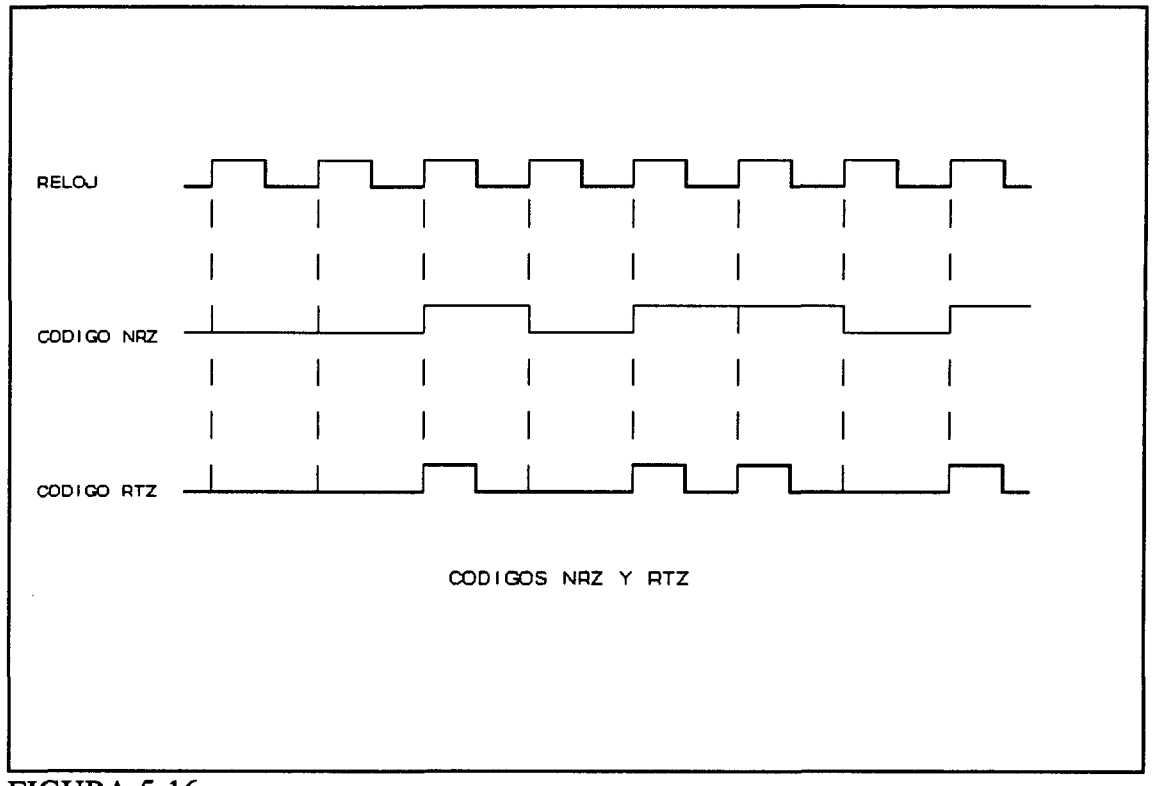


FIGURA 5.16

A diferencia de la difundida especificación RS-232 (=V.24), en la que los niveles absolutos de tensión se deben observar para poder reconocer el '1' ó el '0' lógico, en la transmisión RS-485, los caracteres se identifican solamente por la polaridad de la tensión. Incluso con señales muy ruidosas se puede regenerar en el receptor por medio de amplificadores operacionales. Esta técnica es muy adecuada para grandes tasas de transmisión (hasta 10Mbit/s) en un medio ruidoso.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Codificación del bit. Para reducir los esfuerzos de la transmisión y no depender de componentes específicos de transmisión y recepción, se transmiten los bits sencillos de un carácter generados por una UART. Esta codificación es la llamada NRZ (non-return-to-zero) y es muy común en la tecnología digital. Un bit consiste en un pulso rectangular con el ancho correspondiente del período de reloj. Un voltage o cero es '1', no voltage es '0'.

En el código RTZ (return-to-zero), no obstante, no se identifica un '1' lógico por el nivel de voltage, sino por el borde bajante en el centro del bit (figura 5.16). En este código el ancho del bit es la mitad del pulso de reloj.

Sincronización del bit. PROFIBUS usa una técnica de transmisión asíncrona, lo que significa que las frecuencias de reloj del transmisor y receptor no están sincronizadas. La transmisión es orientada a carácter. Así que, siempre que el receptor reconoce un bit de start, en el comienzo de un nuevo carácter, debe resincronizar su muestreo. La figura 5.17 muestra la estructura de tal carácter según DIN 66022/66203. Un carácter tiene un total de 11 bits:

* 1 bit de start que es siempre un '0' lógico.

* 1 bit de stop que es siempre un '1' lógico.

* 1 bit de paridad.

* 8 bits de datos.

La paridad par significa que los bits datos contienen un número par de '1'. En este caso la paridad es cero. En el caso de que el número de unos sea impar, el bit de paridad se coloca a uno.

Para asegurar una recepción segura de los datos, el bit transmitido debe muestrearse

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

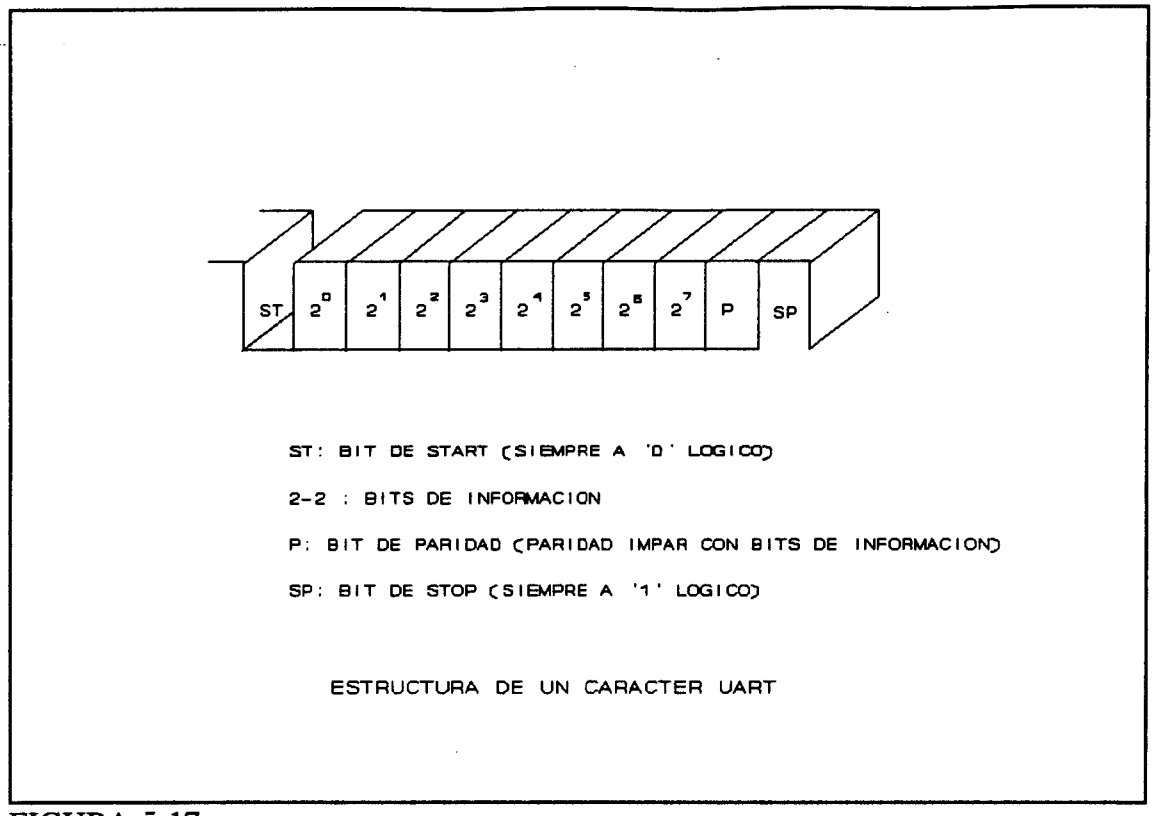


FIGURA 5.17

a la mitad de la duración del bits desde su comienzo. Esto requiere una frecuencia de muestreo del receptor que sea al menos el doble que la del transmisor. Ya que el transmisor y el receptor tienen distintos períodos de tiempo, se debe tener en cuenta las tolerancias entre las frecuencias del transmisor y del receptor. El estándar PROFIBUS permite una desviación de +/- 0.3%.

Los caracteres sencillos se transmiten según la especificación. Por ejemplo, el bit de stop del carácter nº en una trama, debe seguirse de un bit de start del carácter (n+1)º. Entre dos tramas individuales con un número de caracteres, se debe observar el tiempo inútil. Cuando no se transmite, existe un '1' lógico en el bus. Su longitud mínima es de 3 caracteres, p.e., 33 tiempos de bit.

El tiempo inútil ofrece dos ventajas:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

1. Las estaciones pasivas activan sus receptores solamente cuando el tiempo muerto ha transcurrido. Después, chequean la dirección de destino de la siguiente trama y apagan el receptor si no es la suya. Esta 'escucha' selectiva reduce el protocolo superior, así que la capacidad de procesamiento adicional se dispone para el programa de aplicación.

2. La estación que está transmitiendo puede incapacitar su transmisor durante el tiempo muerto. Después de apagarlo, la terminación de la línea proporciona el '1' lógico requerido en el medio.

Todas las estaciones activas controlan el tiempo muerto por temporizadores controladores del hardware. Un '0' lógico detectado durante este momento (causado por posibles ruidos) se interpreta como un error y normalmente se dispara de nuevo el temporizador.

5.3.4 CARACTERISTICAS ELECTRICAS

La pila de voltage es acoplado o desacoplado galvánicamente. La impedancia característica de la línea suele estar entre 100 y 150 ohm. con una f > 100 Khz. El calibre del cable es al menos de 0.22mm^2, la capacidad por debajo de 60 pF/m (EIA RS-485).

En la versión más simple consistente en la conexión de RxD/TxD y RxD/TxD-N, la diferencia de voltage entre la tierra de los datos y todas las conexiones no debe exceder de +/- 7v. Si no se conoce, se necesita introducir un equalizador de voltage (por ejemplo se usa el mismo potencial de tierra).

Además de la terminación de la línea, con una impedancia característica de 120 ohm., proporcionada por el estándar EIA RS-485, el estándar PROFIBUS especifica dos resistencias. Se debe conectar una de ellas, R_D (pulldown) con la tierra de datos, la otra, R_U (pullup), con el voltage positivo de la pila (VP) (figura 5.18). Estas resistencias adicionales

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

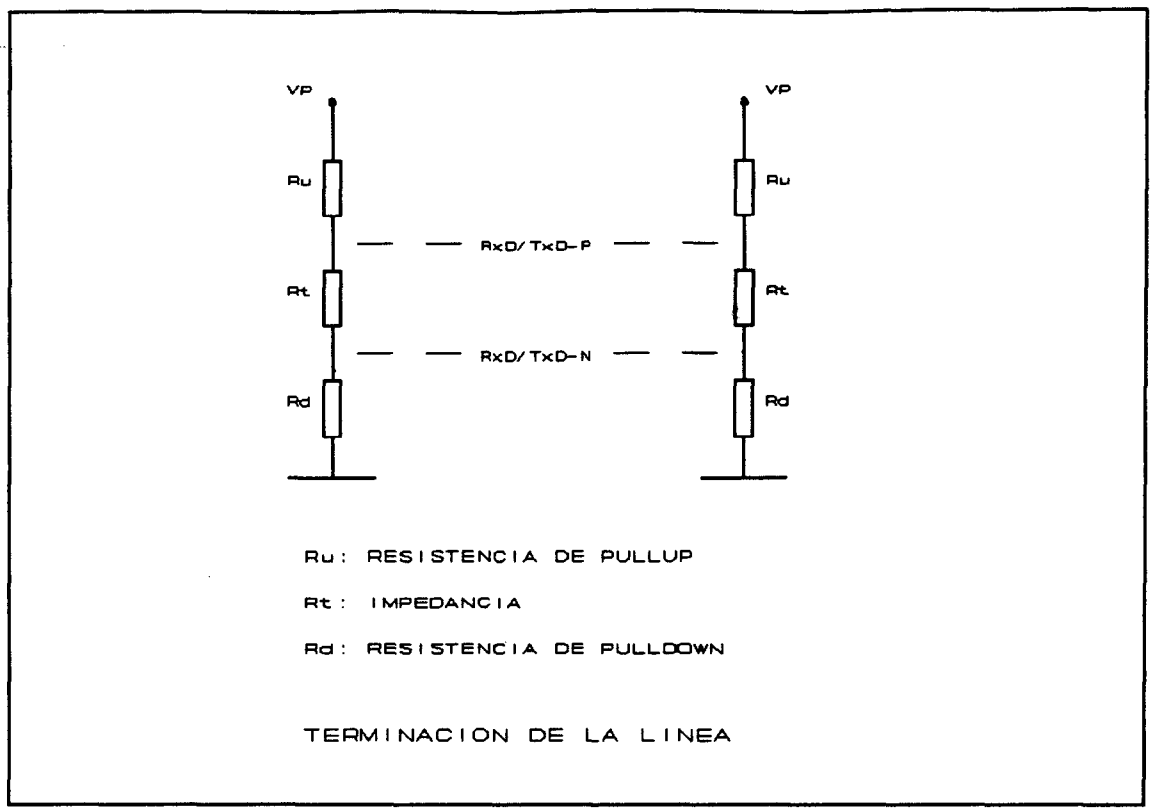


FIGURA 5.18

aseguran que la línea tenga un voltage estacionario mientras transmitan las estaciones. Si una estación tiene una terminación de línea, el voltage de la pila positivo debe proporcionarse en el pin 6 del conector SUB-D. En el pin 5D, se debe aplicar GND.

Un ejemplo concreto: cuando el voltage de la pila es 5V +/- 5%, el estándar recomienda dos resistencias adicionales de terminación de 390 ohm. con +/-2%. LA impedancia de terminación será 150 ohm. La fuente de voltage sería capaz de dirigir una corriente de un pequeño circuito de 10mA.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

ASPECTOS DE LA IMPLEMENTACION

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

6 ASPECTOS DE LA IMPLEMENTACION

6.1 PRERREQUISITOS PARA LA INTEGRACION DE SISTEMAS

Están interesadas dos tipos de personas en cuestiones de integración de sistemas: aquellas que planean integrar un sistema de campo en la organización de producción de sus compañías, y aquellas a las que les gustaría averiguarlo si y cómo un bus de campo puede satisfacer las necesidades de comunicación de sus propios productos.

El primer paso está definiendo el status quo y los pequeños, medianos y largos objetivos que una compañía desea resolver. En este análisis son de particular interés, los aspectos económicos como la competitividad de los productos, tendencias de mercado y de desarrollo, situación de la competencia y flexibilidad en las soluciones ofrecidas en la tecnología de la automatización.

El siguiente paso es identificar las necesidades técnicas para un bus de campo. La adecuación de una red para un propósito específico depende mucho de la topología de la red. Se puede adaptar, por ejemplo, a la estructura organizadora original de la compañía lo que facilitaría el período de restructuramiento. Esta estructura, no obstante, no suele ser muy eficiente, así que la adaptación de la estructura de la compañía a la topología de campo óptima sería lo conveniente.

Cuando se ha encontrado un topología adecuada, se deben considerar los problemas de funcionamiento, como:

* Velocidad de transmisión

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* longitud de red máxima
* máxima distancia entre estaciones simples

Estos tres puntos influencian, por varios motivos, en criterios para determinar la adecuada técnica de un bus de campo que incluya las necesidades de tiempo que se encuentran en el medio de comunicación por el medio de producción o la función de un equipo individual.

Después de que se determinen las necesidades de funcionamiento, se debe definir la funcionalidad del bus de campo. Esto también incluye información como qué clase de servicios de operación de un sistema de bus de campo necesita, qué métodos de acceso deben ser posibles y cómo se distribuirá la carga en la red. En muchos casos, una configuración de una maestra con muchas esclavas es suficiente ('sistema monomaestro'). Un área de aplicación importante de los sistemas monomaestros son los controles de direccionamiento rápidos.

Un sistema monomaestro se puede extender a varios maestros, así que cada maestro podría comunicar con los esclavos correspondientes pero no con los otros maestros. Este sistema puede ser fácil de configurar manualmente. Los equipos de campo, por ejemplo, se pueden controlar por un controlador programable (PLC) y transmitir datos actuales a un PC para visualización de proceso. En ambos casos, el PLC y el PC son maestras, pero no necesitan comunicarse.

Finalmente, las posibles soluciones se deben evaluar bajo aspectos económicos. Se debe analizar la complejidad de integración y se debe determinar la disponibilidad de productos necesarios para el bus de campo. Se debe tomar una decisión lo más apropiada posible.

6.2 PLANIFICANDO Y CONFIGURANDO UNA RED PROFIBUS

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

6.2.1 PARAMETRIZACION DE ESTACIONES DE BUS

Planificar y diseñar un sistema PROFIBUS significa establecer su estructura lógica. La configuración de una red PROFIBUS incluye la creación de Listas de Relaciones de Comunicación, Diccionarios Objetos y parámetros de bus y el paso subsiguiente de los perfiles de configuración a las estaciones del bus. Como opuesto a configuración, el diseño describe la planificación de los valores en los perfiles de configuración. No siempre existe una buena distinción entre estos dos términos.

La configuración incluye la selección de la estructura de comunicación lógica y los parámetros correspondientes que sean necesarios para el software del protocolo de PROFIBUS:

* Fijación de los parámetros de bus (baudios, tiempos,...)

* Descripción de relaciones de comunicación entre estaciones (Lista de Referencia de Comunicación).

* Definición de objetos de comunicación de todas las estaciones (Diccionario Objeto).

* Selección de los servicios necesarios para la transferencia de datos

Aunque se generan los perfiles de configuración, se pueden chequear para detectar errores lo antes posible. Un error es una violación de un acuerdo de la regla del estándar PROFIBUS.

Los errores simples ocurren cuando se excede del rango de los valores permitidos para un parámetro. Los errores más complejos, no obstante, pueden ocurrir cuando se configura un sistema completo, en lugar de una estación simple. Este error es especialmente

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

obvio cuando se genera una Lista de Relación de Comunicación. Una relación de comunicación se describe por entradas en la Lista de Relación de Comunicación incluyendo a ambas estaciones. Solamente se puede establecer y usar exitosamente una conexión en operaciones posteriores, si ambas entradas se emparejan una con la otra.

Un establecimiento de conexión se cancela, por ejemplo, si una de las estaciones en esta conexión permite servicios como cliente, pero la otra no los proporciona como servidor. Por lo tanto, es importante establecer consistentemente las Listas de Relación de Comunicación y los Diccionarios Objetos de todos los equipos de la red. Para lograr consistencia, no se necesita en la mayoría de los casos equipo especial. La cantidad de las soluciones posibles se limita por las configuraciones existentes. Además, los parámetros de la mayoría de los equipos de campo los fijan los vendedores, así que en muchos casos la dirección de la estación se debe seleccionar (normalmente por medio de un conmutador). En estos equipos, se suele reconocer y seleccionar automáticamente las cantidades de transmisión.

En los sistemas monomaestros o en las redes que no proporcionan comunicación maestra-maestra, cada maestra debe configurarse según sus esclavas que tienen fijados los parámetros. La parametrización de tales sistemas es en la mayoría de los casos tan fácil que se puede hacer manualmente in situ. Los parámetros de los esclavos pueden ser vistos desde las respectivas hojas de datos o configuraciones. El equipo de simple configuración para tales aplicaciones debe incluso ser parte del software de aplicación de los Pcs o PLCs.

El vendedor puede proporcionar diferentes posibilidades para parametrizar los datos de configuración en un equipo. Uno de estos es para generar datos ASCII con un editor convencional. Los datos de configuración sólo pueden entrarse por teclado (por ejemplo un PC). Para este propósito, es posible usar relaciones de comunicación preconfiguradas (por ejemplo, de configuraciones de sensores/actuadores).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Los datos de configuración se pueden pasar también de una estación a otra por la red durante la operación. Para esto, se pueden usar los servicios de manejo remoto del FMA7.

El estudio positivo no solamente asegura que la red trabaje, sino que también satisfaga los siguientes criterios:

* Optimización del rendimiento total (cantidad de datos de usuario)

* Tiempos de respuesta mínimos

* tiempo de entrada mínimo de las nuevas estaciones en el anillo lógico

* Necesidades de recursos mínimas

6.2.2 RELACIONES DE COMUNICACION

La planificación de la Lista de Relación de Comunicación significa determinar entre qué estaciones existirá relaciones de comunicación. Sólo se transfiere información entre dos estaciones si se estableció una relación de comunicación entre ellas.

Así, la primera tarea del diseñador es determinar entre qué estaciones habrá intercambio. Por supuesto, sería fácil para el proyectista establecer una relación entre todas las estaciones de la red. Para las redes con algunas maestras y varias esclavas es un buen paso. Pero, con un gran número de estaciones es prácticamente imposible ya que cada relación de comunicación necesita un cierto espacio de memoria, dependiendo de la implementación.

Esta solución no es específica porque el sistema de comunicación no está adaptado a específicamente a las necesidades de la aplicación. Solamente se puede diseñar

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

razonablemente los parámetros que describen una relación de comunicación, si se conoce las necesidades exactas de la aplicación.

Pero, ¿Cuáles son las necesidades de una aplicación? La siguiente lista de control los determina:

1. ¿Se debe transmitir cierta información a una o más estaciones?

Cuando cierta información debe alcanzar a varias estaciones simultáneamente, es apropiada una comunicación sin conexión por broadcast y multicast. Un tipo BRCT (broadcast) o MULT (multicast) de comunicación se puede usar entonces.

2. ¿Son ambas estaciones maestras o una maestra y otra esclava? ¿Debe ser capaz la esclava de iniciar el envío de mensajes a la maestra?

MMAC	Acíclica maestra/maestra
MSAC	Acíclica maestra/esclava
MSAC_SI	Acíclica maestra/esclava con iniciativa de la esclava

3. ¿Se accede a menudo a un objeto de comunicación o debe ser el tiempo de acceso lo menor posible?

En este caso, se pueden usar los servicios cíclicos de PROFIBUS:

MSCY	cíclica maestra/esclava
MSCY_SI	cíclica maestra/esclava con iniciativa de la esclava

4. ¿Ambos usan frecuentemente la relación de comunicación?

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Si el cliente usa servicios paralelos, puede transmitir varias peticiones de una vez sin esperar respuesta del servidor. Esto aumenta significativamente la eficiencia de las aplicaciones que necesitan transmitir un número de peticiones en cortos intervalos de tiempo. El número de servicios paralelos, no obstante, es un factor que necesita considerarse cuando se calcula el espacio de memoria de la relación de comunicación.

5. ¿Es importante disponer rápidamente de la relación de comunicación?

Si es así, es razonable mantener la conexión permanentemente del control. Cuando no se transmitan datos de usuario sobre la conexión, la vigilancia de conexión controla el mantenimiento de la conexión generando cortas tramas inútiles. El intervalo de tiempo para esto, se puede determinar para cada conexión con el Intervalo de Control Acíclico (ACI) o el Intervalo de Control Cíclico (CCI).

6. ¿Necesita una esclava ser polleado más frecuentemente que otros conectados a una esclava?

A través de múltiples entradas de una esclava en la Lista de Polling, puede direccionarse con mucha más frecuencia. El factor de multiplicación se puede seleccionar por el parámetro MULT en la relación de comunicación.

7. ¿Existe una aplicación que cambie de pareja de comunicación pero que sólo tenga una a la vez? ¿La aplicación es exclusivamente servidor o también cliente?

En una configuración en la que una estación intercambie datos con otras estaciones, sería teóricamente necesario establecer relaciones con todas las estaciones. Esto, no obstante, implicaría un gran número de entradas en la Lista de Relación de Comunicación de esta estación, así que requeriría mucho espacio de memoria. Si las condiciones mencionadas previamente se conocen, los recursos disponibles se pueden usar por varias estaciones una

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

después de otra. El atributo de conexión 'I' se introduciría en la Lista de Relación de Comunicación para un cliente con iniciativa abierta y 'o' para un respondedor abierto. En los cálculos disponibles, no obstante, se debe tener en cuenta el tiempo necesario para establecer una conexión y liberarla.

8. ¿Qué tamaño tienen los objetos de comunicación que se transmitirán en esta relación de comunicación?

Para cada relación de comunicación, se debe reservar un cierto número de memoria. Su tamaño depende del tamaño máximo de PDU que se puede transmitir sobre la conexión. Entrando el tamaño máximo de la PDUs en la Lista de Relación Comunicación, se pueden adaptar los tamaños de buffer a los objetos de comunicación.

9. ¿Qué servicios soporta la aplicación en la relación de comunicación?

Los servicios que soporta la aplicación como servidor o cliente se pueden introducir usando el parámetro de Servicios Soportados. Es una forma de controlar si un servidor soporta los servicios que la estación pareja proporciona como cliente.

Si se han considerado las cuestiones anteriores, el diseño específico del software de comunicación prestará óptimos resultados con respecto a la utilización de los recursos, tiempos de respuesta, cantidad y eliminación de errores. Sería posible también añadir nuevas aplicaciones en las estaciones existentes sin mayores modificaciones al diseño de los sistemas de comunicación. Por esta razón, y también para distinguir los procesos de aplicación pertenecientes a una estación por sus referencias de comunicación respectivas, se podría establecer más de una relación de comunicación entre dos estaciones.

Puede ser muy complicado establecer manualmente la Lista de Referencia de Comunicación para una estación con un editor convencional. Se aplica particularmente a las

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

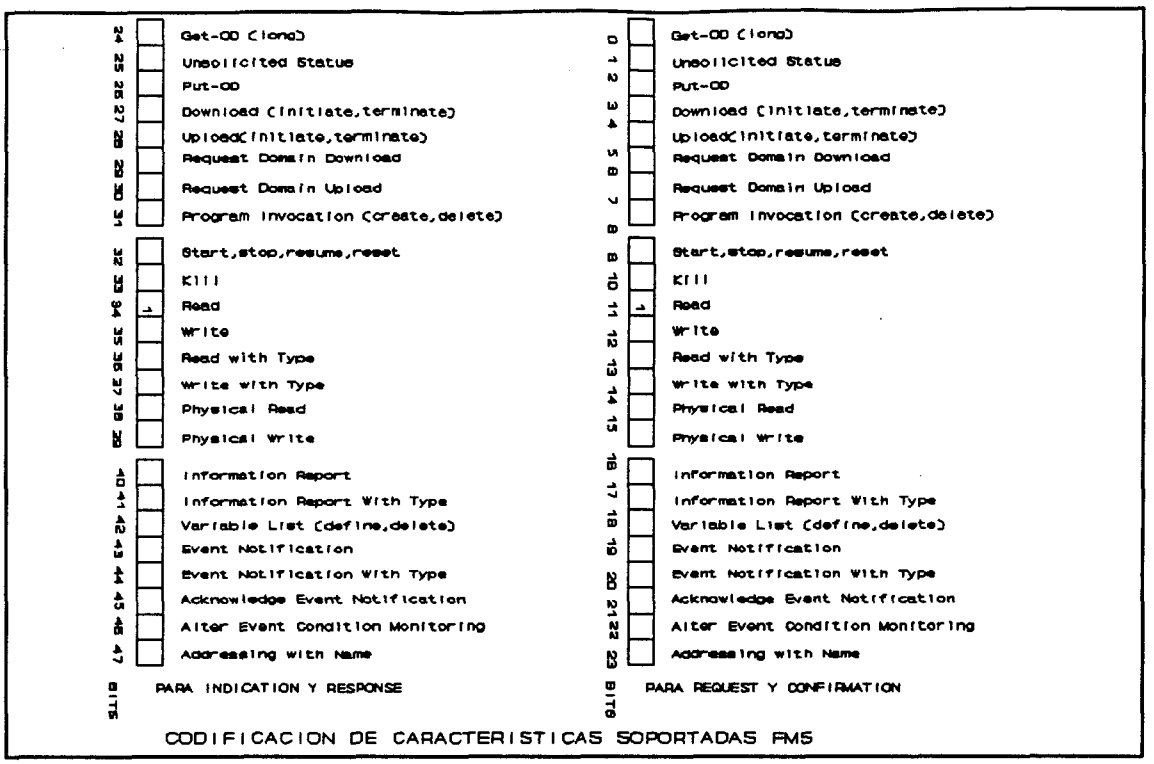


FIGURA 6.1

entradas en las columnas FMS de características soportadas por el servidor o cliente. La codificación de los parámetros FMS de características soportadas se muestra bit a bit en el ejemplo de la figura 6.1. (En este caso corresponde a 00 20 00 20 00= Sólo lectura).

Si, además, la red necesita interconexiones complicadas, el diseñador necesita equipo adicional para diseñar su red. Puede ser una herramienta de configuración que sería parte del firmware de aplicación. Para el diseño de la red que considere varias estaciones al mismo tiempo, se puede usar un configurador, que trabaje tanto como una estación PROFIBUS independiente o como parte de un host central. En la estación de configuración central, se generan los conjuntos de datos, la herramienta de configuración asegura consistencia de los datos de configuración. Cada estación tiene sus propios conjuntos de datos de configuración.

6.2.3 DICCIONARIO OBJETO

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

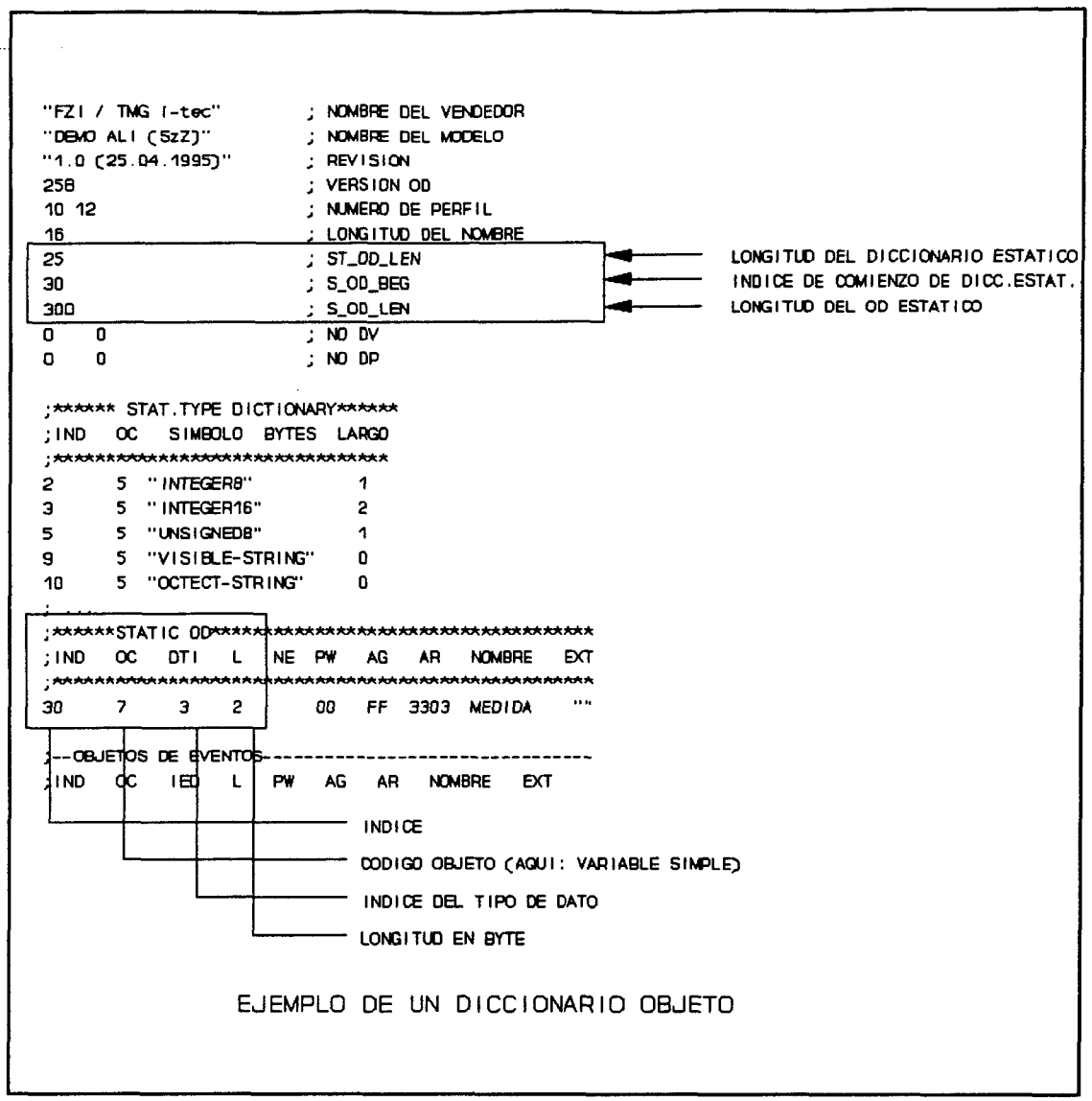


FIGURA 6.2

El Diccionario Objeto (OD) ya tiene sus propios datos de configuración. En el diseño del OD, el diseñador debe determinar qué datos, p.e. objetos de proceso, se pueden disponer en otras estaciones durante la operación última, lo que significa que los objetos de proceso deben convertirse en objetos de comunicación. El diseñador crea una OD fuente para cada estación en la que se listan todos los objetos de comunicación que una estación hace disponible a otra.

Varios equipos de campo han fijado los Ods fuentes, que el usuario encontrará en la hoja de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

datos del equipo.

Durante la operación de la red, una estación puede usar el servicio GetOD para obtener una copia del Diccionario Objeto de la entidad pareja. Esta copia proporciona toda la información necesaria para el acceso a los objetos de comunicación de la estación pareja.

Si los datos son sensibles, se pueden proteger por mecanismos de control de acceso proporcionados por PROFIBUS. Los objetos de comunicación protegidos se pueden solamente acceder por grupos de accesos seleccionados o personas que conocen la clave. Además, es posible limitar los servicios de objetos protegidos por cada estación que tenga Derecho de Acceso (p.e., sólo lectura). En la figura 6.2. se muestra un ejemplo de OD que se puede leer como datos ASCII por un PC. Se señalan las entradas más importantes.

6.2.4 PARAMETROS DE BUS

Los parámetros operacionales de la capa 2 incluyen los parámetros de bus y parámetros específicos de la estación. Después del encendido, cuando se inicializan los parámetros de operación según el estándar PROFIBUS parte 1 de [DIN 91a].

Los parámetros son los mostrados en la tabla 6.1.

Los parámetros de bus deben ser iguales para todas las estaciones del bus, mientras que los parámetros específicos de la estación se pueden fijar independientemente para cada estación. Los siguientes parámetros son obligatorios para cada estación:

* Dirección TS de la estación ('This Station').

* Volumen de transmisión (Reconocido y seleccionado automáticamente por el

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

TABLA 6.1

PARAMETRO	SIMBOLO	CLASE
TARGET ROTATION TIME	TTR	PARAMETRO DE BUS
GAP UPDATE FACTOR	G	PARAMETRO DE BUS
DIRECCION DE LA ESTACION	TS	ESPECIFICO ESTAC.
DATA SIGNALING RATE	BAUD_RATE	PARAMETRO DE BUS
REDUNDANCY	MEDIUM_RED	PARAMETRO DE BUS
STATE OF HARDWARE RELEASE	HW-RELEASE	ESPECIFICO ESTAC.
STATE OF SOFTWARE RELEASE	SW-RELEASE	ESPECIFICO ESTAC.
SLOT TIME	TSL	PARAMETRO DE BUS
MINIMUM STATION DELAY TIME	MIN_TSDR	PARAMETRO DE BUS
MAXIMUNM STATION DELAY TIME	MAX_TSDR	PARAMETRO DE BUS
REPEATER SWITCH TIME	TQUI	PARAMETRO DE BUS
DESIRE TO ENTER RING	IN_RING_DESIRED	ESPECIFICO ESTAC.
HIGHEST STATION ADDRESS	HSA	PARAMETRO DE BUS

PARAMETROS DE OPERACION PROFIBUS

esclavo).

* Maestra o esclava (maestra: in_ring_desired=1)

Para optimizar el sistema, los parámetros más importantes son el tiempo T_{TR} (Target Rotation TIme) y el factor G (GAP Update Factor). El T_{TR} depende del número de estaciones activas en el bus y de la cantidad de datos para transmitirse. Se seleccionarían de qué forma cada estación activa es capaz de ejecutar un número suficiente de ciclos de mensaje cuando recibe el testigo. G influye directamente en el tiempo que una nueva estación activa necesita para entrar en el anillo lógico.

En un esclavo, estos dos parámetros no importan. Para configurar óptimamente un esclavo, sólo se debe fijar el Tiempo mínimo de retraso de la estación.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

En la figura 6.3. se muestra un ejemplo que, después del comienzo, pasa los parámetros del bus a la estación (p.e. un PC). Se resaltan las entradas más significativas. En la estación, la dirección se lee también sobre los datos de configuración. Algunas veces es posible también fijar la dirección mediante un conmutador. El volumen de baudios (velocidad de transmisión) se fija según el valor leído de la serie de datos. Para el parámetro in_ring_desired, un flag determina si es un maestro o esclavo.

```
; PARAMETROS DEL BUS
C800          ; DIRECCION DEL SEGMENTO DE LA TARJETA CONTROLADORA
4             ; BAUDRATE = 500 KBAUD
500           ; Tsdr
500           ; Tidle2
2000          ; Tslot
65535         ; Ttr
1             ; GAP UPDATE
63            ; HSA
0             ; DIRECCION DE SEGMENTO
1             ; DIRECCION DE ESTACION
1             ; EN ANILLO DESEADO
500           ; Tidle1
0             ; Tqui
1             ; MAX RETRY LIMIT
0             ; PREAMBULO


FICHERO PARA LA LECTURA DE LOS PARAMETROS DEL BUS POR UN PC
```

FIGURA 6.3

6.2.5 EJEMPLO

Los tres controladores (maestros) se conectan con un máximo de 24 equipos de campo (sensores/actuadores como se muestra en la figura 6.4). Ya que sólo se demuestran los principios de procedimiento, se hace las siguientes simplificaciones:

Como las herramientas de la máquina para controlarse deben trabajar con gran

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

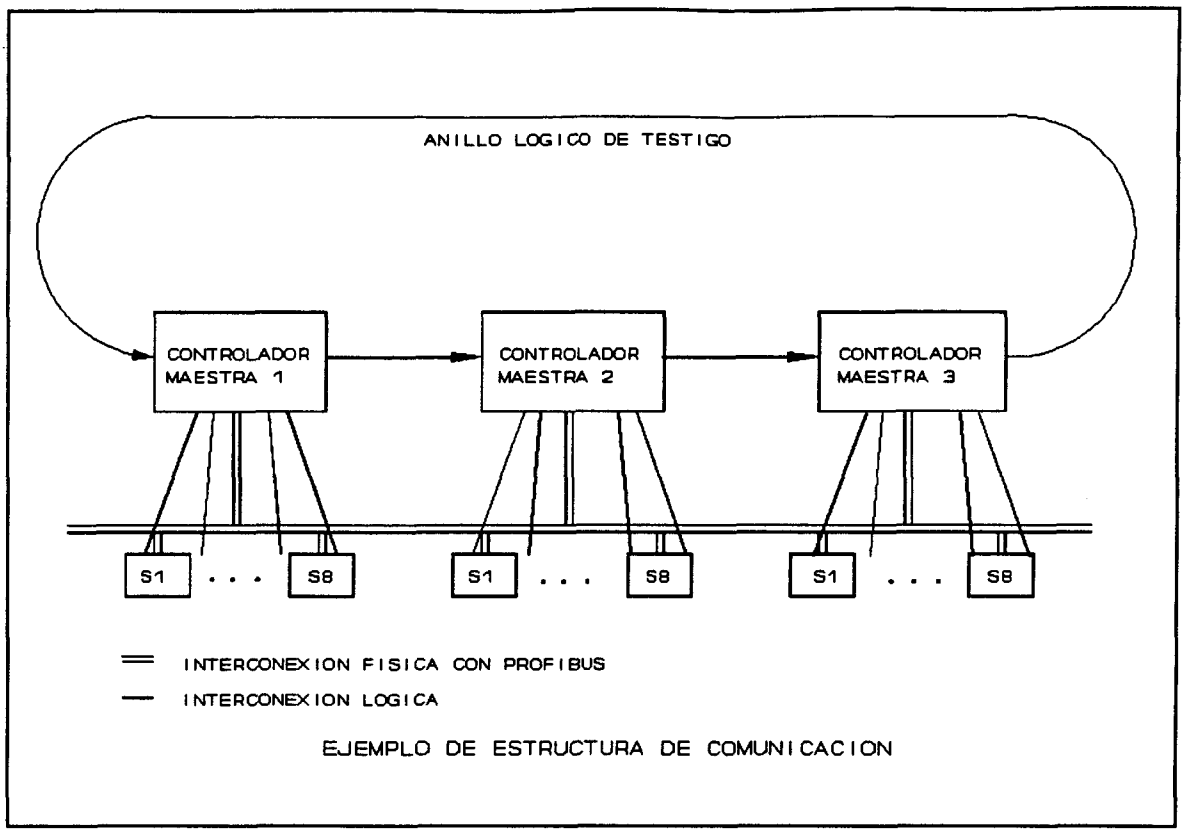


FIGURA 6.4

precisión, el tiempo de ciclo total no debe exceder de 200ms., p.e., después cada tiempo de los sensores conectados debe haberse polleado como mínimo una vez. Además, en este tiempo se debe posibilitar reactuar por los valores de sensores cambiados. Esto significa que en nuestra red PROFIBUS, en 200ms se debe haber completad un ciclo entero de comunicación. Para mantener la simplicidad de los cálculos, siempre se asume una longitud de trama fijada de 14 bytes (incluida la cabecera de la trama y el final).

Ya que cada byte transmitido consiste en 11 bits, la trama tiene un tamaño total de 154 bits. Si asumimos un volumen de transmisión de 500Kbits/s con 2us cada bit, resulta un intervalo total de 308us. En este modelo, cada una de las tres maestras (controladores) debe hacer un polling a 8 esclavas (sistemas sensores/actuadores). De esta forma, necesitan 8 tramas de peticiones y de respuesta respectivamente. El polling se realiza sobre conexiones cíclicas.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Si para el control restante, el procesamiento de trama y tiempos de respuesta, se asume un total de 120us, el tiempo T_{MX} para cada maestra es:

$$T_{MX} = 8 * (2 * 308 \text{ us} + 120 \text{ us}) = 5888 \text{ us} ---> 6\text{ms}$$

Consecuentemente, cada una de las tres maestras necesita al menos 6ms para la comunicación con las esclavas conectadas. Ya que cada trama puede reintentarse por segunda vez después de un error de transmisión, en el peor de los casos tres veces, el tiempo de transmisión, apróx. 18ms, debe calcularse.

Si el tiempo de mantenimiento de testigo se parametriza a 30ms, se puede descuidar la pequeña cabecera para el manejo del testigo. Significa, por ejemplo, que después de un máximo de 3 * 30ms=90ms, todos los sistemas sensores/actuadores se han polleado como mínimo una vez por las maestras correspondientes. El tiempo de ciclo total previamente mencionado de un máximo de 200ms puede ser logrado fácilmente.

6.2.6 HERRAMIENTAS

La aceptación difundida de PROFIBUS como un estándar de campo abierto depende de las herramientas adecuadas que los usuarios soporten en el desarrollo e implementación de los componentes y sistemas PROFIBUS.

Las herramientas ayudan al usuario para:

* planificar

* diseñar

* configurar

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* actualizar

* servicio

sistemas abiertos de PROFIBUS. Incluso en el nivel de campo, un diseñador necesita normalmente estas herramientas de desarrollo. Un factor mayor para considerarse es la integración del medio de comunicación en un concepto de aplicación específica.

Frecuentemente, los usuarios ya tienen herramientas de vendedores específicos. Pero la flexibilidad y la simple integración son propiedades importantes de una herramienta de desarrollo.

Control PROFIBUS.

El control PROFIBUS permite una red terminal y análisis de protocolo. Un controlador de bus es una estación pasiva que registra las señales eléctricas en el canal de transmisión, da a cada trama recibida una marca de tiempo (medido en tiempos de bits y milisegundos)y las analiza. Dependiendo de la conveniencia de la herramienta, el análisis incluye varios niveles de protocolo. En PROFIBUS, tienen importancia los protocolos de las capas 2 y 7, y el nivel de bit en el bus. Los errores de protocolo correspondientes se graban y producen estadísticas.

El control de PROFIBUS capacita al usuario a ver la comunicación en el canal en tiempo real y a visualizar las estaciones de bus activas. La transmisión de la trama se registra y se puede recapitular después paso a paso. Esto permite la detección de hardware de red defectuoso y errores en el software o el programa de aplicación. El volumen de datos para registrarse se puede opcionalmente delimitar por filtrado de cierta clase de tramas.

En la mayoría de los casos, es útil que una estación pasiva realice el control de bus

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

y no afecte a la transmisión (manejo de testigo,...). Para actualización del sistema, es beneficioso en algunos casos que el controlador mantenga las relaciones de comunicación como una estación activa e intercambiar tramas de datos con otras estaciones, así se comporta como una estación regular y afecta el temporizador y las secuencias lógicas en la red PROFIBUS.

Para el soporte de control de servicio y sistema, los controladores PROFIBUS frecuentemente tienen una interfase para almacenar datos estadísticos regulares o sucesos extraordinarios en una memoria externa, o registrarlos en una impresión.

Otras funciones importantes de un controlador de bus son:

* detección de error

* condiciones de disparo seleccionables

* macros de mesas de mando seleccionables

* Función de almacenamiento por protocolos

Herramienta de Configuración de PROFIBUS.

Como una herramienta de software, la herramienta de configuración facilita el diseño y parametrización de red definiendo ayudas de todos los datos de configuración requeridos por PROFIBUS. Soporta, por ejemplo, la generación y modificación de conjuntos de datos específicos de comunicación como parámetros de bus, CRL u OD. La instrumentación controla la consistencia de los datos entrados y notifica inconsistencias, en cada entrada individual, en la lista (error CRL u OD) o en el sistema (p.e. diferentes baudios en distintas estaciones). A través de una dirección de usuario apropiada, los errores pueden quitarse

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

incluso después de haberse producido.

Desde el punto de vista del usuario, existen dos requisitos básicos que una herramienta de configuración debe conocer:

* EL uso de la herramienta debe requerir como sea posible conocimiento técnico de las operaciones de protocolo interno PROFIBUS.

* Sólo existe una herramienta estándar para todos los componentes de la red, incluso de diferentes vendedores.

La llamada a una herramienta de configuración universal necesita una interfase estándar para la transferencia del dato designado. Ya que todos los equipos tienen una interfase PROFIBUS, la configuración en línea central sobre el bus sería ideal. Para este propósito, no obstante, cada estación debe soportar ciertos servicios de manejo de red para configurarse. Si no, es necesario un gran esfuerzo para configurarla a las especificaciones del vendedor.

Herramientas para puesta en marcha y servicio.

La herramienta de puesta en marcha genera un conjunto de datos de configuración para cada estación PROFIBUS de la red en estrecha cooperación con la herramienta de configuración mencionada anteriormente. Para este propósito, dirige una base de datos que contiene toda la información específica relevante vendedor-equipo y los datos de configuración de la red existente.

Es deseable actualizar el proceso de diseño por medio del uso de la herramienta. Para este propósito, deben ser todas las estaciones capaces de leer y escribir los grupos de datos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

vía PROFIBUS. La herramienta debe tener completa capacidad de manejo PROFIBUS según el estándar. Además, los servicios de manejo necesitan complementarse en todas las estaciones involucradas.

Es importante actualizar la red paso a paso. Los componentes de la red se integran en la red uno a uno, y en el caso ideal, completamente testeados después de que cada estación se ha conectado. Para tales tests, la herramienta de puesta en marcha debe emular a las otras estaciones que no se han integrado todavía en la red pero que más tarde establecerán una relación de comunicación. El componente bajo testeo emula las siguientes operaciones: establece conexiones con las estaciones emuladas y transfiere datos sobre el bus. Para testeo real y detección de error, la herramienta de puesta en marcha debe permitir simulación directa de los equipos de campo.

6.3 FUNCIONAMIENTO DE MEDIDA Y EVALUACION

6.3.1 RAZONES PARA EL ANALISIS DEL FUNCIONAMIENTO

Es la tarea del ingeniero de automatización cuando planea y diseña una red PROFIBUS para determinar los valores de varios parámetros PROFIBUS tan óptimamente como sea posible. El número de posibilidades aumenta en proporción con el tamaño de la red. Si se ajustan algunos parámetros vitales incorrectamente, la capacidad de la comunicación de la red se reduce, si no es completamente dañada. Por lo tanto, el diseñador necesita una herramienta que la ayude en el ajuste de los parámetros consistente, correcta y óptimamente. Para esto, no obstante, se debe conocer la influencia de cada parámetro en la función de red de PROFIBUS.

Mientras se ajusta el sistema de bus de campo, se hace obvio que su funcionamiento real es más bajo de lo esperado, se deben encontrar y eliminar la existencia de puntos inseguros y cuellos de botella. En este caso, el análisis de funcionamiento trata a los

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

componentes de campo y al sistema de comunicación.

Si se planea una expansión de la red existente por los nuevos componentes, se debe analizar por otro lado si la red seguirá siendo capaz de satisfacer las necesidades futuras mínimas. Para este propósito, se usa una herramienta de pronóstico que analiza el funcionamiento cambiado en una configuración modificada.

En conclusión, se necesita también en el área de campo un análisis bien fundado y de funcionamiento cuidado.

6.3.2 METODOS Y ESTRATEGIAS

La razón para el análisis de funcionamiento es determinar cómo el cambio de los parámetros del sistema influyen a las variables de funcionamiento definidas previamente (figura 6.5) y describirlo de la forma más compacta posible. Estos parámetros son también conocidos como variables del sistema. Las variables de entrada son parámetros de carga de trabajo que describen el número de cantidad de comandos transmitidos y que expresan la capacidad de procesamiento del sistema bajo testeo.

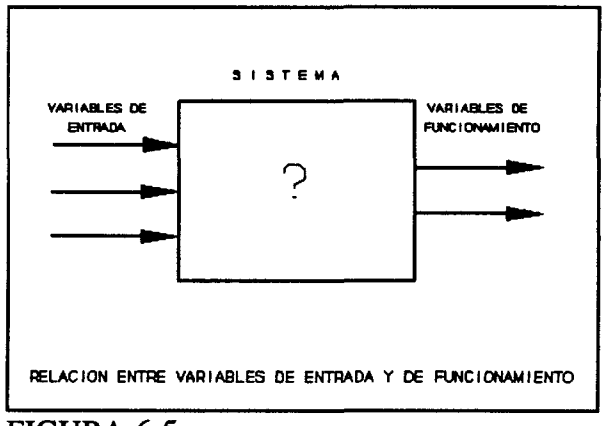


FIGURA 6.5

El funcionamiento de un sistema se puede, por lo general, analizar por las siguientes aproximaciones aceptadas (figura 6.6):

1. En la primera aproximación, modelado, se deben determinar primero las características básicas del sistema. Se combinan en un modelo abstracto de sistema, desde

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

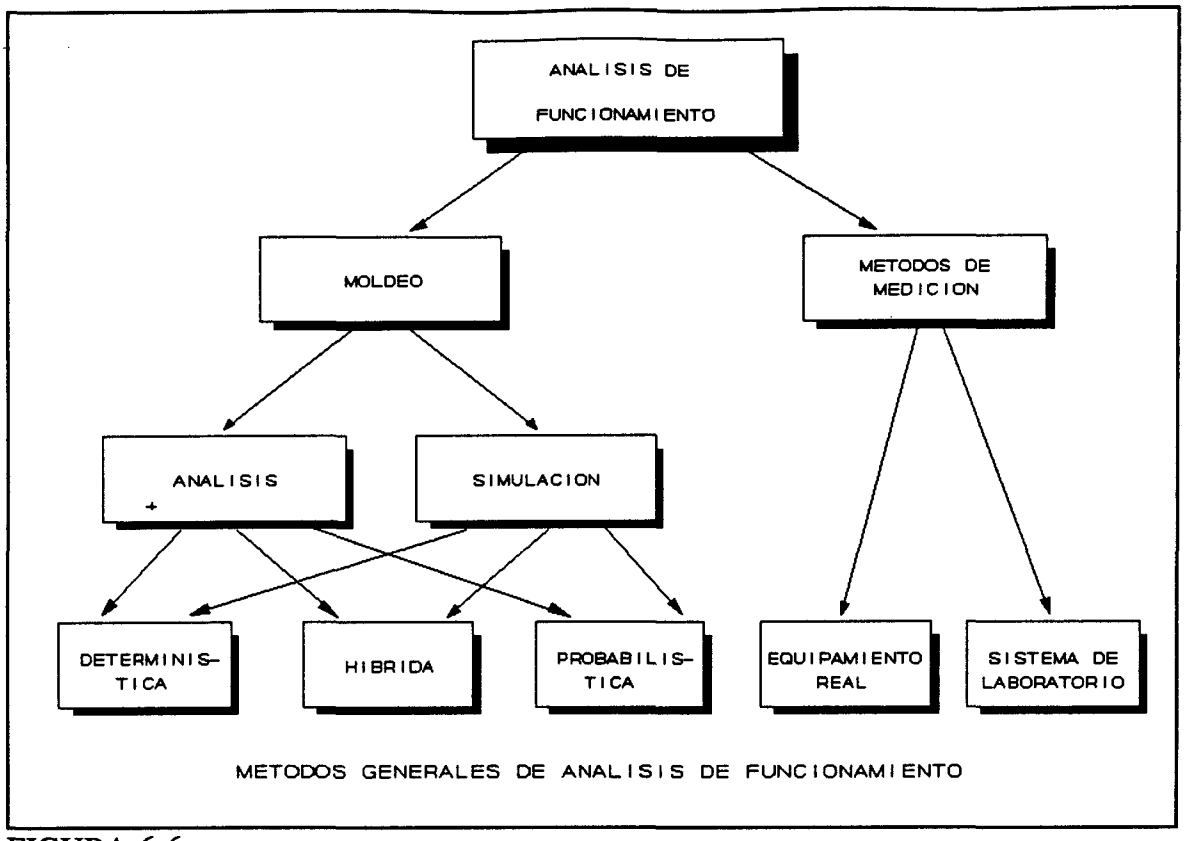


FIGURA 6.6

el que se puede analizar o simular cada funcionamiento:

. En la aproximación analítica se hace por métodos matemáticos.

. En la simulación, se emulan operaciones reales por medio de un ordenador.

No obstante, en este método es relativamente complicado un análisis correcto de hechos esenciales. Frecuentemente, son tan complejos que necesitan primero abstraerse y simplificarse, lo que es usual en donde se hacen los supuestos erróneos que conducen a errores apenas detectables. Por esto, cada modelo debe validarse y compararse con la realidad.

2. El funcionamiento se puede analizar también por la medición. En esta aproximación, se da por sentado la existencia física del sistema, qué debe ser la red real bajo

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

estudio o sistema de laboratorio. Los estados y reacciones del sistema se pueden recoger muy precisamente por herramientas de medidas específica, normalmente monitores. Con esta aproximación, se debe notar que las características de implementación siempre influencian los resultados de medidas. Por esto, no pueden hacerse los estamentos generales que se hacen típicamente en el modelo de aproximación.

Antes de elegir cualquiera de las aproximaciones arriba mencionadas, se deben determinar las propiedades, por las que se describirá el funcionamiento del sistema bajo estudio. Estas variables de funcionamiento indican cómo de bien puede un sistema hacer frente a una carga dada. Algunas variables de funcionamiento usadas comúnmente son capacidad de producción, retraso de transmisión, tiempo de respuesta, rendimiento, utilización de capacidad y utilización de recursos.

En particular, la descripción de carga contiene la tasa llegante de las respuestas, las necesidades de trabajo de las respuestas que los componentes del sistema deben satisfacer y el flujo de datos entre los componentes.

Los métodos usados normalmente para evaluar el funcionamiento de PROFIBUS se basan en una aproximación combinada de medición y simulación: mientras se simulan ciertas situaciones de carga, se miden las variables interesantes de funcionamiento en redes específicas de medición, la red de medida de funcionamiento. Las situaciones reales de carga de redes de campo se modelan antes de medición y la simulación mientras la medición.

6.3.3 HERRAMIENTAS PARA MEDIDA DE FUNCIONAMIENTO

El complejo medio de testeo que se usa normalmente para los sistemas PROFIBUS y para ejecución de muchas medidas, llamado red de medida de funcionamiento, es una combinación de una o más herramientas de medida y uno o mas generadores de carga (figura 6.7). Ambas estaciones de la red de medida

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

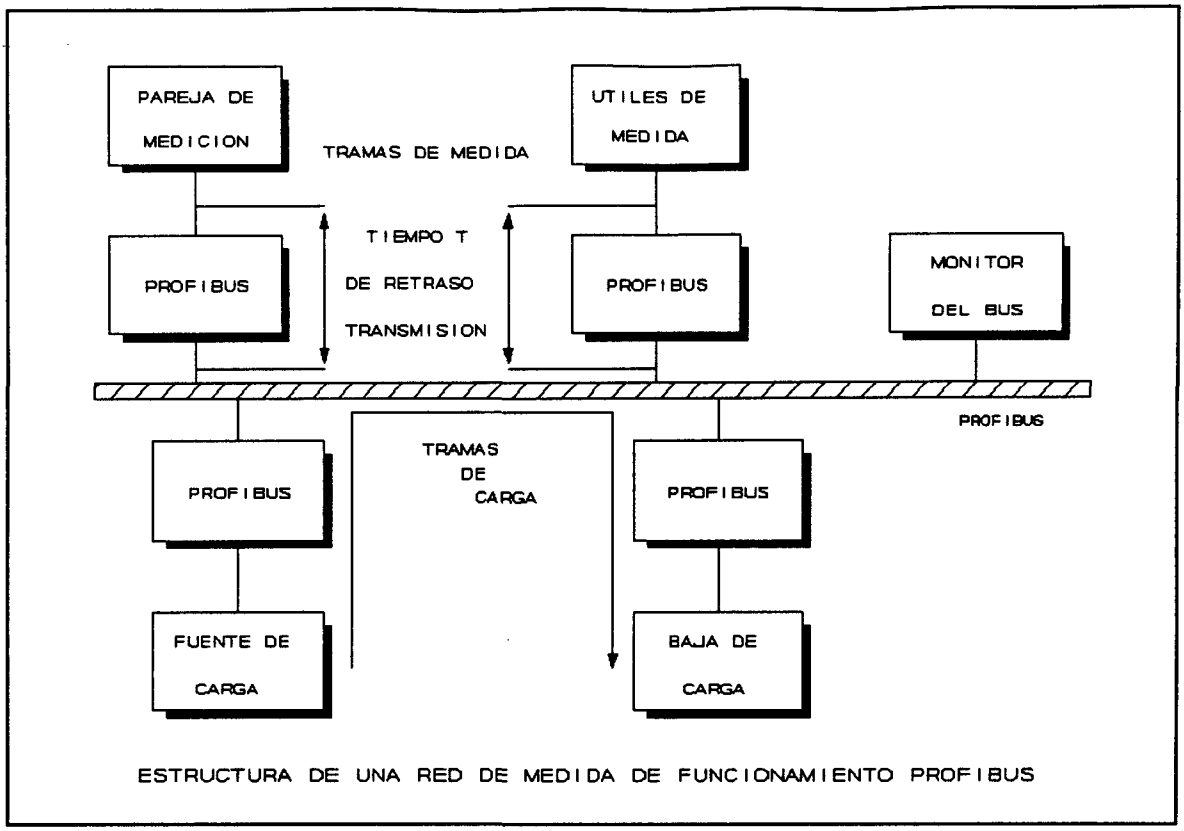


FIGURA 6.7

de funcionamiento se comportan como otra estación normal del bus. Además, un monitor de bus puede registrar opcionalmente el tráfico de datos y los intervalos en los que ocurrió la transferencia de datos.

La medida de la mayoría de las variables de funcionamiento se basa en la medición de retrasos de transmisión. A través de varios servicios, las herramientas de medida intercambian tramas con su entidad par. Estas tramas contienen un tiempo de marca, así que el tiempo de transmisión necesitado se puede registrar exactamente después de la llegada de la trama a su destino.

La herramienta de medición en la red de medida de ejecución controla todas las operaciones de medida, ejecuta todas las mediciones y evalúa los resultados en parte en la línea. Permite gran medida de variables de funcionamiento como una función de todos los

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

parámetros disponibles por ella. La compleja estructura modular de la herramienta de medición, combina varias ventajas en un gran paquete de herramientas (p.e., medida del tiempo real, acceso a tramas en el medio de transmisión y en el nivel de aplicación). Así, permite ejecutar una serie completa de mediciones bajo condiciones variables.

Las características importantes de la herramienta de medida son su interfase de usuario simple y conveniente, control remoto de los generadores de carga y su alta flexibilidad y exactitud de las funciones de medida usadas. El hardware de un monitor PROFIBUS incluye un ordenador, la placa controladora V25 PROFIBUS y una unidad de hardware especial del monitor.

El generador de carga tiene dos tareas principales: genera una carga definida en la red por tramas de transmisión y recepción. La simulación propia de la carga ya incluye movimiento de las tramas desde el sistema. Esto representa la pareja para la herramienta de medida, por la que genera o recibe tramas de medida. Un generador de carga es un controlador PROFIBUS, un ordenador y el software del generador de carga.

La medida física de productos software está ganando importancia. A este nuevo campo ya pertenece la evaluación de funcionamiento de implementaciones PROFIBUS, para lo que se usa un analizador de soft. Esta poderosa herramienta ofrece varias características de análisis de software local en tiempo real con altas opciones de aproximación y evaluación en el nivel fuente de lenguaje, que ya son integrables en la herramienta. Esta herramienta permite que el usuario detecte y optimice.

6.3.4 MEDICIONES TIPICAS Y RESULTADOS

Estas mediciones fueron propuestas para automatización de edificios, ya que existía un perfil necesario y concreto para esta área. Basado en estos requisitos, se puede controlar la aptitud de PROFIBUS para aplicaciones de control de edificios. Para este propósito, se usa

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

una red de laboratorio relativamente pequeña con distancias cortas y 5 estaciones activas.

La carga de comunicación se define como el número medio de acciones por hora de diferentes tamaños de sistema (sistemas pequeños de hasta 5, sistemas medianos de hasta 15 y grandes sistemas de hasta 30 estaciones).

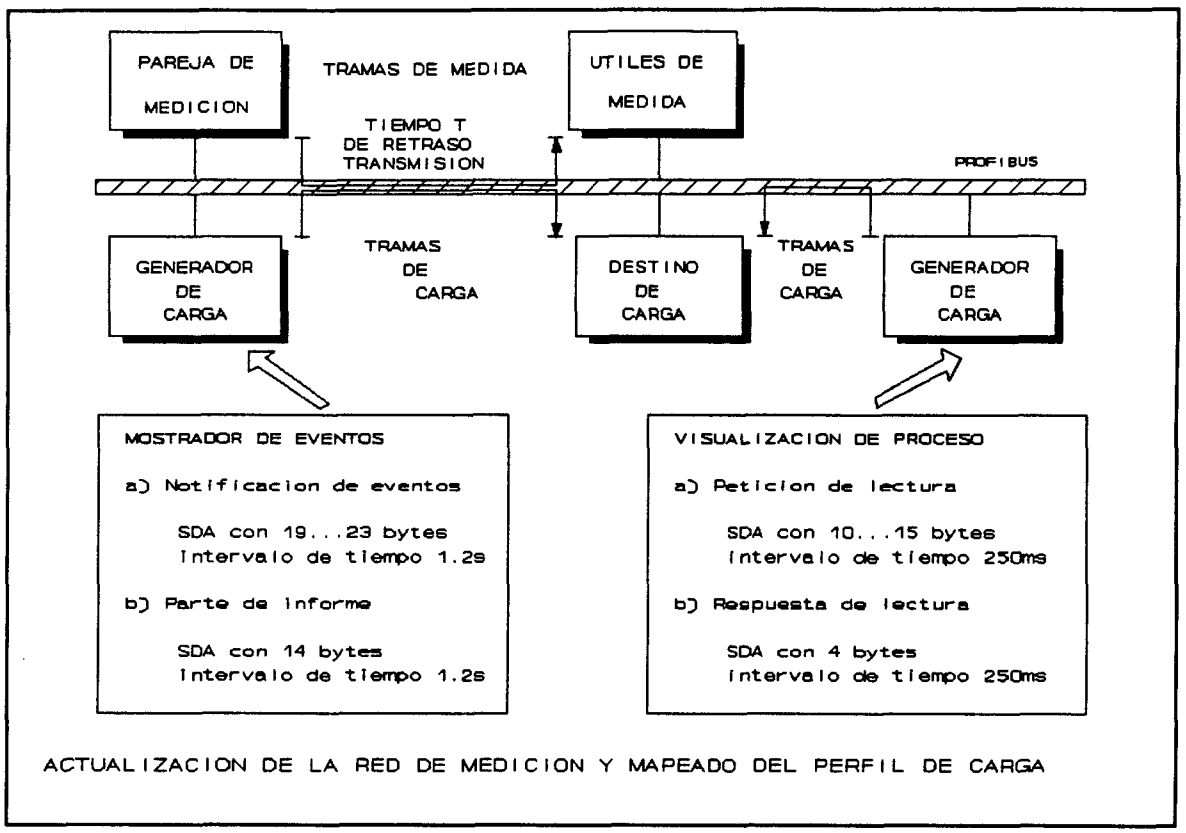


FIGURA 6.8

El perfil de requisitos en automatización de edificios se monta con las acciones individuales y varias combinaciones de acciones ocurridas en la práctica. Todas las acciones se indican como servicios de la capa 7; por la capa 2 se mapean en tramas SDA. Se transmiten en la capa 7 entre 2 y 17 bytes de datos de usuario.

Se supone una tasa de transmisión requerida de 19,2Kbit/s. El tiempo de respuesta de la red sería lo más corto posible, pero no excediendo de 1 segundo en el nivel de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

aplicación. Las siguientes variables de funcionamiento son importantes debido a los perfiles requeridos obtenidos en la automatización de edificios: el tiempo de retraso de transmisión desde la capa 7 de una estación a la entidad pareja y el volumen de datos.

Del tiempo de retraso de transmisión de una trama desde una estación a otra, se pueden derivar los tiempos de respuesta como les ocurre a programas de aplicación que corren. Además, los cálculos han mostrado que algunas de las combinaciones de acción mencionadas requieren una gran capacidad de datos.

Para la simulación de las cargas de red reales, los generadores de carga emulan varias acciones individuales y combinaciones de acciones como las derivadas de los perfiles de necesidades de la automatización de edificios. Bajo las condiciones respectivas de carga, los tiempos de retraso de transmisión se analizan y optimizan por variación de los parámetros.

El generador de carga se instala en cinco estaciones (figura 6.8); en una la herramienta de medición.

El diseño del perfil requerido dado en la red de medida de funcionamiento se indica en la parte baja de la figura 6.8. Se debe resaltar que se genera el perfil de carga correcto por dos generadores de carga solamente. Debido a restricciones físicas, no son idénticas completamente a las condiciones reales en las relaciones de comunicación y flujo de datos en la red simuladora. Este efecto se puede eliminar por una distorsión correctamente adecuada.

Los generadores de carga simulan el perfil de comunicación en automatización de edificios. Cada generador de carga puede establecer hasta 4 conexiones lógicas y sobre cada una de estas conexiones genera un perfil de carga definido y separado. Durante la simulación de tal petición de comunicación, es particularmente evaluada la influencia de la carga en el tiempo de retardo de las tramas. Por otro lado, no obstante, se deben hacer algunas medidas

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

sin carga en la red para encontrar los mejores parámetros para la operación de red bajo los requisitos dados.

El procedimiento completo se muestra en la figura 6.9. Primero, es importante determinar las necesidades funcionales y temporales según el área de aplicación. Para este propósito, puede ayudar la lista de preguntas del punto 6.2.2. Como resultado, se obtienen la estructura física, como mínimo un perfil de carga y los objetivos para las variables de funcionamiento bajo consideración. Después del diseño de las características de la red de medida de funcionamiento, puede empezar la evaluación. Se modifica cada vez los parámetros de bus, las variables de funcionamiento se determinan de nuevo y se comparan con los originales. La repetición de este procedimiento iterativamente optimiza los parámetros de bus. Este método erróneo y experimental, no obstante consume mucho tiempo. Sería mejor consultar a un experto cuando se modificaran los parámetros del bus.

La influencia exacta que los diferentes parámetros de bus tienen en las variables de funcionalidad, deben determinarse analizando sus sensibilidades hacia estas influencias. Si se conocen ya, los parámetros se pueden optimizar mucho más rápidamente. Desafortunadamente, los parámetros individuales están en parte correlacionados, lo que significa que se influencian mutuamente uno al otro. Ya que las dependencias son tan complejas, es difícil determinar la exacta influencia de los parámetros.

Las mediciones han mostrado que la red de medida de funcionalidad PROFIBUS en la capa 2 con 5 estaciones activas fue claramente capaz de satisfacer las necesidades de funcionalidad para pequeños sistemas de control de edificios, e incluso proporcionar alguna capacidad adicional inusual. Solamente son críticas para combinaciones especiales solicitacitadas por algunos usuarios con un volumen de datos significativamente superior de la capacidad teórica de transferencia de 19,2kbit/s. Como se podría esperar, la funcionalidad de la primera generación del hardware y software PROFIBUS fue limitada.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

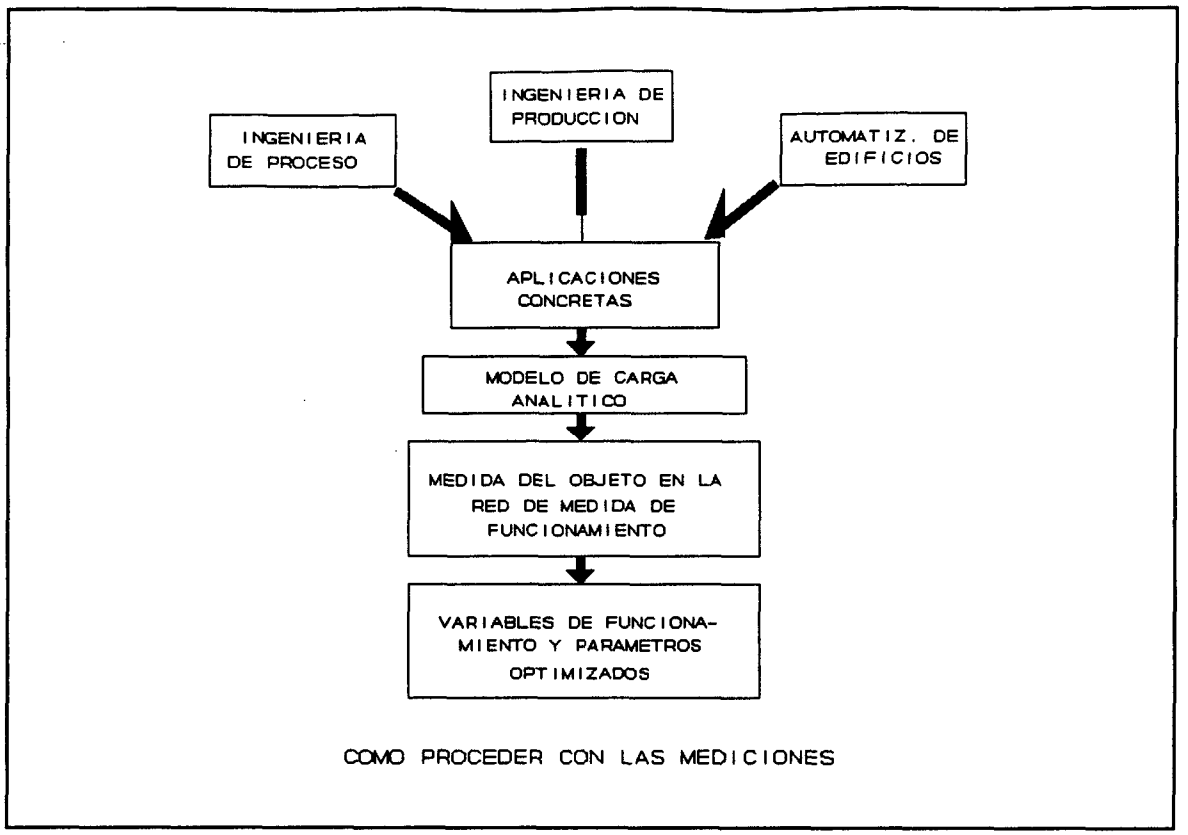


FIGURA 6.9

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

REQUISITOS DE IMPLEMENTACION DEL SOFTWARE

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

7 REQUISITOS DE IMPLEMENTACION DEL SOFTWARE

7.1 INTRODUCCION

El requisito principal, que el software Interface de Intercambio de Datos (DEI) debe cumplir, es el de intercambiar datos a través de la memoria compartida, situada en el módulo común de comunicaciones. Dicho software debe ser diseñado de forma que la comunicación y sincronización, entre la placa base de usuario y el módulo común de comunicaciones, se realice a través de dicha área de memoria compartida.

El Interface de Intercambio de Datos proporciona las siguientes funciones:

- Función para enviar peticiones y respuestas
- Función para recibir confirmaciones y respuestas
- Función para inicializar el software de comunicaciones y el driver

La transferencia de datos entre la estaciones, se realizará a través de dos bloques:

- Bloque de descripción del servicio, que describirá el servicio a realizar por el protocolo de comunicaciones.
- Bloque de datos, que contendrá los datos y parámetros específicos del servicio solicitado.

7.2 REQUISITOS DE CONFIGURACION

La especificación de los requisitos de configuración abarca cuatro aspectos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

fundamentales:

- Especificación de requisitos de configuración de los parámetros de bus que determinarán los aspectos relacionados con la transmisión de datos: velocidades, tiempos de retardo, ...

- Especificación de requisitos de configuración de las relaciones de comunicación que determinarán el tipo de estaciones que se conectarán y el modo de conectarse entre ellas.

- Especificación de requisitos de configuración de los objetos de comunicación que se intercambiarán entre las estaciones conectadas a la red PROFIBUS.

- Especificación de requisitos de los servicios que deberán ser soportados para configurar localmente las estaciones, establecer relaciones de comunicación entre ellas y por último, intercambiar variables entre todas las estaciones.

7.2.1 PARAMETROS DEL BUS

Los requisitos de configuración para los parámetros del bus son los siguientes:

- Velocidad mínima de transmisión de la estación: 500Kbit/s.
- No se soporta la redundancia del medio.
- Todas las estaciones se configurarán como maestras.

Los valores de los parámetros de bus deben ser los de la tabla 7.1.

Dicho valores se pueden optimizar en función del número de estaciones que se conecten a la red PROFIBUS. El resto de parámetros se pueden configurar individualmente en cada estación ya, que al ser específicos de cada una de ellas, no influyen en el

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

TABLA 7.1

PARAMETROS DEL BUS

PARAMETROS DE BUS DE LA ESTACION

ATRIBUTOS	VALORES
HSA	6
BAUD RATE	500 KBIT/S
TSL	2025
TQUI	0
min TSDR	50
max TSDR	2000
TSET	50
TTR	100000
G	1
in_ring_desired	TRUE
max_retry_limit	1

funcionamiento de la red PROFIBUS.

7.2.2 RELACIONES DE COMUNICACION

Las relaciones de comunicación que se configuran en la estación presentan las siguientes características:

- Se configura en la lista de relaciones de comunicación (CRL) cinco relaciones.

- La dirección 4 es la del equipo de ISOLUX-WAT y la del PC es la 5.

- Las relaciones son orientadas a conexión.

- El tipo de las relaciones de comunicación son maestro-maestro con transferencia acíclica de datos (MMAC).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

- Las relaciones de comunicación se definen en el momento del establecimiento de la conexión, es decir, son definidas con atributo de conexión igual a D (Defined connection).

- El usuario de las relaciones de comunicación se configura como FMS.

- Los servicios soportados por las relaciones de comunicación son FMS.

Los valores de la CRL-Header de la estación son los que se pueden observar en la tabla 7.2.

TABLA 7.2

CRL-Header	
ATRIBUTOS	VALORES
CR	0
Nº DE CRL DE ENTRADA	5
POLL LIST SAP	255
ASS/ABT CI	4096
LONGITUD DEL SIMBOLO	20
PUNTERO VFD	0 (VFD UNICO)

CRL-HEADER DE LAS ESTACIONES

Los valores de la CRL-Entries para el equipo W-90 son los de la tabla 7.3.y para el PC son los de la tabla 7.4. Nótese que la diferencia radica en la dirección remota de la estación, ya que la dirección del equipo W-90 es 4 y la del PC 5 como decíamos anteriormente.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

TABLA 7.3

CRL-ENTRIES	
ATRIBUTOS	VALORES
CR	5
LOCAL LSAP	5
REMOTE ADDRESS	5
REMOTE LSAP	5
TYPE	MMAZ
LLI SAP	0
CONNECTION ATTRIBUTE	D
MAX SCC	3
MAX RCC	3
MAX SAC	3
MAX RAC	3
CONTROL INTERVAL (CI)	0
MULTIPLIER	0
MAX PDU SEND. HIGH/LOW PRIO	241/241
MAX PDU RECE. HIGH/LOW PRIO	241/241
SERVICES SUPPORT4ED	80 30 01 80 30 01
SYMBOL	[ISO-CTR]
VFD POINTER (NUMBER)	1

CRL-ENTRIES DE LA ESTACION ISOLUX-WAT

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

TABLA 7.4

CRL-ENTRIES	
ATRIBUTOS	VALORES
CR	5
LOCAL LSAP	5
REMOTE ADDRESS	4
REMOTE LSAP	5
TYPE	MMAZ
LLI SAP	0
CONNECTION ATTRIBUTE	D
MAX SCC	3
MAX RCC	3
MAX SAC	3
MAX RAC	3
CONTROL INTERVAL (CI)	0
MULTIPLIER	0
MAX PDU SEND. HIGH/LOW PRIO	241/241
MAX PDU RECE. HIGH/LOW PRIO	241/241
SERVICES SUPPORT4ED	80 30 01 80 30 01
SYMBOL	[CTR-ISO]
VFD POINTER (NUMBER)	1

CRL-ENTRIES DE LA ESTACION PC

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

7.2.3 DICCIONARIO DE OBJETOS

Los requisitos de configuración para el diccionario de objetos son los siguientes:

- El diccionario de objetos está asociado a un único VFD.

- La parte estática de tipos del OD (ST-OD) está formada por los datos estandarizados por PROFIBUS como son: Boolean, Integer, Unsigned, Floating Point, Visible String, Octect String, Bit String, Date, Time of day y Time
Difference.

- La parte estática del OD (S-OD) está formada por la definición de objetos de tipo Simple Variable.

- El OD no tiene parte dinámica de listas de variables (DV-OD) ni de invocación a programas (DP-OD).

- Las estaciones pueden acceder libremente a los objetos, es decir, no se impondrán derechos de acceso mediante passwords ni grupos de acceso.

- Cada diccionario Objeto contiene 14 objetos de tipo Simple Variable, cada uno de tipo Data Type. Los índices de dichas variables serán correlativos y el primer dígito se corresponderá con la dirección de la estación. Es decir, las del equipo W-90 son 4xx y las del PC 5xx. Cada variable tiene asociado un nombre simbólico.

A continuación se muestra la cabecera (OD-Header) y los objetos del diccionario de objetos de la estación W-90 y del PC.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

ISOLUX-WAT (Simple Variables)

Atributos	Valores													
Index	400	401	402	403	404	405	406	407	408	409	410	411	412	413
Type	BOOL	INT8	INT16	INT32	USIGN8	USIGN16	USIGN32	FLOAT	VSTRING	OSTRING	DATE	TOFDAY	TDIFF	BSTRING
Length	1	1	2	4	1	2	4	4	n	n	7	4 6 6	4 6 6	n
Pass	00	00	00	00	00	00	00	00	00	00	00	00	00	00
AccGrp	00	00	00	00	00	00	00	00	00	00	00	00	00	00
AccR	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
Addr	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
SymName	[ISO-1]	[ISO-2]	[ISO-3]	[ISO-4]	[ISO-5]	[ISO-6]	[ISO-7]	[ISO-8]	[ISO-9]	[ISO-10]	[ISO-11	[ISO-12]	[ISO-13]	[ISO-14]

PC (Simple Variables)

Atributos	Valores													
Index	500	501	502	503	504	505	506	507	508	509	510	511	512	513
Type	BOOL	INT8	INT16	INT32	USIGN8	USIGN16	USIGN32	FLOAT	VSTRING	OSTRING	DATE	TOFDAY	TDIFF	BSTRING
Length	1	1	2	4	1	2	4	4	n	n	7	4 6 6	4 6 6	n
Pass	00	00	00	00	00	00	00	00	00	00	00	00	00	00
AccGrp	00	00	00	00	00	00	00	00	00	00	00	00	00	00
AccR	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
Addr	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
SymName	[CTR-1]	[CTR-2]	[CTR-3]	[CTR-4]	[CTR-5]	[CTR-6]	[CTR-7]	[CTR-8]	[CTR-9]	CTR-10]	[CTR-11]	[CTR-12]	[CTR-13]	[CTR-14]

OD- HEADER.-

Atributos	ISOLUX-WAT	PC
OD Header object code	1	1
OD Header index	0	0
ROM/RAM flag	255	255
Symbol name length	32	32
Access protection	0	0
OD version	1	1
Local address OD-DES	FFFFFFFF	FFFFFFFF
ST-OD Length	20	20
Local address ST-OD	FFFFFFFF	FFFFFFFF
First index S-OD	400	500
S-OD Length	50	50
Local address S-OD	FFFFFFFF	FFFFFFFF
First index DV-OD	0	0
DV-OD Length	0	0
Local address DV-OD	FFFFFFFF	FFFFFFFF
First index DP-OD	0	0
DP-OD Length	0	0
Local address DP-OD	FFFFFFFF	FFFFFFFF

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

7.2.4 SERVICIOS

Los servicios que se soportan son los siguientes:

- Servicios obligatorios según el estándar PROFIBUS (DIN 19245 Part 2):

  * Initiate, Abort, Reject

  * Status, Identify

  * Get OD (short).

- Servicios para configuración local de una estación PROFIBUS:

  * Create VFD Local, Set Physical Status VFD

  * Set Bus Parameters

  * Initiate Load OD Local, Load OD Local, Terminate Load OD Local

  * Initiate Load CRL Local, Load CRL Local, Terminate Load CRL Local

- Servicios para gestión OD-remoto:

  * Get OD (long)

- Servicios para intercambio de variables:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* Read, Write

Todas estas funciones se explicarán en el siguiente capítulo 8.

7.3 ESPECIFICACION DE LAS PRUEBAS DE INTEROPERACION

7.3.1 NOMENCLATURA DE LAS PRUEBAS

La identificación de las pruebas se realizará teniendo en cuenta la siguiente nomenclatura:

XXX_YYY_ZZZ_i

donde:

XXX: Representa los tres primeros carácteres de la estación que actúa como cliente. (ISO o CTR).

YYY: Representa los tres primeros carácteres de la estación que actúa como servidor. (CTR o ISO)

ZZZ: Representa el tipo de servicio que se va a ejecutar. (Initiate, Get-OD (long), Write, Read)

i: Representa el número de la prueba, donde i podrá tomar valores de 1 a 14 dependiendo del tipo de prueba.

7.3.2 DESCRIPCION DE LAS PRUEBAS

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* Prueba CTR_ISO_INITIATE

Esta prueba verifica cómo el PC establece una conexión con la estación de ISOLUX a través del servicio Initiate.

* Prueba CTR_ISO_GET-OD

Verifica cómo el PC obtiene de la estación W-90 una descripción de todos sus objetos a través del servicio Get-OD.

* Prueba CRT_ISO_WRITE_1 ... 14

Por medio de esta prueba se verifica la escritura por parte del PC en las variables de la estación W-90 de índice 400 hasta 413 o de nombres ISO-1 hasta ISO-14.

* Prueba CRT_ISO_READ_1 ... 14

Por medio de esta prueba se verifica la lectura del PC de las variables de la estación W-90 de índice 400 hasta 413 o de nombres ISO-1 hasta ISO-14.

* Prueba CTR_ISO_ABORT

El PC cierra la conexión que había establecido previamente con el equipo W-90.

* Prueba ISO_CTR_INITIATE

Esta prueba verifica cómo la estación de ISOLUX establece una conexión con el PC a través del servicio Initiate.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

* Prueba ISO_CTR_GET-OD

Verifica cómo la estación W-90 obtiene del PC una descripción de todos sus objetos a través del servicio Get-OD.

* Prueba ISO_CRT_WRITE_1 ... 14

Por medio de esta prueba se verifica la escritura por parte de la estación ISOLUX en las variables del PC de índice 500 hasta 513 o de nombres CTR-1 hasta CTR-14.

* Prueba ISO_CTR_READ_1 ... 14

Por medio de esta prueba se verifica la lectura de la estación de las variables del PC de índice 500 hasta 513 o de nombres CTR-1 hasta CTR-14.

* Prueba ISO_CTR_ABORT

La estación W-90 cierra la conexión que había establecido previamente con el PC.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

PRIMITIVAS DE SERVICIO

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

8 PRIMITIVAS DE SERVICIO

8.1 INTRODUCCION

Como en todo protocolo de comunicación basado en el modelo OSI, PROFIBUS distingue cuatro interacciones de aplicación/comunicación. Estos cuatro tipos básicos de interacción se denominan primitivas de servicio:

Request: La estación PROFIBUS "A" inicia un servicio PROFIBUS y pasa los parámetros relevantes y los datos al software de comunicación.

Indication: Llegan los datos pasados por la petición anterior sobre el bus a la estación PROFIBUS "B" y se tratan en la aplicación de la "B".

Response: Para contestar los servicios PROFIBUS, la aplicación de la estación B debe confirmar la recepción de la indicación. Algunos servicios como READ, GET-OD y READ-VALUE causan datos adicionales de respuesta para mandar de vuelta junto con la confirmación. Esto se realiza por medio de esta primitiva de servicio.

Confirmation: Para confirmación de los servicios PROFIBUS, la confirmación o respuesta de la estación "B" se transforma en la aplicación de "A" en la primitiva de servicio "confirmation".

Existen por lo tanto dos tipos de primitivas de servicio, request y response, por la que una estación activa (corriendo), comienza o responde a los servicios PROFIBUS. La

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

dirección del flujo de control y del flujo de datos es desde la aplicación hacia el programa de comunicación.

En el caso de las otras dos primitivas, indication y confirmation, se leen los datos para la aplicación que vienen del programa de comunicación. El flujo de control en este caso depende del mecanismo de sincronización comunicación/aplicación.

La función **profi_snd_req_res** es para el envío de peticiones y respuestas. La función **profi_rcv_con_ind** se puede usar para hacer un polling de confirmaciones e indicaciones.

La transferencia de datos en la interfase de envío/recepción tiene lugar sobre un servicio: SERVICE-DESCRIPTION-BLOCK que contiene los mismos parámetros para todos los servicios, y sobre un DATA BLOCK que contiene parámetros específicos de servicio y datos.

La función **init_profibus** inicializa el software de comunicación y driver.

El bloque descriptor de servicio es el siguiente:

```
typedef struct T_PROFI_SERVICE_DESCR

{
USIGN16          comm_ref;
USIGN8           layer;
USIGN8           service;
USIGN8           primitive;
INT8             invoke_id;
INT16                  result;
} T_PROFI_SERVICE_DESCR;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Donde los elemento son los siguientes;

- comm_ref : referencia de comunicación (Canal lógico)

- layer : capa específica a la que se dirige la tarea (FMS,FMA7,USER)

- service_id : servicio a tratar en la capa

- primitive : primitivas de servicio (request, indication, response, confirmation)

- invoke_id : ID para la tarea

- result : resultado positivo o negativo

El bloque de datos contiene los datos específicos de servicio.

8.2 FUNCION PROFI_SND_REQ

extern INT16 profi_snd_req_res

```
(
IN T_PROFI_SERVICE_DESCR  FAR  *sdb_ptr,
IN VOID                   FAR  *data_ptr,
IN BOOL                        dummy
)
```

El usuario pasa el parámetro sdb_ptr para especificar un puntero a una estructura de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

datos de tipo T_PROFI_SERVICE_DESCR. Esta estructura de datos describe el servicio a utilizar por el software de protocolo.

El usuario proporciona el parámetro data_ptr que contiene un puntero a una estructura de datos donde se almacenan los parámetros de servicio específico y los datos.

El tercer parámetro ha sido definido para mantener la compatibilidad con versiones anteriores y no tiene ninguna otra función,

Cuando la función se ejecuta correctamente devuelve:

- E_OK (0) función ejecutada correctamente

Cuando existe error de software:

- E_FATAL_CMI_ERROR (05) irrecobrable error en driver del controlador
- E_FATAL_LAYER2_ERROR (06) irrecobrable error en la capa 2
- E_FATAL_LAYER7_ERROR (07) irrecobrable error en la capa 7
- E_FATAL_OS_ERROR (08) irrecobrable error en el sistema operativo
- E_INVALID_LAYER (12) caso específico incorrecto (FMS o FMA7)
- E_INVALID_SERVICE (13) servicio específico incorrecto

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
- E_INVALID_PRIMITIVE          (14)  primitiva específica incorrecta

- E_INVALID_COMM_REF           (16)   CR no existente

- E_INVALID_FMS_COMM_REF       (17)   CR no es FMS-CR

- E_INVALID_FMA_COMM_REF       (18)   CR no es FMA7-CR

- E_RESOURCE_UNAVAILABLE       (21)   recurs.no dispon.para procesamiento

- E_NO_PARALLEL_SERVICES       (22)  servicio paralelo no permitido

- E_SERVICE_CONSTR_CONFLICT    (23)   servicio no ejecutable

- E_SERVICE_NOT_SUPPORTED      (24)   no existe serv.en el subconj.soportado

- E_SERVICE_NOT_EXECUTABLE     (25)   no se puede ejecutar el servicio
```

Y por condiciones de error en el driver o en el CMI:

```
- E_NO_CNTRL_RES               (10)  no responde el controlador

- E_INVALID_CMI_CALL           (19)  función de driver inválida

- E_CMI_ERROR                  (20)  error serio en CMI o driver
```

8.3 FUNCION PROFI_RCV_CON_IND

Usando la función profi_rcv_con_ind, la aplicación permite al driver a al programa de comunicación conocer que espera una confirmación o indicación.

```
extern INT16 profi_rcv_con_ind
        (
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
INOUT T_PROFI_SERVICE_DESCR    FAR   *sdb_ptr,
INOUT VOID                     FAR   *data_ptr,
INOUT USIGN16                        *data_len
)
```

Por medio de COND_IND_RECEIVED devuelve un valor de profi_rcv_con_ind, se le indica a la aplicación si la confirmación o indicación es positiva.

Mediante el parámetro específico de usuario sdb_ptr se especifica un puntero a una estructura de datos del tipo indicado. Una confirmación o indicación positiva se introduce dentro de esta estructura por el programa de comunicación.

El parámetro de usuario data_ptr es un puntero a un área de memoria en la que el programa de comunicación puede introducir los parámetros específicos y datos de una confirmación o indicación.

El tercer parámetro indica el tamaño del área de la memoria que se ha dispuesto, y que contiene la longitud actual del bloque de datos.

La función devuelve:

- NO_CON_IND_RECEIVED (00) no hay confirmación ni indicación
- CON_IND_RECEIVED (01) se pasó una confirmación o indicación
- E_INVALID_DATA_SIZE (15) demas.pequeño tamaño bloq.datos pasado
- E_FATAL_CMI_ERROR (05) irrecobrable error en driver del controlador
- E_FATAL_LAYER2_ERROR (06) irrecobrable error en la capa 2

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

- E_FATAL_LAYER7_ERROR (07) irrecobrable error en la capa 7

- E_FATAL_OS_ERROR (08) irrecobrable error en el sistema operativo

Y por condiciones de error en el driver o en el CMI:

- E_NO_CNTRL_RES (10) no responde el controlador

- E_INVALID_CMI_CALL (19) función de driver inválida

- E_CMI_ERROR (20) error serio en CMI o driver

8.4 FUNCION INIT_PROFIBUS

La función init_profibus se usa para inicializar la interfase de comunicación PROFIBUS y el driver de CMI.

```
extern INT16 init_profibus
        (
        IN USIGN32          h_dpr_base_address,
        IN USIGN16          dummy,
        IN BOOL             hw_reset
        )
```

h_dpr_dase_address proporciona la dirección de comienzo del puerto dual RAM para la interfase del controlador CMI.

El parámetro dummy se ha definido por compatibilidad con las versiones de otros controladores.

El parámetro hw_reset determina si se borrará el firmware PROFIBUS o no.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Los posibles valores devueltos son:

- E_OK	(0)	inicialización ejecutada exitosamente
- E_NO_CNTRL_RES	(10)	no responde el controlador
- E_INVALID_CNTRL_TYPE_VERSION	(11)	Driver incompatible con el tipo o con la versión de software del controlador PROFIBUS.

A continuación, se va a explicar los servicios usados en el programa de aplicación desarrollado, así como las correspondientes primitivas de dichos sevicios.

8.5 SERVICIOS DE MANEJO DE CONTEXTO FMS

8.5.1 INITIATE

Este servicio se usa para construir una conexión entre dos estaciones de comunicación. Un parámetro de chequeo controla según las opciones, los servicios y las longitudes de las PDU FMS se realiza también. Más aún, se transfiere el número de versión del OD asignado a la conexión.

Son posibles varias reacciones para una petición de inicialización. Si la pareja de comunicación no está disponible o los puntos de entrada del nivel 2 están mal parametrizados, entonces la capa 2 responde con un abort. Si el chequeo de contexto LLI es negativo, responde la LLI pareja con un abort. Solamente si la FMS pareja de comunicación o la aplicación rechaza el establecimiento de la conexión por un control de chequeo negativo, se devuelve una respuesta de inicialización negativa.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

SERVICE-DESCRIPTION-BLOCK PARA REQUEST E INDICATION

USIGN16	comm_ref	2..MAX_COMREF
USIGN8	layer	FMS/FMS_USR
USIGN8	service	INITIATE
USIGN8	primitive	REQ/IND
INT8	invoke_id	not used
INT16	result	not used

BLOQUE DE DATOS PARA REQUEST E INDICATION

Data structure T_CTXT_INIT_REQ

USIGN8	profile_number[2]	profile number
BOOL	protection	access protection
USIGN8	password	password
USIGN8	access_groups	access_groups
USIGN8	dummy	aligmnment byte
INT16	ov_version	od_version
USIGN8	snd_len_h	max.tamaño a mandar alta prioridad
USIGN8	snd_len_l	max.tamaño a mandar baja prioridad
USIGN8	rcv_len_h	max.tamaño a recibir alta priorid.
USIGN8	rcv_len_l	max.tamaño a recibir baja priorid.
USIGN8	supported_features[FEAT_SUPP_LEN]	

El usuario necesita los parámetros profile_number, password y access_groups. Los parámetros definidos se preparan por el software de comunicación.

SERVICE-DESCRIPTION-BLOCK PARA RESPONSE Y CONFIRMATION

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN16	comm_ref	2..MAX_COMREF
USIGN8	layer	FMS/FMS_USR
USIGN8	service	INITIATE
USIGN8	primitive	RES/CON
INT8	invoke_id	not used
INT16	result	POS/NEG

BLOQUE DE DATOS PARA RESPONSE Y CONFIRMATION

result = POS:

Data structure T_CTXT_INIT_CNF

USIGN8	profile_number[2]	profile number
INT16	ov_version	od_version
BOOL	protection	access protection
USIGN8	password	password
USIGN8	access_groups	access_groups
USIGN8	dummy	aligmnment byte

result = NEG:

Data structure T_CTXT_INIT_ERR_CNF

USIGN16	class_code	error y clase de error
USIGN8	snd_len_h	max.tamaño a mandar alta prioridad
USIGN8	snd_len_l	max.tamaño a mandar baja prioridad
USIGN8	rcv_len_h	max.tamaño a recibir alta priorid.
USIGN8	rcv_len_l	max.tamaño a recibir baja priorid.
USIGN8	supported_features[FEAT_SUPP_LEN]	

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

El usuario necesita los parámetros de profile_number, password y access_groups o error_class y error_code.

8.5.2 ABORT

Este servicio se usa para liberar una conexión de comunicación entre dos compañeros de comunicación. La conexión puede ser rota tanto por el cliente como por el servidor.

El servicio abort también se produce por una llamada del software de comunicación como reacción a una situación de error. Por lo tanto, es también el mecanismo de tratamiento de error estándar PROFIBUS.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST E INDICATION

USIGN16	comm_ref	2..MAX_COMREF
USIGN8	layer	FMS/FMS_USR
USIGN8	service	ABORT
USIGN8	primitive	REQ/IND
INT8	invoke_id	not used
INT16	result	not used

DATA BLOCK PARA REQUEST E INDICATION

Data structure		T_CTXT_ABORT_REQ
BOOL	local	gener.local o remota (TRUE = local)
USIGN8	abort_id	ident.aborto (USR,LLI_USR,LLI,FDL)
USIGN8	reason	código de razón del aborto
USIGN8	detail_length	longitud
USIGN8	detail[DETAIL_LENGTH]	información detallada sobre la razón

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Identificadores de aborto:

USR	0	identificador de USUARIO
FMS	1	identificador de FMS (LLI_USR)
LLI	2	identificador LLI
FDL	3	identificador FDL

Códigos de aborto de usuario:

USR_ABT_RC1	0	desconectado
USR_ABT_RC2	1	incompatible la versión del OD
USR_ABT_RC3	2	error de password
USR_ABT_RC4	3	incompatible el número de perfil
USR_ABT_RC5	4	permitido el servicio limitado
USR_ABT_RC6	5	interact. de carga de OD permitida

Códigos de aborto de FMS:

FMS_ABT_RC1	0	error de FMS-CRL
FMS_ABT_RC2	1	error de usuario
FMS_ABT_RC3	2	error de FMS-PDU
FMS_ABT_RC4	3	estado de con.conflictivo para LLI
FMS_ABT_RC5	4	error de LLI
FMS_ABT_RC6	5	tamaño de PDU
FMS_ABT_RC7	6	cualidad no soportada
FMS_ABT_RC8	7	respuesta errónea de invoke id
FMS_ABT_RC9	8	sevicios máximos sobrecargados
FMS_ABT_RC10	9	estado de con.conflictivo para FMS
FMS_ABT_RC11	10	error de servicio
FMS_ABT_RC12	11	petición errónea de invoke id
FMS_ABT_RC13	12	FMS desconectada

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Códigos de aborto de LLI:

LLI_ABT_RC1	0	cont.neg.context.LLI(cont.rem.AD)
LLI_ABT_RC2	1	LLI-PDU inval.en asociac.o aborto
LLI_ABT_RC3	2	LLI-PDU inválida fase transf.datos
LLI_ABT_RC4	3	LLI-PDU recib.inválida o descon.
LLI_ABT_RC5	4	DTA-ACK-PDU recibida y SCA=0
LLI_ABT_RC6	5	exc.n* máx.serv.paral(serv.rem.s)
LLI_ABT_RC7	6	invoke id desconocido
LLI_ABT_RC8	7	error de prioridad
LLI_ABT_RC9	8	error local en la estación remota
LLI_ABT_RC10	9	timeout durante la asociación
LLI_ABT_RC11	10	timeout en conexión cíclica
LLI_ABT_RC12	11	timeout en tiempo recepción idle
LLI_ABT_RC13	12	error activación LSAP (est.en AD)
LLI_ABT_RC14	13	primitiva FDL ilegal en ASS o ABT
LLI_ABT_RC15	14	primit.FDL ilegal transf.de datos
LLI_ABT_RC16	15	primitiva FDL desconocida
LLI_ABT_RC17	16	primitiva LLI desconocida
LLI_ABT_RC18	17	primitiva LLI ilegal en ASS o ABTD
LLI_ABT_RC19	18	prim.LLI ilegal en transf.de datos
LLI_ABT_RC20	19	inválida entrada de CRL
LLI_ABT_RC21	20	conflictivo estado de conexión ASS
LLI_ABT_RC22	21	error procedimiento conex. cíclica
LLI_ABT_RC23	22	exced.n* máx.serv.paral.(por FMS)
LLI_ABT_RC24	23	CRL en carga, LLI desconectada
LLI_ABT_RC25	24	error modo confirmación/indicación
LLI_ABT_RC26	25	primitiva FM1/2 ilegal
LLI_ABT_RC27	26	serv.ilegal en conexión cíclica
LLI_ABT_RC28	27	FMS-PDU demas.larga conex.cícl.
LLI_ABT_RC29	28	error recurso durante asociación

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

LLI_ABT_RC30	29	error recurso en fase de transf.
LLI_ABT_RC31	30	error de recurso durante el aborto
LLI_ABT_RC32	31	error de estado LLI
LLI_ABT_RC33	32	error de timer LLI
LLI_ABT_RC34	33	fallo en la transf.recursos a FDL

Detalles de aborto LLI:

LLI_ABT_AD1	0	error en actual. del buffer
LLI_ABT_AD2	1	error activ.de entrada lista poll
LLI_ABT_AD3	2	error en desact.de entr.lista poll
LLI_ABT_AD4	3	error en transmisión (SDA.con)
LLI_ABT_AD5	4	error en transmisión (CSRD.con)
LLI_ABT_AD6	5	error en transmisión (SRD.con)
LLI_ABT_AD7	6	error en recepción (CSRD.con)

Códigos de aborto FDL:

UE	1	error de interf.de usuario remota
RR	2	no dispon.los recursos remotos
RS	3	servicio no activado en remota SAP
RDL	12	no recurso mandar baja resp.datos
RDH	13	no recurso mandar alta resp.datos
LS	16	servicio no activado en SAP local
NA	17	no reacción de la estación remota
NLT	18	estación desconectada
NO	19	servicio FDL no correcto
LR	20	no disponible los recursos locales
IV	21	parámetros de petición inválidos

8.5.3 REJECT

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

El servicio reject lo usa FMS para rechazar PDUs inaceptables.

SERVICE-DESCRIPTION-BLOCK PARA INDICATION

USIGN16	comm_ref	2..MAX_COMREF
USIGN8	layer	FMS_USR
USIGN8	service	REJECT
USIGN8	primitive	IND
INT8	invoke_id	not used
INT16	result	not used

DATA BLOCK PARA INDICATION

Data structure	T_CTXT_REJECT_IND

BOOL	detected_here	detectado local o remota
INT8	orig_invoke_id	original invoke ID
USIGN8	pdu_type	tipos de PDU rechazada
USIGN8	reject_code	codigo de rechazo

Tipos de PDU:

CONFIRMED_REQUEST_PDU	1	PDU de confirm. de petición
CONFIRMED_RESPONSE_PDU	2	PDU de confirm. de respuesta
UNCONFIRMED_PDU	3	PDU incorfimada
ONKNOWN_PDU_TYPE	4	tipo de PDU desconocida

Códigos de razón:

REJ_RC0	0	otro
REJ_RC1	1	existe invoke id

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

REJ_RC2	2	sobrecarg.los serv.máximos
REJ_RC3	3	cual.no soport.orient.conex.
REJ_RC4	4	cualidad no soport.sin conex.
REJ_RC5	5	tamaño de PDU
REJ_RC6	6	error de usuario sin conexión

8.5.4 SERVICIOS DE SOPORTE VFD

8.5.4.1 Servicios locales

8.5.4.1.1 CREATE_VFD

Este servicio lo usa el usuario para crear un Equipo Virtual de Campo (VFD) que se asigna a un diccionario objeto. El parámetro vfd_pointer en las entradas de la lista de relación de comunicación corresponde al parámetro vfd_number de la creación del servicio VFD.

Cuando se inicializa una estación PROFIBUS tienen lugar los siguientes sucesos:

- Se crean uno o mas VFDs
- Se generan los diccionarios objeto asociados
- Se carga la lista de relación de comunicación

Si el parámetro de cabecera CRL vfd_pointer_supported se fija a FALSE, solamente es válido el primer VFD, y todas las otras relaciones de comunicación acaban en este VFD.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN16	comm_ref	0
USIGN8	layer	FMS
USIGN8	service	CREATE_VFD_LOC
USIGN8	primitive	REQ
INT8	invoke_id	not used
INT16	result	not used

DATA BLOCK PARA REQUEST

Data structure T_VFD_CREATE_REQ

USIGN32	vfd_number	número de VFD
STRINGV	vendor_name[MAX_VFD_STRING_LENGTH]	vendedor
STRINGV	model_name[MAX_VFD_STRING_LENGTH]	modelo
STRINGV	revision[MAX_VFD_STRING_LENGTH]	rev.equipo
USIGN8	profile_number[2]	núm.de perfil

SERVICE-DESCRIPTION-BLOCK PARA CONFIRMATION

USIGN16	comm_ref	0
USIGN8	layer	FMS_USR
USIGN8	service	CREATE_VFD_LOC
USIGN8	primitive	CON
INT8	invoke_id	not used
INT16	result	POS/NEG

DATA BLOCK PARA CONFIRMACION

result = POS:

Data structure T_VFD_CREATE_CNF

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN32	vfd_number	número de VFD

result = NEG:

Data structure		T_VFD_ERROR
T_ERROR	error	estr.error estánd.
USIGN32	vfd_number	número de VFD

8.5.4.1.2 VFD-SET-PHYSICAL-STATUS

Con este servicio el usuario puede fijar el estado físico de la aplicación en el objeto VFD.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST

USIGN16	comm_ref	0
USIGN8	layer	FMS
USIGN8	service	VFD_SET_PHYS_STATUS_LOC
USIGN8	primitive	REQ
INT8	invoke_id	not used
INT16	result	not used

DATA BLOCK PARA REQUEST

Data structure		T_VFD_SET_PHYS_STATUS_REQ
USIGN32	vfd_number	número de VFD
USIGN8	physical_status	estado físico
USIGN8	dummy	byte de alineación

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

SERVICE-DESCRIPTION-BLOCK PARA CONFIRMATION

USIGN16	comm_ref	0
USIGN8	layer	FMS_USR
USIGN8	service	VFD_SET_PHYS_STATUS_LOC
USIGN8	primitive	CON
INT8	invoke_id	not used
INT16	result	POS/NEG

DATA BLOCK PARA CONFIRMACION

result = POS:

Data structure		T_VFD_SET_PHYS_STATUS_CNF
USIGN32	vfd_number	número de VFD

result = NEG:

Data structure		T_VFD_ERROR
T_ERROR	error	estr.error estánd.(expl.8.6)
USIGN32	vfd_number	número de VFD

8.5.4.2 Servicios remotos

No se han usado en el programa de aplicación, así que nos limitaremos a enumerarlos:

- Status

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

- Unsolicited-Status

- Identify

8.5.5 MANEJO DEL OD

8.5.5.1 Servicios locales

8.5.5.1.1 Cargando un diccionario objeto.

Se cargan una o más descripciones de objeto en un diccionario objeto local por medio de la secuencia de servicio Initiate-Load-OD-LOC, Load-OD-LOC y Terminate-Load-OD-LOC. Se hace una distinción entre carga con y sin efecto retrospectivo.

El proceso de carga es sin efecto retrospectivo cuando las modificaciones no afectan a las relaciones de comunicación de otras estaciones. Es el caso, por ejemplo, cuando se incluye o se borra una descripción de objeto privada (consecuencia 0).

El proceso de carga tiene efecto retrospectivo cuando otra relación de comunicación accede a la descripción modificada del objeto. En este caso la aplicación debe liberar todas las relaciones de comunicación, entonces construirlas otra vez con el número modificado de versión OD.

Si se especifica la consecuencia 1 en el caso de carga con efecto retrospectivo, entonces se pueden modificar solamente entradas individuales y se consigue el nuevo número de versión cuando se cargue la cabecera. En cambio, para consecuencia 2 se borra la totalidad del diccionario objeto y se debe recargar incluyendo la cabecera.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

INITIATE-LOAD-OD-LOC

Este servicio comienza el proceso de carga.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST

USIGN16	comm_ref	0
USIGN8	layer	FMS
USIGN8	service	INITIATE_LOAD_OV_LOC
USIGN8	primitive	REQ
INT8	invoke_id	not used
INT16	result	not used

DATA BLOCK PARA REQUEST

Data structure T_INIT_LOAD_OV_REQ

USIGN32	vfd_number	número de VFD
INT8	consequence	interactúa o no la carga
INT8	dummy	byte de alineación

Consequence:

CONSEQUENCE_0 0	no-interactúa
CONSEQUENCE_1 1	interactúa (sólo cambios)
CONSEQUENCE_2 2	interactúa (nuevo OD)

SERVICE-DESCRIPTION-BLOCK PARA CONFIRMATION

USIGN16	comm_ref	0
USIGN8	layer	FMS_USR

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN8	service	INITIATE_LOAD_OV_LOC
USIGN8	primitive	CON
INT8	invoke_id	not used
INT16	result	POS/NEG

DATA BLOCK PARA CONFIRMACION

result = POS:

Data structure		T_INIT_LOAD_OV_CNF
USIGN32	vfd_number	número de VFD

result = NEG:

Data structure		T_SRC_OV_ERROR
T_ERROR	error	estr.de error estándar
USIGN32	vfd_number	número de VFD
USIGN16	index	índice

LOAD-OD-LOC

Este servicio se usa para cargar una descripción de diccionario objeto en un diccionario objeto local.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST

USIGN16	comm_ref	0
USIGN8	layer	FMS
USIGN8	service	LOAD_OV_LOC

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN8	primitive	REQ
INT8	invoke_id	not used
INT16	result	not used

DATA BLOCK PARA REQUEST

Data structure		T_LOAD_OV_REQ
USIGN32	vfd_number	número de VFD
T_OBJECT_DESCR	obj_descr	descrip.obj.(expl.8.7.3.1.1)

SERVICE-DESCRIPTION-BLOCK PARA CONFIRMATION

USIGN16	comm_ref	0
USIGN8	layer	FMS_USR
USIGN8	service	LOAD_OV_LOC
USIGN8	primitive	CON
INT8	invoke_id	not used
INT16	result	POS/NEG

DATA BLOCK PARA CONFIRMACION

result = POS:

Data structure		T_LOAD_OV_CNF
USIGN32	vfd_number	número de VFD

result = NEG:

Data structure		T_SRC_OV_ERROR

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

T_ERROR	error	estr.error estándar
USIGN32	vfd_number	número de VFD
USIGN16	index	índice

TERMINATE-LOAD-OD-LOC

SERVICE-DESCRIPTION-BLOCK PARA REQUEST

USIGN16	comm_ref	0
USIGN8	layer	FMS
USIGN8	service	TERMINATE_LOAD_OV_LOC
USIGN8	primitive	REQ
INT8	invoke_id	not used
INT16	result	not used

DATA BLOCK PARA REQUEST

Data structure		T_TERM_LOAD_OV_REQ
USIGN32	vfd_number	número de VFD

SERVICE-DESCRIPTION-BLOCK PARA CONFIRMATION

USIGN16	comm_ref	0
USIGN8	layer	FMS_USR
USIGN8	service	TERMINATE_LOAD_OV_LOC
USIGN8	primitive	CON
INT8	invoke_id	not used
INT16	result	POS/NEG

DATA BLOCK PARA CONFIRMACION

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

result = POS:

Data structure		T_TERM_LOAD_OV_CNF
USIGN32	vfd_number	número de VFD

result = NEG:

Data structure		T_SRC_OV_ERROR
T_ERROR	error	estr.error estándar
USIGN32	vfd_number	número de VFD
USIGN16	index	índice

8.5.5.1.2 LEYENDO UNA DESCRIPCION DE OBJETO

Como este servicio no se usó, sólo citaremos que está función se realiza mediante el servicio OD-Read.

8.5.5.2 Servicios remotos

8.5.5.2.1 GET-OD

Se pueden leer una o más descripciones de objeto con el servicio GET-OD. El servicio tiene una forma corta y otra larga, siendo esta última opcional.

Se debe especificar el nombre o el índice para la lectura de una descripción de objeto sencilla. Por los nombres se inspecciona en la clase de objeto (access mode).

Para leer más de una, o todas las descripciones de objeto, se debe

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

especificar el índice de la primera descripción de objeto a leer (start index). Para seleccionar la totalidad del diccionario objeto normalmente se debe llamar a este servicio más una vez.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST E INDICATION

USIGN16	comm_ref	2..MAX_COMREF
USIGN8	layer	FMS/FMS_USR
USIGN8	service	GETOV
USIGN8	primitive	REQ/IND
INT8	invoke_id	0...127
INT16	result	not used

DATA BLOCK PARA REQUEST

Data structure		T_GET_OV_REQ
BOOL	format	TRUE = long, FALSE = formato short
USIGN8	dummy	byte de alineación
T_ACC_SPEC	acc_spec	especificación de acceso

Data structure		T_ACC_SPEC
USIGN8	tag	modo de acceso
USIGN8	dummy	byte de alineación

union
{

USIGN16	index	índice del objeto
STRINGV	name[MAX_ACCESS_NAME_LENGTH]	nomb.simbólico objeto

} id

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Modo de acceso:

INDEX_ACCESS	1	acceso por índice
VAR_NAME_ACCESS	2	acceso por nombre de variable
VAR_LIST_NAME_ACCESS	3	acc.por nombr.lista variable
DOMAIN_NAME_ACCESS	4	ac.por el nombre del dominio
PI_NAME_ACCESS	5	ac.por nombre invoc.programa
EVENT_NAME_ACCESS	6	acceso por nombre del evento
START_INDEX_ACCESS	7	acceso por índice de comienzo

SERVICE-DESCRIPTION-BLOCK PARA RESPONSE Y CONFIRMATION

USIGN16	comm_ref	0
USIGN8	layer	FMS/FMS_USR
USIGN8	service	GETOV
USIGN8	primitive	RES/CON
INT8	invoke_id	0...127
INT16	result	POS/NEG

DATA BLOCK PARA CONFIRMACION

result = POS:

Data structure		T_GET_OV_CNF
BOOL	more_follows	siguen más descrip.de objeto
USIGN8	no_of_ov_descr	número de descrip.de objeto
USIGN8	packed_object_descr[no_of_ov_descr]array	paq.descr.obj.

result = NEG:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Data estructura T_ERROR

8.5.5.2.2 SERVICIOS PUT-OD

Como no se han utilizado, sólo diremos que la secuencia es la siguiente:

- Initiate-Put-OD
- Put-OD
- Terminate-Put-OD

8.5.5.3 Estructura de objeto de comunicación

8.5.5.3.1 OBJETOS EN UN DICCIONARIO OBJETO LOCAL

Se transfiere una descripción de objeto completa con el servicio Load-OD-LOC u OD-Read cuando se carga localmente o se lee el diccionario objeto.

La descripción de objeto general es una unión de descripciones de objeto específicas:

- cabecera del diccionario objeto
- objeto nulo
- tipo de dato
- descripción de la estructura del tipo de dato
- variable simple

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

- array

- registro

- lista de variable

- dominio

- alarma (evento)

- invocación de programa

DESCRIPCION GENERAL DEL OBJETO

```
Data structure          T_OBJECT_DESCR

union
{
T_OV_OBJ_DESCR_HDR  ov_obj_descr
T_OV_NULL_OBJECT    null_obj_descr
T_OV_ST_DT_DESCR    dt_obj_descr
T_OV_ST_DS_DESCR    ds_obj_descr
T_SIMPLE_VAR_OBJECT s_var_obj_descr
T_ARRAY_OBJECT          a_var_obj_descr
T_RECORD_OBJECT     r_var_obj_descr
T_VAR_LIST_OBJECT   vlist_obj_descr
T_DOM_OBJECT        dom_obj_descr
T_EVENT_OBJECT          evn_obj_descr
T_PI_OBJECT         pi_obj_descr
} id
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

A) La cabecera del diccionario objeto contiene datos que describen modificablemente, longitud de símbolo, derechos de acceso y versión para todos los índices de comienzo, longitud y dirección interna para cada una de las cuatro partes del OD:

- diccionario tipo estático:Los objetos de la descripción de tipo de datos se almacenan en el diccionario de tipo estático. No se pueden modificar en PROFIBUS.

PROFIBUS reconoce 14 tipos de datos predefinidos que se localizan en índices desde 1 a 14 en el diccionario de tipo estático como sigue:

-BOOLEAN 1

-INTEGER8 2

-INTEGER16 3

-INTEGER32 4

-UNSIGNED8 5

-UNSIGNED16 6

-UNSIGNED32 7

-FLOATING_POINT 8

-VISIBLE_STRING 9

-OCTET_STRING 10

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

-DATE_TYPE 11

-TIME_OF_DAY 12

-TIME_DIFFERENCE 13

-BIT_STRING 14

Los tipos de datos diseñados por uno mismo, se pueden almacenar a partir del índice 15.

- diccionario objeto estático: aquí se almacenan todas los objetos variables exepto lista de variable, dominios y eventos. Todos los servicios de variable se pueden aplicar a los objetos de variable, servicios de eventos a los eventos y servicios de dominio a los dominios

- diccionario dinámico de lista de variables: Se define una lista de variable como un conjunto de objetos que existen y y representan un nuevo objeto que se introduce en el diccionario dinámico de lista de variable.

- diccionario dinámico de invocación de programas: Una invocación de programa es un programa ejecutable que consiste en varios dominios que contienen código de programa y datos. El primer dominio en la lista de dominio asociada, debe contener el código ejecutable. Un objeto PI se puede crear dinámicamente y entonces se introduce en el diccionario dinámico de invocación de programa como nuevo objeto.

Data structure		T_OV_OBJ_DESCR_HDR
USIGN16	index	índice = 0
USIGN8	obj_code	object-code = 1

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

BOOL	flag	FALSE = Protegido de escritura
USIGN8	length	tamaño de los nombres (0-32)
BOOL	protection	soport.protección de obj.
INT16	version	versión
INT16	len_st_ov	long.descripción tipo estát.
USIGN16	first_index_s_ov	índ.com.desc.obj.estát.
INT16	len_s_ov	long.descripción obj.estática
USIGN16	first_index_dv_ov	índ.com.desc.lista.var.dinam.
INT16	len_dv_ov	long.descr.lista var.dinam.
USIGN16	first_index_dp_ov	índ.com.de desc.pi dinámico
INT16	len_dp_ov	long.de descrip. pi dinámico
USIGN32	int_addr	direc.interna de descrip.od
USIGN32	int_addr_st_ov	direc.int.descr.tipo dinám.
USIGN32	int_addr_s_ov	direc.int.descr.obj.dinám.
USIGN32	int_addr_dv_ov	dir.int.descr.lista var.din.
USIGN32	int_addr_dp_ov	dir.interna descr.pi dinámico

B) El objeto nulo es un lugar para tipos de datos estándar no previstos en el diccionario de tipo estático. Los objetos borrados se sitúan en lugar de los objetos nulos.

Data structure T_OV_NULL_OBJECT

USIGN16	index	índice
USIGN8	obj_code	código del objeto
USIGN8	dummy	byte alineación

C) Descripción del tipo de dato:

Data structure T_OV_ST_DT_DESCR

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN16	index	índice
USIGN8	obj_code	código del objeto
USIGN8	dummy	byte alineación
STRINGV	meaning[MAX_OBJECT_NAME_LENGTH]	signif.del tipo

D) Los objetos de descripción de estructuras de tipo de datos describen los tipos de datos, ej. registros. Se almacenan en el mismo lugar que loas anteriores, en el diccionario de tipo estático. Los objetos registros deben contener un puntero a un descriptor de estructura de tipo de datos en sus descripciones de objeto. Una descripción simple de la estructura de tipos de datos puede ser válida para más de un objeto registro.

Data structure		T_OV_ST_DS_DESCR
USIGN16	index	índice
USIGN8	obj_code	código del objeto
USIGN32	reserved	para uso interno
T_OV_DT_LIST	dt_list[no_of_elements]	lista de tipo de dato

Data structure		T_OV_DT_DESCR
USIGN16	index_of_type	índ.de la descr.tipo relatado
USIGN8	length	long.del elemento en octetos
USIGN8	dummy	byte de alineación

E) Variable simple:

Data structure		T_SIMPLE_VAR_OBJECT
USIGN16	index	índice
USIGN8	obj_code	código de objeto

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN8	length	long.del objeto en octetos
USIGN16	index_of_type	dirección lógica del tipo
T_ACCESS	access	estruct.de derecho de acceso
USIGN32	local_address	dirección local
STRINGV	name[MAX_OBJECT_NAME_LENGTH]nombre	
USIGN8	extension[MAX_EXTENSION_LENGTH]extensión	

F) Campo simple de tipo de datos (Array):

Data structure		T_ARRAY_OBJECT
USIGN16	index	índice
USIGN8	obj_code	código de objeto
USIGN8	length	long.del objeto en octetos
USIGN16	index_of_type	dirección lógica del tipo
USIGN8	no_elements	número de elementos array
USIGN8	dummy	byte de alineación
T_ACCESS	access	estruct.de derecho de acceso
USIGN32	local_address	dirección local
STRINGV	name[MAX_OBJECT_NAME_LENGTH]nombre	
USIGN8	extension[MAX_EXTENSION_LENGTH]extensión	

G) Variable estructurada (Registro):

Data structure		T_RECORD_OBJECT
USIGN16	index	índice
USIGN8	obj_code	código de objeto
USIGN8	no_of_address	núm.direccionamientos locales
USIGN16	index_of_type	dirección lógica del tipo
T_ACCESS	access	estruct.de derecho de acceso

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

STRINGV	name[MAX_OBJECT_NAME_LENGTH]nombre	
USIGN8	extension[MAX_EXTENSION_LENGTH]extensión	
USIGN32	reserved	para uso interno
USIGN32	local_address_list[no_of_address]lista de dirección local	

H) Lista de variable

Data structure		T_VAR_LIST_OBJECT
USIGN16	index	índice
USIGN8	obj_code	código de objeto
USIGN8	no_of_var	número de variables
T_ACCESS	access	derecho de acceso
BOOL	deletable	TRUE = se puede borrar
USIGN8	dummy	byte de alineación
STRINGV	name[MAX_OBJECT_NAME_LENGTH]nombre	
USIGN8	extension[MAX_EXTENSION_LENGTH]extensión	
USIGN32	reserved	para uso interno
USIGN16	var_list[no_of_var] lista de variables	

I) Domain

Data structure		T_DOM_OBJECT
USIGN16	index	índice
USIGN8	obj_code	código de objeto
USIGN8	state	estado del dominio
USIGN8	upload_state	estado de carga
INT8	counter	contador en uso
USIGN16	max_octets	longitud máxima de dominio
T_ACCESS	access	protección de acceso

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN32	local_address	dirección local
STRINGV	name[MAX_OBJECT_NAME_LENGTH]	*nombre simbólico*
USIGN8	extension[MAX_EXTENSION_LENGTH]	extensión

J) Alarm (Event):

Data structure T_EVENT_OBJECT

USIGN16	index_event	índice
USIGN8	obj_code	código de objeto
USIGN8	data_length	tamaño de datos de eventos
USIGN16	index_event_data	índice de dato de evento
T_ACCESS	access	protección de acceso
BOOL	enabled	TRUE = evento activado
USIGN8	dummy	byte de alineación
STRINGV	name[MAX_OBJECT_NAME_LENGTH]	nombre simbólico
USIGN8	extension[MAX_EXTENSION_LENGTH]	extensión

K) Program Invocation:

Data structure T_PI_OBJECT

USIGN16	*index*	*pi_índice en OD*
USIGN8	obj_code	código de objeto para OD
USIGN8	*cnt_dom*	*número de dominio*
T_ACCESS	access	acceso
BOOL	deletable	TRUE = se puede borrar
BOOL	reusable	reusable
USIGN8	pi_state	estado de pi
USIGN8	dummy	byte de alineación
STRINGV	name[MAX_OBJECT_NAME_LENGTH]	nomb.simbólico de pi

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN8	extension[MAX_EXTENSION_LENGTH]	extensión
USIGN32	reserved	para uso interno
USIGN16	dom_list[cnt_dom]	lista de índice de dominio

ESTRUCT. DE ACCESO PARA DIRECCIONAR UN OBJETO

Data structure		T_ACC_SPEC
USIGN8	tag	modo de acceso
USIGN8	dummy	byte de alineación
union		
{		
USIGN16	index	índice del objeto
STRINGV	name[MAX_ACCESS_NAME_LENGTH]	nombre simbólico obj.
} id		

Modo de acceso (todos los servicios excepto Get-OD):

ACCESS_INDEX	1	acceso por índice
ACCESS_NAME	2	acceso por nombre simbólico
NAME_LIST	3	acc.nombre de lista variab.

Modo de acceso para servicio Get-OD:

INDEX_ACCES	1	acceso por índice
VAR_NAME_ACCESS	2	acceso por nombre de variable
VAR_LIST_NAME_ACCESS	3	acc.por nombre de lista var.
DOMAIN_NAME_ACCESS	4	ac.por nombre del dominio
PI_NAME_ACCESS	5	acc.por nombre invoc.progr.
EVENT_NAME_ACCESS	6	acc.por el nombre del evento

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

START_INDEX_ACCESS	7	acc.por el índice de comienzo

ESPECIFICACION DE PROTECCION DE ACCESO

Data structure T_ACCESS

USIGN8	pass_word	password
USIGN8	acc_groups	grupos de acceso
USIGN16	acc_right	derechos de acceso

Códigos de objeto:

NULL_OBJECT	0
OV_OBJECT	1
DOMAIN_OBJECT	2
INVOCATION_OBJECT	3
EVENT_OBJECT	4
TYPE_OBJECT	5
TYPE_STRUCT_OBJECT	6
SIMPLE_VAR_OBJECT	7
ARRAY_OBJECT	8
RECORD_OBJECT	9
VAR_LIST_OBJECT	10

Implementación específica:

VAR_OBJECT	11
ALL_OBJECT	12

TRANSFERENCIA DE DESCRIPCION DE OBJETOS PROFIBUS

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Los servicios FMS GET-OD y PUT-OD se pueden usar para leer o modificar los diccionarios objetos de la pareja de comunicación. Ambos servicios son capaces de transferir una o más descripciones objetos. El número de descripciones de objeto actuales transferidas se indica en el parámetro no_of_ov_descr.

La descripción de objeto que se codifica para estos servicios se define en PROFIBUS DIN 19245 part 2, así como la codificación para los tipos de datos básicos. Esta codificación no puede representarse directamente en las descripciones de objeto locales, ya que las descripciones se transfieren en el bus sin separaciones y con formato "Motorola" (high byte first).

La respuesta GET-OD se codifica en el software de con el formato prescrito según el diccionario objeto de datos. La confirmación GET-OD contiene una o más descripciones de objeto en formato string de byte, para interpretarse por la aplicación según el conjunto de normas.

De la misma forma, el servicio PUT-OD transfiere una o más descripciones de objeto como strings de byte y la aplicación debe manejar el estructuramiento y la interpretación según el conjunto de normas.

Cada descripción de objetos individual consiste en un item de longitud de datos y la propia descripción actual del objeto según PROFIBUS DIN19245 part2:

Data structure		T_PACKED_OBJECT_DESCR
USIGN8	length	long.de descr.del paq.de obj.
USIGN8	packed_obj_descr[length]	descrip.paquete del obj.

Esta estructura muestra el formato de una descripción de objeto. Esta estructura en C solamente trabajará correctamente cuando se use una alineación buena de byte. La alineación en delimitadores de palabras o delimitadores largos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

de palabras causa separaciones entre la longitud de datos y la descripción del objeto, lo cual no se permite. En estos casos se recomienda que los campos de byte se definan en las posiciones de datos en las que se introduce las longitudes.

8.5.6 ACCESO A VARIABLES

8.5.6.1 Read

Este servicio lee los valores de los objetos de la pareja de comunicación. Las clases de objetos son variable simple, array, registros y lista de variables. En los casos de arrays y registros, los elementos individuales se pueden acceder usando un subíndice.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST E INDICATION

USIGN16	comm_ref	2..MAX_COMREF
USIGN8	layer	FMS/FMS_USR
USIGN8	service	READ
USIGN8	primitive	REQ/IND
INT8	invoke_id	0...127
INT16	result	not used

DATA BLOCK PARA REQUEST E INDICATION

Data structure		T_VAR_READ_REQ
T_ACC_SPEC	acc_spec	especificación de acceso
INT8	subindex	subíndice
USIGN8	dummy	byte de alineación

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Data structure T_ACC_SPEC

USIGN8	tag	modo de acceso
USIGN8	dummy	byte de alineación

union
{

USIGN16	index	índice del objeto
STRINGV	name[MAX_ACCESS_NAME_LENGTH]	nombre simból.objeto

} id

Modo de acceso:

ACCESS_INDEX	0	acceso por índice
ACCESS_NAME	1	acceso por nombre simbólico
ACCESS_NAME_LIST	2	acc. por nombre de lista variable

SERVICE-DESCRIPTION-BLOCK PARA RESPONSE Y CONFIRMATION

USIGN16	comm_ref	2..MAX_COMREF
USIGN8	layer	FMS/FMS_USR
USIGN8	service	READ
USIGN8	primitive	RES/CON
INT8	invoke_id	0...127
INT16	result	POS/NEG

DATA BLOCK PARA RESPONSE Y CONFIRMATION

result = POS:

Data structure T_VAR_READ_CNF

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN8	dummy	byte de alineación
USIGN8	length	long.del campo de datos en octetos
USIGN8	value[length]	campo de datos

result = NEG:

Data structure T_ERROR

8.5.6.2 Write

Este servicio modifica los valores de los objetos de la pareja de comunicación de las clases de objetos de variable simple, arrays, registros y lista de variables. Para acceder a los elementos simples de los arrays y registros, se puede hacer mediante el subíndice.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST E INDICATION

USIGN16	comm_ref	2..MAX_COMREF
USIGN8	layer	FMS/FMS_USR
USIGN8	service	WRITE
USIGN8	primitive	REQ/IND
INT8	invoke_id	0...127
INT16	result	not used

DATA BLOCK PARA REQUEST E INDICATION

Data structure T_VAR_WRITE_REQ

T_ACC_SPEC	acc_spec	especificación de acceso
INT8	subindex	subíndice
USIGN8	length	long.del campo de datos en octetos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN8	value[length] campo de datos	

Data structure		T_ACC_SPEC
USIGN8	tag	modo de acceso
USIGN8	dummy	byte de alineación
union		
{		
USIGN16	index	índice del objeto
STRINGV	name[MAX_ACCESS_NAME_LENGTH]	nombre simbólico obj.
} id		

Modo de acceso:

ACCESS_INDEX	0	acceso por índice
ACCESS_NAME	1	acc.por el nombre simbólico
ACCESS_NAME_LIST	2	acc.por nombre lista variab.

SERVICE-DESCRIPTION-BLOCK PARA RESPONSE Y CONFIRMATION

USIGN16	comm_ref	2..MAX_COMREF
USIGN8	layer	FMS/FMS_USR
USIGN8	service	WRITE
USIGN8	primitive	RES/CON
INT8	invoke_id	0...127
INT16	result	POS/NEG

DATA BLOCK PARA RESPONSE Y CONFIRMATION

result = POS:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

nada

result = NEG:

Data structure T_ERROR

8.5.6.3 Read-with_type

Este servicio lee los valores de los objetos de la pareja de comunicación y las descripciones de tipos de datos. Las clases de objetos son variable simple, array, registros y lista de variables. En los casos de arrays y registros, los elementos individuales se pueden acceder usando un subíndice.

Como no se ha utilizado, no entraremos en más detalles.

8.5.6.4 Write-with-type

Este servicio modifica los valores y tipos de datos de los objetos de la pareja de comunicación de las clases de objetos de variable simple, arrays, registros y lista de variables. Para acceder a los elementos simples de los arrays y registros, se puede hacer mediante el subíndice.

Tampoco se ha utilizado.

Existen otros tantos que no han sido usados y que se enumeran a continuación: INFORMATION REPORT, INFORMATION-REPORT-WITH-TYPE, PHYSICAL-READ, PHYSICAL-WRITE, DEFINE-VARIABLE-LIST Y DELETE-VARIABLE-LIST.

8.5.7 SERVICIOS DE MANEJO DE DOMINIO

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Como tampoco se han utilizado, los enumeraremos:

- DOWNLOAD SERVICES

- UPLOAD SERVICES

8.5.8 SERVICIOS DE MANEJO DE INVOCACION DE PROGRAMAS

Son los siguientes:

- CREATE-PROGRAM-INVOCATION

- DELETE-PROGRAM-INVOCATION

- START-PROGRAM-INVOCATION

- STOP-PROGRAM-INVOCATION

- RESUME-PROGRAM-INVOCATION

- RESET-PROGRAM-INVOCATION

- KILL-PROGRAM-INVOCATION

- PI-SET-STATE-LOC

8.5.9 SERVICIOS DE MANEJO DE EVENTOS

También pasaremos a enumerarlos:

- EVENT-NOTIFICATION

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

- EVENT-NOTIFICATION-WITH-TYPE

- ACKNOWLEDGE-EVENT-NOTIFICATION

- ALTER-EVENT-CONDITION-MONITORING

8.6 PARAMETROS Y DATOS DE SERVICIO ESPECIFICO FMA7

Antes de empezar, debemos definir la estructura estándar de error, que ya se ha nombrado con anterioridad y que se volverá a utilizar.

Data structure		T_ERROR
USIGN16	class_code	clase de error y código
INT16	add_detail	detalle adicional
STRINGV	add_description[MAX_ERROR_DESCR_LENGTH]	descrip.adic.

8.6.1 SERVICIOS FMA7 LOCALES

8.6.1.1 Set-Bus-Parameters

Este servicio fija todos los parámetros operacionales de la capa 2 con sólo una llamada. Se debe invocar después de la inicialización del software de protocolo.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST

USIGN16	comm_ref	0
USIGN8	layer	FMA7
USIGN8	service	FMA7_SET_BUSPARAMETER
USIGN8	primitive	REQ
INT8	invoke_id	not used

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

INT16 result not used

DATA BLOCK PARA REQUEST

Data structure T_SET_BUSPARAMETER_REQ

USIGN8	loc_add	dirección de la estación local (0..126)
USIGN8	loc_segm	segmento local(0..63,NO_SEGMENT (255))
USIGN8	baud_rate	baudios
USIGN8	medium_red	redundancia del medio
USIGN16	tsl	tiempo de slot
USIGN16	min_tsdr	tiempo de respuesta mín.de la estación
USIGN16	max_tsdr	tiempo de respuesta máx.de la estación
USIGN8	tqui	tiempo callada
USIGN8	tset	tiempo de setup
USIGN32	ttr	tiempo de rotación del testigo
USIGN8	g	factor de separ.entre actualizaciones
BOOL	in_ring_desired	estación activa o pasiva
USIGN8	hsa	dirección de la estación superior
USIGN8	max_retry_limit	límite máximo de reintentos
USIGN16	reserved	0
USIGN8	ident[202]	string FDL

Baud rate:

C_KBAUD_9	0	9,6 KBAUD
C_KBAUD_19	1	19,2 KBAUD
C_KBAUD_93	2	93,75 KBAUD
C_KBAUD_187	3	187,5 KBAUD
C_KBAUD_500	4	500 KBAUD
C_KBAUD_1_5	5	1,5 KBAUD

© Del documento, los autores. Digitalización realizada por ULPGC. Bibliotecas Universitaria, 2006

Redundancia del medio:

C_NO_REDUNDANCY	0	no redundancia del bus
C_BUS_A_HIGHPRIOR	1	el bus A tiene prioridad
C_BUS_B_HIGHPRIOR	2	el bus B tiene prioridad

SERVICE-DESCRIPTION-BLOCK PARA CONFIRMATION

USIGN16	comm_ref	0
USIGN8	layer	FMA7_USR
USIGN8	service	FMA7_SET_BUSPARAMETER
USIGN8	primitive	CON
INT8	invoke_id	not used
INT16	result	POS/NEG

DATA BLOCK PARA CONFIRMATION

result = POS:

nada

result = NEG:

Data structure T_ERROR

8.6.1.2 Cargando la lista de relaciones de comunicación

La carga local de la CRL se hace mediante la siguiente secuencia: Initiate-Load-CRL-Loc, uno a uno los servicios Load-CRL-Loc y finalmente Terminate-Load-CRL-Loc. Todas las relaciones de comunicación excepto la conexión en defecto de manejo se liberan por FMS y LLI y la comunicación se cierra hasta que Terminate-

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Load-CRL-Loc se complete. Una vez se ha ejecutado Initiate-Load-CRL-Loc, la cabecera CRL y las entradas CRL se pueden cargar usando Load-CRL-Loc.

8.6.1.2.1 INITIATE-LOAD-CRL-LOC

Este servicio se encarga de inicializar la carga local del CRL.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST

USIGN16	comm_ref	0
USIGN8	layer	FMA7
USIGN8	service	FMA7_INIT_LOAD_KBL_LOC
USIGN8	primitive	REQ
INT8	invoke_id	not used
INT16	result	not used

DATA BLOCK PARA REQUEST

nada

SERVICE-DESCRIPTION-BLOCK PARA CONFIRMATION

USIGN16	comm_ref	0
USIGN8	layer	FMA7_USR
USIGN8	service	FMA7_INIT_LOAD_KBL_LOC
USIGN8	primitive	CON
INT8	invoke_id	not used
INT16	result	POS/NEG

DATA BLOCK PARA CONFIRMATION

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

result = POS:

nada

result = NEG:

Data structure T_ERROR

8.6.1.2.2 LOAD-CRL-LOC

El servicio Load-CRL-Loc carga la cabecera CRL o la parte estática de una entrada CRL localmente. La cabecera CRL se carga como una entrada CRL con referencia de comunicación 0.

Para cargar el CRL, se debe llamar repetidamente al servicio Load-CRL-Loc. Primero se carga la cabecera, y entonceslas entradas en orden secuencial ascendente.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST

USIGN16	comm_ref	0
USIGN8	layer	FMA7
USIGN8	service	FMA7_LOAD_KBL_LOC
USIGN8	primitive	REQ
INT8	invoke_id	not used
INT16	result	not used

DATA BLOCK PARA REQUEST

Data structure T_LOAD_KBL_REQ

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN16	desired_cr	referencia de comunicación para cargarse
union		
{		
T_KBL_HDR	kbl_hdr	cabecera del CRL
T_KBL_STATIC	kbl_static	entrada estática de CRL
} id		

Data structure		T_KBL_HDR
INT16	nr_of_entries	número de entradas CRL
USIGN8	poll_sap	LSAP para la lista de poll
USIGN8	symbol_length	máxima longitud de símbolo
USIGN32	ass_abt_ci	intervalo de control ASS/ABT
BOOL	vfd_pointer_supported	TRUE:soportado múltiple VFDs
USIGN8	dummy	byte de alineación

Data structure		T_KBL_STATIC
USIGN8	loc_lsap	LSAP local(0..62,BRCT_SAP,DEFAULT_SAP)
USIGN8	rem_add	dirección remota (0..126, ALL)
USIGN8	rem_segm	segmento remota (0..63, NO_SEGMENT)
USIGN8	rem_lsap	rem.LSAP (0..62, BRCT_SAP,DEFAULT_SAP)
USIGN8	conn_type	tipo de conexión (MMAZ..MULT)
USIGN8	lli_sap	LLI SAP (FMS_SAP, FMA7_SAP)
USIGN8	multiplier	multiplicador en conexiones cíclicas
USIGN8	conn_attr	atrib.de conex.(D_CONN,I_CONN,O_CONN)
USIGN8	max_scc	contador de envío confirmado
USIGN8	max_rcc	contador de recepción confirmado
USIGN8	max_sac	contador de envío con conocimiento
USIGN8	max_rac	contador de recepción con conocimiento
USIGN32	ci	intervalo de control

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN8	max_pdu_snd_high
USIGN8	max_pdu_snd_low
USIGN8	max_pdu_rcv_high
USIGN8	max_pdu_rcv_low
USIGN8	feature_supp[FEAT_SUPP_LEN]
STRINGV	symbol[MAX_KBL_SYMBOL_LENGTH]
USIGN32	vfd_pointer
USIGN8	extension[MAX_KBL_EXTENSION_LENGTH]

Atributo de conexión:

D_CONN	0	conexión definida
I_CONN	1	conexión abierta al iniciador
O_CONN	2	conexión abierta al respondedor

Tipo de conexión:

MMAZ	0x00	acíclica maestra/maestra
MSAZ	0x01	acíclica maestra/esclava
MSAZ_SI	0x05	acíclica maestra/esclava con SI
MSZY	0x03	cíclica maestra/esclava
MSZY_SI	0x07	cíclica maestra/esclava con SI
BRCT	0x08	Broadcast
MULT	0x0A	multicast

LLI-SAP:

FMS_SAP	0x00	LLI-SAP para FMS
FMA7_SAP	0x01	LLI-SAP para FMA7

Otros:

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

BRCT_SAP	0x3F	SAP de Broadcast
DEFAULT_SAP	0x80	
GLOBAL_ADDRESS	0x7F	direc.global para broad y multicast
ALL	0xFF	dirección remota en conexiones abiertas
NO_SEGMENT	0xFF	dirección del número de segmento

SERVICE-DESCRIPTION-BLOCK PARA CONFIRMATION

USIGN16	comm_ref	0
USIGN8	layer	FMA7_USR
USIGN8	service	FMA7_LOAD_KBL_LOC
USIGN8	primitive	CON
INT8	invoke_id	not used
INT16	result	POS/NEG

DATA BLOCK PARA CONFIRMATION

result = POS:

nada

result = NEG:

Data structure T_ERROR

8.6.1.2.3 TERMINATE-LOAD-CRL-LOC

Termina la escritura local del CRL.

SERVICE-DESCRIPTION-BLOCK PARA REQUEST

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN16	comm_ref	0
USIGN8	layer	FMA7
USIGN8	service	FMA7_TERM_LOAD_KBL_LOC
USIGN8	primitive	REQ
INT8	invoke_id	not used
INT16	result	not used

DATA BLOCK PARA REQUEST

nada

SERVICE-DESCRIPTION-BLOCK PARA CONFIRMATION

USIGN16	comm_ref	0
USIGN8	layer	FMA7_USR
USIGN8	service	FMA7_TERM_LOAD_KBL_LOC
USIGN8	primitive	CON
INT8	invoke_id	not used
INT16	result	POS/NEG

DATA BLOCK PARA CONFIRMATION

result = POS:

nada

result = NEG:

Data structure	T_KBL_ERROR
T_ERROR	error

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

USIGN16 error_cr referencia de comunicación

Enumeramos otras existentes:

- READ-CRL-LOC
- SET-VALUE-LOC-SERVICES
- READ-VALUE-LOC-SERVICES
- LSAP-STATUS-LOC
- IDENT-LOC
- GET-LIVE-LIST
- FMA7-RESET
- FMA7-EVENT
- PROFIBUS-EXIT
- SET-CONFIGURATION

8.6.2 SERVICIOS REMOTOS

Aquí tenemos los siguientes:

- FMA7-INITIATE
- FMA7-ABORT

© Del documento, los autores. Digitalización realizada por ULPGC. Bibliotecas Universitaria, 2006

- LOAD-CRL-REM-SERVICES

- READ-CRL-REM

- SET-VALUE-REM

- READ-VALUE-REM

- LSAP-STATUS-REM

- IDENT-REM

En el anexo se muestra una relación de todos estos servicios, según el código y según grupo de servicio, así como los códigos de error y clases de FMS y FMA7.

Además, podemos encontrar en el capítulo 10 los listados de todas las librerías, donde se muestran las estructuras y servicios aquí explicados. También los solamente enumerados, ya que aunque no se use la estructura, en muchos casos se ha usado el servicio.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

IMPLEMENTACION

PRACTICA

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

9 IMPLEMENTACION PRACTICA

9.1 OBJETIVOS DE LOS PROGRAMAS

El objetivo de los programas es permitir la comunicación entre un PC y una placa ARD del equipo W-90 de la empresa ISOLUX-WAT. Por lo tanto, se permitirá el intercambio de datos entre las dos estaciones, tanto de lectura como de escritura, así como petición del diccionario objeto para comprobar los tipos de variables que tiene definidas dicha estación.

9.2 CONEXIONES DEL HARDWARD

El esquema de enlace de los Pcs y de la estación es el de la figura 9.1.

Como es comprueba, un PC hace de estación y el otro sirve para visualizar las posiciones de memoria del equipo W_90, las cuales se explican más adelante.

9.3 PASOS IMPORTANTES PARA CONSEGUIR LOS OBJETIVOS

Para lograr el desarrollo de este programa de aplicación, se ha tenido que tener en cuenta unos pasos previos al desarrollo del diagrama de flujos que se describe a continuación.

El programa tiene que hacer posible los siguientes puntos antes de intentar una transmisión:

- Inicialización de la interfase de comunicación de la capa 7

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

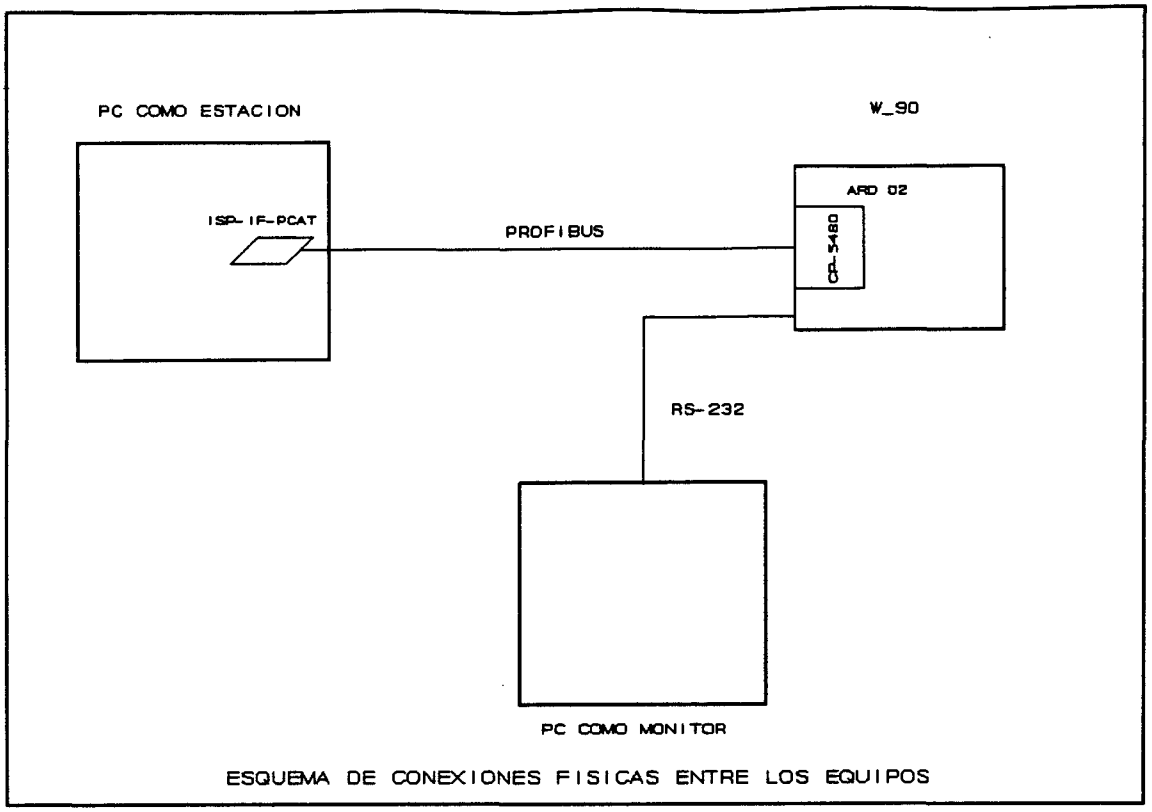


FIGURA 9.1

- Creación de un VFD (Virtual-Field-Deviced)
- Carga del diccionario objeto
- Carga de los parámetros de bus y de la lista de relación de comunicación
- Establecimiento y liberación de la comunicación
- Lectura de un objeto variable
- Escritura de un objeto variable
- Get-OD remoto

La función init_profibus inicializa el controlador PROFIBUS y la parte del host del

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

driver, consistiendo en las siguientes acciones:

- Inicialización de la interfase de comunicación PROFIBUS y del software de comunicación PROFIBUS

- Configuración de la memoria del sistema

- Creación de una o más VFDs

- Carga de uno o más directorios objeto

- Carga de los parámetros de bus

- Carga de la lista de relación de comunicación

- Fijación del "Physical-Status" del VFD

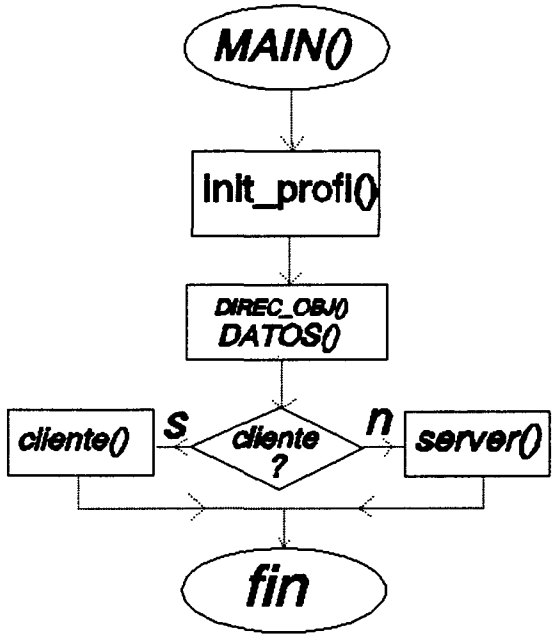
Los parámetros seleccionados para la inicialización del equipo y de la conexión, así como del bus, son los citados en el tema 7. Además, se puede encontrar en el anexo adjunto, más información al respecto.

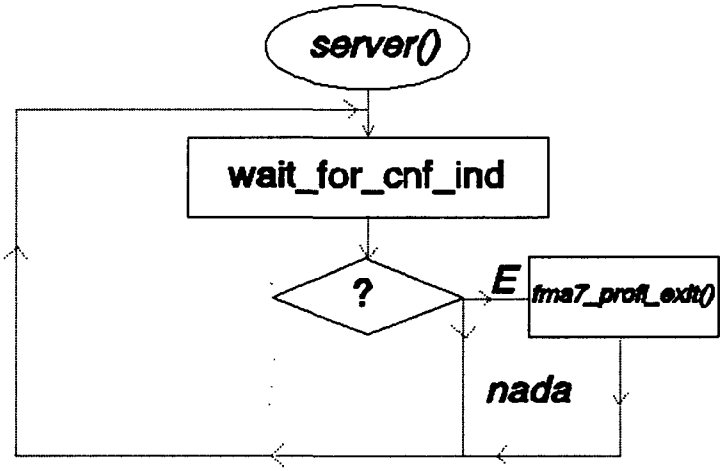
9.4 DIAGRAMAS DE BLOQUES

Se presentan los diagramas de bloque de las funciones que a su vez llaman a otras.

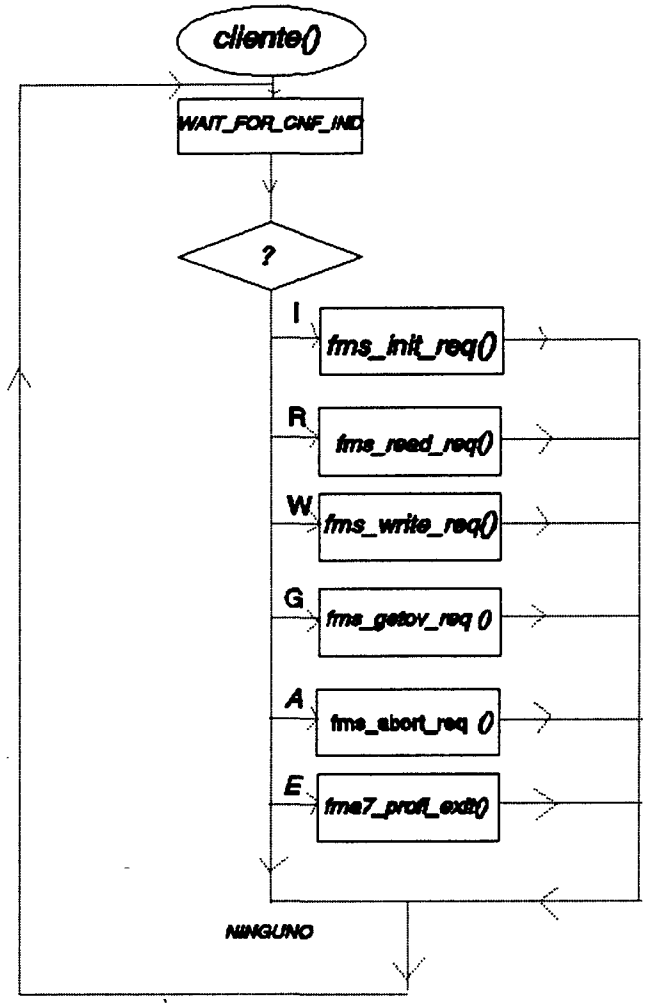
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

DIAGRAMAS DE BLOQUES

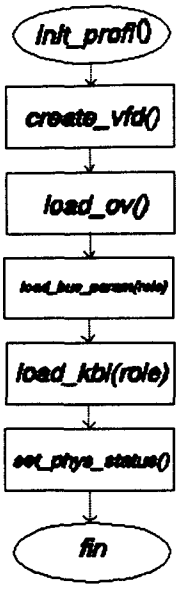




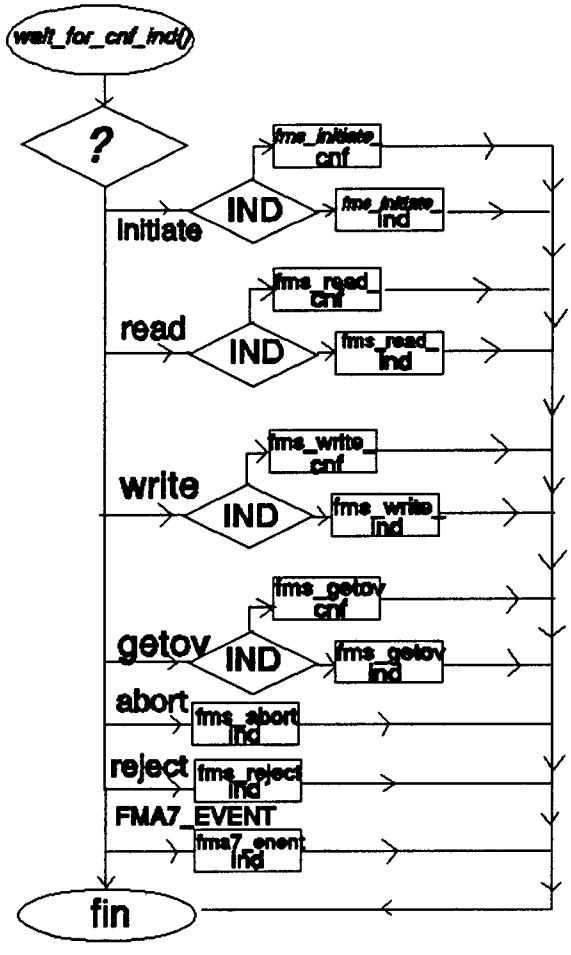
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



Si cualquiera de estas llamadas falla, se llama a fma7_profi_exit().

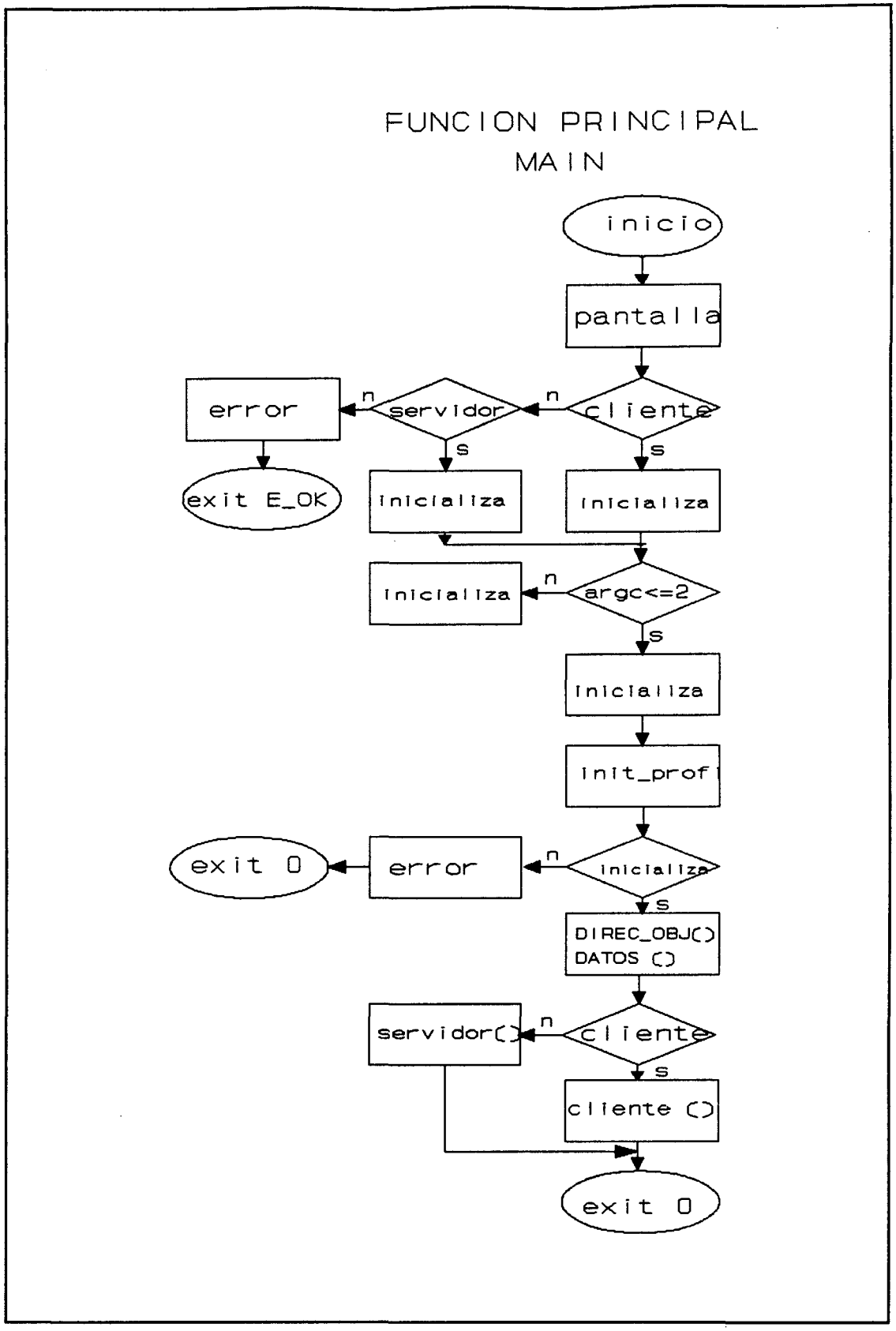


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

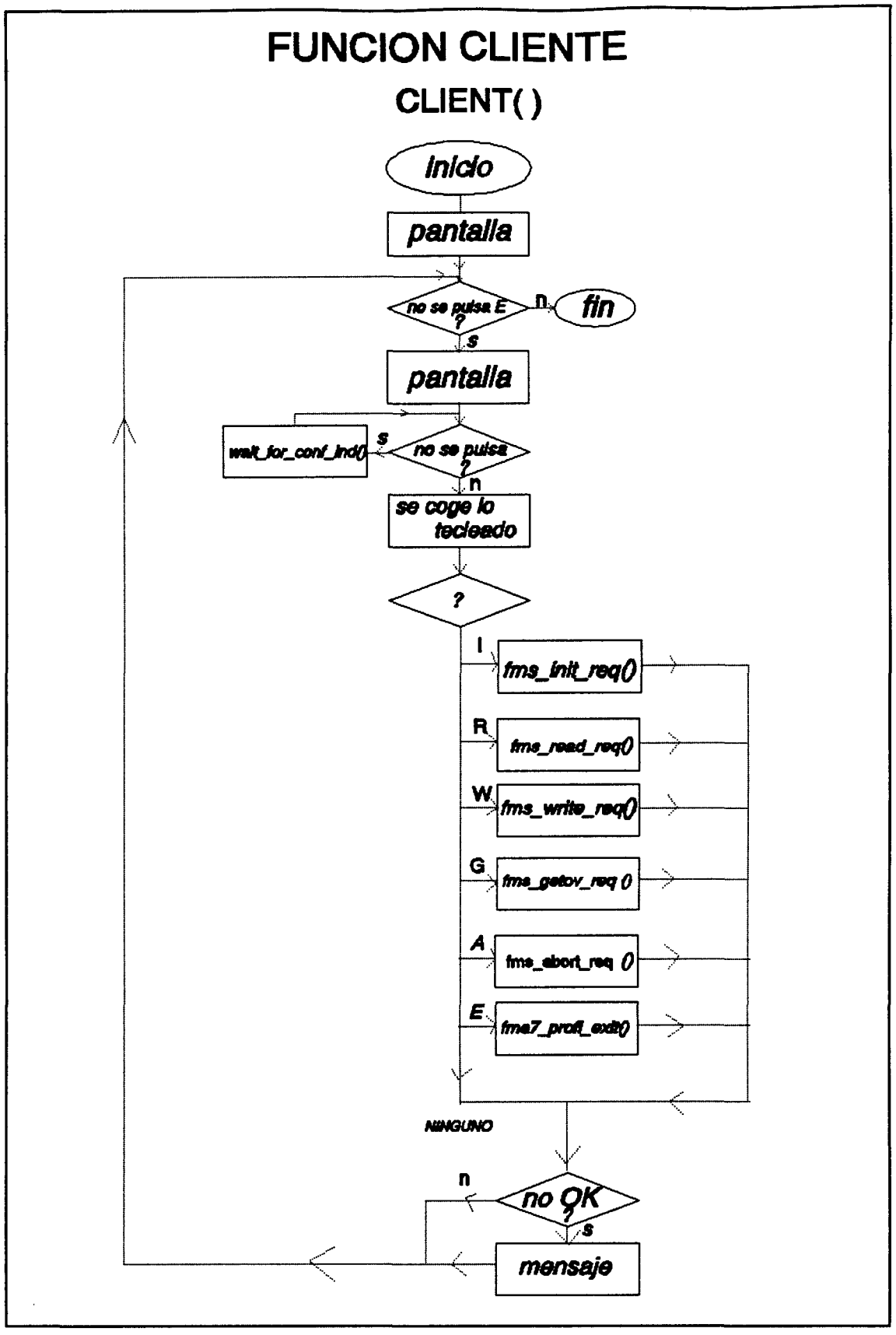
9.5 DIAGRAMAS DE FLUJOS DE LAS FUNCIONES PRINCIPALES

A continuación se presentan los diagramas de flujo de las funciones principales. Para el programa de aplicación del equipo W_90 hay que tener en cuanta que los bloques de pantalla se cambian por introducciones de estado en ciertas posiciones de la SHARED.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

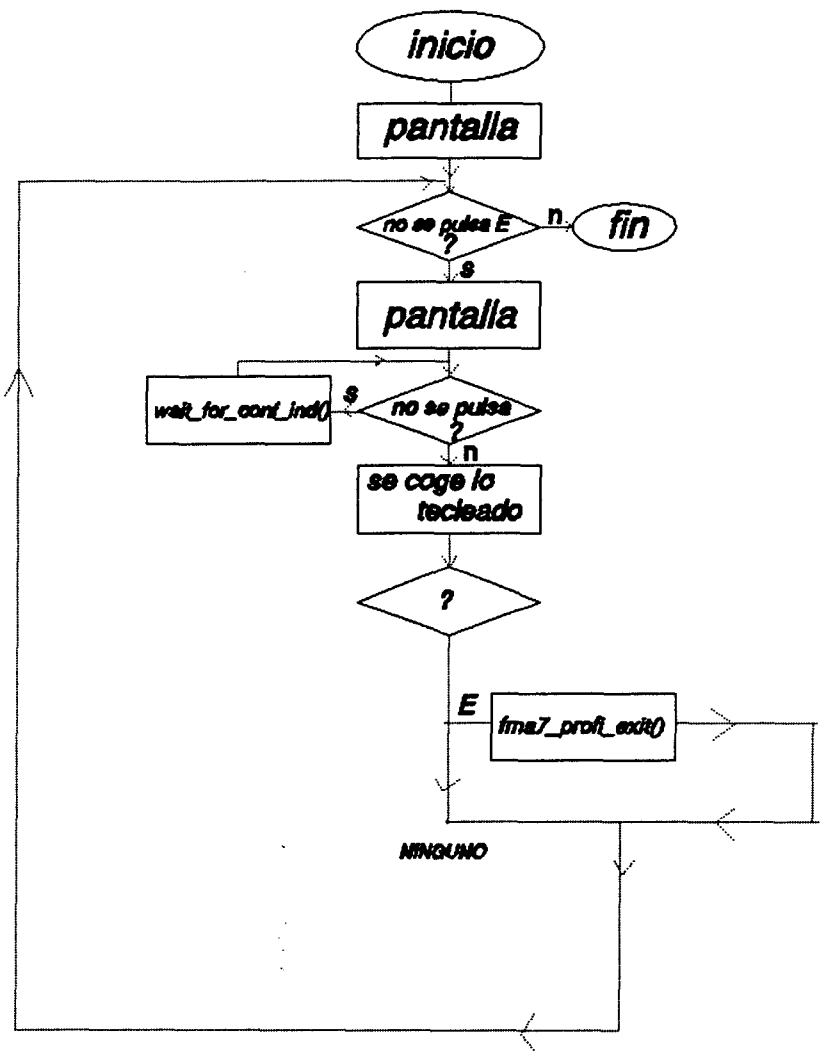


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

FUNCION SERVIDOR SERVER ( )



© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

9.6 PROGRAMAS DE APLICACION

Antes de listar todos los programas de aplicación, debemos explicar el por qué de tres programas y la función de cada uno de ellos.

El primer programa que se muestra, es el programa de aplicación para el PC, el cual se encarga de comunicar e intercambiar información con la estación W_90.

El segundo programa es el programa de aplicación para la estación W_90, el cual se encarga de comunicar e intercambiar información con el PC. Como se puede observar, aquí ya no existen avisos en pantalla y se trabaja con 1000 posiciones de memoria de 2 byte cada una, que la estación define como SHARED. Desde la posición 0 hasta la 835 son para utilización en general (programas lógicos). De la posición 1 a la 100, se encuentran las variables del programa de aplicación PROFIBUS. Vamos a especificarlo mejor:

SHARED [5] = INSTRUCCION:

1-INITIATE	4-GET-OD
2-READ	5-ABORT
3-WRITE	6-EXIT

SHARED [6] = INDICE DEL ELEMENTO CON EL QUE SE VA A TRABAJAR

SHARED [7], [8], [9], [10] = VALOR LEIDO O QUE SE VA A ESCRIBIR

SHARED [20] = ETAPA DE LA INSTRUCCION:

1-REQUEST
2-INDICATION
3-CONFIRMATION
4-ERROR

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

5-NO SUPPORTED

SHARED [22], [23], [24], [25], ... [50] = DETALLES EN ABORT, REJECT, FMA7_EVENT,...

SHARED [51] = FUNCION EN QUE SE ENCUENTRA CUANDO INICIALIZA LA CONEXION:

1-INIT_PROFIBUS	8-LOAD_KBL
2-CREATE_VFD	9-SET_PHYSICAL_STATUS
3-LOAD_HEADER	10-INIT_PROFI
4-LOAD_OV_DATATYPES	11-DIRECCIONES DE OBJETO
5-LOAD_OV_SIMPLEVAR	12-DATOS
6-LOAD_OV	13-MAIN
7-LOAD_BUS_PARAMETERS	14-FMA7_PROFI_EXIT

SHARED[70], ..., [99] = PDU DE GET-OD

De la 111 a la 341 encontramos el diccionario.

A partir de la 836 son variables de uso interno, como direcciones de las 4 interfases, relojes, contadores, ...

Por último, el tercer programa es una librería que creamos basándonos en la original, ya que la original escribía en la memoria de doble puerta de la tarjeta de PROFIBUS words, y por lo tanto las posiciones impares se perdían. Por medio de software se consiguió que escribiera en bytes, es decir, se escribía siempre en posiciones pares. Esto se consiguió buscando todos los accesos a dicha memoria y multiplicando la dirección por dos. Se puede comprobar perfectamente en el programa.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

A continuación presentamos dichos programas, aclarando que las restantes librerías y los ficheros incluidos para definición de variables y estructuras, se encuentran en el anexo.

9.7 LISTADOS DE LOS PROGRAMAS DE APLICACION

9.7.1 PROGRAMA DE APLICACION DE LA ESTACION PC

```
/*****************************************************************************************************************
*                                                *
*                    PROFIBUS                    *
*                                                *
*****************************************************************************************************************
*                                                                                                               *
*                    ISOLUX WAT                                                                                 *
*                    Alcocer 41                                                                                 *
*                    28041 MADRID                                                                               *
*                                                                                                               *
*      Copyright (C) ISOLUX WAT S.A. 1995  All Rights Reserved                                                  *
*                                                                                                               *
*****************************************************************************************************************

FILE_NAME        DEMO.C

PROJECT_NAME     DEMO APPLICATION for PROFIBUS

MODULE           DEMO

COMPONENT_LIBRARY  DEMO.LIB

AUTHOR           Arantxa Alday

VERSION          1.0

DATE             24.02.1995

STATUS

REVIEWER

TESTER

TERMINOLOGY

FUNCTIONAL_MODULE_DESCRIPTION


FUNCTIONAL_SPECIFICATION

DESIGN_SPECIFICATION

CHANGE_NOTES

RELATED_DOCUMENTS

-----------------------------------------------------------------------------------------------------------------*/
#include <keywords.h>

INCLUDES

#include <stdio.h>
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <graph.h>

#include <pb_type.h>
#include <pb_if.h>
#include <pb_fms.h>
#include <pb_fma7.h>
#include <pb_err.h>


GLOBAL_DEFINES

LOCAL_DEFINES
#undef INTEL

/* ---------- internal constants ----------------------------------*/
#define H_DPR_BASE_ADDRESS          0xD0000000
#define CLIENT                      100
#define SERVER                      200

#define WAIT                        0xFF
#define DONT_WAIT                   0x00


/*---------- default values to set the busparameter ----------------*/

#define LEN_ST_OV                   0x000F
#define FIRST_INDEX_S_OV            500
#define LEN_S_OV                    0x000F
#define BOOLEAN                     1
#define I_8                         2
#define I_16                        3
#define I_32                        4
#define U_8                         5
#define U_16                        6
#define U_32                        7
#define F_P                         8
#define V_S                         9
#define O_S                         10
#define D_T                         11
#define T_D                         12
#define T_DF                        13
#define B_S                         14

#define VFD_NUMBER                  0x00000001

#define CLIENT_LOC_ADD              0x05
#define SERVER_LOC_ADD              0x04
#define BAUDRATE                    0x04    /* 500 KBaud      */
#define TSET                        0x05
#define MEDIUM_RED                  0x00    /* Keine Redundanz  */
#define TSL                         2025
#define MIN_TSDR                    50
#define MAX_TSDR                    2000
#define TQUI                        0x00
#define TTR                         100000
#define G                           0x01
#define HSA                         0x06
#define MAX_RETRY_LIMIT             0x01

/*---------- default values of a KBL entry -----------------------*/
#define NIL                         0x3E

#define NR_OF_KBL_ENTRIES           0x0001
#define KR                          0x0005
#define LOC_SAP                     0x05
#define REM_SAP                     0x05
#define CI                          0x00
#define MULTIPLIER                  0x00
#define MAX_SCC                     0x03
#define MAX_RCC                     0x03
#define MAX_SAC                     0x00
#define MAX_RAC                     0x00
#define MAX_FMS_PDU_SIZE            0xF1
#define KBL_SYMBOL                  "MMAZ"
```


© Del documento, los autores. Digitalización realizada por ULPGC. Bibliotecas Universitaria, 2006

```
#define ASS_ABT_CI                4096
#define SYMBOL_LENGTH             0x020

EXPORT_TYPEDEFS

LOCAL_TYPEDEFS

FUNCTION_DECLARATIONS

static VOID    wait_for_cnf_ind(USIGN8);
static INT8    invoke_id;

EXPORT_DATA

IMPORT_DATA


LOCAL_DATA

static T_PROFI_SERVICE_DESCR  sdb;
static USIGN8          cnf_ind_buffer[512];
static INT16           ret_val;
static USIGN16         default_variable_object_data;
static USIGN16         SHARED[813];
static USIGN16         tabla[613];


/* ------------------------------------------------------------------------*/
/* FMS functions:                                                          */
/* - to establish a FMS connection                                         */
/* - to deestablish  a FMS connection                                      */
/* - to read a simple varibale object                                      */
/* - to indicate an abort                                                  */
/* - to indicate a reject                                                  */
/* ------------------------------------------------------------------------*/

FUNCTION static VOID DIRECCIONES_OBJETOS(VOID)
{
LOCAL_VARIABLES

FUNCTION_BODY

tabla[400] = 100;
tabla[401] = 101;
tabla[402] = 102;
tabla[403] = 103;
tabla[404] = 105;
tabla[405] = 106;
tabla[406] = 107;
tabla[407] = 109;
tabla[408] = 111;
tabla[409] = 115;
tabla[410] = 119;
tabla[411] = 123;
tabla[412] = 126;
tabla[413] = 129;

tabla[500] = 100;
tabla[501] = 101;
tabla[502] = 102;
tabla[503] = 103;
tabla[504] = 105;
tabla[505] = 106;
tabla[506] = 107;
tabla[507] = 109;
tabla[508] = 111;
tabla[509] = 115;
tabla[510] = 119;
tabla[511] = 123;
tabla[512] = 126;
tabla[513] = 129;

return;

}
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
FUNCTION static VOID DATOS(VOID)
{
LOCAL_VARIABLES
. . .
FUNCTION_BODY

SHARED[100]=0x01;
SHARED[101]=0x02;
SHARED[102]=0x3333;
SHARED[103]=0x4444;
SHARED[104]=0X4040;
SHARED[105]=0x55;
SHARED[106]=0x6666;
SHARED[107]=0x7777;
SHARED[108]=0X7070;
SHARED[109]=0x8888;
SHARED[110]=0X8080;
SHARED[111]=0x9999;
SHARED[115]=0xAAAA;
SHARED[119]=0xBBBB;
SHARED[123]=0xCCC;
SHARED[126]=0xDDD;
SHARED[129]=0xEEEE;
SHARED[130]=0x0000;
SHARED[131]=0x0000;
SHARED[132]=0XEEEE;

return;

}

FUNCTION static INT16 fms_initiate_req(VOID)

/*-----------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to establish the default connection

possible return values:

== E_OK   -> no error
!= E_OK   -> interface error

-----------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_CTXT_INIT_REQ   init_req;

FUNCTION_BODY

/* --- set data block parameters ------------------------------------*/
init_req.profile_number[0] = 0;
init_req.profile_number[1] = 0;
init_req.password          = 0;
init_req.access_groups     = 0xFF;

/* --- set parameter block parameters -------------------------------*/
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = INITIATE;
sdb.primitive  = REQ;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) &init_req,TRUE));
}


FUNCTION extern INT16 fms_initiate_ind(VOID)

/*-----------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive the initiate indication to establish
the default connection

possible return values:
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
 == E_OK    -> no error
 != E_OK    -> interface error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_CTXT_INIT_REQ FAR*  init_ind = (T_CTXT_INIT_REQ FAR*) cnf_ind_buffer;
T_CTXT_INIT_CNF FAR*  init_rsp = (T_CTXT_INIT_CNF FAR*) cnf_ind_buffer;

FUNCTION_BODY


printf("\nInitiate Indication\n");

/* --- NOTE: the server user has to analyse the indication data structure
          to check profile, password and access groups ---------------- */

/* --- send initiate response ------------------------------------------*/

/* --- set data block parameters ---------------------------------------*/
init_rsp->profile_number[0] = 0;
init_rsp->profile_number[1] = 0;
init_rsp->password          = 0;
init_rsp->access_groups     = 0xFF;


/* --- set parameter block parameters ----------------------------------*/
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = INITIATE;
sdb.primitive  = RES;
sdb.invoke_id  = invoke_id = 0;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) init_rsp,TRUE));
}



FUNCTION static VOID fms_initiate_cnf(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive the confirmation of the FMS-Initiate service

possible return values:

- NONE

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_CTXT_INIT_CNF     FAR* init_cnf     = (T_CTXT_INIT_CNF     FAR*) cnf_ind_buffer;
T_CTXT_INIT_ERR_CNF FAR* init_err_cnf = (T_CTXT_INIT_ERR_CNF FAR*) cnf_ind_buffer;

FUNCTION_BODY

if (sdb.result == POS)
{
  /* --- positive confirmation ------------------------------------------*/
  printf("positive initiate confirmation\n");
  printf("default connection is established\n");
}
else
{
  /* --- negative confirmation ------------------------------------------*/
  printf("negative initiate confirmation\n");
  printf("error class code: %X\n",init_err_cnf->class_code);
}
return;
}



FUNCTION static INT16 fms_read_req(VOID)
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to read the data of the default variable object

possible return values:

- == E_OK   -> no error
- != E_OK   -> interface error

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VAR_READ_REQ    read_req;
USIGN16           VAR_INDEX;
USIGN16           LECTURA;
FUNCTION_BODY

printf("\nIntroduzca el índice del objeto a leer : ");
scanf ("%u",&LECTURA);
VAR_INDEX=LECTURA;
printf("Indice: %u\n",VAR_INDEX);
scanf("%u",&LECTURA);

/*--------Set data block parameters ----------------------------------------*/

read_req.acc_spec.tag      = ACCESS_INDEX;
read_req.acc_spec.id.index = (USIGN16) VAR_INDEX;
read_req.subindex          = 0;

/* --- set parameter block parameters ----------------------------------- */
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = READ;
sdb.primitive  = REQ;
sdb.invoke_id  = invoke_id++;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) &read_req,TRUE));
}


FUNCTION extern INT16 fms_read_ind(VOID)

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive the read indication to read data of the
default variable object

possible return values:

== E_OK   -> no error
!= E_OK   -> interface error

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VAR_READ_REQ FAR *read_ind  = (T_VAR_READ_REQ FAR*) cnf_ind_buffer;
T_VAR_READ_CNF FAR *read_rsp  = (T_VAR_READ_CNF FAR*) cnf_ind_buffer;

USIGN8       FAR *read_data = (USIGN8       FAR*) (read_rsp + 1);
USIGN16          value[4];
FUNCTION_BODY

printf("\nRead Indication\n");
printf("\nRead index: %u\n",read_ind->acc_spec.id.index);

/* --- set data block parameters ---------------------------------------- */

if (read_ind->acc_spec.id.index==413) read_rsp->length=8;
else read_rsp->length=(tabla[read_ind->acc_spec.id.index+1]-tabla[read_ind->acc_spec.id.index])*2;
switch (read_rsp->length)
{
      case 2: value[0]=SHARED[tabla[read_ind->acc_spec.id.index]];
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
            *(USIGN16 FAR*) read_data=value[0];
            printf("\nRead data :%4xh\n",value[0]);
            break;

      case 4: value[0]=SHARED[tabla[read_ind->acc_spec.id.index]];
            value[1]=SHARED[tabla[read_ind->acc_spec.id.index]+1];
            *(USIGN16 FAR*) read_data=value[0];
            *((USIGN16 FAR*) read_data+1)=value[1];
            printf("\nRead data :%4x%4xh\n",value[0],value[1]);
            break;

      case 6: value[0]=SHARED[tabla[read_ind->acc_spec.id.index]];
            value[1]=SHARED[tabla[read_ind->acc_spec.id.index]+1];
            value[2]=SHARED[tabla[read_ind->acc_spec.id.index]+2];
            *(USIGN16 FAR*) read_data=value[0];
            *((USIGN16 FAR*) read_data+1)=value[1];
            *((USIGN16 FAR*) read_data+2)=value[2];
            printf("\nRead data :%4x%4x%4xh\n",value[0],value[1],value[2]);
            break;

      case 8: value[0]=SHARED[tabla[read_ind->acc_spec.id.index]];
            value[1]=SHARED[tabla[read_ind->acc_spec.id.index]+1];
            value[2]=SHARED[tabla[read_ind->acc_spec.id.index]+2];
            value[3]=SHARED[tabla[read_ind->acc_spec.id.index]+3];
            *(USIGN16 FAR*) read_data=value[0];
            *((USIGN16 FAR*) read_data+1)=value[1];
            *((USIGN16 FAR*) read_data+2)=value[2];
            *((USIGN16 FAR*) read_data+3)=value[3];
            printf("\nRead data :%4x%4x%4x%4xh\n",value[0],value[1],value[2],value[3]);
            break;
}
/* --- set parameter block parameters ---------------------------------------*/
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = READ;
sdb.primitive  = RES;

/* --- send read response ---------------------------------------------------*/
return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) read_rsp,TRUE));
}



FUNCTION static VOID fms_read_cnf(VOID)

/*-----------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive the confirmation of the FMS-Read service

possible return values:

- NONE

-----------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VAR_READ_CNF  FAR* read_cnf     = (T_VAR_READ_CNF FAR*) cnf_ind_buffer;
T_ERROR         FAR* read_err_cnf = (T_ERROR        FAR*) cnf_ind_buffer;

USIGN8          FAR* read_data    = (USIGN8         FAR*) (read_cnf + 1);
USIGN16              value[4];


FUNCTION_BODY

if (sdb.result == POS)
{
  /* --- positive confirmation ----------------------------------------------*/

printf("positive read confirmation\n");
switch (read_cnf->length)
{
      case 2 :value[0]=*(USIGN16 FAR*) read_data;
            printf("\nRead data :%4xh\n",value[0]);
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
            break;

      case 4 :value[0]=*(USIGN16 FAR*) read_data;
            value[1]=*((USIGN16 FAR*) read_data+1);
            printf("\nRead data :%4x%4xh\n",value[0],value[1]);
            break;

      case 6 :value[0]=*(USIGN16 FAR*) read_data;
            value[1]=*((USIGN16 FAR*) read_data+1);
            value[2]=*((USIGN16 FAR*) read_data+2);
            printf("\nRead data :%4x%4x%4xh\n",value[0],value[1],value[2]);
            break;

      default :value[0]=*(USIGN16 FAR*) read_data;
            value[1]=*((USIGN16 FAR*) read_data+1);
            value[2]=*((USIGN16 FAR*) read_data+2);
            value[3]=*((USIGN16 FAR*) read_data+3);
            printf("\nRead data :%4x%4x%4x%4xh\n",value[0],value[1],value[2],value[3]);
            break;
}


}
else
{
  /* --- negative confirmation ------------------------------------------*/
  printf("negative read confirmation\n");
  printf("error class code %X\n",read_err_cnf->class_code);
}

return;
}

FUNCTION static INT16 fms_write_req(VOID)
/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to write a date of the default variable object

possible return values:

- == E_OK   -> no error
- != E_OK   -> interface error

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VAR_WRITE_REQ  FAR*   write_req=(T_VAR_WRITE_REQ FAR*) cnf_ind_buffer;
USIGN8          FAR*   write_data=(USIGN8 FAR*) (write_req + 1);
USIGN16                VAR_INDEX;
USIGN16                indice;
USIGN16                value[4];
USIGN16                LECTURA[4];

FUNCTION_BODY

printf("\nIntroduzca el índice del objeto a escribir : ");
scanf ("%u",&indice);
VAR_INDEX=indice;

if (VAR_INDEX==413) write_req->length=8;
else write_req->length=(tabla[VAR_INDEX+1]-tabla[VAR_INDEX])*2;

/*---set data block paramaters-------------------------------------------*/
write_req->acc_spec.tag      = ACCESS_INDEX;
write_req->acc_spec.id.index= (USIGN16) VAR_INDEX;
write_req->subindex          = 0;

/*---set parameter block parameters--------------------------------------*/
sdb.comm_ref =KR;
sdb.layer    =FMS;
sdb.service  =WRITE;
sdb.primitive=REQ;
sdb.invoke_id=invoke_id++;

printf("\nIntroduzca el valor que desea escribir :");
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
switch (write_req->length)
{
      case 2 :scanf("%4x",LECTURA);
            value[0]=LECTURA[0];
            *(USIGN16 FAR*) write_data=value[0];
            printf("\nWrited data :%4xh\n",value[0]);
            scanf ("%u",&indice);
            break;

      case 4 :scanf("%4x%4x",&LECTURA[0],&LECTURA[1]);
            value[0]=LECTURA[0];
            value[1]=LECTURA[1];
            *(USIGN16 FAR*) write_data=value[0];
            *((USIGN16 FAR*)write_data+1)=value[1];
            printf("\nWrited data :%4x%4xh\n",value[0],value[1]);
            scanf ("%u",&indice);
            break;

      case 6 :scanf("%4x%4x%4x",&LECTURA[0],&LECTURA[1],&LECTURA[2]);
            value[0]=LECTURA[0];
            value[1]=LECTURA[1];
            value[2]=LECTURA[2];
            *(USIGN16 FAR*) write_data=value[0];
            *((USIGN16 FAR*) write_data+1)=value[1];
            *((USIGN16 FAR*) write_data+2)=value[2];
            printf("\nWrited data :%4x%4x%4xh\n",value[0],value[1],value[2]);
            scanf ("%u",&indice);
            break;

      case 8 :scanf("%4x%4x%4x%4x",&LECTURA[0],&LECTURA[1],&LECTURA[2],&LECTURA[3]);
            value[0]=LECTURA[0];
            value[1]=LECTURA[1];
            value[2]=LECTURA[2];
            value[3]=LECTURA[3];
            *(USIGN16 FAR*) write_data=value[0];
            *((USIGN16 FAR*) write_data+1)=value[1];
            *((USIGN16 FAR*) write_data+2)=value[2];
            *((USIGN16 FAR*) write_data+3)=value[3];
            printf("\nWrited data :%4x%4x%4x%4xh\n",value[0],value[1],value[2],value[3]);
            scanf ("%u",&indice);
            break;

}

printf("\nPetición de escritura enviada\n");

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) write_req,TRUE));
}

FUNCTION extern INT16 fms_write_ind(VOID)

/*-------------------------------------------------------------------------
FUNCTIONAL DESCRIPTION

this function is used to receive the write indication to write data.
possible return values:

- ==E_OK    ->no error
- !=E_OK    ->interface error
-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES
T_VAR_WRITE_REQ FAR*    write_ind=(T_VAR_WRITE_REQ FAR*) cnf_ind_buffer;
T_VAR_WRITE_REQ FAR*    write_rsp=(T_VAR_WRITE_REQ FAR*) cnf_ind_buffer;
USIGN8          FAR*    write_data=(USIGN8 FAR*) (write_ind+1);
USIGN16         value[4];
USIGN16         direccion;

FUNCTION_BODY
printf ("\nWrite Indication\n");
printf ("\nWrited index: %u\n",write_ind->acc_spec.id.index);
if (write_ind->acc_spec.id.index==413) write_ind->length=8;
else write_ind->length=(tabla[write_ind->acc_spec.id.index+1]-tabla[write_ind->acc_spec.id.index])*2;

/*---set parameter block parameters-----------------------------------------*/
sdb.comm_ref   =KR;
sdb.layer      =FMS;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
sdb.service    =WRITE;
sdb.primitive  =RES;
direccion=tabla[write_ind->acc_spec.id.index];
switch (write_ind->length)
{
      case 2 :value[0]=*(USIGN16 FAR*) write_data;
              SHARED[direccion]=value[0];
              printf("\nWrited data :%4xh\n",value[0]);
              break;

      case 4 :value[0]=*(USIGN16 FAR*) write_data;
              value[1]=*((USIGN16 FAR*) write_data+1);
              SHARED[direccion]=value[0];
              SHARED[direccion+1]=value[1];
              printf("\nWrited data :%4x%4xh\n",value[0],value[1]);
              break;

      case 6 :value[0]=*(USIGN16 FAR*) write_data;
              value[1]=*((USIGN16 FAR*) write_data+1);
              value[2]=*((USIGN16 FAR*) write_data+2);
              SHARED[direccion]=value[0];
              SHARED[direccion+1]=value[1];
              SHARED[direccion+2]=value[2];
              printf("\nWrited data :%4x%4x%4xh\n",value[0],value[1],value[2]);
              break;

      case 8 :value[0]=*(USIGN16 FAR*) write_data;
              value[1]=*((USIGN16 FAR*) write_data+1);
              value[2]=*((USIGN16 FAR*) write_data+2);
              value[3]=*((USIGN16 FAR*) write_data+3);
              SHARED[direccion]=value[0];
              SHARED[direccion+1]=value[1];
              SHARED[direccion+2]=value[2];
              SHARED[direccion+3]=value[3];
              printf("\nWrited data :%4x%4x%4x%4xh\n",value[0],value[1],value[2],value[3]);
              break;

}

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb, (VOID FAR*) write_rsp,TRUE));
}

FUNCTION static VOID fms_write_cnf(VOID)

/*-------------------------------------------------------------------------
FUNCTIONAL DESCRIPTION

this function is used to receivethe confirmation of the FMS-Read-Service.

possible returns value

- NONE
-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VAR_WRITE_REQ FAR* write_cnf=(T_VAR_WRITE_REQ FAR*) cnf_ind_buffer;
T_ERROR         FAR* write_err_cnf=(T_ERROR FAR*) cnf_ind_buffer;

FUNCTION_BODY
if (sdb.result==POS) printf ("positive write confirmation \n");
else
{ printf("\nNegative write confirmation\n");
  printf("\nError class code %x\n",write_err_cnf->class_code);
}
return;
}

FUNCTION static INT16 fms_getov_req(VOID)

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to read the data of the default variable object.

possible return:
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
- == E_OK -> no error
- != E_OK -> interface error

-----------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_GET_OV_REQ    getov_req;
USIGN16         VAR_INDEX;

FUNCTION_BODY

printf("Introduzca el índice del objeto : ");
scanf ("%d",&VAR_INDEX);

/*--- set data block parameters ------------------------------------------*/
getov_req.format = TRUE;
getov_req.acc_spec.tag = INDEX_ACCESS;
getov_req.acc_spec.id.index= (USIGN16) VAR_INDEX;

/*--- set parameter block parameters -------------------------------------*/
sdb.comm_ref = KR;
sdb.layer    = FMS;
sdb.service  = GETOV;
sdb.primitive= REQ;
sdb.invoke_id= invoke_id++;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) &getov_req,TRUE));
}

FUNCTION extern INT16 fms_getov_ind(VOID)

/*-----------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

thid function is used to receive the read indication to get object
description from the object dictionary.

possible return values:

- == E_OK  -> no error
- != E_OK  -> interface error

-----------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_GET_OV_REQ   FAR    *getov_ind  = (T_GET_OV_REQ  FAR*) cnf_ind_buffer;
T_GET_OV_CNF   FAR    *getov_rsp  = (T_GET_OV_CNF  FAR*) cnf_ind_buffer;

FUNCTION_BODY

printf("\nGet_OD Indication\n");

/*--- set parameter block parameters -------------------------------------*/
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = GETOV;
sdb.primitive  = RES;
sdb.result     = POS;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb, (VOID FAR*) getov_rsp,TRUE));
}

FUNCTION static VOID fms_getov_cnf(VOID)

/*-----------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive the confirmation of the FMS-GET_OD service

possible return values:

- NONE

-----------------------------------------------------------------------*/
{
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
LOCAL_VARIABLES

T_GET_OV_CNF    FAR*    getov_cnf       = (T_GET_OV_CNF FAR*) cnf_ind_buffer;
T_ERROR         FAR*    getov_err_cnf   = (T_ERROR      FAR*) cnf_ind_buffer;
USIGN8          FAR*    get_ov          = (USIGN8       FAR*) (getov_cnf +1);
USIGN8          i;
USIGN8          j;

FUNCTION_BODY

i=0;
j=1;

if (sdb.result == POS)
{
  printf("\nPositive get_od confirmation\n");
  if (getov_cnf->no_of_ov_descr == 0) printf("\nNo hay objeto\n");

  else
  {
  while (j<=getov_cnf->no_of_ov_descr)
   {
      printf(" Objeto solicitado :");
      while (i<=get_ov[0])
      {
            printf(" %x ",get_ov[i]);
            i=i+1;
      }
      i=0;
      printf("\n En carácter ASCII : ");

      while (i<=get_ov[0])
      {
         if ((get_ov[i]>=0x20) & (get_ov[i]<=0x7E)) printf("%c",get_ov[i]);
         i=i+1;
      }

      *get_ov=*get_ov + get_ov[0] + 1;
      j=j+1;
      printf("\n");
   }
  }
}
else
{
  printf("\nNegative get_od confirmation\n");
  printf("\nError class code %x\n",getov_err_cnf->class_code);
}

return;
}

FUNCTION static INT16 fms_abort_req(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to deestablish the open default connection

possible return values:

- == E_OK    -> no error
- != E_OK    -> interface error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_CTXT_ABORT_REQ abort_req;

FUNCTION_BODY

/* --- set data block parameters ----------------------------------*/
abort_req.abort_id = USR;
abort_req.reason   = USR_ABT_RC1;        /* ---- disconect ----- */
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = ABORT;
sdb.primitive  = REQ;
sdb.invoke_id  = invoke_id = 0;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) &abort_req,TRUE));
}


FUNCTION static VOID fms_abort_ind(VOID)

/*-----------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive an FMS abort indication

possible return values:

- NONE

-----------------------------------------------------------------------*/
{
LOCAL_VARIABLES


T_CTXT_ABORT_REQ FAR *abort = (T_CTXT_ABORT_REQ FAR*) cnf_ind_buffer;
USIGN8 i;
FUNCTION_BODY

printf("Abort Indication\n");
printf("local                    : %d\n",abort->local);
printf("id                       : %d\n",abort->abort_id);
printf("reason                   : %d\n",abort->reason);
printf("Longitud                 : %d\n",abort->detail_length);
printf("Information about abort reason:");
i=0;
while(i!=abort->detail_length)
{
  printf("%d",abort->detail[i]);
  i=i+1;
}
return;
}


FUNCTION static VOID fms_reject_ind(VOID)

/*-----------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive an FMS reject indication

possible return values:

- NONE

-----------------------------------------------------------------------*/
{
LOCAL_VARIABLES


T_CTXT_REJECT_IND FAR *reject = (T_CTXT_REJECT_IND FAR*) cnf_ind_buffer;

FUNCTION_BODY

printf("Reject Indication\n");
printf("local     : %d\n",reject->detected_here);
printf("invoke id : %d\n",reject->orig_invoke_id);
printf("pdu type  : %d\n",reject->pdu_type);
printf("reason    : %d\n",reject->reject_code);

return;
}
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* ------------------------------------------------------------------*/
/* FMA7 functions:                                                   */
/* - to indicate FMA7 events                                         */
/* - lli fault indications                                           */
/* - to reset and leave the Demo application                         */
/* ------------------------------------------------------------------*/


FUNCTION static VOID fma7_event_ind(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive a fma7 event indication
- fma2 event
- lli fault indication
- lli event

possible return values:

- NONE

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_FMA7_EVENT_IND FAR* event = (T_FMA7_EVENT_IND FAR*) cnf_ind_buffer;

FUNCTION_BODY

printf("FMA7 Event Indication\n");
printf("comm_ref: %d\n",event->comm_ref);
printf("id      : %d\n",event->instance_id);
printf("reason  : %d\n",event->reason);

return;
}


FUNCTION static INT16 fma7_profi_exit(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function is used to reset and leave the PROFIBUS communication software

Note: For a new start up of the PROFIBUS communication software, initialize
      the PROFIBUS communication interface and PROFIBUS communication software
      using by the init_profibus() function


possible return values:

- == E_OK   -> no error
- != E_OK   -> error


-------------------------------------------------------------------*/
{
LOCAL_VARIABLES


FUNCTION_BODY

/* --- set parameter block parameters ------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_PROFIBUS_EXIT;
sdb.primitive  = REQ;

if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                          NULL,
                          (USIGN8) TRUE )) != E_OK )
{
  exit(ret_val);
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
}


/* --- wait for confirmation and leave the program ---------------------- */
wait_for_cnf_ind(WAIT);

exit(0);
return(E_OK);
}


/* ------------------------------------------------------------------------*/
/* scheduling functions:                                                   */
/* - to enter client funtionality                                          */
/* - to enter server funtionality                                          */
/* - to schedule the demo application and PROFIBUS communication interface */
/* ------------------------------------------------------------------------*/

FUNCTION static VOID wait_for_cnf_ind
        (
          IN USIGN8 wait
        )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function is used to receive an indication or confirmation from
the PROFIBUS communication interface


-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES
T_VFD_ERROR FAR*  error = (T_VFD_ERROR FAR*) cnf_ind_buffer;
USIGN16 buffer_len;

FUNCTION_BODY


for(;;)
{
  buffer_len = sizeof(cnf_ind_buffer);
  ret_val = profi_rcv_con_ind((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                              (VOID FAR *) cnf_ind_buffer,
                              &buffer_len);

  if (ret_val == CON_IND_RECEIVED)
  {
    if (sdb.layer == FMS_USR)
    {
      switch(sdb.service)
      {
        case INITIATE:
            if (sdb.primitive == IND) fms_initiate_ind();
            else                      fms_initiate_cnf();
            break;

        case READ:
            if (sdb.primitive == IND) fms_read_ind();
            else                      fms_read_cnf();
            break;

        case WRITE:
            if (sdb.primitive == IND) fms_write_ind();
            else                      fms_write_cnf();
            break;

        case GETOV:
            if (sdb.primitive == IND) fms_getov_ind();
            else                      fms_getov_cnf();
            break;

        case ABORT:
            fms_abort_ind();
            break;

        case REJECT:
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
            fms_reject_ind();
            break;

        case INITIATE_LOAD_OV_LOC:
        case LOAD_OV_LOC:
        case TERMINATE_LOAD_OV_LOC:
        case CREATE_VFD_LOC:
        case VFD_SET_PHYS_STATUS_LOC:
            if (sdb.result == POS)
            {
              printf("\npositive confirmation\n");
            }
            else
            {
              printf("\nnegative confirmation\n");
              printf("\nservice    : %d\n",sdb.service);
              printf("\nclass code : %X\n",error->error.class_code);
              exit(1);
            }
            if (wait == WAIT) return;
            break;

        default:
            printf("service not supported\n");
            break;
      }
      if (wait == DONT_WAIT) return;
    }
    else
    {
      switch(sdb.service)
      {
        case FMA7_SET_BUSPARAMETER:
        case FMA7_INIT_LOAD_KBL_LOC:
        case FMA7_LOAD_KBL_LOC:
        case FMA7_TERM_LOAD_KBL_LOC:
        case FMA7_PROFIBUS_EXIT:
            if (sdb.result == NEG)
            {
              printf("\nnegative confirmation\n");
              printf("\nservice : %d\n",sdb.service);
              printf("\nservice : %X\n",error->error.class_code);
              exit(1);
            }

            if (wait == WAIT) return;
            break;

        case FMA7_EVENT:
            fma7_event_ind();
            break;

        default:
            printf("service not supported\n");
            break;
      }
      if (wait == DONT_WAIT) return;
    }
  }
  else
  {
   if (ret_val == NO_CON_IND_RECEIVED)
   {
     if (wait == DONT_WAIT) return;
   }
   else
   {
    printf("PROFIBUS interface error %d\n",ret_val);
    exit(ret_val);
   }
  }
}
return;
}
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
FUNCTION static VOID client(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

CLIENT waits for keyboard inputs of the user

- with the demo application the user can do the following FMS services

  1.) INITIATE:        establish the default connection
  2.) READ VARIABLE:   read the data of the default variable object
  3.) WRITE VARIABLE:  write the data from the default variable object
  4.) GET_OD           get object descriptions
  5.) ABORT:           deestablish the open default connection
  6.) EXIT:            leave the demo application


-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

USIGN8 c = '\0';

FUNCTION_BODY

 _clearscreen ( _GCLEARSCREEN );

 while (toupper(c) != 'E')
 {
  ret_val = E_OK;

  _clearscreen ( _GCLEARSCREEN );
  printf("\n");
  printf("\n");
  printf("\n");
  printf("              'I' : INITIATE\n");
  printf("\n");
  printf("              'R' : READ VARIABLE\n");
  printf("\n");
  printf("              'W' : WRITE VARIABLE\n");
  printf("\n");
  printf("              'G' : GET OBJECT DESCRIPTIONS\n");
  printf("\n");
  printf("              'A' : ABORT\n");
  printf("\n");
  printf("              'E' : EXIT\n\n\n");

  while (!kbhit()) wait_for_cnf_ind(DONT_WAIT);

  c = (USIGN8) getch();

  switch (toupper(c))
  {
   case 'I':
        ret_val = fms_initiate_req();
        break;

   case 'R':
        ret_val = fms_read_req();
        break;

   case 'W':
        ret_val = fms_write_req();
        break;

   case 'G':
        ret_val = fms_getov_req();
        break;

   case 'A':
        ret_val = fms_abort_req();
        break;

   case 'E':
        fma7_profi_exit();
        break;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
    default:
        break;
   }

   if (ret_val != E_OK) printf("\n profi_snd_req_res failed %d\n",ret_val);

}
return;
}


FUNCTION static VOID server(VOID)

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

SERVER waits for keyboard inputs of the user

- with the demo application the user can do the following FMS services

 1.) EXIT:          leave the demo application


-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

USIGN8 c = '\0';

FUNCTION_BODY

 _clearscreen ( _GCLEARSCREEN );

 while (toupper(c) != 'E')
 {
  _clearscreen ( _GCLEARSCREEN );
  printf("\n");
  printf("\n");
  printf("\n");
  printf("\n");
  printf("          'E' : EXIT\n");

  while (!kbhit()) wait_for_cnf_ind(DONT_WAIT);

  c = (USIGN8) getch();

  switch (toupper(c))
  {
   case 'E':
       fma7_profi_exit();
       break;

   default:
       break;
  }
}

return;
}


/* ------------------------------------------------------------------------*/
/* intialization                                                           */
/* - initialize the PROFIBUS communication interface                       */
/* - download PROFIBUS firmware                                            */
/* - create a vfd                                                          */
/* - load the ov                                                           */
/* - load the bus parameter                                                */
/* - load the kbl                                                          */
/* - set the physical device status in the vfd                             */
/* ------------------------------------------------------------------------*/


/*=========================================================================*/
```


© Del documento, los autores. Digitalización realizada por ULPGC. Bibliotecas Universitaria, 2006

```
FUNCTION static INT16 create_vfd(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to create a VFD

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VFD_CREATE_REQ  create_vfd;

FUNCTION_BODY


/* --- set data block parameters ---------------------------------*/

create_vfd.vfd_number = (USIGN32) VFD_NUMBER;

/* vendor name (byte 0 contains the length of the vendor name) ---------- */
create_vfd.vendor_name[0] = 10;
strcpy(&create_vfd.vendor_name[1],"ISOLUX WAT");

/* model name (byte 0 contains the length of the model name) ------------ */
create_vfd.model_name[0]  = 20;
strcpy(&create_vfd.model_name[1],"CREACION DICCIONARIO");

/* revision (byte 0 contains the length of the revision) ---------------- */
create_vfd.revision[0] = 3;
strcpy(&create_vfd.revision[1],"1.0");

/* -- profile number (no profile is specified) ---------------------- */
create_vfd.profile_number[0] = 0;
create_vfd.profile_number[1] = 0;


/* --- set parameter block parameters ----------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = CREATE_VFD_LOC;
sdb.primitive  = REQ;

/* --- send request ----------------------------------------------*/
if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*)
                          &sdb,(VOID FAR*) &create_vfd,
                          TRUE
                         )) != E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);
return(E_OK);
}


FUNCTION static INT16 load_ov_header(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to load the OV header

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_LOAD_OV_REQ  load_ov_req;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
FUNCTION_BODY

/* --- set the data block parameters ------------------------------------*/

load_ov_req.vfd_number  = VFD_NUMBER;

load_ov_req.obj_descr.id.ov_obj_descr.index           = 0;
load_ov_req.obj_descr.id.ov_obj_descr.obj_code        = OV_OBJECT;
load_ov_req.obj_descr.id.ov_obj_descr.flag            = TRUE;
load_ov_req.obj_descr.id.ov_obj_descr.length          = 32;
load_ov_req.obj_descr.id.ov_obj_descr.protection      = 0;
load_ov_req.obj_descr.id.ov_obj_descr.version         = 01;
load_ov_req.obj_descr.id.ov_obj_descr.len_st_ov       = LEN_ST_OV;
load_ov_req.obj_descr.id.ov_obj_descr.first_index_s_ov = FIRST_INDEX_S_OV;
load_ov_req.obj_descr.id.ov_obj_descr.len_s_ov        = LEN_S_OV;
load_ov_req.obj_descr.id.ov_obj_descr.first_index_dv_ov= 0;
load_ov_req.obj_descr.id.ov_obj_descr.len_dv_ov       = 0;
load_ov_req.obj_descr.id.ov_obj_descr.first_index_dp_ov= 0;
load_ov_req.obj_descr.id.ov_obj_descr.len_dp_ov       = 0;
load_ov_req.obj_descr.id.ov_obj_descr.int_addr        = 0xFFFFFFFF;
load_ov_req.obj_descr.id.ov_obj_descr.int_addr_st_ov  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.ov_obj_descr.int_addr_s_ov   = 0xFFFFFFFF;
load_ov_req.obj_descr.id.ov_obj_descr.int_addr_dv_ov  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.ov_obj_descr.int_addr_dp_ov  = 0xFFFFFFFF;

/* --- set the parameter block parameter --------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;


if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 (VOID FAR *) &load_ov_req,
                                 (USIGN8) TRUE )) != E_OK )
  return(ret_val);

/* --- wait for confirmation --------------------------------------------*/

wait_for_cnf_ind(WAIT);

return(E_OK);

}

FUNCTION static INT16 load_ov_datatypes(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to load the OV datatypes

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_LOAD_OV_REQ  load_ov_req;

FUNCTION_BODY

/* --- set data block parameters ----------------------------------------*/

load_ov_req.vfd_number              = VFD_NUMBER;

load_ov_req.obj_descr.id.dt_obj_descr.index     = BOOLEAN;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code  = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 7;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "BOOLEAN");
/* --- set parameter block parameters -----------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE )) ;
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = I_8;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "I_8");
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

load_ov_req.obj_descr.id.dt_obj_descr.index      = I_16;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 4;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "I_16");
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

load_ov_req.obj_descr.id.dt_obj_descr.index      = I_32;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 4;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "I_32");
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE )) ;

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

load_ov_req.obj_descr.id.dt_obj_descr.index      = U_8;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "U_8");
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
load_ov_req.obj_descr.id.dt_obj_descr.index      = U_16;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 4;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "U_16");
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = U_32;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 4;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "U_32");
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = F_P;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "F_P");
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = V_S;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "V_S");
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = O_S;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "O_S");
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = D_T;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "D_T");
/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = T_D;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "T_D");
/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = T_DF;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 4;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "T_DF");
/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;
(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = B_S;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
strcpy (&load_ov_req.obj_descr.id.dt_obj_descr.meaning[1], "B_S");

/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

/* --- wait for confirmation ---------------------------------------------*/

return(E_OK);
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
}

FUNCTION static INT16 load_ov_simplevar(VOID)
-- --
/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to load the OV simple variable

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_LOAD_OV_REQ  load_ov_req;

FUNCTION_BODY

default_variable_object_data = 0;

/* --- set data block parameters ------------------------------------*/
load_ov_req.vfd_number              = VFD_NUMBER;

load_ov_req.obj_descr.id.s_var_obj_descr.index          = 500;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 1;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = BOOLEAN;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 05;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-1");

/* --- set parameter block parameters -------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index          = 501;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 1;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = I_8;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 05;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-2");

/* --- set parameter block parameters -------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
load_ov_req.obj_descr.id.s_var_obj_descr.index          = 502;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 2;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = I_16;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 05;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-3");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index          = 503;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 4;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = I_32;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 05;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-4");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index          = 504;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 1;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = U_8;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 05;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-5");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index          = 505;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 2;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = U_16;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 05;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-6");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;
(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index           = 506;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 4;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = U_32;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 05;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-7");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;
(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index           = 507;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 4;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = F_P;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 05;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-8");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index           = 508;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 8;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = V_S;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 05;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-9");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index           = 509;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 8;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = O_S;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 06;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-10");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index           = 510;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 7;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = D_T;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 06;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-11");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index           = 511;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 4;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = T_D;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 06;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-12");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

load_ov_req.obj_descr.id.s_var_obj_descr.index            = 512;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code         = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length           = 4;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type    = T_DF;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 06;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-13");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

load_ov_req.obj_descr.id.s_var_obj_descr.index            = 513;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code         = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length           = 8;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type    = B_S;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 06;
strcpy(&load_ov_req.obj_descr.id.s_var_obj_descr.name[1],"CTR-14");

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

/* --- wait for confirmation --------------------------------------------- */

return(E_OK);
}

FUNCTION static INT16 load_ov(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
this function is used to load the OV

possible return values:
- == E_OK    -> no error
- != E_OK    -> error

------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_INIT_LOAD_OV_REQ     init_load_ov_req;
T_TERM_LOAD_OV_REQ     term_load_ov_req;

FUNCTION_BODY

/* ------------------------------------------------------------------------*/
/* ---------------- initiate load OV ------------------------------------- */
/* ------------------------------------------------------------------------*/

/* --- set data block parameters ---------------------------------------- */
init_load_ov_req.vfd_number    = VFD_NUMBER;
init_load_ov_req.consequence   = CONSEQUENCE_2;

/* --- set parameter block parameters ----------------------------------- */
sdb.comm_ref    = 0;
sdb.layer       = FMS;
sdb.service     = INITIATE_LOAD_OV_LOC;
sdb.primitive   = REQ;

if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 (VOID FAR *) &init_load_ov_req,
                                 (USIGN8) TRUE)) != E_OK)
{
  return(ret_val);
}


/* --- wait for confirmation -------------------------------------------- */
wait_for_cnf_ind(WAIT);


/* ------------------------------------------------------------------------*/
/* ----------------load OV descriptions ------------------------------------ */
/* ------------------------------------------------------------------------*/

/* ---------------- load OV header ------------------------------------- */
if ((ret_val = load_ov_header()) != E_OK) return(ret_val);

/* -------- load OV standard datatypes ------------------------------------ */
if ((ret_val = load_ov_datatypes()) != E_OK) return(ret_val);

/* -------- load OV simple variable ------------------------------------ */
if ((ret_val = load_ov_simplevar()) != E_OK) return(ret_val);


/* ------------------------------------------------------------------------*/
/* ----------------terminate load OV  ------------------------------------ */
/* ------------------------------------------------------------------------*/


/* -- set data block parameters ---------------------------------------- */
term_load_ov_req.vfd_number    = VFD_NUMBER;


/* -- set parameter block parameter ------------------------------------ */
sdb.comm_ref    = 0;
sdb.layer       = FMS;
sdb.service     = TERMINATE_LOAD_OV_LOC;
sdb.primitive   = REQ;

if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 (VOID FAR *) &term_load_ov_req,
                                 (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* --- wait for confirmation ------------------------------------------*/
wait_for_cnf_ind(WAIT);

return(E_OK);
}


FUNCTION static INT16 load_bus_param
        (
          IN USIGN16 role
        )

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to load the bus parameter

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_SET_BUSPARAMETER_REQ bus_par_req;

FUNCTION_BODY

/* --- set data block parameters ---------------------------------------*/
/* the ident will set by the communication software ------------------- */

if (role == CLIENT) bus_par_req.loc_add = CLIENT_LOC_ADD;
else                bus_par_req.loc_add = SERVER_LOC_ADD;

bus_par_req.in_ring_desired = (BOOL)    TRUE;
bus_par_req.baud_rate       = (USIGN8)  BAUDRATE;
bus_par_req.tset            = (USIGN8)  TSET;
bus_par_req.loc_segm        = (USIGN8)  NO_SEGMENT;
bus_par_req.medium_red      = (USIGN8)  MEDIUM_RED;
bus_par_req.tsl             = (USIGN16) TSL;
bus_par_req.min_tsdr        = (USIGN16) MIN_TSDR;
bus_par_req.max_tsdr        = (USIGN16) MAX_TSDR;
bus_par_req.tqui            = (USIGN8)  TQUI;
bus_par_req.ttr             = (USIGN32) TTR;
bus_par_req.g               = (USIGN8)  G;
bus_par_req.hsa             = (USIGN8)  HSA;
bus_par_req.max_retry_limit = (USIGN8)  MAX_RETRY_LIMIT;
bus_par_req.reserved        = (USIGN8)  0;


/* --- set parameter block parameters ----------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_SET_BUSPARAMETER;
sdb.primitive  = REQ;

if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 (VOID FAR *) &bus_par_req,
                                 (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}


/* --- wait for confirmation ------------------------------------------*/
wait_for_cnf_ind(WAIT);

return(E_OK);
}


FUNCTION static INT16 load_kbl
        (
          IN USIGN16 role
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
        )

/*-------------------------------------------------------------------
-FUNCTIONAL_DESCRIPTION

this function is used to load the kbl

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_LOAD_KBL_REQ  kbl_req;

FUNCTION_BODY

/* ------------------------------------------------------------------*/
/* ---------------- initiate load kbl ------------------------------*/
/* ------------------------------------------------------------------*/

/* --- set parameter block parameters ------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_INIT_LOAD_KBL_LOC;
sdb.primitive  = REQ;
printf("initiate load kbl\n");
if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                          NULL,
                          (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}

/* --- wait for confirmation ---------------------------------------*/
wait_for_cnf_ind(WAIT);

/* ------------------------------------------------------------------*/
/* ---------------- load kbl entries -------------------------------*/
/* ------------------------------------------------------------------*/

/* --- kbl header --------------------------------------------------*/
/* --- set data block parameters -----------------------------------*/
kbl_req.desired_cr                      = 0;
kbl_req.id.kbl_hdr.nr_of_entries          = (INT16) NR_OF_KBL_ENTRIES;
kbl_req.id.kbl_hdr.poll_sap               = 0;
kbl_req.id.kbl_hdr.ass_abt_ci             = (USIGN32) ASS_ABT_CI;
kbl_req.id.kbl_hdr.symbol_length          = 9;
kbl_req.id.kbl_hdr.vfd_pointer_supported    = FALSE;


/* --- set parameter block parameters ------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_LOAD_KBL_LOC;
sdb.primitive  = REQ;
printf("load kbl entries\n");
if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                          (VOID FAR*) &kbl_req,
                          (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}

/* --- wait for confirmation ---------------------------------------*/
wait_for_cnf_ind(WAIT);


/* --- KBL entry ---------------------------------------------------*/
/* --- set data block parameters -----------------------------------*/

if (role == CLIENT)
{
   kbl_req.id.kbl_static.rem_add            = (USIGN8) SERVER_LOC_ADD;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
    kbl_req.id.kbl_static.feature_supp[0]    = (USIGN8) 0x80;
    kbl_req.id.kbl_static.feature_supp[1]    = (USIGN8) 0x30;
    kbl_req.id.kbl_static.feature_supp[2]    = (USIGN8) 0x01;
    kbl_req.id.kbl_static.feature_supp[3]    = (USIGN8) 0x80;
    kbl_req.id.kbl_static.feature_supp[4]    = (USIGN8) 0x30;
    kbl_req.id.kbl_static.feature_supp[5]    = (USIGN8) 0x01;
}
else
{
    kbl_req.id.kbl_static.rem_add            = (USIGN8) CLIENT_LOC_ADD;
    kbl_req.id.kbl_static.feature_supp[0]    = (USIGN8) 0x80;
    kbl_req.id.kbl_static.feature_supp[1]    = (USIGN8) 0x30;
    kbl_req.id.kbl_static.feature_supp[2]    = (USIGN8) 0x01;
    kbl_req.id.kbl_static.feature_supp[3]    = (USIGN8) 0x80;
    kbl_req.id.kbl_static.feature_supp[4]    = (USIGN8) 0x30;
    kbl_req.id.kbl_static.feature_supp[5]    = (USIGN8) 0x01;
}

kbl_req.desired_cr                       = (USIGN16) KR;

kbl_req.id.kbl_static.loc_lsap            = (USIGN8)  LOC_SAP;
kbl_req.id.kbl_static.rem_segm             = (USIGN8)  NO_SEGMENT;
kbl_req.id.kbl_static.rem_lsap             = (USIGN8)  REM_SAP;
kbl_req.id.kbl_static.lli_sap             = (USIGN8)  0;
kbl_req.id.kbl_static.conn_type            = (USIGN8)  MMAZ;
kbl_req.id.kbl_static.conn_attr            = (USIGN8)  D_CONN;
kbl_req.id.kbl_static.max_scc              = (USIGN8)  MAX_SCC;
kbl_req.id.kbl_static.max_rcc              = (USIGN8)  MAX_RCC;
kbl_req.id.kbl_static.max_sac              = (USIGN8)  MAX_SAC;
kbl_req.id.kbl_static.max_rac              = (USIGN8)  MAX_RAC;
kbl_req.id.kbl_static.ci                  = (USIGN32) CI;
kbl_req.id.kbl_static.multiplier           = (USIGN8)  MULTIPLIER;
kbl_req.id.kbl_static.max_pdu_snd_high      = 0;
kbl_req.id.kbl_static.max_pdu_snd_low       = (USIGN8)  MAX_FMS_PDU_SIZE;
kbl_req.id.kbl_static.max_pdu_rcv_high      = 0;
kbl_req.id.kbl_static.max_pdu_rcv_low       = (USIGN8)  MAX_FMS_PDU_SIZE;
kbl_req.id.kbl_static.vfd_pointer          = (USIGN32) VFD_NUMBER;
kbl_req.id.kbl_static.extension[0]         = 0;        /* no extension */
kbl_req.id.kbl_static.symbol[0]            = (STRINGV) strlen(KBL_SYMBOL);
strcpy(&kbl_req.id.kbl_static.symbol[1],KBL_SYMBOL);

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_LOAD_KBL_LOC;
sdb.primitive  = REQ;
printf("set data lock parameters\n");
if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 (VOID FAR*) &kbl_req,
                                 (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}

/* --- wait for confirmation --------------------------------------------- */
wait_for_cnf_ind(WAIT);

/* ----------------------------------------------------------------------- */
/* ---------------- terminate load KBL ----------------------------------- */
/* ----------------------------------------------------------------------- */

/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_TERM_LOAD_KBL_LOC;
sdb.primitive  = REQ;
printf("terminate load KBL\n");
if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 NULL,
                                 (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}

/* --- wait for confirmation --------------------------------------------- */
wait_for_cnf_ind(WAIT);
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
return(E_OK);
}

FUNCTION static INT16 set_phys_status(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to set the physical device status in the vfd object

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VFD_SET_PHYS_STATUS_REQ  vfd_phys_status;

FUNCTION_BODY


vfd_phys_status.vfd_number      = (USIGN32) VFD_NUMBER;
vfd_phys_status.physical_status = (USIGN8)  OPERATIONAL;



/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = VFD_SET_PHYS_STATUS_LOC;
sdb.primitive  = REQ;

if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                          (VOID FAR *) &vfd_phys_status,
                          (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}


/* --- wait for confirmation ---------------------------------------------*/
wait_for_cnf_ind(WAIT);

return(E_OK);
}



FUNCTION static INT16 init_profi
       (
         IN USIGN16 role,
         IN BOOL    reset_load_fw
       )

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to initialize the PROFIBUS communication interface

- download PROFIBUS firmware
- create a VFD
- load the OV
- load the busparameter
- load the KBL
- set the physical status of the VFD

possible return values:

- == E_OK   -> no error

-------------------------------------------------------------------*/
{
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
LOCAL_VARIABLES

USIGN16 dummy;

FUNCTION_BODY

/* --- download firmware ------------------------------------------------*/
_setbkcolor(0);
_settextposition(24,24);
_settextcolor(7+16);
_displaycursor(_GCURSOROFF);
_outtext(" INITIALIZE PROFIBUS CONTROLLER ");

ret_val = init_profibus(H_DPR_BASE_ADDRESS,dummy,reset_load_fw);

_settextposition(24,24);
_settextcolor(7);
_outtext("                              ");
_displaycursor(_GCURSORON);

if (ret_val != E_OK) return(ret_val);


/* --- create a VFD -----------------------------------------------------*/
printf("\n create VFD\n");
if ((ret_val = create_vfd()) != E_OK)
{
  printf("\n profi_snd_req_res failed %d\n",ret_val);
  fma7_profi_exit();
  return(ret_val);
}

/* --- load the OV ------------------------------------------------------*/
printf("\n load OV\n");
if ((ret_val = load_ov()) != E_OK)
{
  printf("\n profi_snd_req_res failed %d\n",ret_val);
  fma7_profi_exit();
  return(ret_val);
}

/* --- load the bus parameter -------------------------------------------*/
printf("\n load bus parameter\n");
if ((ret_val = load_bus_param(role)) != E_OK)
{
  printf("\n profi_snd_req_res failed %d\n",ret_val);
  fma7_profi_exit();
  return(ret_val);
}

/* --- load the KBL -----------------------------------------------------*/
printf("\n load KBL\n");
if ((ret_val = load_kbl(role)) != E_OK)
{
  printf("\n profi_snd_req_res failed %d\n",ret_val);
  fma7_profi_exit();
  return(ret_val);
}

/* --- set the physical status in the VFD -------------------------------*/
printf("\n set physical status\n");
if ((ret_val = set_phys_status()) != E_OK)
{
  printf("\n profi_snd_req_res failed %d\n",ret_val);
  fma7_profi_exit();
  return(ret_val);
}
return( E_OK );
}


/* ----------------------------------------------------------------------*/
/* main program                                                          */
/* ----------------------------------------------------------------------*/

FUNCTION extern INT16 main
      (
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
        IN  INT16 argc,
        IN  USIGN8 *argv[]
      )
...
/*---------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

main procedure of the PROFIBUS demo application

possible return values:

- == E_OK   -> no error

---------------------------------------------------------------------*/
{
LOCAL_VARIABLES

USIGN16   role;
BOOL      reset_load_fw;

FUNCTION_BODY

_clearscreen(_GCLEARSCREEN);

/* selection client- or server role ------------------------------------*/
if (!stricmp("CLIENT", argv[1]))
{
   role = CLIENT;
   printf("\nAct as client\n\n");
}
else
{
  if (!stricmp("SERVER", argv[1]))
  {
    role = SERVER;
    printf("\nAct as server\n\n");
  }
  else
  {
     printf("\nUSAGE: %s [Client | Server]\n", argv[0]);
     printf("EXAMPLE: %s Client\n",argv[0]);
     exit(E_OK);
  }
}

if (argc <= 2)
{
  reset_load_fw = TRUE;
}
else
{
  reset_load_fw = (BOOL) atoi(argv[2]);
  if (reset_load_fw != FALSE) reset_load_fw = TRUE;
}

/* initialize the PROFIBUS communication interface ---------------------*/
if ((ret_val = init_profi(role,reset_load_fw)) == E_OK)
{
   printf("\nInitialization successfully\n");
   DIRECCIONES_OBJETOS();
   DATOS();

   /* schedule the application and communication ---------------------*/
   if (role == CLIENT) client();
   else                server();
   exit(0);
}
else
{
   printf("\nInitialization failed %d\n",ret_val);
   exit(0);
}
exit(0);
return(E_OK);
}
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

9.7.2 PROGRAMA DE APLICACION DE LA ESTACION W_90

```
/*****************************************************************************************************************
*                                                                                                                *
*                        PROFIBUS                                                                                *
*                                                                                                                *
*****************************************************************************************************************
*                                                                                                                *
*                        ISOLUX WAT                                                                              *
*                        Alcocer 41                                                                              *
*                        28041 MADRID                                                                            *
*                                                                                                                *
*       Copyright (C) ISOLUX WAT S.A. 1995  All Rights Reserved                                                  *
*                                                                                                                *
*****************************************************************************************************************

FILE_NAME          W_90.C

PROJECT_NAME       APPLICATION for PROFIBUS

AUTHOR             Arantxa Alday

VERSION            1.0

DATE               8.03.1995

================================================================================================================*/
#include <keywords.h>

INCLUDES

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

#include <pb_type.h>
#include <pb_if.h>
#include <pb_fms.h>
#include <pb_fma7.h>
#include <pb_err.h>


GLOBAL_DEFINES

LOCAL_DEFINES
#undef INTEL

/* ---------- internal constants ----------------------------------*/
#define H_DPR_BASE_ADDRESS          0xA0000000
#define CLIENT                      100

#define WAIT                        0xFF
#define DONT_WAIT                   0x00


/*--------- default values to set the busparameter ----------------*/

#define LEN_ST_OV                   0x000F
#define FIRST_INDEX_S_OV            400
#define LEN_S_OV                    0x000F
#define BOOLEAN                     1
#define I_8                         2
#define I_16                        3
#define I_32                        4
#define U_8                         5
#define U_16                        6
#define U_32                        7
#define F_P                         8
#define V_S                         9
#define O_S                         10
#define D_T                         11
#define T_D                         12
#define T_DF                        13
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#define B_S                     14

#define VFD_NUMBER              0x00000001
.. ..
#define CLIENT_LOC_ADD          0x04
#define SERVER_LOC_ADD          0x05
#define BAUDRATE                0x04    /* 500 KBaud      */
#define TSET                    0x05
#define MEDIUM_RED              0x00    /* Keine Redundanz */
#define TSL                     2025
#define MIN_TSDR                50
#define MAX_TSDR                2000
#define TQUI                    0x00
#define TTR                     100000
#define G                       0x01
#define HSA                     0x06
#define MAX_RETRY_LIMIT         0x01

/*---------- default values of a KBL entry ----------*/
#define NIL                     0x3E

#define NR_OF_KBL_ENTRIES       0x0001
#define KR                      0x0005
#define LOC_SAP                 0x05
#define REM_SAP                 0x05
#define CI                      0x00
#define MULTIPLIER              0x00
#define MAX_SCC                 0x03
#define MAX_RCC                 0x03
#define MAX_SAC                 0x00
#define MAX_RAC                 0x00
#define MAX_FMS_PDU_SIZE        0xF1
#define KBL_SYMBOL              "MMAZ"
#define ASS_ABT_CI              4096
#define SYMBOL_LENGTH           0x020


EXPORT_TYPEDEFS

LOCAL_TYPEDEFS

FUNCTION_DECLARATIONS

static VOID     wait_for_cnf_ind(USIGN8);
static INT8     invoke_id;

EXPORT_DATA

IMPORT_DATA


LOCAL_DATA

static T_PROFI_SERVICE_DESCR  sdb;
static USIGN8           cnf_ind_buffer[512];
static INT16            ret_val;
static USIGN16          default_variable_object_data;
extern unsigned         SHARED[1000];
static USIGN16          tabla[614];
static USIGN16          VAR_INDEX;
static USIGN8           arranque;


/* ------------------------------------------------------------------*/
/* FMS functions:                                                    */
/* - to establish a FMS connection                                   */
/* - to deestablish  a FMS connection                                */
/* - to read a simple varibale object                                */
/* - to indicate an abort                                            */
/* - to indicate a reject                                            */
/* ------------------------------------------------------------------*/

FUNCTION static VOID DIRECCIONES_OBJETOS(VOID)
{
LOCAL_VARIABLES

FUNCTION_BODY
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
tabla[400]=231;
tabla[401]=232;
tabla[402]=233;
tabla[403]=234;
tabla[404]=236;
tabla[405]=237;
tabla[406]=238;
tabla[407]=240;
tabla[408]=242;
tabla[409]=246;
tabla[410]=250;
tabla[411]=254;
tabla[412]=256;
tabla[413]=258;

tabla[500]=231;
tabla[501]=232;
tabla[502]=233;
tabla[503]=234;
tabla[504]=236;
tabla[505]=237;
tabla[506]=238;
tabla[507]=240;
tabla[508]=242;
tabla[509]=246;
tabla[510]=250;
tabla[511]=254;
tabla[512]=256;
tabla[513]=258;

return;

}


FUNCTION static INT16 fms_initiate_req(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to establish the default connection

possible return values:

== E_OK   -> no error
!= E_OK   -> interface error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_CTXT_INIT_REQ   init_req;

FUNCTION_BODY

/* --- set data block parameters ------------------------------------*/
init_req.profile_number[0] = 0;
init_req.profile_number[1] = 0;
init_req.password          = 0;
init_req.access_groups     = 0xFF;

/* --- set parameter block parameters -------------------------------*/
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = INITIATE;
sdb.primitive  = REQ;
SHARED[20]=1;
return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) &init_req,TRUE));
}


FUNCTION extern INT16 fms_initiate_ind(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
this function is used to receive the initiate indication to establish
the default connection

possible return values:

== E_OK   -> no error
!= E_OK   -> interface error

------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_CTXT_INIT_REQ FAR* init_ind = (T_CTXT_INIT_REQ FAR*) cnf_ind_buffer;
T_CTXT_INIT_CNF FAR* init_rsp = (T_CTXT_INIT_CNF FAR*) cnf_ind_buffer;

FUNCTION_BODY


/* --- NOTE: the server user has to analyse the indication data structure
         to check profile, password and access groups ---------- */

/* --- send initiate response ------------------------------------*/

/* --- set data block parameters ---------------------------------*/
init_rsp->profile_number[0] = 0;
init_rsp->profile_number[1] = 0;
init_rsp->password          = 0;
init_rsp->access_groups     = 0xFF;


/* --- set parameter block parameters ----------------------------*/
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = INITIATE;
sdb.primitive  = RES;
sdb.invoke_id  = invoke_id = 0;
SHARED[20]=2;
return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) init_rsp,TRUE));
}


FUNCTION static VOID fms_initiate_cnf(VOID)

/*-----------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive the confirmation of the FMS-Initiate service

possible return values:

- NONE

------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_CTXT_INIT_CNF     FAR* init_cnf     = (T_CTXT_INIT_CNF     FAR*) cnf_ind_buffer;
T_CTXT_INIT_ERR_CNF FAR* init_err_cnf = (T_CTXT_INIT_ERR_CNF FAR*) cnf_ind_buffer;

FUNCTION_BODY

if (sdb.result == POS)
{
  /* --- positive confirmation ------------------------------------*/
  SHARED[20]=3;
}
else
{
  /* --- negative confirmation ------------------------------------*/
  SHARED[20]=4;
  SHARED[21]=init_err_cnf->class_code;
}
return;
}
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
FUNCTION static INT16 fms_read_req(VOID)

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to read the data of the default variable object

possible return values:

- == E_OK   -> no error
- != E_OK   -> interface error

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VAR_READ_REQ    read_req;

FUNCTION_BODY


VAR_INDEX=SHARED[6];
SHARED[20]=1;

/*--------Set data block parameters ----------------------------------------*/

read_req.acc_spec.tag      = ACCESS_INDEX;
read_req.acc_spec.id.index = (USIGN16) VAR_INDEX;
read_req.subindex          = 0;

/* --- set parameter block parameters ----------------------------------*/
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = READ;
sdb.primitive  = REQ;
sdb.invoke_id  = invoke_id++;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) &read_req,TRUE));
}


FUNCTION extern INT16 fms_read_ind(VOID)

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive the read indication to read data of the
default variable object

possible return values:

== E_OK   -> no error
!= E_OK   -> interface error

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VAR_READ_REQ FAR *read_ind  = (T_VAR_READ_REQ FAR*) cnf_ind_buffer;
T_VAR_READ_CNF FAR *read_rsp  = (T_VAR_READ_CNF FAR*) cnf_ind_buffer;

USIGN8       FAR *read_data = (USIGN8       FAR*) (read_rsp + 1);
USIGN16          value[4];
FUNCTION_BODY

SHARED[20]=2;

/* --- set data block parameters ----------------------------------------*/

if (read_ind->acc_spec.id.index==413) read_rsp->length=8;
else read_rsp->length=(tabla[read_ind->acc_spec.id.index+1]-tabla[read_ind->acc_spec.id.index])*2;
switch (read_rsp->length)
{
      case 2: value[0]=SHARED[tabla[read_ind->acc_spec.id.index]];
              *(USIGN16 FAR*) read_data=value[0];
              break;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
        case 4: value[0]=SHARED[tabla[read_ind->acc_spec.id.index]];
                value[1]=SHARED[tabla[read_ind->acc_spec.id.index]+1];
                *(USIGN16 FAR*) read_data=value[0];
                *((USIGN16 FAR*) read_data+1)=value[1];
                break;

        case 6: value[0]=SHARED[tabla[read_ind->acc_spec.id.index]];
                value[1]=SHARED[tabla[read_ind->acc_spec.id.index]+1];
                value[2]=SHARED[tabla[read_ind->acc_spec.id.index]+2];
                *(USIGN16 FAR*) read_data=value[0];
                *((USIGN16 FAR*) read_data+1)=value[1];
                *((USIGN16 FAR*) read_data+2)=value[2];
                break;

        case 8: value[0]=SHARED[tabla[read_ind->acc_spec.id.index]];
                value[1]=SHARED[tabla[read_ind->acc_spec.id.index]+1];
                value[2]=SHARED[tabla[read_ind->acc_spec.id.index]+2];
                value[3]=SHARED[tabla[read_ind->acc_spec.id.index]+3];
                *(USIGN16 FAR*) read_data=value[0];
                *((USIGN16 FAR*) read_data+1)=value[1];
                *((USIGN16 FAR*) read_data+2)=value[2];
                *((USIGN16 FAR*) read_data+3)=value[3];
                break;
}

/* --- set parameter block parameters ---------------------------------------*/
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = READ;
sdb.primitive  = RES;

/* --- send read response ---------------------------------------------------*/
return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) read_rsp,TRUE));
}



FUNCTION static VOID fms_read_cnf(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive the confirmation of the FMS-Read service

possible return values:

- NONE

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VAR_READ_CNF  FAR* read_cnf     = (T_VAR_READ_CNF FAR*) cnf_ind_buffer;
T_ERROR         FAR* read_err_cnf = (T_ERROR        FAR*) cnf_ind_buffer;
USIGN16         FAR* read_data    = (USIGN16        FAR*) (read_cnf + 1);

FUNCTION_BODY
if (VAR_INDEX==513) read_cnf->length=8;
else read_cnf->length=(tabla[VAR_INDEX+1]-tabla[VAR_INDEX])*2;

if (sdb.result == POS)
{
  /* --- positive confirmation ----------------------------------------------*/

SHARED[20]=3;
switch (read_cnf->length)
{
      case 2 :if ((VAR_INDEX==500)||(VAR_INDEX==501)||(VAR_INDEX==504))
              SHARED[7]=(*(USIGN16 FAR*) read_data) & (0x00FF);
              else SHARED[7]=*(USIGN16 FAR*) read_data;
              SHARED[8]=0;
              SHARED[9]=0;
              SHARED[10]=0;
              break;

      case 4 :SHARED[7]=*(USIGN16 FAR*) read_data;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
            SHARED[8]=*((USIGN16 FAR*) read_data+1);
            SHARED[9]=0;
            SHARED[10]=0;
            break;

      case 6 :SHARED[7]=*(USIGN16 FAR*) read_data;
            SHARED[8]=*((USIGN16 FAR*) read_data+1);
            SHARED[9]=*((USIGN16 FAR*) read_data+2);
            SHARED[10]=0;
            break;

      case 8 :SHARED[7]=*(USIGN16 FAR*) read_data;
            SHARED[8]=*((USIGN16 FAR*) read_data+1);
            SHARED[9]=*((USIGN16 FAR*) read_data+2);
            SHARED[10]=*((USIGN16 FAR*) read_data+3);
            break;
}

}
else
{
  /* --- negative confirmation ------------------------------------------*/
  SHARED[20]=4;
  SHARED[21]=read_err_cnf->class_code;
}

return;
}

FUNCTION static INT16 fms_write_req(VOID)
/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to write a date of the default variable object

possible return values:

- == E_OK   -> no error
- != E_OK   -> interface error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VAR_WRITE_REQ  FAR*   write_req=(T_VAR_WRITE_REQ FAR*) cnf_ind_buffer;
USIGN8           FAR*   write_data=(USIGN8 FAR*) (write_req + 1);

FUNCTION_BODY

VAR_INDEX=SHARED[6];

if (VAR_INDEX==513) write_req->length=8;
else write_req->length=(tabla[VAR_INDEX+1]-tabla[VAR_INDEX])*2;

/*---set data block paramaters-------------------------------------------*/
write_req->acc_spec.tag      = ACCESS_INDEX;
write_req->acc_spec.id.index= (USIGN16) VAR_INDEX;
write_req->subindex          = 0;

/*---set parameter block parameters--------------------------------------*/
sdb.comm_ref =KR;
sdb.layer    =FMS;
sdb.service  =WRITE;
sdb.primitive=REQ;
sdb.invoke_id=invoke_id++;

*(USIGN16 FAR*)write_data=SHARED[7];
*((USIGN16 FAR*)write_data+1)=SHARED[8];
*((USIGN16 FAR*)write_data+2)=SHARED[9];
*((USIGN16 FAR*)write_data+3)=SHARED[10];
SHARED[20]=1;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) write_req,TRUE));
}

FUNCTION extern INT16 fms_write_ind(VOID)
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*-------------------------------------------------------------------
FUNCTIONAL DESCRIPTION

- this function is used to receive the write indication to write data.
  possible return values:

- ==E_OK    ->no error
- !=E_OK    ->interface error
-------------------------------------------------------------------*/
{
LOCAL_VARIABLES
T_VAR_WRITE_REQ FAR*    write_ind=(T_VAR_WRITE_REQ FAR*) cnf_ind_buffer;
T_VAR_WRITE_REQ FAR*    write_rsp=(T_VAR_WRITE_REQ FAR*) cnf_ind_buffer;
USIGN8          FAR*    write_data=(USIGN8 FAR*) (write_ind +1);
USIGN16         value[4];
USIGN16         direccion;

FUNCTION_BODY
SHARED[20]=2;

/*---set parameter block parameters-----------------------------------*/
sdb.comm_ref   =KR;
sdb.layer      =FMS;
sdb.service    =WRITE;
sdb.primitive  =RES;
if (write_ind->acc_spec.id.index==413) write_ind->length=8;
else write_ind->length=(tabla[write_ind->acc_spec.id.index+1]-tabla[write_ind->acc_spec.id.index])*2;
direccion=tabla[write_ind->acc_spec.id.index];
switch (write_ind->length)
{
      case 2 :SHARED[direccion]=*(USIGN16 FAR*) write_data;
              break;

      case 4 :SHARED[direccion]=*(USIGN16 FAR*) write_data;
              SHARED[direccion+1]=*((USIGN16 FAR*) write_data+1);
              break;

      case 6 :SHARED[direccion]=*(USIGN16 FAR*) write_data;
              SHARED[direccion+1]=*((USIGN16 FAR*) write_data+1);
              SHARED[direccion+2]=*((USIGN16 FAR*) write_data+2);
              break;

      case 8 :SHARED[direccion]=*(USIGN16 FAR*) write_data;
              SHARED[direccion+1]=*((USIGN16 FAR*) write_data+1);
              SHARED[direccion+2]=*((USIGN16 FAR*) write_data+2);
              SHARED[direccion+3]=*((USIGN16 FAR*) write_data+3);
              break;

}


return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb, (VOID FAR*) write_rsp,TRUE));
}

FUNCTION static VOID fms_write_cnf(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL DESCRIPTION

this function is used to receivethe confirmation of the FMS-Read-Service.

possible returns value

- NONE
-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VAR_WRITE_REQ FAR* write_cnf=(T_VAR_WRITE_REQ FAR*) cnf_ind_buffer;
T_ERROR         FAR* write_err_cnf=(T_ERROR FAR*) cnf_ind_buffer;

FUNCTION_BODY
if (sdb.result==POS)
{
  SHARED[20]=3;
}
else
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
{
  SHARED[20]=4;
  SHARED[21]=write_err_cnf->class_code;
}
return;
}

FUNCTION static INT16 fms_getov_req(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to read the data of the default variable object.

possible return:

- == E_OK -> no error
- != E_OK -> interface error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_GET_OV_REQ    getov_req;

FUNCTION_BODY

VAR_INDEX=SHARED[6];

/*--- set data block parameters ----------------------------------------*/
getov_req.format = TRUE;
getov_req.acc_spec.tag = INDEX_ACCESS;
getov_req.acc_spec.id.index= (USIGN16) VAR_INDEX;

/*--- set parameter block parameters -----------------------------------*/
sdb.comm_ref = KR;
sdb.layer    = FMS;
sdb.service  = GETOV;
sdb.primitive= REQ;
sdb.invoke_id= invoke_id++;
SHARED[20]=1;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) &getov_req,TRUE));
}

FUNCTION extern INT16 fms_getov_ind(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

thid function is used to receive the read indication to get object
description from the object dictionary.

possible return values:

- == E_OK  -> no error
- != E_OK  -> interface error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_GET_OV_REQ   FAR    *getov_ind  = (T_GET_OV_REQ  FAR*) cnf_ind_buffer;
T_GET_OV_CNF   FAR    *getov_rsp  = (T_GET_OV_CNF  FAR*) cnf_ind_buffer;

FUNCTION_BODY

SHARED[20]=2;

/*--- set parameter block parameters -----------------------------------*/
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = GETOV;
sdb.primitive  = RES;
sdb.result     = POS;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb, (VOID FAR*) getov_rsp,TRUE));
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
}

FUNCTION static VOID fms_getov_cnf(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive the confirmation of the FMS-GET_OD service

possible return values:

- NONE

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_GET_OV_CNF    FAR*    getov_cnf       = (T_GET_OV_CNF FAR*) cnf_ind_buffer;
T_ERROR         FAR*    getov_err_cnf   = (T_ERROR      FAR*) cnf_ind_buffer;
USIGN8          FAR*    get_ov          = (USIGN8       FAR*) (getov_cnf +1);
USIGN8          i;
USIGN8          j;

FUNCTION_BODY

i=0;
j=1;

if (sdb.result == POS)
{

  SHARED[20]=3;
  if (getov_cnf->no_of_ov_descr == 0) SHARED[70]=0;

  else
  {
  while (j <=getov_cnf->no_of_ov_descr)
   {
      while (i <=get_ov[0])
      {
            SHARED[70+i]=get_ov[i];
            i=i+1;
      }
      *get_ov=*get_ov + get_ov[0] + 1;
      j=j+1;
   }
  }
}
else
{
  SHARED[20]=4;
  SHARED[21]=getov_err_cnf->class_code;
}

return;
}

FUNCTION static INT16 fms_abort_req(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to deestablish the open default connection

possible return values:

- == E_OK   -> no error
- != E_OK   -> interface error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_CTXT_ABORT_REQ abort_req;

FUNCTION_BODY
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* --- set data block parameters ------------------------------------*/
abort_req.abort_id = USR;
abort_req.reason   = USR_ABT_RC1;        /* ---- disconect ----- */
. . ..

/* --- set parameter block parameters -------------------------------*/
sdb.comm_ref   = KR;
sdb.layer      = FMS;
sdb.service    = ABORT;
sdb.primitive  = REQ;
sdb.invoke_id  = invoke_id = 0;

return(profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,(VOID FAR*) &abort_req,TRUE));
}


FUNCTION static VOID fms_abort_ind(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive an FMS abort indication

possible return values:

- NONE

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES


T_CTXT_ABORT_REQ FAR *abort = (T_CTXT_ABORT_REQ FAR*) cnf_ind_buffer;
USIGN8 i;
FUNCTION_BODY

SHARED[20]=2;
SHARED[21]=abort->local;
SHARED[22]=abort->abort_id;
SHARED[23]=abort->reason;
SHARED[24]=abort->detail_length;
return;
}


FUNCTION static VOID fms_reject_ind(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive an FMS reject indication

possible return values:

- NONE

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES


T_CTXT_REJECT_IND FAR *reject = (T_CTXT_REJECT_IND FAR*) cnf_ind_buffer;

FUNCTION_BODY

SHARED[20]=2;
SHARED[21]=reject->detected_here;
SHARED[22]=reject->orig_invoke_id;
SHARED[23]=reject->pdu_type;
SHARED[24]=reject->reject_code;

return;
}


/* ------------------------------------------------------------------*/
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* FMA7 functions:                                                    */
/* - to indicate FMA7 events                                          */
/* - lli fault indications                                            */
/* - to reset and leave the Demo application                          */
/* ------------------------------------------------------------------*/


FUNCTION static VOID fma7_event_ind(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to receive a fma7 event indication
- fma2 event
- lli fault indication
- lli event

possible return values:

- NONE

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_FMA7_EVENT_IND FAR* event = (T_FMA7_EVENT_IND FAR*) cnf_ind_buffer;

FUNCTION_BODY

SHARED[20]=4;
SHARED[21]=event->comm_ref;
SHARED[22]=event->instance_id;
SHARED[23]=event->reason;
return;
}


FUNCTION static INT16 fma7_profi_exit(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function is used to reset and leave the PROFIBUS communication software

Note: For a new start up of the PROFIBUS communication software, initialize
      the PROFIBUS communication interface and PROFIBUS communication software
      using by the init_profibus() function


possible return values:

- == E_OK   -> no error
- != E_OK   -> error


-------------------------------------------------------------------*/
{
LOCAL_VARIABLES


FUNCTION_BODY

/* --- set parameter block parameters ------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_PROFIBUS_EXIT;
sdb.primitive  = REQ;

if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                          NULL,
                          (USIGN8) TRUE)) != E_OK)
{
  return(ret_val);
}
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* --- wait for confirmation and leave the program ---------------------- */
wait_for_cnf_ind(WAIT);
SHARED[51]=12;

return(E_OK);
}


/* ------------------------------------------------------------------------*/
/* scheduling functions:                                                   */
/* - to enter client funtionality                                          */
/* - to enter server funtionality                                          */
/* - to schedule the demo application and PROFIBUS communication interface */
/* ------------------------------------------------------------------------*/

FUNCTION static VOID wait_for_cnf_ind
        (
          IN USIGN8 wait
        )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function is used to receive an indication or confirmation from
the PROFIBUS communication interface


-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES
T_VFD_ERROR FAR* error = (T_VFD_ERROR FAR*) cnf_ind_buffer;
USIGN16 buffer_len;

FUNCTION_BODY


for(;;)
{
  buffer_len = sizeof(cnf_ind_buffer);
  ret_val = profi_rcv_con_ind((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                              (VOID FAR *) cnf_ind_buffer,
                              &buffer_len);

  if (ret_val == CON_IND_RECEIVED)
  {
    if (sdb.layer == FMS_USR)
    {
      switch(sdb.service)
      {
        case INITIATE:
            if (sdb.primitive == IND) fms_initiate_ind();
            else                      fms_initiate_cnf();
            break;

        case READ:
            if (sdb.primitive == IND) fms_read_ind();
            else                      fms_read_cnf();
            break;

        case WRITE:
            if (sdb.primitive == IND) fms_write_ind();
            else                      fms_write_cnf();
            break;

        case GETOV:
            if (sdb.primitive == IND) fms_getov_ind();
            else                      fms_getov_cnf();
            break;

        case ABORT:
            fms_abort_ind();
            break;

        case REJECT:
            fms_reject_ind();
            break;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
        case INITIATE_LOAD_OV_LOC:
        case LOAD_OV_LOC:
        case TERMINATE_LOAD_OV_LOC:
        case CREATE_VFD_LOC:
        case VFD_SET_PHYS_STATUS_LOC:
            if (sdb.result == POS)
            {
              SHARED[20]=3;
            }
            else
            {
              SHARED[20]=4;
              SHARED[21]=sdb.service;
              SHARED[22]=error->error.class_code;
              return;
            }
            if (wait == WAIT) return;
            break;

        default:
            SHARED[20]=5;
            break;
      }
      if (wait == DONT_WAIT) return;
    }
    else
    {
      switch(sdb.service)
      {
        case FMA7_SET_BUSPARAMETER:
        case FMA7_INIT_LOAD_KBL_LOC:
        case FMA7_LOAD_KBL_LOC:
        case FMA7_TERM_LOAD_KBL_LOC:
        case FMA7_PROFIBUS_EXIT:
            if (sdb.result == NEG)
            {
              SHARED[20]=4;
              SHARED[21]=sdb.service;
              SHARED[22]=error->error.class_code;
              return;
            }

            if (wait == WAIT) return;
            break;

        case FMA7_EVENT:
            fma7_event_ind();
            break;

        default:
            SHARED[20]=5;
            break;
      }
      if (wait == DONT_WAIT) return;
    }
  }
  else
  {
    if (ret_val == NO_CON_IND_RECEIVED)
    {
      if (wait == DONT_WAIT) return;
    }
    else
    {
      SHARED[20]=4;
      SHARED[21]=ret_val;
      return;
    }
  }
}
return;
}
```

FUNCTION static VOID client(VOID)

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

CLIENT waits for keyboard inputs of the user

- with the demo application the user can do the following FMS services

  1.) INITIATE:        establish the default connection
  2.) READ VARIABLE:   read the data of the default variable object
  3.) WRITE VARIABLE:  write the data from the default variable object
  4.) GET_OD           get object descriptions
  5.) ABORT:           deestablish the open default connection
  6.) EXIT:            leave the demo application


-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

FUNCTION_BODY
 wait_for_cnf_ind(DONT_WAIT);
 ret_val = E_OK;
 switch (SHARED[5])
  {
    case 1:
        ret_val = fms_initiate_req();
        SHARED[5]=0;
        break;

    case 2:
        ret_val = fms_read_req();
        SHARED[5]=0;
        break;

    case 3:
        ret_val = fms_write_req();
        SHARED[5]=0;
        break;

    case 4:
        ret_val = fms_getov_req();
        SHARED[5]=0;
        break;

    case 5:
        ret_val = fms_abort_req();
        SHARED[5]=0;
        break;

    case 6:
        fma7_profi_exit();
        SHARED[5]=0;
        break;

    default:
        break;
 }

 if (ret_val != E_OK)
  {
     SHARED[20]=4;
     SHARED[21]=ret_val;
  }
return;
}



/* -------------------------------------------------------------------*/
/* intialization                                                      */
/* - initialize the PROFIBUS communication interface                  */
/* - download PROFIBUS firmware                                       */
/* - create a vfd                                                     */
/* - load the ov                                                      */
/* - load the bus parameter                                           */
/* - load the kbl                                                     */
/* - set the physical device status in the vfd                        */
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* ------------------------------------------------------------------*/

/*==================================================================*/

FUNCTION static INT16 create_vfd(VOID)

/*------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to create a VFD

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VFD_CREATE_REQ  create_vfd;

FUNCTION_BODY

/* --- set data block parameters ------------------------------------*/

create_vfd.vfd_number = (USIGN32) VFD_NUMBER;

/* vendor name (byte 0 contains the length of the vendor name) ------- */
create_vfd.vendor_name[0] = 10;
create_vfd.vendor_name[1]=73;
create_vfd.vendor_name[2]=83;
create_vfd.vendor_name[3]=79;
create_vfd.vendor_name[4]=76;
create_vfd.vendor_name[5]=85;
create_vfd.vendor_name[6]=88;
create_vfd.vendor_name[7]=32;
create_vfd.vendor_name[8]=87;
create_vfd.vendor_name[9]=65;
create_vfd.vendor_name[10]=84;

/* model name (byte 0 contains the length of the model name) --------- */
create_vfd.model_name[0]  = 20;
create_vfd.model_name[1]=80;
create_vfd.model_name[2]=82;
create_vfd.model_name[3]=79;
create_vfd.model_name[4]=70;
create_vfd.model_name[5]=73;
create_vfd.model_name[6]=66;
create_vfd.model_name[7]=85;
create_vfd.model_name[8]=83;

/* revision (byte 0 contains the length of the revision) ------------- */
create_vfd.revision[0] = 3;
create_vfd.revision[1]=49;
create_vfd.revision[2]=46;
create_vfd.revision[3]=48;

/* -- profile number (no profile is specified) --------------------- */
create_vfd.profile_number[0] = 0;
create_vfd.profile_number[1] = 0;


/* --- set parameter block parameters ------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = CREATE_VFD_LOC;
sdb.primitive  = REQ;

/* --- send request ------------------------------------------------- */
if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*)
                          &sdb,(VOID FAR*) &create_vfd,
                          TRUE
                         )) != E_OK) return(ret_val);
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
wait_for_cnf_ind(WAIT);
SHARED[51]=2;
return(E_OK);
}


FUNCTION static INT16 load_ov_header(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to load the OV header

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_LOAD_OV_REQ  load_ov_req;

FUNCTION_BODY

/* --- set the data block parameters ------------------------------*/

load_ov_req.vfd_number  = VFD_NUMBER;

load_ov_req.obj_descr.id.ov_obj_descr.index           = 0;
load_ov_req.obj_descr.id.ov_obj_descr.obj_code        = OV_OBJECT;
load_ov_req.obj_descr.id.ov_obj_descr.flag            = TRUE;
load_ov_req.obj_descr.id.ov_obj_descr.length          = 32;
load_ov_req.obj_descr.id.ov_obj_descr.protection      = 0;
load_ov_req.obj_descr.id.ov_obj_descr.version         = 01;
load_ov_req.obj_descr.id.ov_obj_descr.len_st_ov       = LEN_ST_OV;
load_ov_req.obj_descr.id.ov_obj_descr.first_index_s_ov = FIRST_INDEX_S_OV;
load_ov_req.obj_descr.id.ov_obj_descr.len_s_ov        = LEN_S_OV;
load_ov_req.obj_descr.id.ov_obj_descr.first_index_dv_ov= 0;
load_ov_req.obj_descr.id.ov_obj_descr.len_dv_ov       = 0;
load_ov_req.obj_descr.id.ov_obj_descr.first_index_dp_ov= 0;
load_ov_req.obj_descr.id.ov_obj_descr.len_dp_ov       = 0;
load_ov_req.obj_descr.id.ov_obj_descr.int_addr        = 0xFFFFFFFF;
load_ov_req.obj_descr.id.ov_obj_descr.int_addr_st_ov  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.ov_obj_descr.int_addr_s_ov   = 0xFFFFFFFF;
load_ov_req.obj_descr.id.ov_obj_descr.int_addr_dv_ov  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.ov_obj_descr.int_addr_dp_ov  = 0xFFFFFFFF;

/* --- set the parameter block parameter --------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;


if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                        (VOID FAR *) &load_ov_req,
                        (USIGN8) TRUE )) != E_OK )
  return(ret_val);

/* --- wait for confirmation --------------------------------------*/

wait_for_cnf_ind(WAIT);
SHARED[51]=3;
return(E_OK);

}

FUNCTION static INT16 load_ov_datatypes(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to load the OV datatypes
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
possible return values:

- == E_OK    -> no error
- != E_OK    -> error

------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_LOAD_OV_REQ  load_ov_req;

FUNCTION_BODY

/* --- set data block parameters ------------------------------------- */

load_ov_req.vfd_number                  = VFD_NUMBER;

load_ov_req.obj_descr.id.dt_obj_descr.index      = BOOLEAN;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 7;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=66;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=79;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=79;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[4]=76;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[5]=69;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[6]=65;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[7]=78;
/* --- set parameter block parameters -------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE)) ;
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = I_8;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=73;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=56;
/* --- set parameter block parameters -------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

load_ov_req.obj_descr.id.dt_obj_descr.index      = I_16;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 4;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=73;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=49;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[4]=54;
/* --- set parameter block parameters -------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE));

if (ret_val!=E_OK) return(ret_val);
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
wait_for_cnf_ind(WAIT);

load_ov_req.obj_descr.id.dt_obj_descr.index      = I_32;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 4;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=73;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=51;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[4]=50;
/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE)) ;

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

load_ov_req.obj_descr.id.dt_obj_descr.index      = U_8;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=85;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=56;

/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = U_16;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 4;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=85;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=49;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[4]= 54;
/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = U_32;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 4;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=85;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=51;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[4]=50;
/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
                        (USIGN8) TRUE));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);
...

load_ov_req.obj_descr.id.dt_obj_descr.index      = F_P;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=70;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=80;

/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                        (VOID FAR *) &load_ov_req,
                        (USIGN8) TRUE));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = V_S;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=86;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=83;

/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                        (VOID FAR *) &load_ov_req,
                        (USIGN8) TRUE));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = O_S;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=79;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=80;

/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                        (VOID FAR *) &load_ov_req,
                        (USIGN8) TRUE));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = D_T;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=68;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=84;

/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = T_D;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=84;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=68;

/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = T_DF;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 4;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=84;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=68;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[4]=70;
/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;
(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.dt_obj_descr.index      = B_S;
load_ov_req.obj_descr.id.dt_obj_descr.obj_code   = TYPE_OBJECT;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[0] = 3;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[1]=66;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[2]=95;
load_ov_req.obj_descr.id.dt_obj_descr.meaning[3]=83;


/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

/* --- wait for confirmation ------------------------------------------- */
SHARED[51]=4;
return(E_OK);
}

FUNCTION static INT16 load_ov_simplevar(VOID)

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to load the OV simple variable
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
possible return values:

- == E_OK    -> no error
- != E_OK    -> error

-----------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_LOAD_OV_REQ  load_ov_req;

FUNCTION_BODY

default_variable_object_data = 0;

/* --- set data block parameters ------------------------------------------*/
load_ov_req.vfd_number                = VFD_NUMBER;

load_ov_req.obj_descr.id.s_var_obj_descr.index          = 400;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 1;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = BOOLEAN;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 05;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=49;

/* --- set parameter block parameters -------------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index          = 401;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 1;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = I_8;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 05;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=50;

/* --- set parameter block parameters -------------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = LOAD_OV_LOC;
sdb.primitive  = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
load_ov_req.obj_descr.id.s_var_obj_descr.index           = 402;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 2;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = I_16;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address   = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]    = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]         = 05;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=51;


/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index           = 403;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 4;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = I_32;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address   = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]    = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]         = 05;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=52;


/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index           = 404;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 1;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = U_8;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address   = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]    = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]         = 05;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=53;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index          = 405;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 2;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = U_16;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 05;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=54;


/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;
(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index          = 406;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 4;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = U_32;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 05;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=55;


/* --- set parameter block parameters ------------------------------------ */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;
(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index          = 407;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 4;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = F_P;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 05;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=56;


/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index           = 408;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 8;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = V_S;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 05;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=57;


/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index           = 409;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code        = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length          = 8;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type   = O_S;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 06;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=49;
load_ov_req.obj_descr.id.s_var_obj_descr.name[6]=48;

/* --- set parameter block parameters ---------------------------------- */
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));
if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index          = 410;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 7;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = D_T;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 06;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=49;
load_ov_req.obj_descr.id.s_var_obj_descr.name[6]=49;


/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);


load_ov_req.obj_descr.id.s_var_obj_descr.index          = 411;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 4;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = T_D;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address  = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]   = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]        = 06;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=49;
load_ov_req.obj_descr.id.s_var_obj_descr.name[6]=50;


/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE ));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

load_ov_req.obj_descr.id.s_var_obj_descr.index          = 412;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 4;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = T_DF;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 06;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=49;
load_ov_req.obj_descr.id.s_var_obj_descr.name[6]=51;


/* --- set parameter block parameters ----------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

load_ov_req.obj_descr.id.s_var_obj_descr.index          = 413;
load_ov_req.obj_descr.id.s_var_obj_descr.obj_code       = SIMPLE_VAR_OBJECT;
load_ov_req.obj_descr.id.s_var_obj_descr.length         = 8;
load_ov_req.obj_descr.id.s_var_obj_descr.index_of_type  = B_S;
load_ov_req.obj_descr.id.s_var_obj_descr.access.pass_word = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_groups= 0;
load_ov_req.obj_descr.id.s_var_obj_descr.access.acc_right = 255;
load_ov_req.obj_descr.id.s_var_obj_descr.local_address    = 0xFFFFFFFF;
load_ov_req.obj_descr.id.s_var_obj_descr.extension[0]     = 0;
load_ov_req.obj_descr.id.s_var_obj_descr.name[0]          = 06;
load_ov_req.obj_descr.id.s_var_obj_descr.name[1]=73;
load_ov_req.obj_descr.id.s_var_obj_descr.name[2]=83;
load_ov_req.obj_descr.id.s_var_obj_descr.name[3]=79;
load_ov_req.obj_descr.id.s_var_obj_descr.name[4]=45;
load_ov_req.obj_descr.id.s_var_obj_descr.name[5]=49;
load_ov_req.obj_descr.id.s_var_obj_descr.name[6]=52;


/* --- set parameter block parameters ----------------------------------*/
sdb.comm_ref  = 0;
sdb.layer     = FMS;
sdb.service   = LOAD_OV_LOC;
sdb.primitive = REQ;

(ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                             (VOID FAR *) &load_ov_req,
                             (USIGN8) TRUE));

if (ret_val!=E_OK) return(ret_val);
wait_for_cnf_ind(WAIT);

/* --- wait for confirmation -------------------------------------------*/
SHARED[51]=5;
return(E_OK);
}

FUNCTION static INT16 load_ov(VOID)

/*-----------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to load the OV

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

-----------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_INIT_LOAD_OV_REQ    init_load_ov_req;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
T_TERM_LOAD_OV_REQ     term_load_ov_req;

FUNCTION_BODY
...
/* ------------------------------------------------------------------*/
/* ---------------- initiate load OV ---------------------------------*/
/* ------------------------------------------------------------------*/

/* --- set data block parameters ------------------------------------*/
init_load_ov_req.vfd_number   = VFD_NUMBER;
init_load_ov_req.consequence  = CONSEQUENCE_2;

/* --- set parameter block parameters -------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = INITIATE_LOAD_OV_LOC;
sdb.primitive  = REQ;

if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 (VOID FAR *) &init_load_ov_req,
                                 (USIGN8) TRUE)) != E_OK)
{
  return(ret_val);
}


/* --- wait for confirmation ----------------------------------------*/
wait_for_cnf_ind(WAIT);


/* ------------------------------------------------------------------*/
/* ----------------load OV descriptions ----------------------------*/
/* ------------------------------------------------------------------*/

/* ---------------- load OV header ----------------------------------*/
if ((ret_val = load_ov_header()) != E_OK) return(ret_val);

/* -------- load OV standard datatypes ----------------------------*/
if ((ret_val = load_ov_datatypes()) != E_OK) return(ret_val);

/* -------- load OV simple variable -------------------------------*/
if ((ret_val = load_ov_simplevar()) != E_OK) return(ret_val);


/* ------------------------------------------------------------------*/
/* ----------------terminate load OV  ------------------------------*/
/* ------------------------------------------------------------------*/


/* -- set data block parameters -----------------------------------*/
term_load_ov_req.vfd_number   = VFD_NUMBER;


/* -- set parameter block parameter -------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = TERMINATE_LOAD_OV_LOC;
sdb.primitive  = REQ;

if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 (VOID FAR *) &term_load_ov_req,
                                 (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}

/* --- wait for confirmation ----------------------------------------*/
wait_for_cnf_ind(WAIT);
SHARED[51]=6;
return(E_OK);
}


FUNCTION static INT16 load_bus_param(VOID)

/*----------------------------------------------------------------
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
FUNCTIONAL_DESCRIPTION

this function is used to load the bus parameter

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_SET_BUSPARAMETER_REQ bus_par_req;

FUNCTION_BODY

/* --- set data block parameters ------------------------------------*/
/* the ident will set by the communication software ---------------- */

bus_par_req.loc_add = CLIENT_LOC_ADD;

bus_par_req.in_ring_desired = (BOOL)    TRUE;
bus_par_req.baud_rate       = (USIGN8)  BAUDRATE;
bus_par_req.tset            = (USIGN8)  TSET;
bus_par_req.loc_segm        = (USIGN8)  NO_SEGMENT;
bus_par_req.medium_red      = (USIGN8)  MEDIUM_RED;
bus_par_req.tsl             = (USIGN16) TSL;
bus_par_req.min_tsdr        = (USIGN16) MIN_TSDR;
bus_par_req.max_tsdr        = (USIGN16) MAX_TSDR;
bus_par_req.tqui            = (USIGN8)  TQUI;
bus_par_req.ttr             = (USIGN32) TTR;
bus_par_req.g               = (USIGN8)  G;
bus_par_req.hsa             = (USIGN8)  HSA;
bus_par_req.max_retry_limit = (USIGN8)  MAX_RETRY_LIMIT;
bus_par_req.reserved        = (USIGN8)  0;


/* --- set parameter block parameters ------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_SET_BUSPARAMETER;
sdb.primitive  = REQ;

if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                          (VOID FAR *) &bus_par_req,
                          (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}


/* --- wait for confirmation ---------------------------------------*/
wait_for_cnf_ind(WAIT);
SHARED[51]=7;
return(E_OK);
}


FUNCTION static INT16 load_kbl (VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to load the kbl

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_LOAD_KBL_REQ   kbl_req;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
FUNCTION_BODY

/* ------------------------------------------------------------------*/
/* ---------------- initiate load kbl ------------------------------*/
/* ------------------------------------------------------------------*/

/* --- set parameter block parameters ---------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_INIT_LOAD_KBL_LOC;
sdb.primitive  = REQ;
if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                          NULL,
                          (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}

/* --- wait for confirmation ------------------------------------------*/
wait_for_cnf_ind(WAIT);

/* ------------------------------------------------------------------*/
/* ---------------- load kbl entries -------------------------------*/
/* ------------------------------------------------------------------*/

/* --- kbl header -----------------------------------------------------*/
/* --- set data block parameters --------------------------------------*/
kbl_req.desired_cr                      = 0;
kbl_req.id.kbl_hdr.nr_of_entries         = (INT16) NR_OF_KBL_ENTRIES;
kbl_req.id.kbl_hdr.poll_sap              = 0;
kbl_req.id.kbl_hdr.ass_abt_ci            = (USIGN32) ASS_ABT_CI;
kbl_req.id.kbl_hdr.symbol_length          = 9;
kbl_req.id.kbl_hdr.vfd_pointer_supported   = FALSE;


/* --- set parameter block parameters ---------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_LOAD_KBL_LOC;
sdb.primitive  = REQ;
if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                          (VOID FAR*) &kbl_req,
                          (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}

/* --- wait for confirmation ------------------------------------------*/
wait_for_cnf_ind(WAIT);


/* --- KBL entry ------------------------------------------------------*/
/* --- set data block parameters --------------------------------------*/

   kbl_req.id.kbl_static.rem_add          = (USIGN8) SERVER_LOC_ADD;
   kbl_req.id.kbl_static.feature_supp[0]  = (USIGN8) 0x80;
   kbl_req.id.kbl_static.feature_supp[1]  = (USIGN8) 0x30;
   kbl_req.id.kbl_static.feature_supp[2]  = (USIGN8) 0x01;
   kbl_req.id.kbl_static.feature_supp[3]  = (USIGN8) 0x80;
   kbl_req.id.kbl_static.feature_supp[4]  = (USIGN8) 0x30;
   kbl_req.id.kbl_static.feature_supp[5]  = (USIGN8) 0x01;


kbl_req.desired_cr                     = (USIGN16) KR;

kbl_req.id.kbl_static.loc_lsap          = (USIGN8)  LOC_SAP;
kbl_req.id.kbl_static.rem_segm          = (USIGN8)  NO_SEGMENT;
kbl_req.id.kbl_static.rem_lsap          = (USIGN8)  REM_SAP;
kbl_req.id.kbl_static.lli_sap           = (USIGN8)  0;
kbl_req.id.kbl_static.conn_type         = (USIGN8)  MMAZ;
kbl_req.id.kbl_static.conn_attr         = (USIGN8)  D_CONN;
kbl_req.id.kbl_static.max_scc           = (USIGN8)  MAX_SCC;
kbl_req.id.kbl_static.max_rcc           = (USIGN8)  MAX_RCC;
kbl_req.id.kbl_static.max_sac           = (USIGN8)  MAX_SAC;
kbl_req.id.kbl_static.max_rac           = (USIGN8)  MAX_RAC;
kbl_req.id.kbl_static.ci                = (USIGN32) CI;
kbl_req.id.kbl_static.multiplier        = (USIGN8)  MULTIPLIER;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
kbl_req.id.kbl_static.max_pdu_snd_high      = (USIGN8) 0;
kbl_req.id.kbl_static.max_pdu_snd_low       = (USIGN8) MAX_FMS_PDU_SIZE;
kbl_req.id.kbl_static.max_pdu_rcv_high      = (USIGN8) 0;
kbl_req.id.kbl_static.max_pdu_rcv_low       = (USIGN8) MAX_FMS_PDU_SIZE;
kbl_req.id.kbl_static.vfd_pointer           = (USIGN32) VFD_NUMBER;
kbl_req.id.kbl_static.extension[0]          = 0;       /* no extension */
kbl_req.id.kbl_static.symbol[0]             = 4;
kbl_req.id.kbl_static.symbol[1]=77;
kbl_req.id.kbl_static.symbol[2]=77;
kbl_req.id.kbl_static.symbol[3]=65;
kbl_req.id.kbl_static.symbol[4]=90;


/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_LOAD_KBL_LOC;
sdb.primitive  = REQ;
if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 (VOID FAR*) &kbl_req,
                                 (USIGN8) TRUE)) != E_OK )
{
  return(ret_val);
}

/* --- wait for confirmation ---------------------------------------------*/
wait_for_cnf_ind(WAIT);

/* -----------------------------------------------------------------------*/
/* ---------------- terminate load KBL -----------------------------------*/
/* -----------------------------------------------------------------------*/

/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMA7;
sdb.service    = FMA7_TERM_LOAD_KBL_LOC;
sdb.primitive  = REQ;
if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 NULL,
                                 (USIGN8) TRUE)) != E_OK )
{
  return(ret_val);
}

/* --- wait for confirmation ---------------------------------------------*/
wait_for_cnf_ind(WAIT);

SHARED[51]=8;
return(E_OK);
}


FUNCTION static INT16 set_phys_status(VOID)

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to set the physical device status in the vfd object

possible return values:

- == E_OK   -> no error
- != E_OK   -> error

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

T_VFD_SET_PHYS_STATUS_REQ  vfd_phys_status;

FUNCTION_BODY


vfd_phys_status.vfd_number      = (USIGN32) VFD_NUMBER;
vfd_phys_status.physical_status = (USIGN8)  OPERATIONAL;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* --- set parameter block parameters ------------------------------------*/
sdb.comm_ref   = 0;
sdb.layer      = FMS;
sdb.service    = VFD_SET_PHYS_STATUS_LOC;
sdb.primitive  = REQ;

if ((ret_val = profi_snd_req_res((T_PROFI_SERVICE_DESCR FAR*) &sdb,
                                 (VOID FAR *) &vfd_phys_status,
                                 (USIGN8) TRUE )) != E_OK )
{
  return(ret_val);
}


/* --- wait for confirmation ---------------------------------------------*/
wait_for_cnf_ind(WAIT);
SHARED[51]=9;
return(E_OK);
}



FUNCTION static INT16 init_profi(VOID)

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to initialize the PROFIBUS communication interface

- download PROFIBUS firmware
- create a VFD
- load the OV
- load the busparameter
- load the KBL
- set the physical status of the VFD

possible return values:

- == E_OK    -> no error

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

USIGN16 dummy;

FUNCTION_BODY

/* --- download firmware -------------------------------------------------*/

ret_val = init_profibus(H_DPR_BASE_ADDRESS,dummy,TRUE);

if (ret_val != E_OK) return(ret_val);
SHARED[51]=1;

/* --- create a VFD ------------------------------------------------------*/
if ((ret_val = create_vfd()) != E_OK)
{
  fma7_profi_exit();
  return(ret_val);
}

/* --- load the OV -------------------------------------------------------*/
if ((ret_val = load_ov()) != E_OK)
{
  fma7_profi_exit();
  return(ret_val);
}

/* --- load the bus parameter --------------------------------------------*/
if ((ret_val = load_bus_param()) != E_OK)
{
  fma7_profi_exit();
  return(ret_val);
}
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* --- load the KBL ------------------------------------------------*/
if ((ret_val = load_kbl()) != E_OK)
{
  fma7_profi_exit();
  return(ret_val);
}

/* --- set the physical status in the VFD --------------------------*/
if ((ret_val = set_phys_status()) != E_OK)
{
  fma7_profi_exit();
  return(ret_val);
}
return( E_OK );
}


/* ------------------------------------------------------------------*/
/* main program                                                      */
/* ------------------------------------------------------------------*/

FUNCTION extern VOID prof_main(VOID)

/*-------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

main procedure of the PROFIBUS demo application

possible return values:

- == E_OK   -> no error

-------------------------------------------------------------------*/
{
LOCAL_VARIABLES

FUNCTION_BODY

if (arranque==0)
{
/* initialize the PROFIBUS communication interface ---------------- */
if ((ret_val = init_profi()) == E_OK)
{
  arranque=1;
  DIRECCIONES_OBJETOS();

  /* schedule the application and communication ------------------- */
  client();

}
else
{
  return;
}

}
else
  client();
}
```

9.7.3 LIBRERIA CMI_H.C

```
/*****************************************************************************************************************
*                                                                                                                *
*                          profibus                                                                              *
*                                                                                                                *
*****************************************************************************************************************
*                                                                                                                *
*                          ISOLUX-WAT                                                                            *
*                                                                                                                *
*                                                                                                                *
*                                                                                                                *
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
*                                                                    *
*                                                                    *
**********************************************************************

FILE_NAME               CMI_H.C

PROJECT_NAME                PROFIBUS  CP5480-A1

MODULE                  CMI_H

COMPONENT_LIBRARY               PIFL.LIB or PIFM.LIB or PIFS.LIB

AUTHOR                  ARANTXA ALDAY

VERSION                 1.00

DATE                    15.03.1995

STATUS                  finished

FUNCTIONAL_MODULE_DESCRIPTION

COMMON MEMORY INTERFACE driver for PROFIBUS host
More information you will find in the description
"COMMON MEMORY INTERFACE for PROFIBUS CONTROLLER"


FUNCTIONAL_SPECIFICATION

DESIGN_SPECIFICATION

RELATED_DOCUMENTS

"COMMON MEMORY INTERFACE for PROFIBUS CONTROLLER"

=============================================================================*/
#include <keywords.h>

INCLUDES

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

#include <pb_type.h>
#include <pb_err.h>
#include <cmi.h>

#include <pb_if.h>
#include <i86.h>

GLOBAL_DEFINES

LOCAL_DEFINES

/* --- macros for interrupt handling ------------------------------------ */
EXPORT_TYPEDEFS

LOCAL_TYPEDEFS

FUNCTION_DECLARATIONS

#define _INT_TO_CNTRL      jose_antonio=buildptr((selector)h_int_ptr,(void near*)(((long)h_int_ptr)*2)); *jose_antonio=0xFF;
#define _RESET_CNTRL_INT   jose_antonio=buildptr((selector)c_int_ptr,(void near*)(((long)c_int_ptr)*2)); *jose_antonio=0x00;

EXPORT_DATA

IMPORT_DATA

LOCAL_DATA

static T_CMI_DESCRIPTOR      FAR *cmi_descriptor;

static USIGN32               h_base_address;
static USIGN32               c_base_address;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
static T_PROFI_SERVICE_DESCR FAR *h_serv_descr;
static T_PROFI_SERVICE_DESCR FAR *c_serv_descr;

static VOID              FAR *h_data_ptr;
static VOID              FAR *c_data_ptr;

static USIGN8            FAR *h_int_ptr;
static USIGN8            FAR *c_int_ptr;
static USIGN8            FAR *jose_antonio;
static USIGN8            FAR *arantxa;
static USIGN16           data_block_size;
static USIGN16           param_block_size;

long              puntoff;
selector          puntsel;


FUNCTION static USIGN32 swap_32_intel_motorola
          (
           IN USIGN32 value              /* value to swap */
          )

/*---------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function swaps the ready mask from INTEL-FORMAT to MOTOROLA-FORMAT

---------------------------------------------------------------------*/
{
LOCAL_VARIABLES

          USIGN8 i,j;                        /* loop variables      */
volatile USIGN8 tmp_value1[4];               /* temporaer help buffer */
volatile USIGN8 tmp_value2[4];               /* temporaer help buffer */
          USIGN32 *inout_value = (USIGN32*) tmp_value1;  /* temporaer help buffer */

FUNCTION_BODY

*inout_value = value;

for (i=0;     i<4; i++    ) tmp_value2[i] = tmp_value1[i];
for (i=0, j=3; i<4; i++,j--) tmp_value1[i] = tmp_value2[j];

return(*inout_value);

}


FUNCTION static INT16 cmi_h_wait_for_controller(VOID)

/*---------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function reads the the semaphore flag in the host common blocks. The
function returns with E_OK if the semaphore flag is set to _IDLE. If the
semarphore flag is not set after CMI_TIMEOUT seconds the function returns with
E_NO_CNTRL_RES.

---------------------------------------------------------------------*/
{
LOCAL_VARIABLES

INT32  act_time;          /* actual time from AT */
INT16  ret_val = E_OK;
USIGN8 FAR*   cmi_arantxa;

FUNCTION_BODY

puntsel = (selector) &cmi_descriptor->c_int_enable;
puntoff = (long) &cmi_descriptor->c_int_enable;
puntoff = puntoff * 2;
cmi_arantxa= buildptr(puntsel,(void near *)puntoff);
/* --- activate controller interrupt ------------------------------------*/
if (*cmi_arantxa & 0x01){ _INT_TO_CNTRL}

/* --- get the start time -----------------------------------------------*/
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
puntsel = (selector) &cmi_descriptor->h_sema;
puntoff = (long) &cmi_descriptor->h_sema;
puntoff = puntoff * 2;
cmi_arantxa= buildptr(puntsel,(void near *)puntoff);
act_time=0;
while (*cmi_arantxa != _IDLE)
{
        puntsel = (selector) &cmi_descriptor->c_ret_val;
        puntoff = (long) &cmi_descriptor->c_ret_val;
        puntoff = puntoff * 2;
        arantxa= buildptr(puntsel,(void near *)puntoff);
        if (*arantxa!= E_OK)    return(ret_val);
        if (act_time > (CMI_TIMEOUT*5000) ) return(E_NO_CNTRL_RES);
        act_time=act_time+1;
}

return(E_OK);
}


FUNCTION static USIGN32  cmi_h_correct_address
        (
          IN USIGN32  address          /* address to correct */
        )
/*---------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function computes an address of the controller.

NOTE: the address mode of the controller is SEGMENT_OFFSET_INTEL

---------------------------------------------------------------------*/
{
LOCAL_VARIABLES

USIGN8  FAR* h_arantxa;
USIGN8  FAR* c_arantxa;
USIGN8      ad[4];
USIGN8  FAR* general;
USIGN32 FAR* dirx;
FUNCTION_BODY

if (address==0) return(0);
puntsel = (selector) &cmi_descriptor->h_address_swap_mode;
puntoff = (long) &cmi_descriptor->h_address_swap_mode;
puntoff = puntoff * 2;
h_arantxa= buildptr(puntsel,(void near *)puntoff);
switch (*h_arantxa & 0xBF)
{
  case SEGMENT_OFFSET_INTEL:
        puntsel = (selector) &cmi_descriptor->c_address_swap_mode;
        puntoff = (long) &cmi_descriptor->c_address_swap_mode;
        puntoff = puntoff * 2;
        c_arantxa= buildptr(puntsel,(void near *)puntoff);
        switch ( *c_arantxa & 0x7F)
        {
            case SEGMENT_OFFSET_INTEL:
                puntsel = (selector) &cmi_descriptor->c_base_address;
                puntoff = (long) &cmi_descriptor->c_base_address;
                puntoff = puntoff * 2;
                general= buildptr(puntsel,(void near *)puntoff);
                ad[0]=*general;
                ad[1]=*(general+2);
                ad[2]=*(general+4);
                ad[3]=*(general+6);
                dirx=ad;
                address=address - *dirx;

                puntsel = (selector) &cmi_descriptor->h_base_address;
                puntoff = (long) &cmi_descriptor->h_base_address;
                puntoff = puntoff * 2;
                general= buildptr(puntsel,(void near *)puntoff);
                ad[0]=*general;
                ad[1]=*(general+2);
                ad[2]=*(general+4);
                ad[3]=*(general+6);
                dirx=ad;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
                        address=address + *dirx;
                        break;

        default:
                        break;
        }
        break;

  default:
        break;
}

return(address);
}


FUNCTION extern INT16 cmi_h_check_controller_exists
        (
          IN  USIGN32 h_dpr_base_address    /* host base address of DPR  */
        )

/*---------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to check the existence of the controller

possible return values:

- E_OK                  cmi is initialized
- E_NO_CNTRL_RES        controller does not respond

---------------------------------------------------------------------*/
{
LOCAL_VARIABLES

volatile USIGN8  FAR  *controller_mask;

FUNCTION_BODY

if (h_dpr_base_address > 0)
{
  controller_mask = (USIGN8 FAR*) h_dpr_base_address;
  *controller_mask=0x0F;
  *(controller_mask+2)=0x0E;
  *(controller_mask+4)=0xE0;
  *(controller_mask+6)=0xF0;
  if ((*controller_mask==0x0F)&((*(controller_mask+2))==0x0E)
    &((*(controller_mask+4))==0xE0)&((*(controller_mask+6))==0xF0))

    return(E_OK);
}
return(E_NO_CNTRL_RES);
}


FUNCTION extern INT16 cmi_h_init
        (
          IN  USIGN32 h_dpr_base_address,   /* host base address of DPR  */
          IN  USIGN16 controller_type       /* controller type           */
        )

/*---------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function initializes the the host cmi driver

possible return values:

- E_OK                  cmi is initialized
- E_NO_CNTRL_RES        controller does not respond
- E_INVALID_CNTRL_TYPE_VERSION  invalid controller type or software version
---------------------------------------------------------------------*/
{
LOCAL_VARIABLES

USIGN32         h_ready_mask;           /* host ready mask      */
USIGN32         c_ready_mask;           /* controller ready mask */
INT32           act_time;               /* actual time from AT   */
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
USIGN32             palabra;
USIGN8    FAR*      aran[2];
USIGN8    FAR*      apuntador;
USIGN8    FAR*      puntero;
USIGN8    FAR*      arantxa_h;
USIGN8    FAR*      arantxa_c;
USIGN32   FAR*      direc;
USIGN8              elementos[4];
FUNCTION_BODY

/* -- set host/controller ready mask in MOTOROLA Format ---------- */
h_ready_mask = swap_32_intel_motorola((USIGN32) H_READY_MASK);
c_ready_mask = swap_32_intel_motorola((USIGN32) C_READY_MASK);

/* --- compute the interrupt cell address ------------------------- */
h_int_ptr = (USIGN8 FAR*) (h_dpr_base_address + H_INT_OFFSET);
c_int_ptr = (USIGN8 FAR*) (h_dpr_base_address + C_INT_OFFSET);

/* --- reset controller interrrupts ------------------------------- */
_RESET_CNTRL_INT

/* --- set CMI host base address ---------------------------------- */
cmi_descriptor = (T_CMI_DESCRIPTOR FAR *) h_dpr_base_address;

/* --- intialize CMI host and controller ready mask --------------- */
puntsel = (selector) &cmi_descriptor->h_ready_mask;
puntoff = (long) &cmi_descriptor->h_ready_mask;
puntoff = puntoff * 2;
arantxa = buildptr(puntsel,(void near *)puntoff);
*arantxa=0;
*(arantxa+2)=0;
*(arantxa+4)=0;
*(arantxa+6)=0;
puntsel = (selector) &cmi_descriptor->c_ready_mask;
puntoff = (long) &cmi_descriptor->c_ready_mask;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
*arantxa=0;
*(arantxa+2)=0;
*(arantxa+4)=0;
*(arantxa+6)=0;
/* --- intialize CMI host and controller return value ------------- */
puntsel = (selector) &cmi_descriptor->h_ret_val;
puntoff = (long) &cmi_descriptor->h_ret_val;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
*arantxa=0;
puntsel = (selector) &cmi_descriptor->c_ret_val;
puntoff = (long) &cmi_descriptor->c_ret_val;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
*arantxa=0;

/* --- set the cmi host descriptor block parameters --------------- */
puntsel = (selector) &cmi_descriptor->h_ready_mask;
puntoff = (long) &cmi_descriptor->h_ready_mask;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
apuntador=&h_ready_mask;
*arantxa = *apuntador;
*(arantxa+2) = *(apuntador+1);
*(arantxa+4) = *(apuntador+2);
*(arantxa+6) = *(apuntador+3);

puntsel = (selector) &cmi_descriptor->h_base_address;
puntoff = (long) &cmi_descriptor->h_base_address;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
apuntador=&h_dpr_base_address;
*arantxa = *apuntador;
*(arantxa+2) = *(apuntador+1);
*(arantxa+4) = *(apuntador+2);
*(arantxa+6) = *(apuntador+3);

puntsel = (selector) &cmi_descriptor->h_id;
puntoff = (long) &cmi_descriptor->h_id;
puntoff = puntoff * 2;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
arantxa= buildptr(puntsel,(void near *)puntoff);
*arantxa = HOST_TYPE;
puntsel = (selector) &cmi_descriptor->h_int_enable;
puntoff = (long) &cmi_descriptor->h_int_enable;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
*arantxa= POLL_INT_MODE;
puntsel = (selector) &cmi_descriptor->h_address_swap_mode;
puntoff = (long) &cmi_descriptor->h_address_swap_mode;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
*arantxa= ADDRESS_SWAP_MODE;
puntsel = (selector) &cmi_descriptor->h_state;
puntoff = (long) &cmi_descriptor->h_state;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
*arantxa= COMM_MODE;
puntsel = (selector) &cmi_descriptor->h_sema;
puntoff = (long) &cmi_descriptor->h_sema;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
*arantxa= _IDLE;

/* --- get the start time for check timeout ---------- */
act_time=0;
/* --- wait until controller ready mask is set ---------- */
puntsel = (selector) &cmi_descriptor->c_ready_mask;
puntoff = (long) &cmi_descriptor->c_ready_mask;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
apuntador=&c_ready_mask;
while (( *arantxa!= *(apuntador))||( *(arantxa+2)!= *(apuntador+1))||
      ( *(arantxa+4)!= *(apuntador+2))||( *(arantxa+6)!= *(apuntador+3)))
{  /* --- set host ready mask again because controller initializes the ready masks */
   puntsel = (selector) &cmi_descriptor->h_ready_mask;
   puntoff = (long) &cmi_descriptor->h_ready_mask;
   puntoff = puntoff * 2;
   arantxa_h= buildptr(puntsel,(void near *)puntoff);
   puntero=&h_ready_mask;
   *arantxa_h = *puntero;
   *(arantxa_h+2) = *(puntero+1);
   *(arantxa_h+4) = *(puntero+2);
   *(arantxa_h+6) = *(puntero+3);
   /* --- check time out -> controller does not respond ---------- */
   puntsel = (selector) &cmi_descriptor->c_ret_val;
   puntoff = (long) &cmi_descriptor->c_ret_val;
   puntoff = puntoff * 2;
   arantxa_c= buildptr(puntsel,(void near *)puntoff);
   if (*arantxa_c != E_OK)       return(E_NO_CNTRL_RES);
   if (act_time > ( CMI_TIMEOUT*5000 )) return(E_NO_CNTRL_RES);
   act_time=act_time+1;
}

/* --- set host ready mask again ---------- */
puntsel = (selector) &cmi_descriptor->h_ready_mask;
puntoff = (long) &cmi_descriptor->h_ready_mask;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
puntero=&h_ready_mask;
*arantxa = *puntero;
*(arantxa+2) = *(puntero+1);
*(arantxa+4) = *(puntero+2);
*(arantxa+6) = *(puntero+3);
/* --- check the controller type ---------- */
puntsel = (selector) &cmi_descriptor->c_id;
puntoff = (long) &cmi_descriptor->c_id;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
if (*arantxa != (USIGN8) controller_type)
{
  return(E_INVALID_CNTRL_TYPE_VERSION);
}

/* --- determine host param_addr and data_addr ---------- */
puntsel = (selector) &cmi_descriptor->h_param_addr;
puntoff = (long) &cmi_descriptor->h_param_addr;
puntoff = puntoff * 2;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
arantxa= buildptr(puntsel,(void near *)puntoff);
elementos[0]=*arantxa;
elementos[1]=*(arantxa+2);
elementos[2]=*(arantxa+4);
elementos[3]=*(arantxa+6);
direc = elementos;
palabra = cmi_h_correct_address ((USIGN32) *direc);
apuntador=&palabra;
*arantxa=*apuntador;
*(arantxa+2)=*(apuntador+1);
*(arantxa+4)=*(apuntador+2);
*(arantxa+6)=*(apuntador+3);

puntsel = (selector) &cmi_descriptor->h_data_addr;
puntoff = (long) &cmi_descriptor->h_data_addr;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
elementos[0]=*arantxa;
elementos[1]=*(arantxa+2);
elementos[2]=*(arantxa+4);
elementos[3]=*(arantxa+6);
direc = elementos;
palabra = cmi_h_correct_address ((USIGN32) *direc);
apuntador=&palabra;
*arantxa=*apuntador;
*(arantxa+2)=*(apuntador+1);
*(arantxa+4)=*(apuntador+2);
*(arantxa+6)=*(apuntador+3);



puntsel = (selector) &cmi_descriptor->h_param_addr;
puntoff = (long) &cmi_descriptor->h_param_addr;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
elementos[0]=*arantxa;
elementos[1]=*(arantxa+2);
elementos[2]=*(arantxa+4);
elementos[3]=*(arantxa+6);
direc = elementos;
h_serv_descr = (T_PROFI_SERVICE_DESCR FAR *) *direc;


puntsel = (selector) &cmi_descriptor->h_data_addr;
puntoff = (long) &cmi_descriptor->h_data_addr;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
elementos[0]=*arantxa;
elementos[1]=*(arantxa+2);
elementos[2]=*(arantxa+4);
elementos[3]=*(arantxa+6);
direc = elementos;
h_data_ptr = (VOID FAR *) *direc;




puntsel = (selector) &cmi_descriptor->h_param_size;
puntoff = (long) &cmi_descriptor->h_param_size;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
apuntador=&param_block_size;
*apuntador = *arantxa;
*(apuntador+1) = *(arantxa+2);

puntsel = (selector) &cmi_descriptor->h_data_size;
puntoff = (long) &cmi_descriptor->h_data_size;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
apuntador=&data_block_size;
*apuntador = *arantxa;
*(apuntador+1) = *(arantxa+2);

/* --- determine controller param_addr and data_addr ---------- */
puntsel = (selector) &cmi_descriptor->c_param_addr;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
puntoff = (long) &cmi_descriptor->c_param_addr;
puntoff = puntoff * 2;
apuntador= buildptr(puntsel,(void near *)puntoff);
elementos[0]=*apuntador;
elementos[1]=*(apuntador+2);
elementos[2]=*(apuntador+4);
elementos[3]=*(apuntador+6);
direc = elementos;

c_serv_descr   = (T_PROFI_SERVICE_DESCR FAR *)
                        cmi_h_correct_address ((USIGN32) *direc);


puntsel = (selector) &cmi_descriptor->c_data_addr;
puntoff = (long) &cmi_descriptor->c_data_addr;
puntoff = puntoff * 2;
apuntador= buildptr(puntsel,(void near *)puntoff);
elementos[0]=*apuntador;
elementos[1]=*(apuntador+2);
elementos[2]=*(apuntador+4);
elementos[3]=*(apuntador+6);
direc=elementos;
c_data_ptr     = (VOID FAR *)
                        cmi_h_correct_address ((USIGN32) *direc);

puntsel = (selector) &cmi_descriptor->h_ret_val;
puntoff = (long) &cmi_descriptor->h_ret_val;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
*arantxa = E_OK;

return(E_OK);

}
FUNCTION static VOID copia
                        (
                          INOUT USIGN8 FAR* destino,
                          IN USIGN8 FAR* origen,
                          IN USIGN16 i,IN INT8 z
                        )
{
LOCAL_VARIABLES
USIGN8 FAR*   puntdes;
USIGN8 FAR*   puntori;
INT16         cont;
INT16         a;
INT16         b;
FUNCTION_BODY

cont=1;
if (z==1)
{
            a=1;
            b=2;

puntsel = (selector) origen;
puntoff = (long) origen;
puntoff = puntoff * 2;
puntori= buildptr(puntsel,(void near *)puntoff);

puntdes=destino;

}
else
{
            a=2;
            b=1;

puntsel = (selector) destino;
puntoff = (long) destino;
puntoff = puntoff * 2;
puntdes= buildptr(puntsel,(void near *)puntoff);

puntori = origen;

}
while (cont<=i)
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
{
        *puntdes = *puntori;
        puntdes = puntdes + a;
        puntori = puntori + b;
        cont = cont + 1;
}
return;
}


FUNCTION extern INT16 cmi_h_read
        (
         IN    T_PROFI_SERVICE_DESCR FAR   *psd,
         IN    VOID                  FAR   *data_ptr,
         INOUT USIGN16               FAR   *data_len
        )

/*---------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function reads the SERVICE_DESCRIPTION_BLOCK and the SERVICE_DATA_BLOCK
from the common memory if the controller semaphore is set.

possible return values:
- CON_IND_RECEIVED     -> a confirmation or indication is received
- NO_CON_IND_RECEIVED  -> no confirmation or indication is received

- E_INVALID_DATA_SIZE  -> not enough memory to transfer the service specific
                                        datas from CMI data block to user data block

---------------------------------------------------------------------*/
{
LOCAL_VARIABLES

INT16 ret_val;          /* return value */
USIGN16 aran;
USIGN8  FAR* apuntador;
FUNCTION_BODY

ret_val = E_OK;

puntsel = (selector) &cmi_descriptor->c_sema;
puntoff = (long) &cmi_descriptor->c_sema;
puntoff = puntoff * 2;
arantxa = buildptr(puntsel,(void near *)puntoff);
if (*arantxa == _IDLE) return(NO_CON_IND_RECEIVED);

puntsel = (selector) &cmi_descriptor->c_ret_val;
puntoff = (long) &cmi_descriptor->c_ret_val;
puntoff = puntoff * 2;
arantxa = buildptr(puntsel,(void near *)puntoff);
if ( *arantxa != E_OK)
{
  ret_val               = (INT16) *arantxa;
  puntsel = (selector) &cmi_descriptor->c_sema;
  puntoff = (long) &cmi_descriptor->c_sema;
  puntoff = puntoff * 2;
  arantxa = buildptr(puntsel,(void near *)puntoff);
  *arantxa = _IDLE;

  /* --- activate controller interrupt ---------------------------------*/
  puntsel = (selector) &cmi_descriptor->c_int_enable;
  puntoff = (long) &cmi_descriptor->c_int_enable;
  puntoff = puntoff * 2;
  arantxa = buildptr(puntsel,(void near *)puntoff);
  if ( *arantxa & 0x02) {_INT_TO_CNTRL}

  return(ret_val);
}


/* --- copy controller service description block ---------------------- */

copia(psd,c_serv_descr,(sizeof(T_PROFI_SERVICE_DESCR)),1);

/* --- copy controller service data block ----------------------------- */
puntsel = (selector) &cmi_descriptor->c_data_size;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
puntoff = (long) &cmi_descriptor->c_data_size;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
apuntador=&aran;
*apuntador=*arantxa;
*(apuntador+1)=*(arantxa+2);
if ( aran <= *data_len)
{
  copia(data_ptr,c_data_ptr,aran,1);
  *data_len = aran;

  /* --- set controller semaphore return to _IDLE ---------- */
  puntsel = (selector) &cmi_descriptor->c_sema;
  puntoff = (long) &cmi_descriptor->c_sema;
  puntoff = puntoff * 2;
  arantxa= buildptr(puntsel,(void near *)puntoff);
  *arantxa= _IDLE;

  /* --- activate controller interrupt ---------- */
  puntsel = (selector) &cmi_descriptor->c_int_enable;
  puntoff = (long) &cmi_descriptor->c_int_enable;
  puntoff = puntoff * 2;
  arantxa= buildptr(puntsel,(void near *)puntoff);
  if ( *arantxa & 0x02) {_INT_TO_CNTRL}

  return(CON_IND_RECEIVED);
}
else
{
 puntsel = (selector) &cmi_descriptor->c_data_size;
 puntoff = (long) &cmi_descriptor->c_data_size;
 puntoff = puntoff * 2;
 arantxa= buildptr(puntsel,(void near *)puntoff);
 apuntador=data_len;
 *apuntador=*arantxa;
 *(apuntador+1) =*(arantxa+2);
 return(E_INVALID_DATA_SIZE);
}
}



FUNCTION extern INT16 cmi_h_write
            (
             IN T_PROFI_SERVICE_DESCR FAR  *psd,
             IN VOID                  FAR  *data_ptr,
             IN USIGN16                    data_len
            )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function writes the SERVICE_DESRIPTION_BLOCK and the SERVICE_DATA_BLOCK
to the common memory.

possible return values:
- E_OK              -> no error occured
- E_INVALID_CMI_CALL  -> invalid CMI call
- E_INVALID_DATA_SIZE -> not enough cmi data block memory
- E_NO_CNTRL_RES      -> controller does not respond
-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

INT16 ret_val;
USIGN16     aran;
USIGN8 FAR* apuntador;

FUNCTION_BODY

/* --- copy service description parameter block from host to CMI ---------- */
if (psd != NULL)
{
 copia (h_serv_descr,psd,(sizeof(T_PROFI_SERVICE_DESCR)),2);

 puntsel = (selector) &cmi_descriptor->h_param_size;
 puntoff = (long) &cmi_descriptor->h_param_size;
 puntoff = puntoff * 2;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
  arantxa= buildptr(puntsel,(void near *)puntoff);
  aran = (USIGN16) sizeof(T_PROFI_SERVICE_DESCR);
  apuntador=&aran;
  *arantxa=*apuntador;
  *(arantxa+2)=*(apuntador+1);
 }
 else
 {
  return(E_INVALID_CMI_CALL);
 }


/* --- copy data block from host to CMI ---------------------------------- */
if (data_len > data_block_size) return(E_INVALID_DATA_SIZE);


if (data_ptr != NULL)
{
  copia(h_data_ptr,data_ptr,data_len,2);

  puntsel = (selector) &cmi_descriptor->h_data_size;
  puntoff = (long) &cmi_descriptor->h_data_size;
  puntoff = puntoff * 2;
  arantxa= buildptr(puntsel,(void near *)puntoff);
  apuntador=&data_len;
  *arantxa = *apuntador;
  *(arantxa+2) = *(apuntador+1);
}
else
{
  puntsel = (selector) &cmi_descriptor->h_data_size;
  puntoff = (long) &cmi_descriptor->h_data_size;
  puntoff = puntoff * 2;
  arantxa= buildptr(puntsel,(void near *)puntoff);
  *arantxa = 0;
  *(arantxa+2) = 0;
}

/* --- set host semaphore to _BUSY ---------------------------------------- */
puntsel = (selector) &cmi_descriptor->h_sema;
puntoff = (long) &cmi_descriptor->h_sema;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
*arantxa = _BUSY;


/* --- wait until the semaphore in host cmi common block is set to _IDLE --- */
if (cmi_h_wait_for_controller() != E_OK)
{
  *arantxa = _IDLE;
  return(E_NO_CNTRL_RES);
}

/* --- return the result of controller ------------------------------------ */
puntsel = (selector) &cmi_descriptor->h_ret_val;
puntoff = (long) &cmi_descriptor->h_ret_val;
puntoff = puntoff * 2;
arantxa= buildptr(puntsel,(void near *)puntoff);
ret_val = (INT16)*arantxa ;

*arantxa = E_OK;

return(ret_val);
}
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

DOCUMENTO ANEXO

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

PARAMETROS

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

OD - ISOLUX WAT

```
 ISOLUX-WAT: Symbolic name of the network user
      4    : Network user address
1           ; VFD number


OD Header
Parameter  description

_____________________________
            ;OD Header Object code
            ;OD header index
255         ;ROM/RAM flag
32          ;Symbolic name length
0           ;Access protection supported
1           ;OD version
FFFFFFFF    ;Local address OD-OB
14          ;ST-OD length
FFFFFFFF    ;Local address ST-OD
400         ;First index S-OD
14          ;S-OD length
FFFFFFFF    ;Local sddress S-OD
0           ;First index DV-OD
0           ;DV-OD length
FFFFFFFF    ;Local sddress DV-OD
0           ;First index DP-OD
0           ;DP-OD length
FFFFFFFF    ;Local address DP-OD
;
;
;Basic types
;OC index description
;_____________________________
 5   1      [Boolean];
 5   2      [Integer8];
 5   3      [Integer16];
 5   4      [Integer32];
 5   5      [Unsigned8];
 5   6      [Unsigned16];
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
7      [Unsigned32];
8      [Floating point];
9      [Visible string];
0      [Octet string];
1      [Date];
2      [Time of day];
3      [Time difference];
4      [Bit string];


 type structures
index #elems  type/length ( x #elems)
-------------------------------------
ata type structure description found!


ct 'simple variable'
ndex  type length pass  accgrp accr address  symname      ext
-------------------------------------------------------------
400     1     1   00     00    0000 FFFFFFFF [ISO-1                ]   0;
401     2     1   00     00    0000 FFFFFFFF [ISO-2                ]   0;
402     3     2   00     00    0000 FFFFFFFF [ISO-3                ]   0;
403     4     4   00     00    0000 FFFFFFFF [ISO-4                ]   0;
404     5     1   00     00    0000 FFFFFFFF [ISO-5                ]   0;
405     6     2   00     00    0000 FFFFFFFF [ISO-6                ]   0;
406     7     4   00     00    0000 FFFFFFFF [ISO-7                ]   0;
407     8     4   00     00    0000 FFFFFFFF [ISO-8                ]   0;
408     9     8   00     00    0000 FFFFFFFF [ISO-9                ]   0;
409    10     8   00     00    0000 FFFFFFFF [ISO-10               ]   0;
410    11     7   00     00    0000 FFFFFFFF [ISO-11               ]   0;
411    12     4   00     00    0000 FFFFFFFF [ISO-12               ]   0;
412    13     4   00     00    0000 FFFFFFFF [ISO-13               ]   0;
413    14     8   00     00    0000 FFFFFFFF [ISO-14               ]   0;


ct 'Array'
ndex #elems type length pass  accgrp accr address  symname       ext
-------------------------------------------------------------------
rray variable found!


ct 'Record'
ndex  type  pass  accgrp accr address  symname      ext
-------------------------------------------------------
Record found!
```


© Del documento, los derechos: Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
ct 'Event'
ndex  type length pass accgrp accr enable  address  symname      ext
-------------------------------------------------------------------
vent found!


ct 'Domain'
ndex lenght pass  accgrp accr state address  symname      ext
-------------------------------------------------------------
omain found!


ct 'Variable list'
ndex #objects  pass accgrp accr   del    addr symname  ext  object list
-----------------------------------------------------------------------
ariable list found!


ct 'Program invocation'
ndex #doms  pass accgrp accr   del reuse state   address  symname      ext  index list
-----------------------------------------------------------------------------------
rogram invocation found!
```


...del documento, los autores - Tesis Doctoral realizada por [illegible] Politécnica Universitaria, 2006

OD - PC

PC;

5

1

```
OD Header
Parameter  description
-----------------------------------------------------
           ;OD Header Object code
           ;OD header index
55         ;ROM/RAM flag
7          ;Symbolic name length
           ;Access protection supported
           ;OD version
FFFFFFF    ;Local address OD-OB
4          ;ST-OD length
FFFFFFF    ;Local address ST-OD
C0         ;First index S-OD
4          ;S-OD length
FFFFFFF    ;Local sddress S-OD
           ;First index DV-OD
           ;DV-OD length
FFFFFFFF   ;Local sddress DV-OD
           ;First index DP-OD
           ;DP-OD length
FFFFFFFF   ;Local address DP-OD
;
;
;Basic types
;OC index description
;-----------------------------------------------------
 5   1     [Boolean];
 5   2     [Integer8];
 5   3     [Integer16];
 5   4     [Integer32];
 5   5     [Unsigned8];
 5   6     [Unsigned16];
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
;
;Object 'Event'
;OC index  type length pass accgrp accr enable  address  symname      ext
;_____________________________________________________________________
;No Event found!
;
;
;Object 'Domain'
;OC index lenght pass  accgrp accr state address  symname      ext
;_____________________________________________________________________
;No Domain found!
;
;
;Object 'Variable list'
;OC index #objects  pass accgrp accr   del    addr symname  ext  object list
;_____________________________________________________________________
;No variable list found!
;
;
;Object 'Program invocation'
;OC index #doms  pass accgrp accr   del reuse state   address  symname      ext  index list
;_____________________________________________________________________
;No program invocation found!
;
;
;
```


© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

SERVIVIOS FMS Y FMA7

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

MS services error classes and error codes

	class_ code	error class	error code	meaning class	error
E_INIT_OTHER	0x0000	0	0	Initiate	Other
_INIT_MAX_PDU_SIZE_INSUFF	0x0001	0	1		Max_PDU-Size-insufficient
_INIT_FEAT_NOT_SUPPORTED	0x0002	0	2		Feature-Not-Supported
E_INIT_OV_VERSION_INCOMP	0x0003	0	3		Version-OD-Incompatible
E_INIT_USER_DENIED	0x0004	0	4		User-Initiate-Denied
_INIT_PASSWORD_ERROR	0x0005	0	5		Password-Error
_INIT_PROFILE_NUMB_INCOMP	0x0006	0	6		Profile-Number-Incompatible
_VFD_STATE_OTHER	0x0100	1	0	VFD-State	Other
_APPLICATION_OTHER	0x0200	2	0	Application	Other
E_APPLICATION_UNREACHABLE	0x0201	2	1		Unreachable
_DEF_OTHER	0x0300	3	0	Definition	Other
_DEF_OBJ_UNDEF	0x0301	3	1		Object-Undefined
E_DEF_OBJ_ATTR_INCONSIST	0x0302	3	2		Object-Attributes-Inconsistent
E_DEF_OBJECT_ALREADY_EXISTS	0x0303	3	3		Object-Already-Exists
_RESOURCE_OTHER	0x0400	4	0	Resource	Other
E_RESOURCE_MEM_UNAVAILABLE	0x0401	4	1		Memory-Unavailable
_SERV_OTHER	0x0500	5	0	Service	Other
_SERV_OBJ_STATE_CONFLICT	0x0501	5	1		Object-State-Conflict
_SERV_PDU_SIZE	0x0502	5	2		PDU-Size
E_SERV_OBJ_CONSTR_CONFLICT	0x0503	5	3		Object-Constraint-Conflict
_SERV_PARAM_INCONSIST	0x0504	5	4		Parameter-Inconsistent
_SERV_ILLEGAL_PARAM	0x0505	5	5		Illegal-Parameter
E_ACCESS_OTHER	0x0600	6	0	Access	Other
E_ACCESS_OBJ_INVALIDATED	0x0601	6	1		Object-Invalidated
_ACCESS_HARDWARE_FAULT	0x0602	6	2		Hardware-Fault
_ACCESS_OBJ_ACCESS_DENIED	0x0603	6	3		Object-Access-Denied
E_ACCESS_ADDR_INVALID	0x0604	6	4		Invalid-Address
E_ACCESS_OBJ_ATTR_INCONST	0x0605	6	5		Object-Attribute-Inconsistent
_ACCESS_OBJ_ACCESS_UNSUPP	0x0606	6	6		Object-Access-Unsupported
_ACCESS_OBJ_NON_EXIST	0x0607	6	7		Object-Non-Exist
E_ACCESS_TYPE_CONFLICT	0x0608	6	8		Type-Conflict
E_ACCESS_NAME_ACCESS_UNSUP	0x0609	6	9		Name-Access-Unsupported
_OV_OTHER	0x0700	7	0	OD	Other
_OV_NAME_LEN_OVERFLOW	0x0701	7	1		Name-Length-Overflow
E_OV_OVERFLOW	0x0702	7	2		OD-Overflow
_OV_WRITE_PROTECT	0x0703	7	3		OD-Write-Protected
_OV_EXTENSION_LEN_OVERFLOW	0x0704	7	4		Extension-Length-Overflow
_OV_OBJ_DESCR_OVERFLOW	0x0705	7	5		OD-Descr-Length-Overflow
E_OV_OPERAT_PROBLEM	0x0706	7	6		Operational-Problem
_OTHER	0x0800	8	0	Other	Other

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Additional details:

IO_ADD_DETAIL	0x00
AD_LLI_UNEXP_FDL_OR_TIMER_EVT	0x01
AD_LLI_LSAP_ACT_FAILED	0x02
AD_LLI_POLL_LIST_LOAD_FAILED	0x03
AD_LLI_PUT_RESRC_FAILED	0x04
AD_LLI_FDL_RESET_FAILED	0x05

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Overview of FMS services (in service group order)

Service group	Identifier	Code
Context-Management	INITIATE	0
	ABORT	38
	REJECT	39
VFD local	CREATE_VFD_LOC	46
	VFD_SET_PHYS_STATUS_LOC	47
VFD remote	STATUS	2
	UNSOLICITEDSTATUS	34
	IDENTIFY	3
OD local	INITIATE_LOAD_OV_LOC	43
	LOAD_OV_LOC	44
	TERMINATE_LOAD_OV_LOC	45
	OV_READ_LOC	42
OD remote	INITIATE_PUT_OV	30
	PUT_OV	31
	TERMINATE_PUT_OV	32
	GETOV	6
Variable service	READ	4
	READWITHTYPE	7
	WRITE	5
	WRITEWITHTYPE	8
	INFORMATIONREPORT	33
	INFORMATIONREPORTWITHTYPE	36
	DEFINEVARIABLELIST	9
	DELETEVARIABLELIST	10
	PHYS_READ	28
	PHYS_WRITE	29
Domain Download	INITIATEDOWNLOADSEQUENCE	11
	DOWNLOADSEGMENT	12
	TERMINATEDOWNLOADSEQUENCE	13
	REQUESTDOMAINDOWNLOAD	17
Domain Upload	INITIATEUPLOADSEQUENCE	14
	UPLOADSEGMENT	15
	TERMINATEUPLOADSEQUENCE	16
	REQUESTDOMAINUPLOAD	18
Program Invocation	CREATEPROGRAMINVOCATION	19
	DELETEPROGRAMINVOCATION	20
	START	21
	STOP	22
	RESUME	23
	RESET	24
	KILL	25
	PI_SET_STATE_LOC	48
Event-Service	EVENTNOTIFICATION	35
	EVENTNOTIFICATIONWITHTYPE	37
	ACKNOWLEDGEEVENTNOTIFICATION	27
	ALTEREVENTCONDITIONMONITORING	26

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Overview of FMS services (in code order)

Identifier	Code
INITIATE	0
STATUS	2
IDENTIFY	3
READ	4
WRITE	5
GETOV	6
READWITHTYPE	7
WRITEWITHTYPE	8
DEFINEVARIABLELIST	9
DELETEVARIABLELIST	10
INITIATEDOWNLOADSEQUENCE	11
DOWNLOADSEGMENT	12
TERMINATEDOWNLOADSEQUENCE	13
INITIATEUPLOADSEQUENCE	14
UPLOADSEGMENT	15
TERMINATEUPLOADSEQUENCE	16
REQUESTDOMAINDOWNLOAD	17
REQUESTDOMAINUPLOAD	18
CREATEPROGRAMINVOCATION	19
DELETEPROGRAMINVOCATION	20
START	21
STOP	22
RESUME	23
RESET	24
KILL	25
ALTEREVENTCONDITIONMONITORING	26
ACKNOWLEDGEEVENTNOTIFICATION	27
PHYS_READ	28
PHYS_WRITE	29
INITIATE_PUT_OV	30
PUT_OV	31
TERMINATE_PUT_OV	32
INFORMATIONREPORT	33
UNSOLICITEDSTATUS	34
EVENTNOTIFICATION	35
INFORMATIONREPORTWITHTYPE	36
EVENTNOTIFICATIONWITHTYPE	37
ABORT	38
REJECT	39

Local FMS services

OV_READ_LOC	42
INITIATE_LOAD_OV_LOC	43
LOAD_OV_LOC	44
TERMINATE_LOAD_OV_LOC	45
CREATE_VFD_LOC	46
VFD_SET_PHYS_STATUS_LOC	47
PI_SET_STATE_LOC	48

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

FMA7 Services Error Classes and Error Codes

	class_ code	error class	error code	Description Class	Code
E_FMA7_APPLICATION_OTHER	0x0100	1	0	Application	Other
E_FMA7_APPLICATION_UNREACHABLE	0x0101	1	1		Application-Unreachable
E_FMA7_RESOURCE_OTHER	0x0200	2	0	Resource	Other
E_FMA7_RESOURCE_MEM_UNAVAILABLE	0x0201	2	1		Memory-Unavailable
E_FMA7_SERV_OTHER	0x0300	3	0	Service	Other
E_FMA7_SERV_OBJ_STATE_CONFLICT	0x0301	3	1		Object-State-Conflict
E_FMA7_SERV_OBJ_CONSTR_CONFLICT	0x0302	3	2		Object-Constraint-Conflict
E_FMA7_SERV_PARAM_INCONSIST	0x0303	3	3		Parameter-Inconsistent
E_FMA7_SERV_ILLEGAL_PARAM	0x0304	3	4		Illegal-Parameter
E_FMA7_SERV_PERM_INTERN_FAULT	0x0305	3	5		Permanent-Internal-Fault
E_FMA7_USR_OTHER	0x0400	4	0	User	Other
E_FMA7_USR_DONT_WORRY_BE_HAPPY	0x0401	4	1		Don't-Worry-Be-Happy
E_FMA7_USR_MEM_UNAVAILABLE	0x0402	4	2		Memory-Unavailable
E_FMA7_ACCESS_OTHER	0x0500	5	0	Access	Other
E_FMA7_ACCESS_OBJ_ACC_UNSUP	0x0501	5	1		Object-Access-Unsupported
E_FMA7_ACCESS_OBJ_NON_EXIST	0x0502	5	2		Object-Non_Existent
E_FMA7_ACCESS_OBJ_ACCESS_DENIED	0x0503	5	3		Object-Access-Denied
E_FMA7_ACCESS_HARDWARE_FAULT	0x0504	5	4		Hardware-Fault
E_FMA7_ACCESS_TYPE_CONFLICT	0x0505	5	5		Type-Conflict
E_FMA7_KBL_OTHER	0x0600	6	0	CRL	Other
E_FMA7_KBL_INVALID_ENTRY	0x0601	6	1		Invalid-CRL-Entry
E_FMA7_KBL_NO_KBL_ENTRY	0x0602	6	2		No-CRL-Entry
E_FMA7_KBL_INVALID_KBL	0x0603	6	3		Invalid-CRL
E_FMA7_KBL_NO_KBL	0x0604	6	4		No-CRL
E_FMA7_KBL_WRITE_PROTECTED	0x0605	6	5		CRL-Write-Protected
E_FMA7_OTHER	0x0700	7	0	Other	Other

Additional-Details:

NO_ADD_DETAIL	0x00
AD_LLI_UNEXP_FDL_OR_TIMER_EVT	0x01
AD_LLI_LSAP_ACT_FAILED	0x02
AD_LLI_POLL_LIST_LOAD_FAILED	0x03
AD_LLI_PUT_RESRC_FAILED	0x04
AD_LLI_FDL_RESET_FAILED	0x05
AD_FMA7_TO_MANY_KBL_ENTRIES	0x10
AD_FMA7_COMM_REF_NOT_ALLOWED	0x11
AD_FMA7_TO_MANY_PARALLEL_SERV	0x12
AD_FMA7_ILLEGAL_FMS_PDU_SIZE	0x13

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Overview of FMA7 Services in Service Group Order

Local Management

Service Group	Identifier	Code
Set and read all bus parameters	FMA7_SET_BUSPARAMETER	22
	FMA7_READ_BUSPARAMETER	24
Load and read the CRL	FMA7_READ_KBL_LOC	11
	FMA7_INIT_LOAD_KBL_LOC	12
	FMA7_LOAD_KBL_LOC	13
	FMA7_TERM_LOAD_KBL_LOC	14
Set and read individual FDL variables	FMA7_SET_VALUE_LOC	15
	FMA7_READ_VALUE_LOC	16
	FMA7_SET_STATISTIC_CTR	23
	FMA7_READ_STATISTIC_CTR	25
LSAP-Status	FMA7_LSAP_STATUS_LOC	17
Ident	FMA7_IDENT_LOC	18
Live-List	FMA7_GET_LIVE_LIST	26
Error messages	FMA7_EVENT	19
Halt and restart	FMA7_PROFIBUS_EXIT	21
	FMA7_RESET	20
Memory configuration	FMA7_SET_CONFIGURATION	27

Remote Management

Service group	Identifier	Code
FMA7-Context-Management	FMA7_INITIATE	0
	FMA7_ABORT	38
Load CRL	FMA7_READ_KBL_REM	1
	FMA7_INIT_LOAD_KBL_REM	2
	FMA7_LOAD_KBL_REM	3
	FMA7_TERM_LOAD_KBL_REM	4
FDL operational params	FMA7_SET_VALUE_REM	5
	FMA7_READ_VALUE_REM	6
LSAP-Status	FMA7_LSAP_STATUS_REM	7
Ident	FMA7_IDENT_REM	8

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

erview of FMA7 Services in Code Order

...tifier	Code
A7_INITIATE	0
.7_READ_KBL_REM	1
.7_INIT_LOAD_KBL_REM	2
MA7_LOAD_KBL_REM	3
MA7_TERM_LOAD_KBL_REM	4
7_SET_VALUE_REM	5
7_READ_VALUE_REM	6
MA7_LSAP_STATUS_REM	7
MA7_IDENT_REM	8
7_ABORT	38
MA7_READ_KBL_LOC	11
MA7_INIT_LOAD_KBL_LOC	12
7_LOAD_KBL_LOC	13
7_TERM_LOAD_KBL_LOC	14
A7_SET_VALUE_LOC	15
MA7_READ_VALUE_LOC	16
7_LSAP_STATUS_LOC	17
7_IDENT_LOC	18
7_EVENT	19
A7_RESET	20
A7_PROFIBUS_EXIT	21
_SET_BUSPARAMETER	22
_SET_STATISTIC_CTR	23
A7_READ_BUSPARAMETER	24
A7_READ_STATISTIC_CTR	25
_GET_LIVE_LIST	26
_SET_CONFIGURATION	27

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

DESCRIPCION DE LAS PLACAS

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

SOFTING GmbH
Dingolfinger Str. 2
81673 München, Germany
Tel.: + + 49-89-413 004-0 Fax: + + 49-89-491018

September 1993

HARDWARE DESCRIPTION

PROFI-IF-PCAT

ISP-IF-PCAT

SOFTING GmbH

Urheberschutzvermerk DIN 34, vollstandige Fassung
Nachdruck nur mit unserer Genehmigung

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only pursuant to our explicit consent

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

1 INTRODUCTION

The PC-Interface connects PC's and programming devices to ISP/PROFIBUS. It is compatible to the following devices:

- IBM-compatible PCs (according to the release list)
- Siemens PG 730 / 750 / 770

The interface is designed as a "short AT interface" with reduced height, to allow the usage in laptops.

Copyright only pursuant to our explicit consent
Nachdruck nur mit unserer Genehmigung

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only persuant to our explicit consent.
Urheberschutzvermerk DIN 34, vollstandige Fassung.
Nachdruck nur mit unserer Genehmigung

2 GENERAL

The onboard microcontroller is a NEC V25+ (μPD 70325) with a clock frequency of 10 MHz.

The PC-board utilizes a local memory of 512 KByte DRAM. The access is accomplished at 10 MHz with one wait state. Refreshing is done by the V25+ in intervalls of 10 ms with zero wait states.

The interface to the host is a 16 bit Dual Port RAM with a size of 64 KByte.

The coordination of the access is done by the READY signal of the V25+ as well as by the IOCHRDY signal of the host. The access method of the V25 is byte granular, the access method of the host is byte or word granular. The location of the DPRAM within the 16 MByte address space of the PC can be adjusted by a 8-channel DIP-switch.

The use of the ASIC 'SPC' allows transmission rates up to 1.5 MBit per seconds.

The connection to the ISP/PROFIBUS is a RS 485 interface. A 9-pin D-sub connector according to DIN 19245 is used. The connection to SINEC L2 FO is done by a HP-duplex connector for plastic fibre optics. The RS 485 driver and the optical couplers are powered galvanically decoupled by the host.

For ISP, a separate Medium Access Unit (MAU) is needed to interface IEC Physical Layer.

The RS 485 interface provides galvanic insulation, but not by means of VDE. The RS 485 standard itself does not require galvanic insulation.

The complete firmware of the PC-board is located on the host and downloaded during an initialization phase.

A green LED at the rear panel shows the actual state of the interface (ON = CP holds the token).

The V25+ triggers an interrupt on the host side by writing to the address $0FE80_{Hex}$ within the Dual Port RAM. The Interrupt line may be selected by using the jumper array X4.

- IRQ 10
- IRQ 11
- IRQ 12 default setting
- IRQ 15

The host acknowledges the interrupt request by clearing the address $0FE80_{Hex}$ in the Dual Port RAM.

The interface has to wait for the acknowledge to trigger the next interrupt on the host side.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

For copyright the requirements of DIN 34 in its entire wording shall apply.
Copyright only pursuant to our explicit consent.
Urheberschutzvermerk DIN 34, vollstandige Fassung.
Nachdruck nur mit unserer Genehmigung.

3 INSTALLATION OF THE PC-BOARD

Prior to installation the setting of the interrupt vector should be checked.

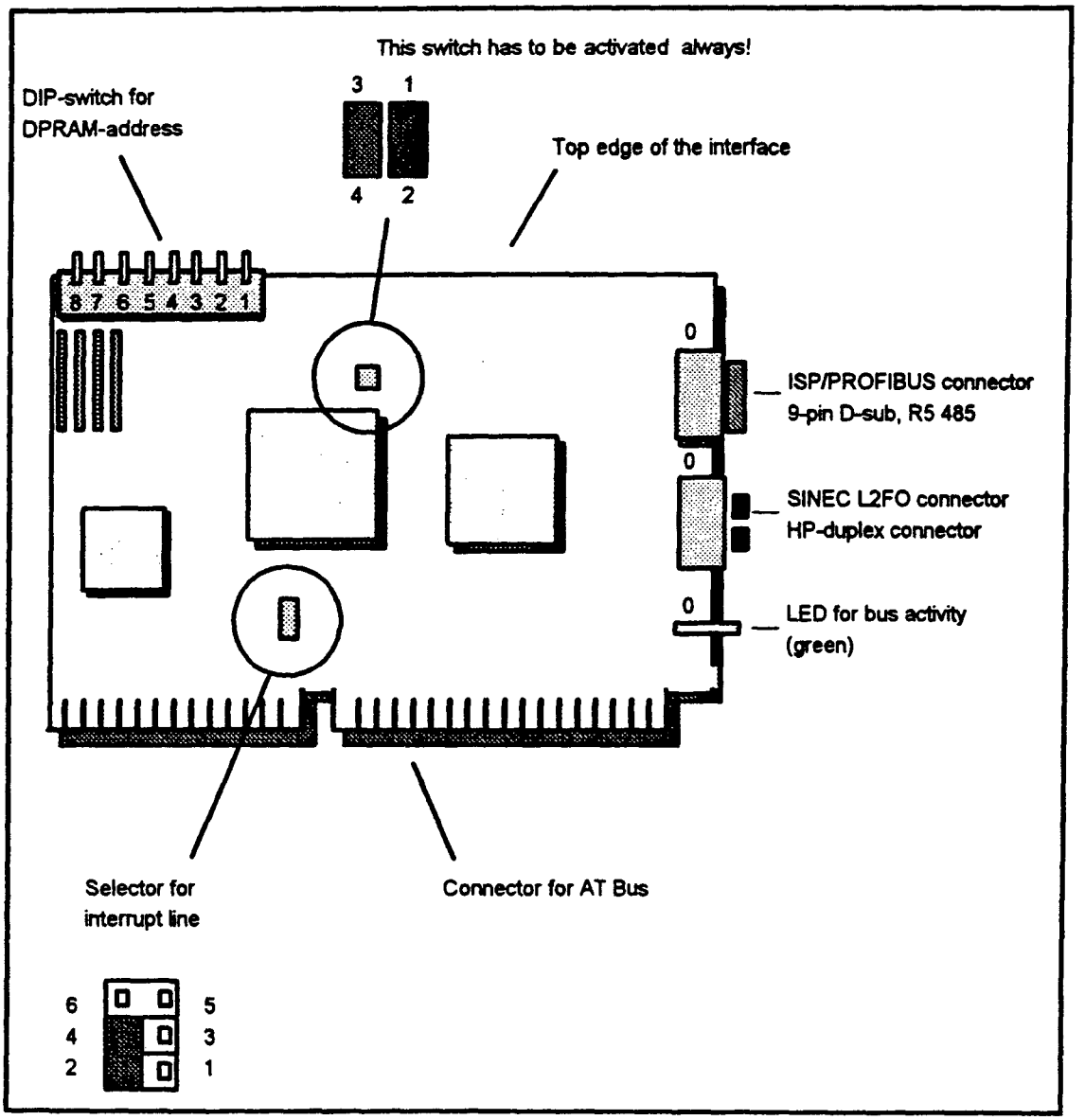


Figure 1: layout of the PC-board.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

BOARD DPRAM

-board provides a DPRAM of 64 KByte, which is used for the data exchange between host and PC-
he DPRAM is mapped into the PC´s address space:

MS-DOS	in the first MByte	(address 00 0000_{Hex} to 0F $FFFF_{Hex}$)
Unix	in the 16th MByte	(address F0 0000_{Hex} to FF $FFFF_{Hex}$

ently there are (for MS-DOS) 15 theoretically possible address locations. However there are
ons for MS-DOS as in the first MByte the normal program memory, BIOS routines and graphic
s are addressed. Practically usable are only two address loacations:

0D 0000_{Hex} to

0E 0000_{Hex} to

-switch selects only the eight most significant bits of the ISA-bus address highest (0D, 0E).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Computers running Unix and more than 16 MByte map the PC-board DPRAM in the 48th MByte.

Computers with an EISA bus and having a physical memory more than 16 MByte have to keep the 16th MByte free using the EISA configuration capability. Please refer to the manuals of your computer for the correct parameters in the configuration file.

3.2 HARDWARE-INTERRUPTS

The interrupt line is used by the PC-board in order to interrupt the operation of the PC. The line is selected by jumpers (see Figure 3). The selection must be unambiguous.

Remark: Only interrupts 10, 11, 12 or 15 are selectable. If these interrupts are not free, the devices using one of the interrupts have to be reconfigured or removed.

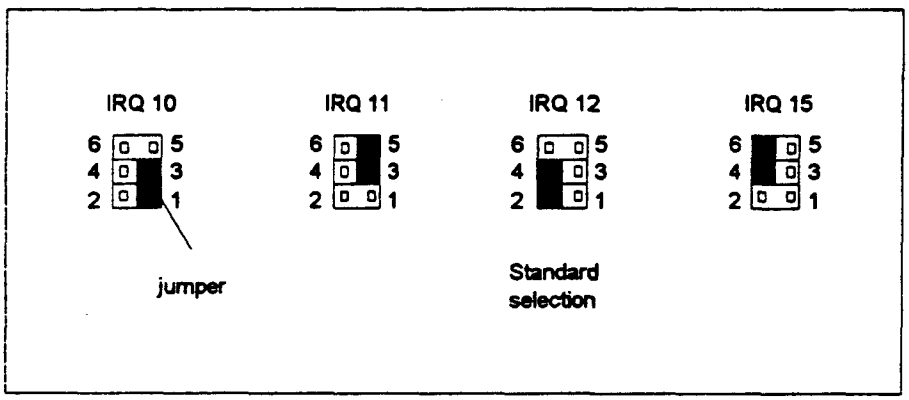


Figure 3: Selection of interrupt vectors

Please ensure that no other device has selected the configured interrupt vector!

Copyright only pursuant to our explicit consent
Nachdruck nur mit unserer Genehmigung
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

3.3 INSTALLATION INSTRUCTION

Please read the installation instructions of your computer how to open your computer, how to choose the appropriate plug and how to install the interface.

Warning: When handling the interface, please be careful to avoid damages resulting from electrostatic charge.

Warning: The maintenance of the device has only to be done by authorized persons only. Unauthorized opening and improper repair may result in a significant danger to the users!

Do not insert or pull out the interface before unpowering your computer!

Warning: Do not use interfaces that appear to be damaged.

Urheberschutzvermerk DIN 34, vollstandige Fassung.
Nachdruck nur mit unserer Genehmigung.
For copyright the requirements of DIN 34 in its entire wording shall apply.
Copyright only persuant to our explicit consent.
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

4 TECHNICAL DATA

Hardware	
- CPU	Nec V25+
- L2 bus controller	SPC
Memory	
- dynamic RAM	512 KByte
- dual port RAM	64 KByte
Connectors	
- PC connection	16 BitAT connector
- ISP/PROFIBUS connection	R5 485, 9-pin D-sub connector
- SINEC L2FO connection	HP duplex socket
Power supply	+ 5V, ± 5 %
-	
Current consumption	typ.650 m A
Protection class as described in system	IP 00 acc. DIN 40050
Maximum Ratings	
- operating temperature	0.. + 55°C
- storage temperature	-20.. + 70°C
- class of humidity	F acc. DIN 40040 (max. 95 % at 25°C)
Max. altitude	
operation	3.000 m over NN
transport	10.000 m over NN
Construction	
- format	short AT format
- size	107mm * 155 mm
- mounting width	1 SEP (= 15,24 mm)
- weight	app. 0,140 kg
Transmission rate	9,6 kBIT/s..1,5 MBit/s
Transmission mode	serial, UART format acc. TC57 FT1.2

1) SINEC L2FO is a trademark of SIEMENS

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only pursuant to our explicit consent.

Urheberschutzvermerk DIN 34, vollstandige Fassung.
Nachdruck nur mit unserer Genehmigung

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

SOFTING GmbH Dingolfinger Str. 2
81673 Munich
Tel.: (++49) (0) 89/413 004-0 Fax: (++49) (0) 89/491018

Date: July 1994

PROFIBUS CONTROLLER

Hardware Description

CP5480 (A1)

©Copyright 1994
Softing GmbH
All Rights Reserved

Unterschutzvermerk DIN 34, vollständige Fassung.
Nachdruck nur mit unserer Genehmigung

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only persuant to our explicit consent.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only persuant to our explicit consent
Urheberrechtvermerk DIN 34, vollständige Fassung
Rechtsbruck nur mit unserer Genehmigung

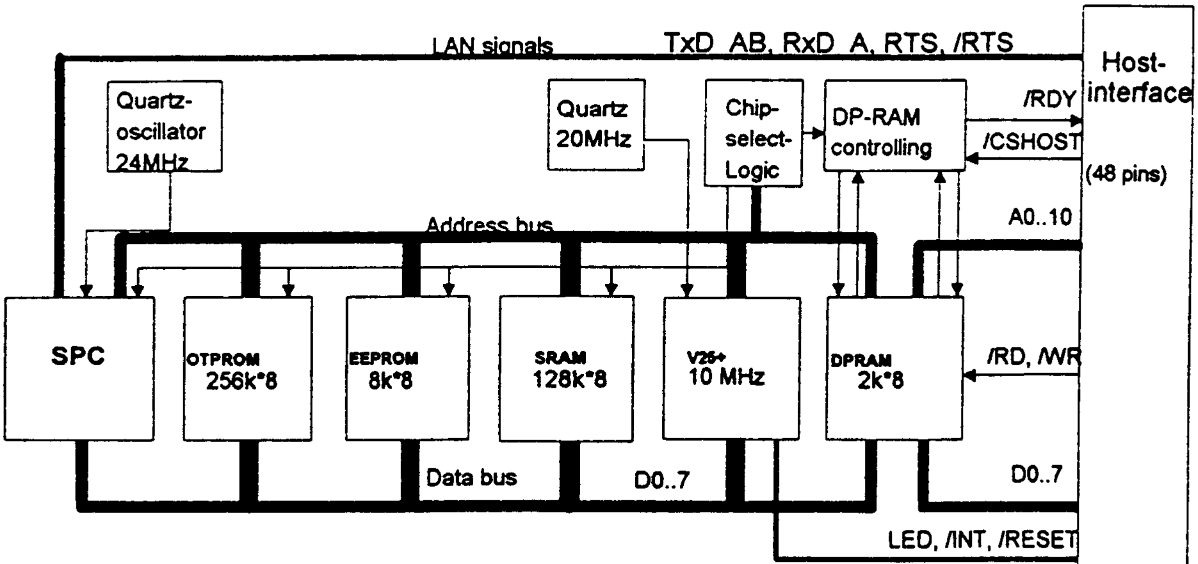
CONTENT

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

1 HARDWARE DESCRIPTION CP5480 A1

1.1 BLOCK DIAGRAM

The following picture displays the block diagram of the CP5480 A1 controller.



Picture 1.1: Block diagram of the CP5480 A1

Explanation of the diagram:

Microprocessor — The on-board micro controller is a NEC V25+ (µPD 70325) with a clock frequency of 10 Mhz. This processor is object compatible to the Intel 80x86 micro controller family. Main parts of the functionality required for communication purposes are already integrated on this chip.

OTPROM — = One Time Programmable Read Only Memory
On the CP5480 A1 a 32 pin PLCC base (Plastic Leaded Chip Carrier) is integrated carrying 256 K OTPROMs in the PLCC-OTP version (OTP = One Time Programmable). The plugged in OTPROM contains the operating system of the controller.

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only persuant to our explicit consent
Urheberschutzvermerk DIN 34, vollständige Fassung
Nachdruck nur mit unserer Genehmigung
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

EEPROM = Electrically Erasable Programmable Read Only Memory
The 32 K EEPROM on the controller board is a non-volatile memory block used for e.g. storing the bus parameters.

SRAM = Static Random-Access Memory
The 128 K of SRAM on the CP5480 A1 board are used as local memory for the V25+ micro controller.

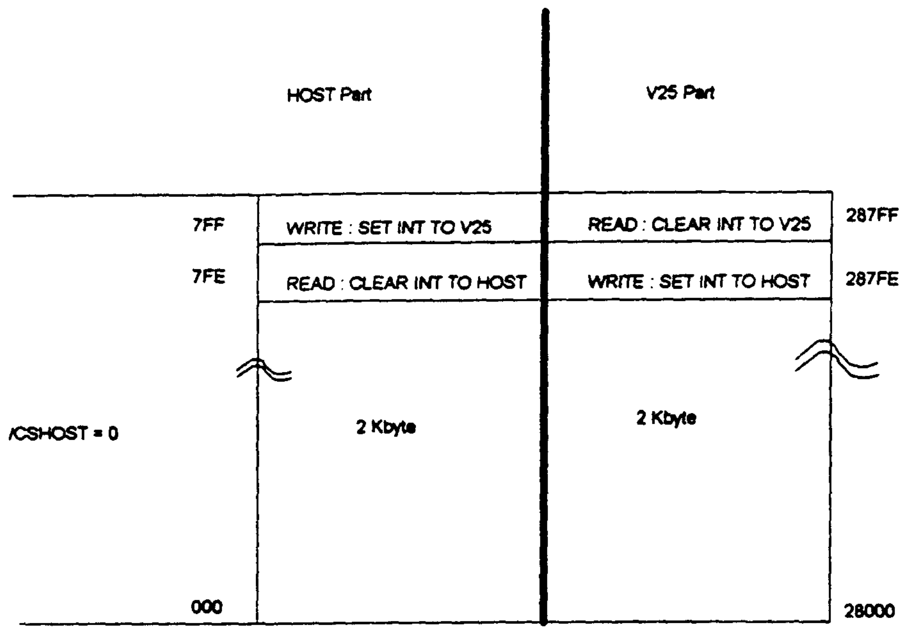
SPC = SIEMENS PROFIBUS Controller
The SPC is an ASIC which provides main parts of layer 1 and layer 2 of the PROFIBUS protocol. The following functionality is integrated in the SPC:

- UART (serial interface)
- Idle-Timer (bus synchronization)
- Syn-Intervall-Timer (receiver synchronization)
- Slot-Timer (supervising of the Wait-For-Receive-Time)
- Token-Rotation-Timer (controlling of the token rotation time)
- Tqui-Timer (controlling of the transmitter fall time)

The SPC filters all telegrams not addressed to the local device. It checks the message telegrams and separates the telegram header and telegram data. The net data is transfered to the micro controller via a 32 byte transmit / receive FIFOs. The handling of the FIFOs is done by the DMA channels of the V25+.

DPRAM The DPRAM (Dual Port RAM) is the interface between the V25+ and the host system. The size of the DPRAM is 2 K (2K * 8 bit). The host selects the DPRAM through the /CSHOST. The description from the host side assumes a linear address space of 2 K (000 to 7FF).

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only pursuant to our explicit consent
Urheberschutzvermerk DIN 34, vollstandige Fassung
Der Inhalt ist nur mit unserer Genehmigung
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



Picture 1.2: Memory image of the Dual-Port-RAM

Simultaneous access of the host and the V25+ to the same DPRAM cell (access conflict) are serialized by an internal mechanism. In this case the access of the host system may be delayed by using the RDY signal.

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only pursuant to our explicit consent

Urheberschutzvermerk DIN 34, vollständige Fassung
Nachdruck nur mit unserer Genehmigung

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

1.2 HOST INTERFACE

1.2.1 Connector Assignment

			Fan out current LOW (mA)	/ High (mA)	Fan in current value max. (uA)	Fan in capacity (pF)	max. capacity (pF)
X1	1	reserved					
X1	2	TxD_AB	8	8			45
X1	3	reserved					
X1	4	RxD_A			10	15	
X1	5	RTS	8	8			35
X1	6	ADR1			5	20	
X1	7	/WR			6	25	
X1	8	ADR2			5	20	
X1	9	ADR5			5	20	
X1	10	D0	4	4	5	20	25
X1	11	D3	4	4	5	20	25
X1	12	D6	4	4	5	20	25
X1	13	ADR4			5	20	
X1	14	D2	4	4	5	20	25
X1	15	D5	4	4	5	20	25
X1	16	/CSHOST			150	10	

Urheberschutzvermerk DIN 34, vollständige Fassung
Nachdruck nur mit unserer Genehmigung
For Copyright the requirements of DIN 34 in the version wording shall apply
Copyright only pursuant to our explicit consent

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

			Fan out current LOW (mA) / High (mA)		Fan in current value max. (uA)	Fan in capacity (pF)	max. capacity (pF)
X2	1	GND					
X2	2	VCC					
X2	3	reserved					
X2	4	reserved					
X2	5	/RTS	24	24			45
X2	6	/RD			6	25	
X2	7	ADR0			5	20	
X2	8	ADR3			5	20	
X2	9	reserved					
X2	10	D1	4	4	5	20	25
X2	11	D4	4	4	5	20	25
X2	12	D7	4	4	5	20	25
X2	13	RDY	24	24			45
X2	14	ADR6			5	20	
X2	15	VCC					
X2	16	GND					
X3	1	reserved					
X3	2	reserved					
X3	3	/RESET			170	30	
X3	4	reserved					
X3	5	reserved					
X3	6	reserved					
X3	7	resrved					
X3	8	LED	1,5	0,4			95
X3	9	reserved					
X3	10	ADR9			5	20	
X3	11	/INT	12	open drain		15	25
X3	12	ADR8			5	20	
X3	13	ADR7			5	20	
X3	14	reserved					
X3	15	reserved					
X3	16	ADR10			5	20	

Für Copyright the requirements of DIN 34 in the entire wording shall apply
Copyright only passed to our explicit consent
Unbefugte Verwertung DIN 34, vollständige Passage
Nachdruck nur mit unserer Genehmigung

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

For copyright the requirements of DIN 34 on its entire wording shall apply
Copyright only pursuant to our explicit consent

Urheberschutzvermerk DIN 34 vollinhaltlich Gültig
Nutzung nur mit unserer Genehmigung

The following connectors are used for test functions and must be wired according to the description in order to fulfil the correct functionality:

Link X3/9 and X3/15 and connect them through 1 kOhm to GND.

Link X1/1, X1/3, X2/3, X3/1 and X3/14 and connect them through 1 kOhm with VCC.

X2/4, X2/9, X3/2, X3/4, X3/5, X3/6 and X3/7 must not be connected.

1.2.2 Signal Description

TxD_AB	Serial data output (channel A) with TTL level for driving the LAN transmitter.
RxD_A	Serial data input (channel A) driven by the LAN receiver with TTL level.
RTS	Signal used to activate the LAN transmitter during the send time. On sending the RTS signal is HIGH.
/RTS	/RTS is the inverted RTS signal. During sending the /RTS signal is low.
ADR0-10	Address bus to the DPRAM.
D0-7	Data bus to the DPRAM.
/WR	Write signal for DPRAM access.
/RD	Read signal for DPRAM access.
/CSHOST	Chip Select input for the DPRAM.
GND	0 (zero) volt connector of the power supply (Both pins have to be connected).
VCC	5 volt connector of the power supply (Both pins have to be connected).
RDY	Ready signal of the CP5480 A1 to be linked with the read signal of the host system. RDY = HIGH ==> "access finished".
/INT	Interrupt from the CP5480 A1 to the host system. The transmission type is 'open drain' and connected to VCC over a 2,2 kOhm resistor within the CP5480 A1.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

/RESET	Hardware reset. The CP5480 A1 is set back to a ground state (restart of the micro controller)
LED	Indicates the state 'station in ring'. Can be used to control a LED driver. LED HIGH signifies 'station in ring'.

1.2.3 Description of the Host Interface

Reset condition

The 48 pin host connector contains the /RESET channel. On power-on this channel must be set to LOW for at least 30 milliseconds. During normal operation of the CP5480 A1 it is sufficient to set the reset channel for only 30 microseconds to LOW to achieve the reset state.

The state of the /RESET channel does not influence the access control of the DPRAM. The host system is able to access the DPRAM during the /RESET = LOW.

Access to the DPRAM

The 48 pin host connector also contains the following signals which are used for driving the DPRAM (see also bus timing):

ADR0 - ADR10	Address bus, drives the DPRAM directly.
D0 - D7	Data bus, drives the DPRAM directly.
/CSHOST	Chip-Select-channel of the DPRAM.
/RD	Read channel for the DPRAM.
/WR	Write channel for the DPRAM.
RDY	Ready signal of the CP5480 A1. LOW during an access conflict between the host system and the V25+ . The host system is not allowed to finish the DPRAM access unless the RDY signal is HIGH. The RDY signal must be observed while accessing the DPRAM.

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only pursuant to our explicit consent
Urheberschutzvermerk DIN 34, vollständige Fassung
Nachdruck nur mit unserer Genehmigung

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only pursuant to our explicit consent

Urheberschutzvermerk DIN 34 vollinhaltlich anzuwenden
Nachdruck nur mit unserer Genehmigung

Interrupt

/INT	Interrupt from the CP5480 A1 to the host system. The transmission type is 'open drain' and connected to +5 volt over a 2,2 kOhm resistor within the CP5480 A1.

LED connector

LED	Indicate the state 'station in ring'. This signal may be used to control a LED driver. LED = HIGH signifies 'station in ring'.

1.2.4 Timing

A write access to the Dual-Port-RAM is indicated by the /CSHOST=LOW and /CSHOST/IWR=LOW. Only if both signals are set to LOW the write access is initiated. As soon as one signal is set to HIGH the write access will be terminated.

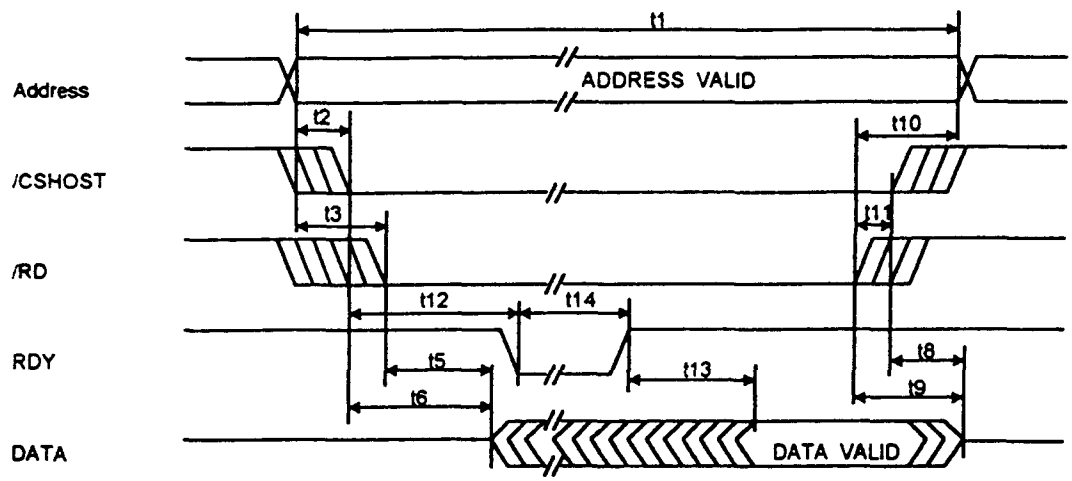
A read access to the Dual-Port-Ram is indicated by the /CSHOST=LOW and /RD=LOW. Only when both signals are set to LOW, the read access is initiated. As soon as one signal is set to HIGH the read access is terminated. During the read access the /WR must be set to HIGH.

The ready signal RDY is only set LOW if both the host processor and the micro controller of the CP5480 A1 try to access concurrently the same cell of the Dual-Port-RAM. Therefore the ready signal is normally set to HIGH (RDY pulse width = 0 ns!).

Definition:

LOW Z:	Data bus signals of the host interface have low impedance.
HIGH Z:	Data bus signals of the host interface have high impedance.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



Dual-Port-RAM read cycle

Test conditions VCC= 5V+/-5%, T_A = 0 to 70° C

Parameter	Description	min.	max.	Unit	Notes
t1	Read Cycle Time	0.1	100	us	/CSHOST = LOW /RD = LOW
t2	ADDRESS VALID to /CSHOST LOW	0		ns	
t3	ADDRESS VALID to /RD LOW	0		ns	
t5	/RD LOW to LOW Z	3		ns	
t6	/CSHOST LOW to LOW Z	5		ns	
t8	/CSHOST HIGH to DATA HIGH Z		25	ns	
t9	/RD HIGH to DATA HIGH Z		25	ns	
t10	/RD HIGH to ADDRESS CHANGE	0		ns	
t11	/RD HIGH tq /CSHOST HIGH	0		ns	
t12	/CSHOST LOW to RDY LOW		55	ns	
t13	RDY HIGH to DATA VALID		45	ns	
t14	RDY Pulse Width	0	550	ns	

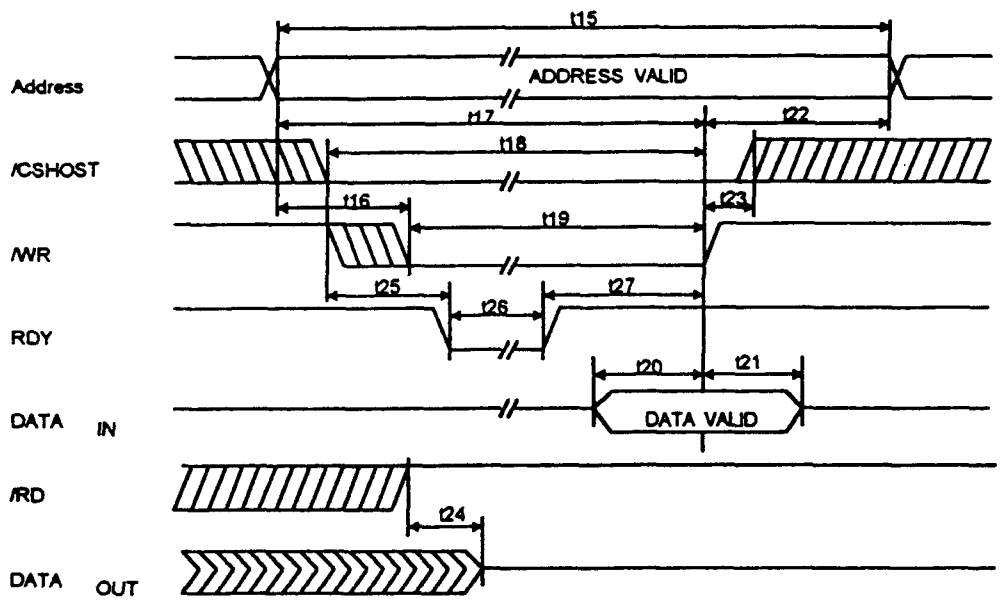
Picture 1.3: Dual-Port-RAM read timing

Die Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten.
Copyright reserved. Duplication, use or disclosure to third parties without express authority is forbidden.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Für evtl. Verletzungen des DIN 34 wird keine Haftung übernommen.
Rechte nur mit unserer Genehmigung

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only persuant to our explicit consent



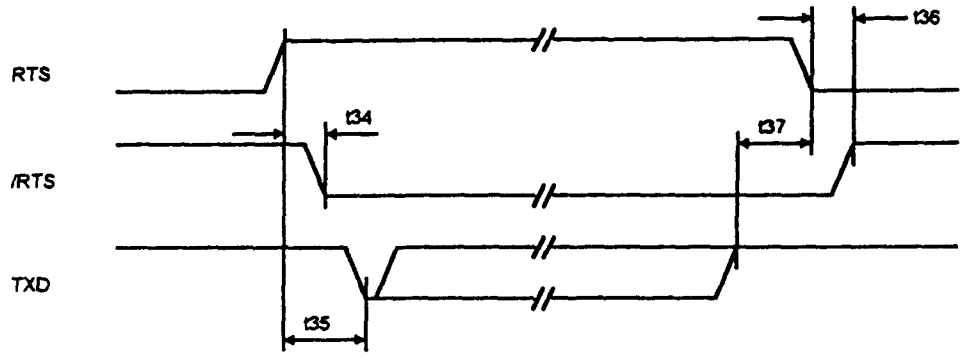
Dual-Port-RAM write cycle

Test requirements VCC= 5V+/-5%, T_A = 0 to 70° C

Parameter	Description	min.	max.	Unit	Notes
t15	Write Cycle Time	0.1	100	us	/CSHOST = LOW /RD = LOW
t16	ADDRESS Set-Up to /WR LOW	0		ns	
t17	ADDRESS Set-Up to /WR HIGH	40		ns	
t18	/CSHOST LOW to /WR HIGH	55		ns	
t19	/WR Pulse Width	30		ns	
t20	DATA Set-Up to /WR HIGH	20		ns	
t21	DATA Hold from /WR HIGH	0		ns	
t22	ADDRESS Hold from /WR HIGH	2		ns	
t23	/WR HIGH to /CSHOST HIGH	15		ns	
t24	/RD HIGH to DATAout HIGH Z		25	ns	
t25	/CSHOST LOW to RDY LOW		55	ns	
t26	RDY Pulse Width	0	550	ns	
t27	RDY HIGH to /WR HIGH	35		ns	

Picture 1.4: Dual-Port-RAM write timing

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



Transmit timing

Test requirements VCC= 5V+/-5%, T_A = 0 to 70° C

Parameter	Description	min.	max.	Unit	Notes
t34	RTS HIGH to /RTS LOW	0	10	ns	
t35	RTS HIGH to TxD LOW	500		ns	Start of Transmission
t36	RTS LOW to /RTS HIGH	0	10	ns	
t37	TxD HIGH to RTS LOW	500		ns	End of Transmission

Picture 1.5: Transmit timing

For copyright the requirements of DIN 34 must be under wording shall apply
Copyright only pursuant to our explicit consent
Urheberschutzvermerk DIN 34, vollständige Fassung
Nachdruck nur mit unserer Genehmigung
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only pursuant to our explicit consent
Urheberschutzvermerk DIN 34, vollständige Fassung
Nachdruck nur mit unserer Genehmigung

1.3 DESCRIPTION OF THE LAN INTERFACE

All signalling lines needed for the LAN interface use TTL level and are included in the 48 pin host interface socket. This enables the user to realize a medium adaption.

The following versions of the LAN interface are possible:

⇒ RS485 DC coupled

⇒ RS485 galvanically insulated

⇒ Fiber Optic with glass wire

⇒ Fiber Optic with plastic wire

TxD_AB serial data output of the LAN interface

RxD_A serial data input of the LAN interface

RTS HIGH if TxD_AB is active, is used to activate the RS485 driver if operating in the RS485 DC coupled mode.

/RTS [1] LOW if TxD_AB is active. Is used to control the opto connectors for activating the RS485 driver if operating in the RS485 galvanically insulated mode.

Bit coding:

Send mode	CP 5480 A1 LAN interface			PROFIBUS coding
	/RTS	RTS	TxD_AB	
	Low	High	High	"1"
	Low	High	Low	"0"
	High	Low	High	Idle ("1")

Receive mode	PROFIBUS coding	CP 5480 A1 LAN interface		
		/RTS	RTS	RxD_a
	"1"	High	Low	High
	"0"	High	Low	Low
	Idle ("1")	High	Low	High

[1] Inverted RTS signal

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

1.4 TECHNICAL DATA

1.4.1 Electrical Data

Distribution voltage VCC:

Power supply	5 Volt DC
Operating interval	4,75 ... 5,25 Volt
Destruction limit	-0,3 ... 6 Volt
Current consumption	500 mA max. 330 mA average

Signal level:

Output High Voltage	min. 2,4 Volt
Output Low Voltage	max. 0,4 Volt
Input High Voltage	min. 2,2 Volt
Input Low Voltage	max. 0,8 Volt

1.4.2 Climate Conditions

Environment temperature:

in operation	0 ... +70° C
storage and transport	-40 ... +85° C
change in temperature	max. 10° C
relative humidity in operation, storage and transport	max. 95% at 25° C, no dew

Max. altitude

in operation	3.000 m
transport	10.000 m

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only pursuant to our explicit consent
Urheberschutzvermerk DIN 34, vollständige Fassung
Nachdruck nur mit unserer Genehmigung
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

1.4.3 LAN Interface

Transmission rate	9,6 kbit/s
	19,2 kbit/s
	93,75 kbit/s
	187,5 kbit/s
	500 kbit/s
	1,5 kbit/s
Transmission mode	TTL level on host plug
	Adaption of media (RS485 or FO) is to be realized on the host board.

1.4.4 Mechanical Data

Mechanical environment conditions

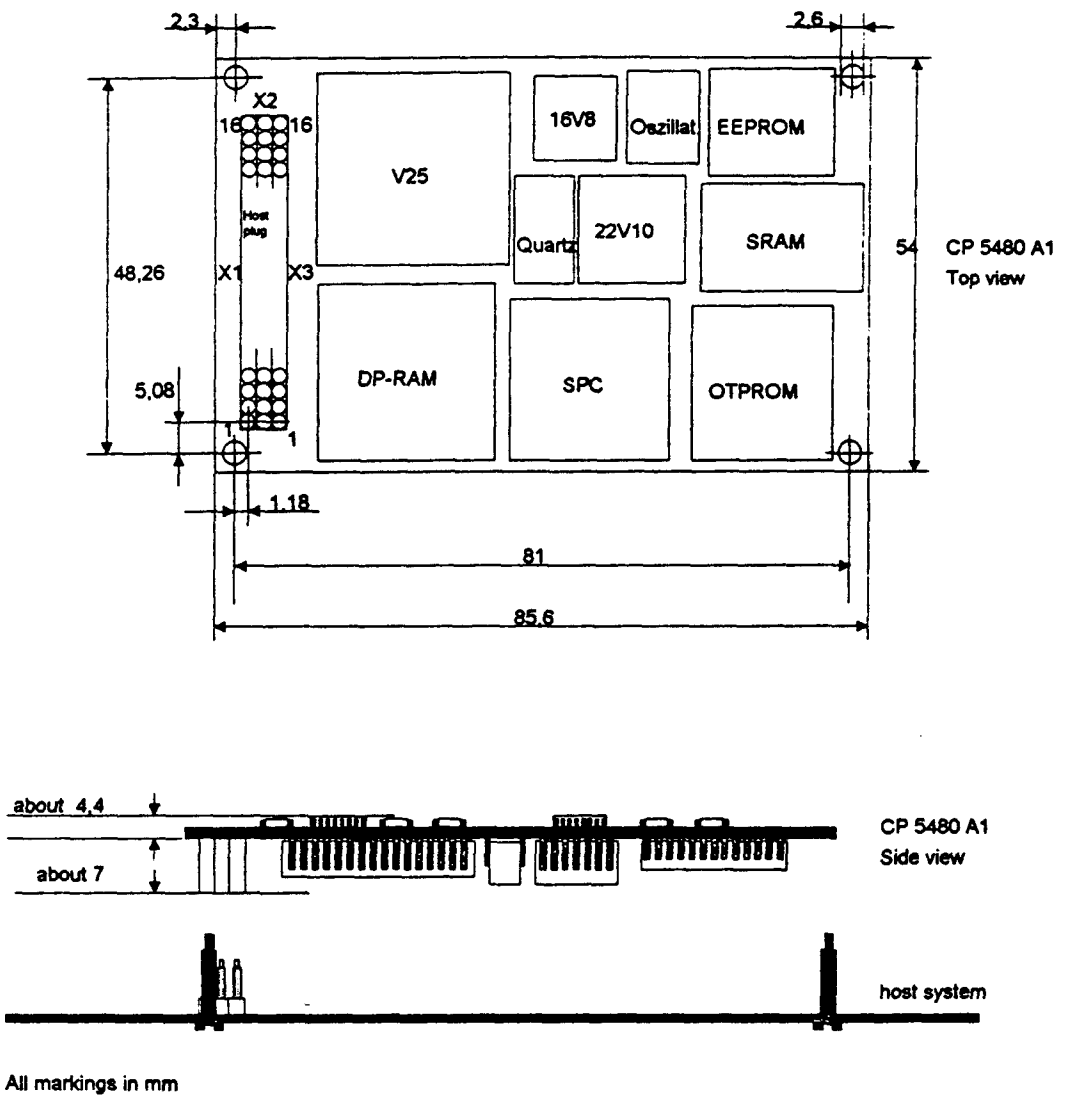
Oscillation	according to IEC 68-2-6
tested while in operation	10 - 58 Hz:
	constant excursion 0,15 mm
	58 - 500 Hz:
	constant accelerations 19,6 m/s^2
Shock	according to IEC 68-2-27
tested while in operation	150 m/s^2; 11 ms: 3 Shocks per direction

Mechanical Data

Protective system	IP 00, open component for assembly on a host system
Size	85,6 mm x 54 mm x 11,4 mm (LxWxH)
Weight	60 Gram

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

For copyright the requirements of DIN 34 must be strictly observed. Copyright only pursuant to our explicit consent
Urheberschutzvermerk DIN 34 verbindlich beachten. Nachdruck nur mit unserer Genehmigung



Picture 1.6: Mechanical design

As plug for the 48 pin socket board on the CP5480 A1 three single-row 16 pin plugs with a 2.54 mm grid and square contacts (0.64 mm^2, gold over nickel) may be used. One company offering the single-row plugs is AMP Deutschland GmbH.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

2 EXAMPLE OF INTEGRATION

The CP5480 A1 communications processor module is intended for installation in a host device. It is designed exclusively for operation when installed. Accordingly, the limit values for interference suppression-interference emission, noise immunity against line disturbances, high-frequency interference and electrostatic discharge only apply to the complete unit with the module installed (host and CP5480 A1).

The achievable limit values are dependant on correct installation of the CP5480 A1 in the host in terms of ESC.

In this respect, the following rules must be adhered to:

- Use a device housing of conductive material and ground the housing.
- Use a shielded bus cable (e.g. SINEC L2-bus cable).
- Connect the shield of the bus cable where it enters the housing with low inductance over a large area. For example, by implementing the bus connection as a 9-pim sub-D socket with a spring shield plate which can be screwed to the housing of front panel making good contact.
- The RS 485 cable should be kept as short as possible within the device, i.e. the RS 485 driver should be located immediately at the entrance to the housing (sub-D socket).
- Internal device signal lines and lines with an external connection such as the RS 485 signals should be routed in strictly separate areas.
- With particular demanding requirements, use an isolating interface or optical fiber.

Ambient temperature:

The host device should be designed so that the heat created by the CP5480 A1 during operation is adequately dissipated. The ambient temperature of the CP5480 A1 must not exceed the limit values stated in the technical data at any point.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Für Copyright die Anforderungen der DIN 34 in ihrer vollständigen Fassung gelten / Copyright only pursuant to our explicit consent
Unbefugte Nutzung/Vermerk DIN 34, vollständige Fassung / Nachdruck nur mit unserer Genehmigung

UL approval:

SIEMENS AG does not intend to apply for approval for the CP5480 A1 component. The materials used, however, were selected so that they will not hinder the approval of a device with an integrated CP5480 A1, e.g. according to UL1950 (computer) or UL508 (industrial controllers)

If necessary, contact your SIEMENS partner.

2.1 HOST INTERFACE

The host interface of the CP5480 A1 is designed as a dual-port RAM interface. This means a straightforward interface that can be matched easily to the host system.

The dual-port RAM is 2 Kbyte long and can be addressed by a host processor as an additional memory. The ready signal (RDY) of the host interface must be checked. This signal varies the duration of access to the dual-port RAM. This is necessary to resolve possible conflicts accessing the dual-port RAM. The data bus is 8 bits wide.

The only other relevant interface signals are the reset and interrupt signals.

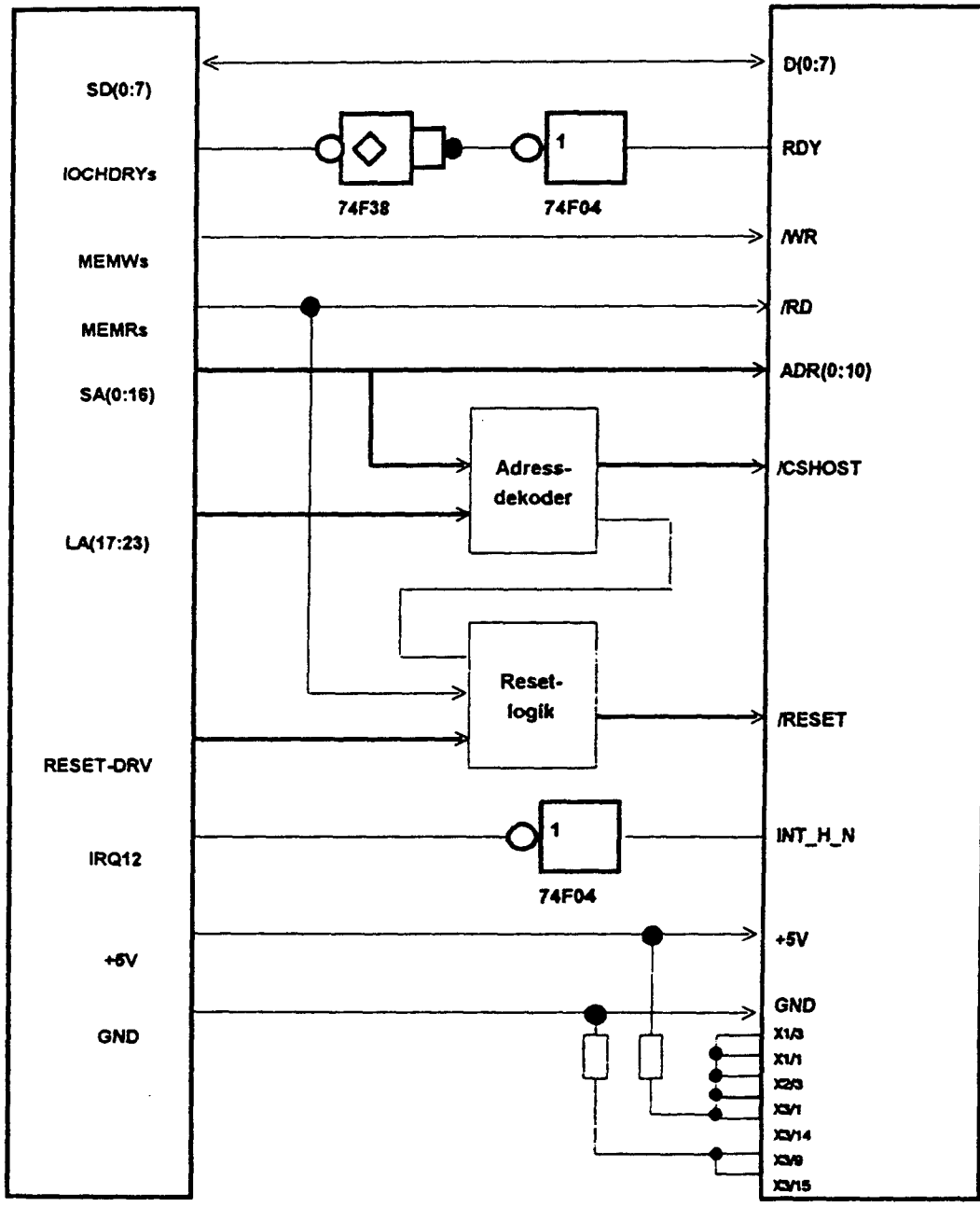
With the reset signal (/RESET), the CP5480 A1 is set to a defined initial status.

The reset time after switching on the module must not be below 30 ms and during operation not below 30 µs.

The interrupt signal (/INT) must be connected to the interrupt controller of the host processor.

The following example illustrates the connection of the CP5480 A1 to a PC/AT.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



Picture 2.1: Block diagram of the interface between the PC/AT and the CP5480 A1

Für Copyright die Anforderungen der DIN 34 und die entsprechenden Vorschriften gelten.
Kopieren nur mit unserer Genehmigung
Copyright only pursuant to our explicit consent
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only persuant to our explicit consent.
Urheberschutzvermerk DIN 34, vollstandige Fassung
Nachdruck nur mit unserer Genehmigung

Functional description:

The data bus (D0:7) and the read/write control signals are connected directly to the corresponding slot signals.

The ready signal must be connected to the slot via an open collector driver. Since this driver is inverted, a further inverter must be connected in the ready signal line to match the signals.

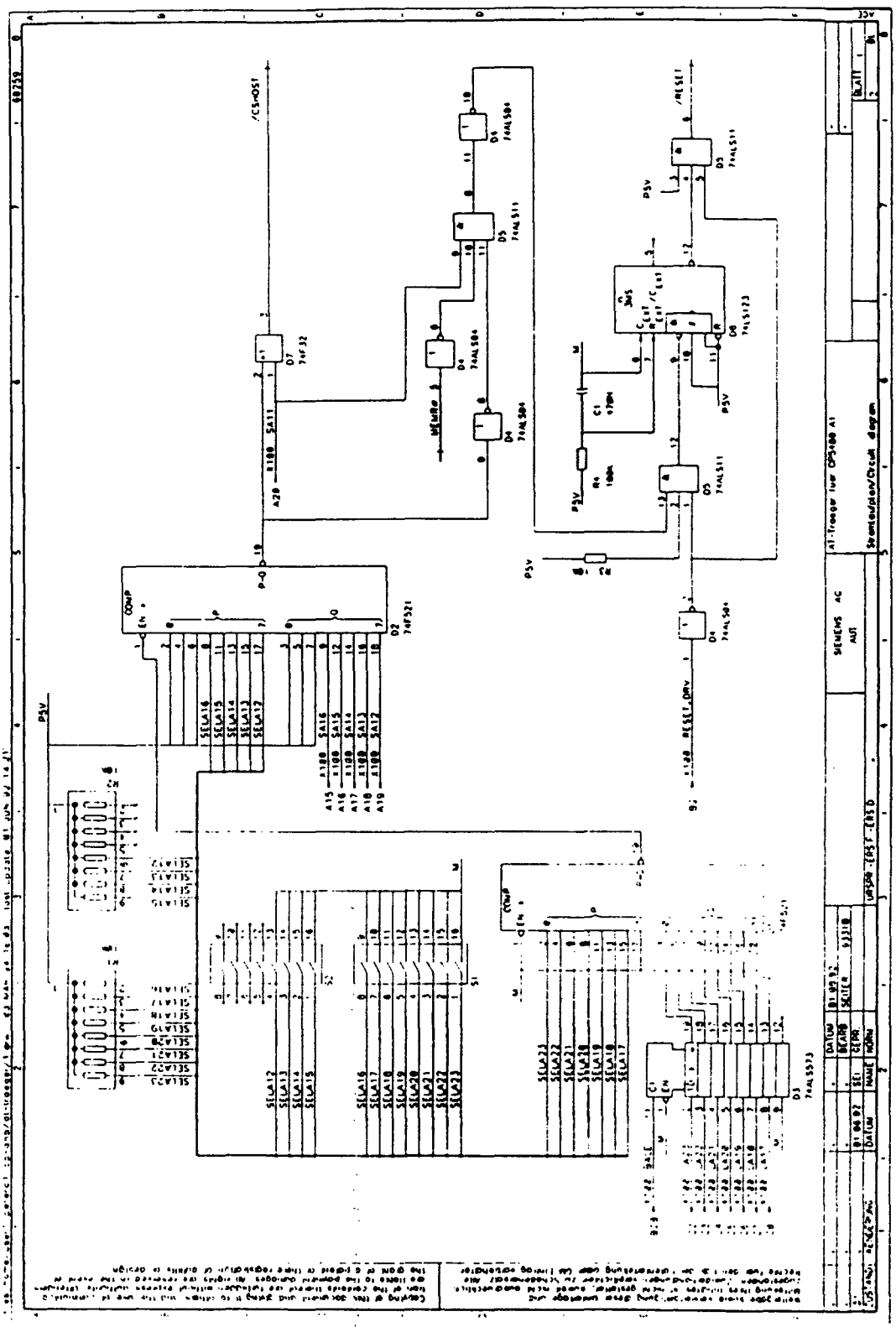
The interrupt output must also be connected via an inverter for signal matching.

The address decoder decodes a 4 Kbyte long address window from the PC/AT address area. The dual-port RAM of the CP5480 A1 can be accessed via the lower 2 Kbytes of this address window, a read access to any address of the upper 2 Kbytes of the address window triggers a hardware reset on the CP5480 A1

The location of the address window in the PC/AT address area can be set in 4 Kbyte steps with address switches.
The following circuit diagrams illustrate a suggested implementation of the connections described above.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



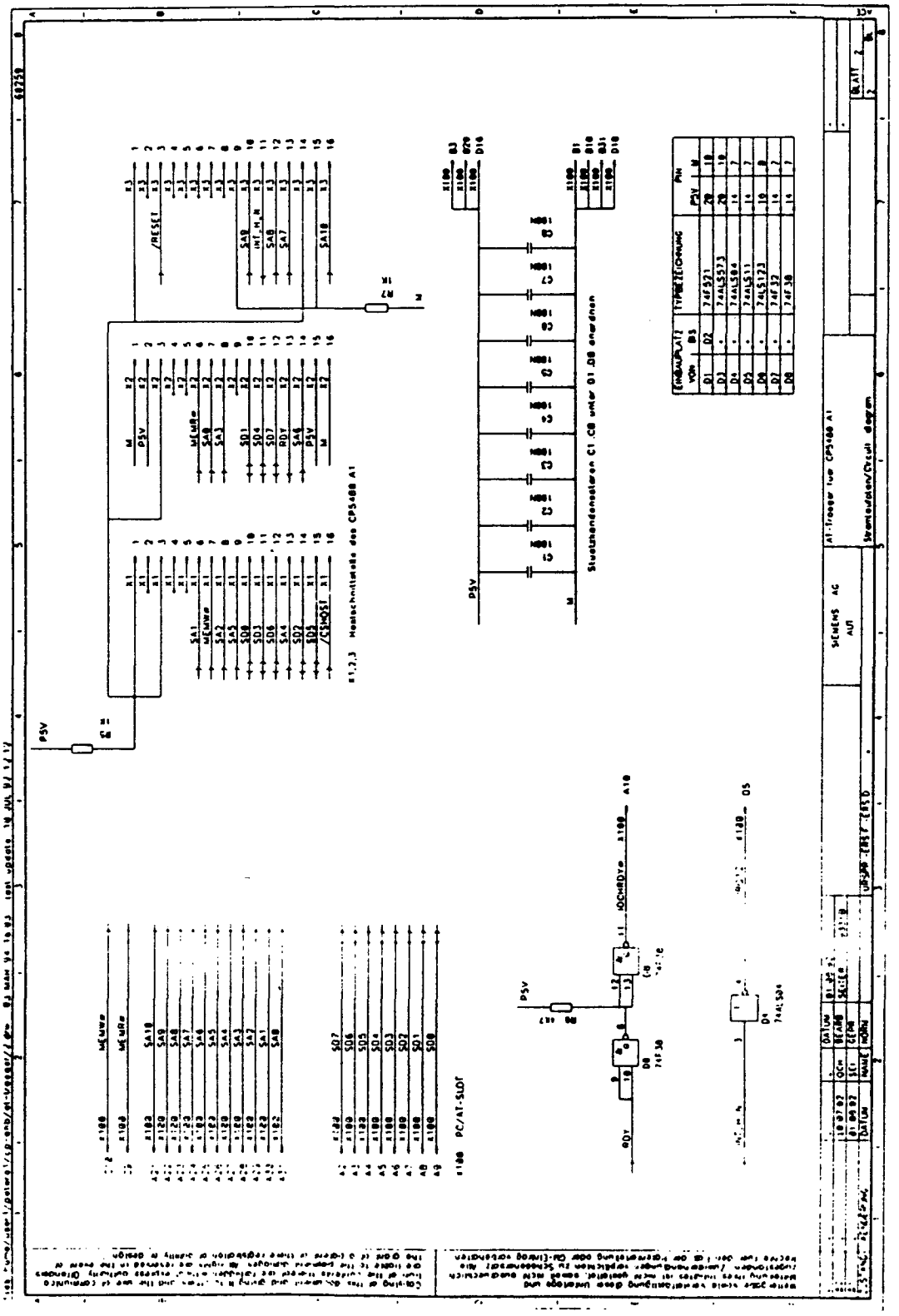
For copyright the requirements of DIN 34 on the order wording shall apply
Copyright only pursuant to our explicit consent

Urheberschutzvermerk DIN 34, vollständige Fassung
Nachdruck nur mit unserer Genehmigung

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

For copyright the requirements of DIN 34 in its entire wording shall apply
Copyright only persuant to our explicit consent

Urheberschutzvermerk DIN 34, vollstandige Fassung
Nachdruck nur mit unserer Genehmigung



2.2 LAN INTERFACE

The LAN interface for the CP5480 A1 must be created by the user with external connections. The required signals with TTL levels must be applied to the 48-pin socket connector. The amount of additional wiring depends mainly on the required transmission medium. A possible implementation is provided for each of the versions described below.

2.2.1 RS485 Interface, Non-floating

With the non-floating version of the L2 connection, the signals RTS, TXD_AB and RXD_A are led directly to the bus transceiver chip.

D21 (75176) is the RS485 bus transceiver chip, which untreated the level conversion for the level 2 interface. The resistors R106 and R107 are required so that the idle level (logical "1") is set at D21 if the L2 connector is disconnected.

The following standards are relevant:

EIA RS485 standard

PROFIBUS standard DIN 19245, Part 1

Die Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhaltes ist nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten.
Copyright only pursuant to our explicit consent
© Der Inhalt dieses Dokuments ist urheberrechtlich geschützt. Digitalisierung freigegeben durch ULB Düsseldorf

2.2.2 RS485 Interface, Floating

With the floating version of the L2 connection, the signals /RTS, TXD_AB and RXD_A are led via an optocoupler. The L2 connection is supplied via a floating d.c. /d.c. converter.

D26 is responsible for increasing the driver current to allow the LED of the optocoupler (HCPL2631) to be controlled R800, R801 and R806 limit the current to a value permitted for the particular LED. R802 - R805 and R809 are pull-up resistors. D21 (75176) is the RS485 bus transceiver chip, which undertakes the level conversion for the L2 interface. The resistors R807 and R808 are required to set the idle level (logical "1") at D21 if the L2 connector is disconnected. /RTS is the inverted RTS signal, D22 is the inverter between the optocoupler and bus transceiver for controlling the transmit enable of the bus transceiver. This inversion before and after the optocoupler is necessary to suppress disturbances on the L2 network which could occur when the operating voltage of this module is switched off. With the floating power supply, the bus transceiver can remain at operating voltage, although the controller of the optocoupler is already deenergized as a result of which the output of the optocoupler is at logical high.

The following standards are relevant:

EIA RS485 standard

PROFIBUS standard DIN 19245, Part 1

Urheberrechtlich geschützt nach DIN 34, soweit nicht anderweitig gestattet
Nachdruck und anderer Gebrauch
For copyright the requirements of DIN 34 must be observed.
Copyright only pursuant to our explicit consent

2.2.3 Interface for Plastic Optical Fibres

When connecting the L2 interface for operation with plastic optical fibres, the signals RTS, /RTS, TXD_AB and RXD_A are required.

U16 (HFBR2521) is the optical receiver chip which convertes the incoming optical signals to electrical signals. R51 is a pull-up resistor. D1 and D2 are used to switch to the receive channel RXD_A when receiving the incoming data and transmitting the outgoing data. The RXD_A channel checks the transmitted data during transmission. D15 (74F3037) is responsible for the current for the transmit LED. R55 and C54 distort the transmitted signal slightly to partly compensate for distortion during the transmission. B1 protects the transmit LED from excessive reverse voltage. R53, R54, R56 and C55 limit the transmit current to a value permitted for the LED (HFBR1521).

Refer also to the Hewlett Packard Optoelectronics Designer's Catalog.

2.2.4 Interface for Glass Optical Fibres

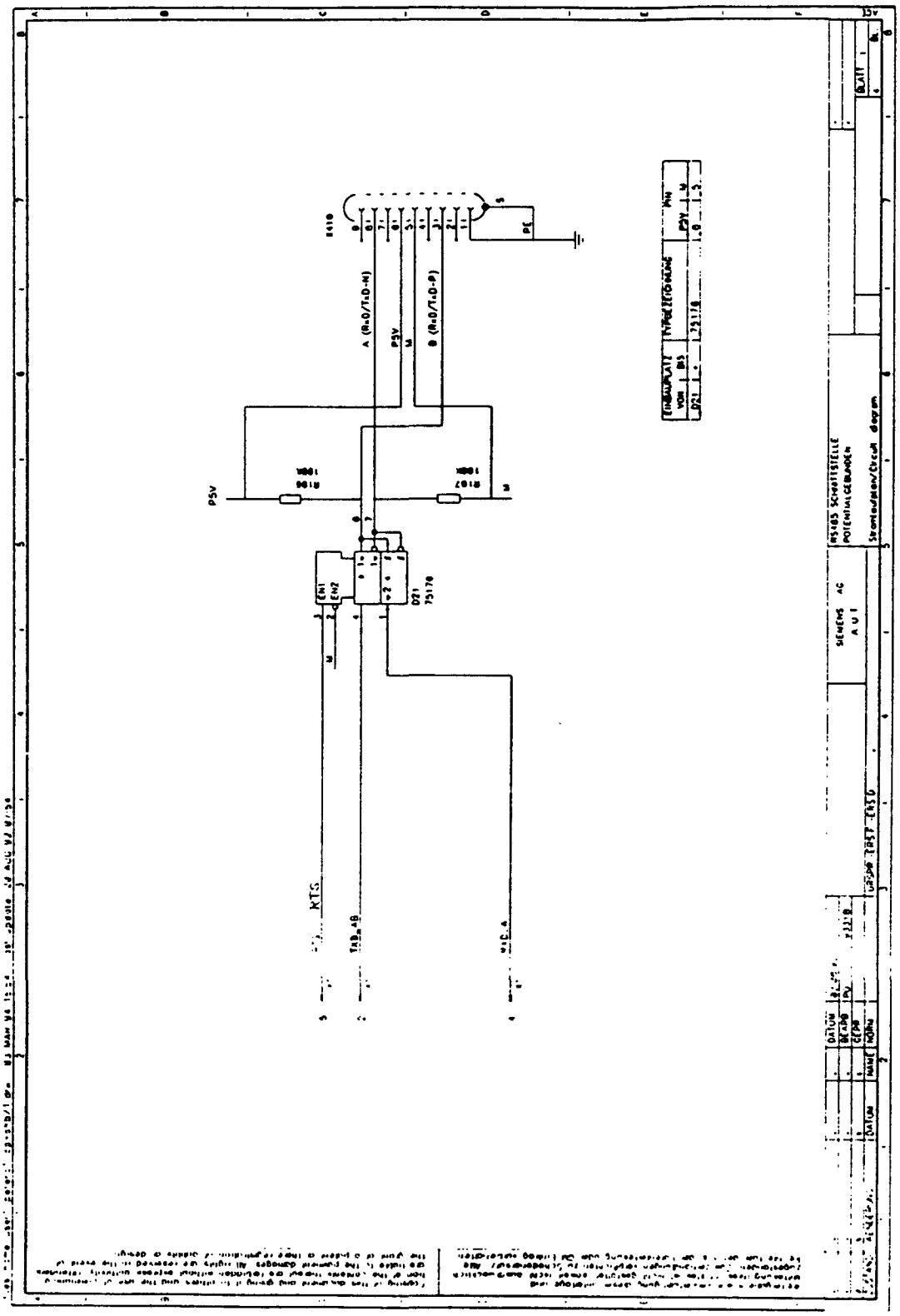
When connecting the L2 interface for operation with glass optical fibres, the signals RTS, /RTS, TXD_AB and RXD_A are required.

U17 (HFBR2412) is the optical receiver chip which converts the incoming optical signals to electrical signals. R51 is a pull-up resistor. D1 and D2 are used to switch to the receive channel RXD_A when receiving the incoming data and transmitting the outgoing data. The RXD_A channel checks the transmitted data during transmission. D15 (74F3037) is responsible for the current for the transmit LED. R55 and C54 distort the transmitted signal slightly to partly compensate for distortion during the transmission. B1 protects the transmit LED from excessive reverse voltage. R53, R54, R56 and C55 limit the transmit current to a value permitted for the LED (HFBR1414).

Refer also to the Hewlett Packard Optoelectronis Designer's Catalog.

Urheberrechtlich geschützt nach DIN 34, vollständige Kopierung
Das Urheberrecht liegt bei der Softing GmbH
For copyright the requirements of DIN 34 in the entire wording shall apply
Copyright only pursuant to our explicit consent

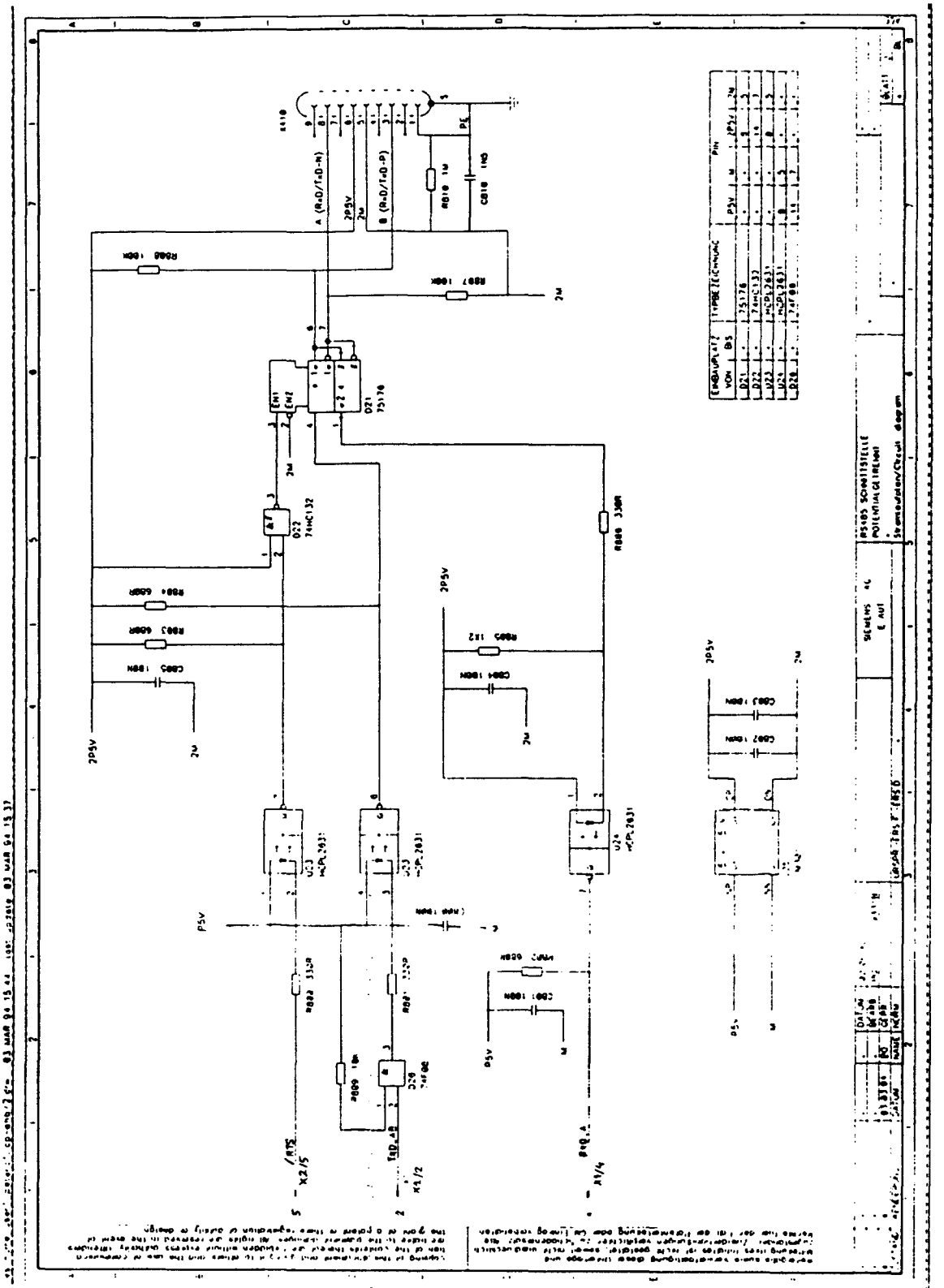
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



Die Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte vorbehalten.

For copyright the requirements of DIN 34 must be under wording shall apply
Copyright only persuant to our explicit consent.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



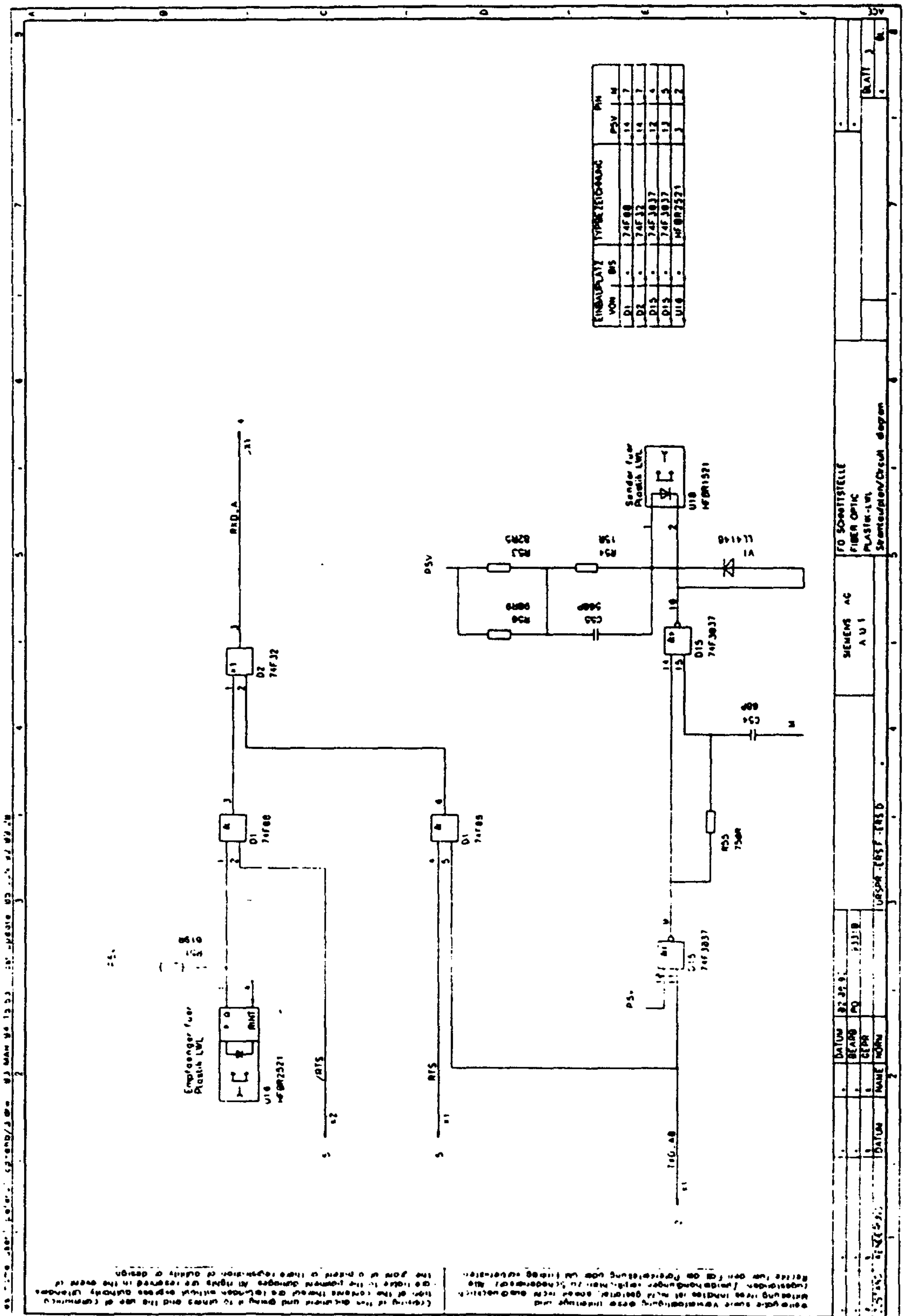
Für Kopien und Weitergabe dieser Unterlage sowie Verwertung ihres Inhalts oder seiner Teile sind ohne unsere ausdrückliche Genehmigung nicht gestattet. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten.

Copyright only pursuant to our explicit consent. For copyright the requirements of DIN 34 must be under Working shall apply

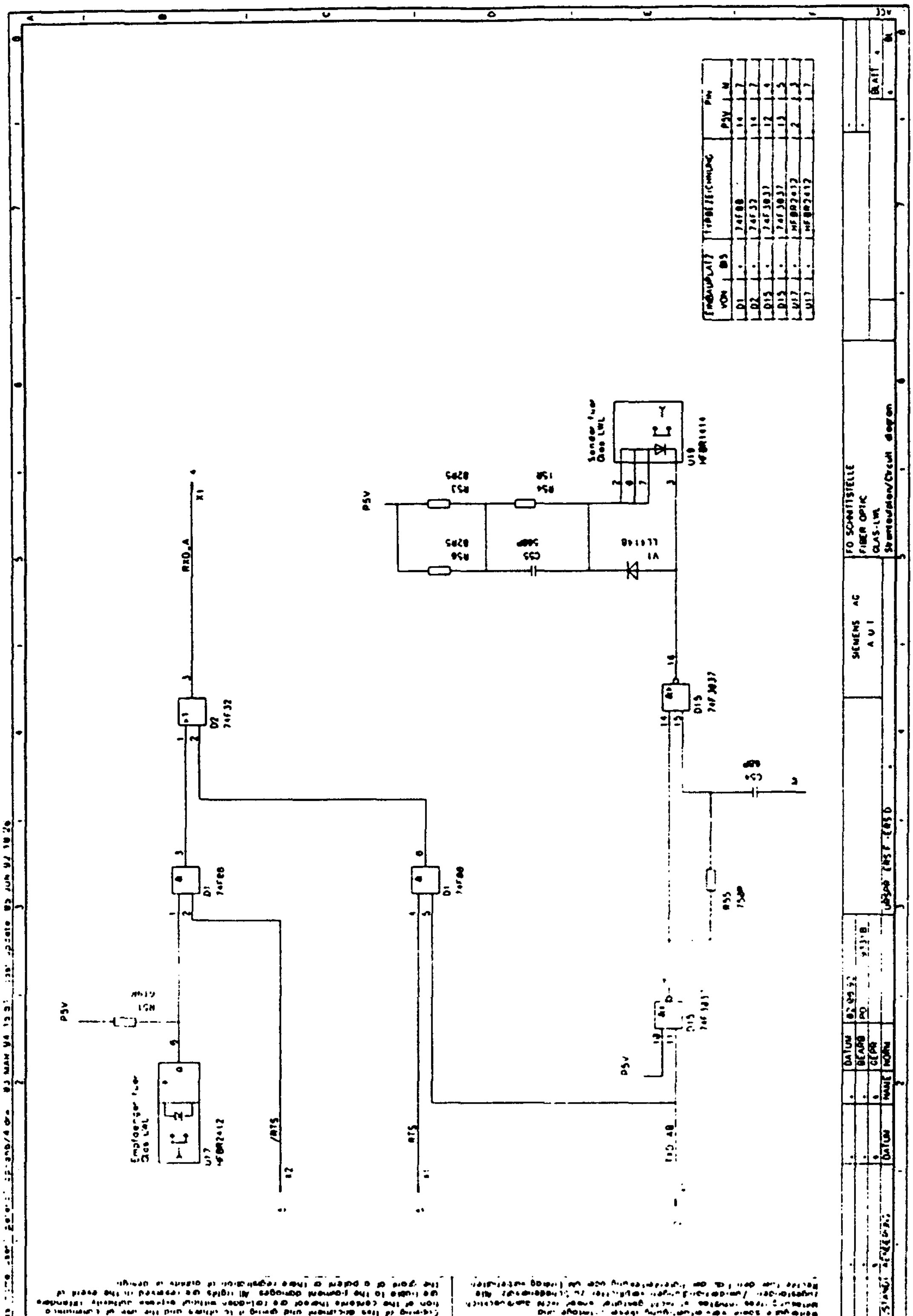
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

Für Copyright the requirements of DIN 34 in its entire wording shall apply. Copyright only pursuant to our explicit consent

Urheberschutzvermerk DIN 34, vollständige Fassung. Nachdruck nur mit unserer Genehmigung



© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006



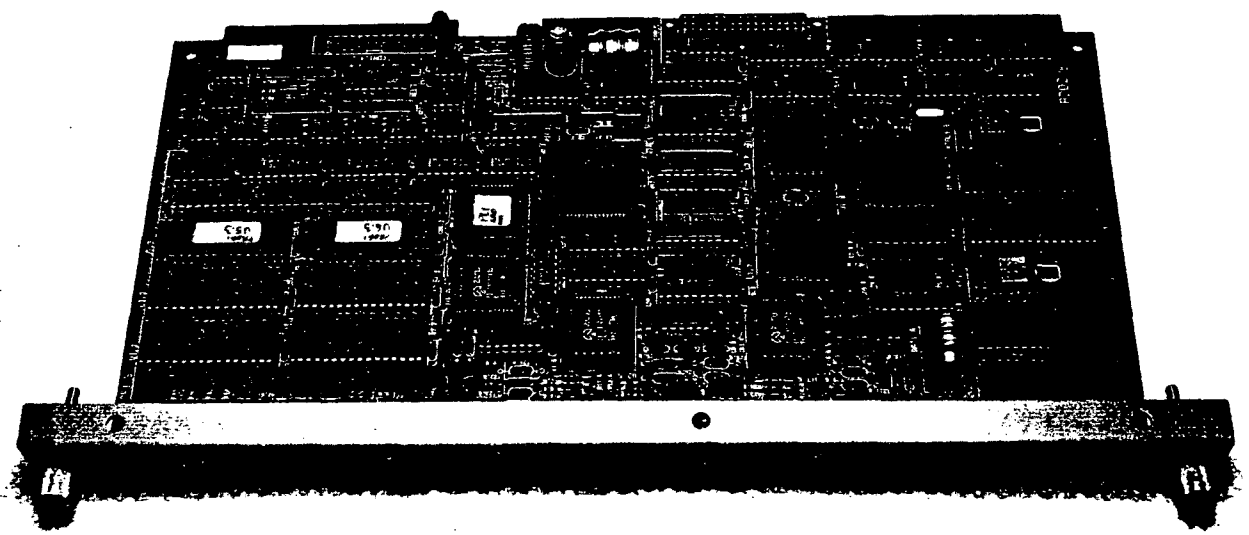
For copyright the requirements of DIN 34 in its entire wording shall apply. Copyright only pursuant to our explicit consent

Urheberschutzvermerk DIN 34, vollständige Fassung. Ein Eindruck nur mit unserer Genehmigung

ISOLUXWAT
GRUPO ISOLUX

Terminal remota programable W - 90

TARJETA DE ACCESO A LA RED DE DATOS (ARD)



CARACTERISTICAS

La tarjeta ARD se ha diseñado utilizando las tecnologías más modernas en el ámbito de los circuitos integrados. Es una tarjeta controladora de interfases de entradas y salidas así como de comunicaciones que actúa como tarjeta de control en el W90. Asume las funciones de mantenimiento de la base de datos de todas las señales de entradas/salidas y parámetros de la terminal remota, comunicaciones con el centro de control o con redes locales y actuando como controladora en las funciones de autómata programable de la W90.

Puede controlar por medio del Bus de entradas y salidas (BES) hasta 32 interfases. El BES es un bus paralelo para comunicaciones con las interfases, específicamente diseñado para proporcionar una alta inmunidad contra las interferencias electromagnéticas y una gran fiabilidad en las comunicaciones.

La ARD tiene en total seis puertas de comunicaciones serie, de propósito general, tres RS232 y tres RS485, dispone de posibilidad de conexión a los buses de campo FIP (Factory Instrumentation Protocol) y PROFIBUS (Process Field Bus). Las comunicaciones por el Bus de Entradas/Salidas se realizan mediante un controlador DMA (Direct Memory Access), lo cual proporciona una gran velocidad en el intercambio de datos con las interfases.

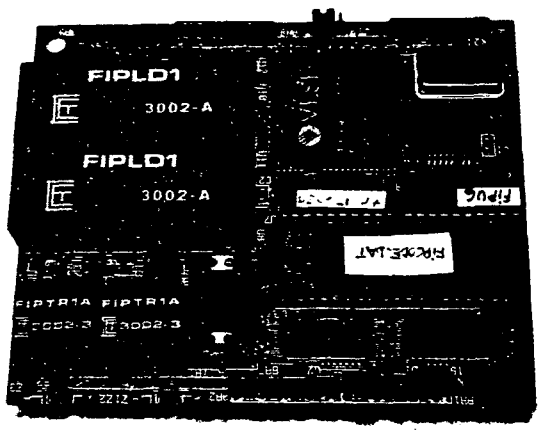
La ARD dispone de dos microprocesadores 80C186 trabajando a 20 MHz, uno de ellos se encarga de controlar las seis puertas de comunicaciones, el BES, los controladores FIP y PROFIBUS y el otro se encarga de realizar los procesos locales. Los dos procesadores se comunican por medio de una memoria de doble puerta.

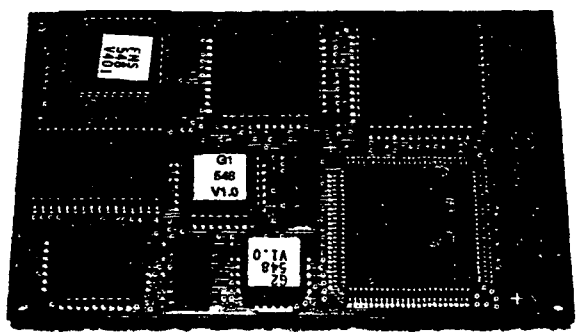
Para funcionamiento redundante se pueden instalar dos ARD en el W90, pudiendo comunicar con el centro por medio de dos redes independientes.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

- Dos procesadores 80C186, uno dedicado a las comunicaciones y otro a la conexión a procesos locales.
- Circuitos de supervisión de tensión y de los procesadores.
- 256 Kbytes de memoria EPROM y 512 Kbytes de memoria RAM en el procesador de comunicaciones y 256 Kbytes de memoria EPROM y 512 Kbytes de memoria RAM en el procesador local.
- Memoria RAM no volátil.
- Tres líneas de comunicaciones RS232C y tres RS485.
- Interfase para comunicador FIP a 1 M.b.p.s. con dos canales redundantes (opción).
- Interfase para comunicador PROFIBUS a 1.5 M.b.p.s. (opción).
- Posibilidad de comunicaciones con DPLL y codificación NRZI.
- Bancos de puentes para configurar opciones.
- LEDs para indicación de estados internos.

* Especificaciones sujetas a cambio sin previo aviso





CARACTERISTICAS FIP

- Estándar: NFC-46-601/1/2/3/4/5.
- Medio portador: Par trenzado apantallado, fibra óptica.
- Velocidad: Hasta 1 M.b.p.s.
- Longitud del bus: Hasta tres km.
- Control de acceso: Polling centralizado.
- Número de estaciones: Hasta 256.
- Servicios de aplicación: Intercambio periódico y aperiódico de variables según mecanismo productor-consumidor.

CARACTERISTICAS PROFIBUS

- Estándar: DIN 19245
- Medio portador: Par trenzado apantallado, fibra óptica.
- Velocidad: Hasta 1.5 M.b.p.s.
- Longitud del bus: 1.2 km sin repetidor.
- Control de acceso: Híbrido entre paso de testigo y polling.
- Número de estaciones: Hasta 127 sin repetidor, de las que 32 pueden ser maestras.
- Servicios de aplicación: Subconjunto de la especificación MMS (gestión de conexión, de variables, dominios y eventos).

ISOLUXWAT
GRUPO ISOLUX

ISOLUX WAT, S.A.
División de Control y Sistemas

Alcocer, 41
28041 MADRID
Tfno.: (91) 796 30 00
Fax: (91) 798 37 70

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

LIBRERIAS DEL PROGRAMA

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*****************************************************************************
*                                                                           *
*                                profibus                                   *
*                                                                           *
*****************************************************************************
*                                                                           *
*                              Softing GmbH                                 *
*                            Dingolfinger Str. 2                            *
*                            D-8000 Muenchen 80                             *
*                                                                           *
*         Copyright (C) SOFTING GmbH 1992, 1993, 1994 All Rights Reserved   *
*                                                                           *
*****************************************************************************

FILE_NAME           FMAGDL.C

PROJECT_NAME            PROFIBUS   CP5480-A1

MODULE                      FMAGDL

COMPONENT_LIBRARY       PIFL.LIB or PIFM.LIB or PIFS.LIB

AUTHOR                      Matthias Boettcher

VERSION             4.00
                    4.01
                    4.01A

DATE                01.02.94

STATUS              finished

REVIEWER            Matthias Boettcher

TESTER              Matthias Boettcher

TERMINOLOGY


FUNCTIONAL_MODULE_DESCRIPTION


This modul contains FMA7-Service-Specific-Functions which return the length
length of the Request- or Response-Datas.



FUNCTIONAL_SPECIFICATION

DESIGN_SPECIFICATION

CHANGE_NOTES

date      name    change
-----------------------------------------------------------------------------


RELATED_DOCUMENTS

============================================================================*/
#include <keywords.h>                                   /* Keyword Comments */
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
INCLUDES

#include <stdio.h>

#include <pb_type.h>
#include <pb_err.h>
#include <pb_fma7.h>


GLOBAL_DEFINES

LOCAL_DEFINES

EXPORT_TYPEDEFS

LOCAL_TYPEDEFS

FUNCTION_DECLARATIONS

EXPORT_DATA

IMPORT_DATA

LOCAL_DATA



FUNCTION static INT16 fmagdl_get_ctxt_data_len
         (
          IN    USIGN8     service,          /* Service              */
          IN    USIGN8     primitive,        /* Service-Primitive    */
          IN    USIGN8 FAR *data_ptr         /* pointer to data      */
         )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following FMA7-Context-Management-Services.

- FMA7_INITIATE
- FMA7_ABORT

possible return values:
- Data-length
-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

FUNCTION_BODY

if (service == FMA7_ABORT && primitive == REQ)
{
  return(sizeof(T_FMA_ABORT_REQ));
}
else
{
  return(0);
}
}


FUNCTION static INT16 fmagdl_get_config_data_len
         (
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
          IN    USIGN8     service,          /* Service              */
          IN    USIGN8     primitive,        /* Service-Primitive    */
          IN    USIGN8 FAR *data_ptr         /* pointer to data      */
         )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function is used to return the length of request-datas of the
of FMA7_SET_CONFIGURATION service.

possible return values:
- Data-length
-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

FUNCTION_BODY

if (primitive == REQ)
{
  return(sizeof(T_SET_CONFIGURATION_REQ));
}
else
{
  return(0);
}
}




FUNCTION static INT16 fmagdl_get_kbl_data_len
         (
          IN    USIGN8     service,          /* Service              */
          IN    USIGN8     primitive,        /* Service-Primitive    */
          IN    USIGN8 FAR *data_ptr         /* pointer to data      */
         )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following KBL-Services.

- FMA7_INIT_LOAD_KBL_LOC
- FMA7_INIT_LOAD_KBL_REM
- FMA7_LOAD_KBL_LOC
- FMA7_LOAD_KBL_REM
- FMA7_TERM_LOAD_KBL_LOC
- FMA7_TERM_LOAD_KBL_REM

- FMA7_READ_KBL_LOC
- FMA7_READ_KBL_REM

possible return values:
- Data-length


-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

FUNCTION_BODY
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
switch(service)
{
   case FMA7_LOAD_KBL_LOC:
   case FMA7_LOAD_KBL_REM:
        if (primitive == REQ) return(sizeof(T_LOAD_KBL_REQ));
        break;

   case FMA7_READ_KBL_LOC:
   case FMA7_READ_KBL_REM:
        if (primitive == REQ)
        {
          return(sizeof(T_READ_KBL_REQ));
        }
        else
        {
          T_READ_KBL_CNF FAR *rsp = (T_READ_KBL_CNF FAR*) data_ptr;

          if (rsp->desired_cr == 0) return(sizeof(T_READ_KBL_CNF));
          else                      return(sizeof(T_READ_KBL_CNF) +
                                    rsp->id.kbl_entry.kbl_status_len);
          break;
        }

   default:
        break;
}
return(0);
}



FUNCTION static INT16 fmagdl_get_s_r_value_data_len
         (
          IN    USIGN8     service,          /* Service              */
          IN    USIGN8     primitive,        /* Service-Primitive    */
          IN    USIGN8 FAR *data_ptr         /* pointer to data      */
         )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following SET- / Read-Value-Services.

- FMA7_SET_VALUE_LOC
- FMA7_SET_VALUE_REM
- FMA7_SET_BUSPARAMETER
- FMA7_READ_VALUE_LOC
- FMA7_READ_VALUE_REM
- FMA7_SET_STATISTIC_CTR
- FMA7_READ_BUSPARAMETER
- FMA7_READ_STATISTIC_CTR

possible return values:
- Data-length


-------------------------------------------------------------------------*

LOCAL_VARIABLES

FUNCTION_BODY
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
switch (service)
{
     case FMA7_SET_VALUE_LOC:
     case FMA7_SET_VALUE_REM:
          if (primitive == REQ)
          {
            T_SET_VALUE_REQ FAR *req = (T_SET_VALUE_REQ FAR*) data_ptr;
            return(sizeof (T_SET_VALUE_REQ) + req->length);
          }
          break;

     case FMA7_SET_BUSPARAMETER:
          if (primitive == REQ) return(sizeof(T_SET_BUSPARAMETER_REQ));
          break;


     case FMA7_READ_VALUE_LOC:
     case FMA7_READ_VALUE_REM:
          if (primitive == REQ)
          {
            return(sizeof(T_READ_VALUE_REQ));
          }
          else
          {
             T_READ_VALUE_CNF FAR *rsp = (T_READ_VALUE_CNF FAR*) data_ptr;
             return(sizeof(T_READ_VALUE_CNF) + rsp->length);
          }
          break;

     default:
          return(0);
          break;
}

return(0);
}




FUNCTION static INT16 fmagdl_get_ident_data_len
         (
          IN    USIGN8     service,          /* Service              */
          IN    USIGN8     primitive,        /* Service-Primitive    */
          IN    USIGN8 FAR *data_ptr         /* pointer to data      */
         )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function is used to return the length of request-datas or response-datas
of the following Ident-Services.

- IDENT_LOC
- IDENT_REM

possible return values:
- Data-length

-------------------------------------------------------------------------*
{
LOCAL_VARIABLES

FUNCTION_BODY
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
if (primitive == REQ) return(sizeof(T_IDENT_REQ));
else                  return(sizeof(T_IDENT_CNF));
}



FUNCTION static INT16 fmagdl_get_live_list_data_len
        (
         IN    USIGN8     service,          /* Service              */
         IN    USIGN8     primitive,        /* Service-Primitive    */
         IN    USIGN8 FAR *data_ptr         /* pointer to data      */
        )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas
of the Get-Live-List Service.

possible return values:
- Data-length

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES


FUNCTION_BODY

return(0);
}



FUNCTION static INT16 fmagdl_get_lsap_status_data_len
        (
         IN    USIGN8     service,          /* Service              */
         IN    USIGN8     primitive,        /* Service-Primitive    */
         IN    USIGN8 FAR *data_ptr         /* pointer to data      */
        )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following LSAP-Status Services.

- LSAP_STATUS_LOC
- LSAP_STATUS_REM

possible return values:
- Data-length

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

FUNCTION_BODY

if (primitive == REQ) return(sizeof (T_LSAP_STATUS_REQ));
else                  return(sizeof (T_LSAP_STATUS_CNF));
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
FUNCTION static INT16 fmagdl_get_fault_mngt_data_len
        (
         IN    USIGN8     service,          /* Service             */
         IN    USIGN8     primitive,        /* Service-Primitive   */
         IN    USIGN8 FAR *data_ptr         /* pointer to data     */
        )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas
of the following Fault-Management Services.

- RESET
- EXIT
- FMA7_EVENT

possible return values:
- Data-length

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

FUNCTION_BODY

return(0);
}




FUNCTION static INT16 fmagdl_get_error_data_len
        (
         IN    USIGN8     service           /* Service  */
        )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of response-error-datas
of the following FMA7-Services.

- FMA7_INITIATE
- FMA7_READ_KBL_REM
- FMA7_INIT_LOAD_KBL_REM
- FMA7_LOAD_KBL_REM
- FMA7_TERM_LOAD_KBL_REM
- FMA7_SET_VALUE_REM
- FMA7_READ_VALUE_REM
- FMA7_LSAP_STATUS_REM
- FMA7_IDENT_REM
- FMA7_READ_KBL_LOC
- FMA7_INIT_LOAD_KBL_LOC
- FMA7_LOAD_KBL_LOC
- FMA7_TERM_LOAD_KBL_LOC
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
    case FMA7_IDENT_LOC:
    case FMA7_IDENT_REM:
         if (result == POS) *data_len = fmagdl_get_ident_data_len(service,prim
         else               *data_len = fmagdl_get_error_data_len(service);
         break;

    case FMA7_EVENT:
    case FMA7_RESET:
    case FMA7_PROFIBUS_EXIT:
         if (result == POS) *data_len = fmagdl_get_fault_mngt_data_len(service
         else               *data_len = fmagdl_get_error_data_len(service);
         break;

    case FMA7_GET_LIVE_LIST:
         if (result == POS) *data_len = fmagdl_get_live_list_data_len(service,p
         else               *data_len = fmagdl_get_error_data_len(service);
         break;

    case FMA7_INITIATE:
    case FMA7_ABORT:
         if (result == POS) *data_len = fmagdl_get_ctxt_data_len(service,primit
         else               *data_len = fmagdl_get_error_data_len(service);
         break;

    case FMA7_SET_CONFIGURATION:
         *data_len = fmagdl_get_config_data_len(service,primitive,data_ptr);
         break;

    default:
         return(E_INVALID_SERVICE);
         break;
}
return(E_OK);
}
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*****************************************************************************
*
*                                  profibus
*
*****************************************************************************
*
*                                 Softing GmbH
*                               Dingolfinger Str. 2
*                               D-8000 Muenchen 80
*
*          Copyright (C) SOFTING GmbH 1992, 1993, 1994 All Rights Reserved
*
*****************************************************************************

FILE_NAME            FMSGDL.C

PROJECT_NAME             PROFIBUS  CP5480-A1

MODULE                       FMSGDL

COMPONENT_LIBRARY        PIFL.LIB or PIFM.LIB or PIFS.LIB

AUTHOR                       Matthias Boettcher

VERSION              4.00
                     4.01
                     4.01A

DATE                 01.02.94

STATUS               finished

REVIEWER             Matthias Boettcher

TESTER               Matthias Boettcher


TERMINOLOGY

FUNCTIONAL_MODULE_DESCRIPTION


This modul contains FMS-Service-Specific-Functions which return the length
length of the Request- or Response-Datas.


FUNCTIONAL_SPECIFICATION

DESIGN_SPECIFICATION

CHANGE_NOTES

date       name    change
-------------------------------------------------------------------------------
29.10.93   Boe     functions: fmsgdl_get_pi_data_len()
                              fmsgdl_get_data_len()
                   - remove statements 'case PI_STOP_LOC:' and
                                       'case PI_EOP_LOC:'.
                   - insert statements 'case PI_SET_STATE_LOC:'
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
RELATED_DOCUMENTS

================================================================================*
#include <keywords.h>

INCLUDES

#include <stdio.h>

#include <pb_type.h>
#include <pb_err.h>

#include <pb_fms.h>


GLOBAL_DEFINES

LOCAL_DEFINES

EXPORT_TYPEDEFS

LOCAL_TYPEDEFS

FUNCTION_DECLARATIONS

EXPORT_DATA

IMPORT_DATA

LOCAL_DATA




FUNCTION  static  INT16  fmsgdl_get_ctxt_data_len
          (
            IN  USIGN8     service,          /* Service               */
            IN  USIGN8     primitive,        /* Service-Primitive     */
            IN  USIGN8 FAR *data_ptr         /* Pointer to data       */
          )

/*--------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following FMS-Context-Management-Services.

- INITIATE
- ABORT

possible return values:
- Data-length

--------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

FUNCTION_BODY

switch (service)
{
  case INITIATE:
       if (primitive == REQ) return(sizeof(T_CTXT_INIT_REQ));
       else                  return(sizeof(T_CTXT_INIT_CNF));
       break;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
  case ABORT:
       if (primitive == REQ) return(sizeof(T_CTXT_ABORT_REQ));
       else                  return(0);
       break;

  default:
       return(0);
       break;
}
}



FUNCTION static INT16 fmsgdl_get_vfd_data_len
          (
            IN  USIGN8     service,          /* Service              */
            IN  USIGN8     primitive,        /* Service-Primitive    */
            IN  USIGN8 FAR *data_ptr         /* Pointer to data      */
          )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following VFD-Services.

- STATUS
- IDENTIFY
- UNSOLICITEDSTATUS
- CREATE_VFD_LOC
- VFD_SET_PHYS_STATUS_LOC

possible return values:
- Data-length

-------------------------------------------------------------------------*
{
LOCAL_VARIABLES

FUNCTION_BODY

switch (service)
{
  case STATUS:
       if (primitive == RES) return(sizeof(T_VFD_STATUS_CNF));
       break;

  case IDENTIFY:
       if (primitive == RES) return(sizeof(T_VFD_IDENTIFY_CNF));
       break;

  case UNSOLICITEDSTATUS:
       if (primitive == REQ) return(sizeof(T_VFD_UNSOL_STATUS_REQ));
       break;

  case CREATE_VFD_LOC:
       if (primitive == REQ) return(sizeof(T_VFD_CREATE_REQ));
       break;

  case VFD_SET_PHYS_STATUS_LOC:
       if (primitive == REQ) return(sizeof(T_VFD_SET_PHYS_STATUS_REQ));
       break;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
  default:
       return(0);
       break;
}

return(0);
}
```

```
FUNCTION  static INT16  fmsgdl_get_ov_data_len
          (
            IN  USIGN8      service,          /* Service              */
            IN  USIGN8      primitive,        /* Service-Primitive    */
            IN  USIGN8 FAR *data_ptr          /* Pointer to data      */
          )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following OV-Services.

- INITIATE_PUT_OV
- PUT_OV
- TERMINATE_PUT_OV
- INITIATE_LOAD_OV_LOC
- LOAD_OV_LOC
- TERMINATE_LOAD_OV_LOC
- GETOV
- OV_READ_LOC


possible return values:
- Data-length

-------------------------------------------------------------------------*
{
LOCAL_VARIABLES

FUNCTION_BODY

switch(service)
{
   case GETOV:
        if (primitive == REQ) return(sizeof(T_GET_OV_REQ));
        break;

   case INITIATE_PUT_OV:
        if (primitive == REQ) return(sizeof(T_INIT_PUT_OV_REQ));
        break;

   case PUT_OV:
        if (primitive == REQ)
        {
          T_PUT_OV_REQ FAR *req              = (T_PUT_OV_REQ FAR*) data_ptr;
          USIGN8       FAR *obj_descr_ptr    = (USIGN8       FAR*) (req+1);
          USIGN16          offset            = 0;
          USIGN16          size              = 0;
          USIGN8           i;

          for (i=0; i<req->no_of_ov_descr; i++)
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
        {
          size   += (obj_descr_ptr[offset] + 1);
          offset =  size;
        }
        return(sizeof(T_PUT_OV_REQ) + size);
      }
      break;

  case INITIATE_LOAD_OV_LOC:
      if (primitive == REQ) return(sizeof(T_INIT_LOAD_OV_REQ));
      break;

  case TERMINATE_LOAD_OV_LOC:
      if (primitive == REQ) return(sizeof(T_TERM_LOAD_OV_REQ));
      break;

  case LOAD_OV_LOC:
      if (primitive == REQ)
      {
        T_LOAD_OV_REQ     FAR *req     = (T_LOAD_OV_REQ     FAR *) data_ptr;
        T_OV_NULL_OBJECT FAR *object = (T_OV_NULL_OBJECT FAR*)
                                       &req->obj_descr.id.null_obj_descr;
        switch (object->obj_code)
        {
          case TYPE_STRUCT_OBJECT:
               return(sizeof(T_LOAD_OV_REQ) +
                      (req->obj_descr.id.ds_obj_descr.no_of_elements *
                       sizeof(T_OV_DT_LIST))
                     );
               break;

           case RECORD_OBJECT:
                return(sizeof(T_LOAD_OV_REQ) +
                       (req->obj_descr.id.r_var_obj_descr.no_of_address *
                       sizeof(USIGN32))
                      );
                break;

          case INVOCATION_OBJECT:
               return(sizeof(T_LOAD_OV_REQ) +
                      (req->obj_descr.id.pi_obj_descr.cnt_dom * sizeof(USIG
                     );
               break;


          default:
               return(sizeof(T_LOAD_OV_REQ));
               break;
        }
      }
      break;

  case OV_READ_LOC:
      if (primitive == REQ) return(sizeof(T_OV_READ_LOC_REQ));
      break;

  default:
      return(0);
      break;

}
return(0);
}
```

Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
FUNCTION static INT16 fmsgdl_get_var_data_len
        (
          IN  USIGN8      service,          /* Service                */
          IN  USIGN8      primitive,        /* Service-Primitive      */
          IN  USIGN8  FAR *data_ptr         /* Pointer to data        */
        )

/*---------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following Variable-Access-Services.

- READ
- READWITHTYPE
- WRITE
- WRITEWITHTYPE
- INFORMATIONREPORT
- INFORMATIONREPORTWITHTYPE
- PHYS_READ
- PHYS_WRITE
- DEFINEVARIABLELIST
- DELETEVARIABLELIST


possible return values:
- Data-length

---------------------------------------------------------------------------*
{
LOCAL_VARIABLES


FUNCTION_BODY

switch(service)
{
  case READ:
      if (primitive == REQ)
      {
        return(sizeof(T_VAR_READ_REQ));
      }
      else
      {
         T_VAR_READ_CNF FAR *rsp = (T_VAR_READ_CNF FAR *) data_ptr;
         return(sizeof(T_VAR_READ_CNF) + rsp->length);
      }
      break;

  case READWITHTYPE:
       if (primitive == REQ)
       {
         return(sizeof(T_VAR_READ_WITH_TYPE_REQ));
       }
       else
       {
         T_VAR_READ_WITH_TYPE_CNF FAR *rsp = (T_VAR_READ_WITH_TYPE_CNF FAR*) c
         return(sizeof(T_VAR_READ_WITH_TYPE_CNF)                +
                (sizeof(T_TYPE_DESCR) * rsp->no_of_type_descr) +
                rsp->length
               );
       }
       break;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
case WRITE:
     if (primitive == REQ)
     {
       T_VAR_WRITE_REQ FAR *req = (T_VAR_WRITE_REQ FAR *) data_ptr;
       return(sizeof(T_VAR_WRITE_REQ) + req->length);
     }
     break;

case WRITEWITHTYPE:
     if (primitive == REQ)
     {
       T_VAR_WRITE_WITH_TYPE_REQ FAR *req = (T_VAR_WRITE_WITH_TYPE_REQ FAR *
       return(sizeof(T_VAR_WRITE_WITH_TYPE_REQ)                +
              (sizeof(T_TYPE_DESCR) * req->no_of_type_descr) +
              req->length
             );
     }
     break;

case INFORMATIONREPORT:
     if (primitive == REQ)
     {
       T_VAR_INFO_RPT_REQ FAR *req = (T_VAR_INFO_RPT_REQ FAR *) data_ptr;
       return(sizeof(T_VAR_INFO_RPT_REQ) + req->length);
     }
     break;

case INFORMATIONREPORTWITHTYPE:

     if (primitive == REQ)
     {
       T_VAR_INFO_RPT_WITH_TYPE_REQ FAR *req = (T_VAR_INFO_RPT_WITH_TYPE_REQ
       return(sizeof(T_VAR_INFO_RPT_WITH_TYPE_REQ)          +
              (sizeof(T_TYPE_DESCR) * req->no_of_type_descr) +
              req->length
             );
     }
     break;

case PHYS_READ:
     if (primitive == REQ)
     {
       return(sizeof(T_VAR_PHYS_READ_REQ));
     }
     else
     {
       T_VAR_PHYS_READ_CNF FAR *rsp = (T_VAR_PHYS_READ_CNF FAR *) data_ptr;
       return(sizeof(T_VAR_PHYS_READ_CNF) + rsp->length);
     }
     break;

case PHYS_WRITE:
     if (primitive == REQ)
     {
       T_VAR_PHYS_WRITE_REQ FAR *req = (T_VAR_PHYS_WRITE_REQ FAR*) data_ptr;
       return(sizeof(T_VAR_PHYS_WRITE_REQ) + req->length);
     }
     break;

case DEFINEVARIABLELIST:
     if (primitive == REQ)
     {
       T_VAR_DEFINE_VAR_LIST_REQ FAR *req = (T_VAR_DEFINE_VAR_LIST_REQ FAR *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
          return(sizeof(T_VAR_DEFINE_VAR_LIST_REQ) +
                 (req->no_of_var * sizeof(T_ACC_SPEC))
                );
        }
        else
        {
          return(sizeof(T_VAR_DEFINE_VAR_LIST_CNF));
        }
        break;

  case DELETEVARIABLELIST:
        if (primitive == REQ) return(sizeof(T_VAR_DELETE_VAR_LIST_REQ));
        break;

  default:
        return(0);
        break;

}
return(0);
}



FUNCTION  static  INT16  fmsgdl_get_evn_data_len
          (
            IN  USIGN8       service,          /* Service              */
            IN  USIGN8       primitive,        /* Service-Primitive    */
            IN  USIGN8  FAR *data_ptr          /* Pointer to data      */
          )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following Event-Management-Services.

- EVENTNOTIFICATION
- EVENTNOTIFICATIONWITHTYPE
- ALTEREVENTCONDITIONMONITORING
- ACKNOWLEDGEEVENTNOTIFICATION

possible return values:
- Data-length

-------------------------------------------------------------------------*
{
LOCAL_VARIABLES

FUNCTION_BODY

switch (service)
{
   case EVENTNOTIFICATION:
        if (primitive == REQ)
        {
          T_EVENT_NOTIFY_REQ FAR *req = (T_EVENT_NOTIFY_REQ FAR *) data_ptr;
          return(sizeof(T_EVENT_NOTIFY_REQ) + req->data_length);
        }
        break;

   case EVENTNOTIFICATIONWITHTYPE:
        if (primitive == REQ)
        {
          T_EVENT_NOTIFY_WITH_TYPE_REQ FAR *req = (T_EVENT_NOTIFY_WITH_TYPE_RE
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
          return(sizeof(T_EVENT_NOTIFY_WITH_TYPE_REQ) + req->data_length);
        }
        break;

    case ALTEREVENTCONDITIONMONITORING:
        if (primitive == REQ) return(sizeof(T_ALT_EVN_CND_MNT_REQ));
        break;

    case ACKNOWLEDGEEVENTNOTIFICATION:
        if (primitive == REQ) return(sizeof(T_ACK_EVN_NOTIFY_REQ));
        break;

    default:
        return(0);

}
return(0);
}



FUNCTION static INT16 fmsgdl_get_dom_data_len
          (
            IN  USIGN8      service,          /* Service                */
            IN  USIGN8      primitive,        /* Service-Primitive      */
            IN  USIGN8  FAR *data_ptr         /* Pointer to data        */
          )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following Domain-Management-Services.

- INITIATEDOWNLOADSEQUENCE
- DOWNLOADSEGMENT
- TERMINATEDOWNLOADSEQUENCE
- INITIATEUPLOADSEQUENCE
- UPLOADSEGMENT
- TERMINATEUPLOADSEQUENCE
- REQUESTDOMAINDOWNLOAD
- REQUESTDOMAINUPLOAD

possible return values:
- Data-length

-------------------------------------------------------------------------*
{
LOCAL_VARIABLES

FUNCTION_BODY

switch (service)
{
    case INITIATEDOWNLOADSEQUENCE:
    case INITIATEUPLOADSEQUENCE:
    case TERMINATEUPLOADSEQUENCE:
        if (primitive == REQ) return(sizeof(T_DOM_REQ));
        break;

    case TERMINATEDOWNLOADSEQUENCE:
        if (primitive == REQ) return(sizeof(T_TERM_DNL_REQ));
        break;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
    case DOWNLOADSEGMENT:
    case UPLOADSEGMENT:
         if (primitive == REQ)
         {
           return(sizeof(T_DOM_REQ));
         }
         else
         {
           T_DNL_UPL_SEG_CNF FAR *rsp = (T_DNL_UPL_SEG_CNF FAR *) data_ptr;
           return(sizeof(T_DNL_UPL_SEG_CNF) + rsp->data_len);
         }
         break;

    case REQUESTDOMAINDOWNLOAD:
    case REQUESTDOMAINUPLOAD:
         if (primitive == REQ)
         {
           T_REQUEST_DOM_REQ FAR *req = (T_REQUEST_DOM_REQ FAR *) data_ptr;
           return(sizeof(T_REQUEST_DOM_REQ) + req->add_info_length);
         }
         break;

    default:
         return(0);
         break;

}
return(0);
}



FUNCTION  static INT16 fmsgdl_get_pi_data_len
          (
            IN  USIGN8      service,          /* Service              */
            IN  USIGN8      primitive,        /* Service-Primitive    */
            IN  USIGN8  FAR *data_ptr         /* Pointer to data      */
          )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of request-datas or response-datas
of the following Program-Invocation-Management-Services.

- CREATEPROGRAMINVOCATION
- DELETEPROGRAMINVOCATION
- START
- STOP
- RESUME
- RESET
- KILL
- PI_SET_STATE_LOC

possible return values:
- Data-length

-------------------------------------------------------------------------*
{
LOCAL_VARIABLES

FUNCTION_BODY

switch (service)
{
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
   case CREATEPROGRAMINVOCATION:
        if (primitive == REQ)
        {
          T_PI_CR8_REQ FAR *cr8_req = (T_PI_CR8_REQ FAR*) data_ptr;

          return(sizeof(T_PI_CR8_REQ) +
                 (cr8_req->cnt_dom * sizeof(T_ACC_SPEC))
                );
        }
        else
        {
          return(sizeof(T_PI_CR8_CNF));
        }
        break;

   case DELETEPROGRAMINVOCATION:
        if (primitive == REQ) return(sizeof(T_PI_DEL_REQ));
        break;

   case START:
        if (primitive == REQ) return(sizeof(T_PI_START_REQ));
        break;

   case STOP:
        if (primitive == REQ) return(sizeof(T_PI_STOP_REQ));
        break;

   case RESUME:
        if (primitive == REQ) return(sizeof(T_PI_RESUME_REQ));
        break;

   case RESET:
        if (primitive == REQ) return(sizeof(T_PI_RESET_REQ));
        break;

   case KILL:
        if (primitive == REQ) return(sizeof(T_PI_KILL_REQ));
        break;

   case PI_SET_STATE_LOC:
        if (primitive == REQ) return(sizeof(T_PI_SET_STATE_REQ));
        break;

   default:
        return(0);
        break;
}
return(0);
}



FUNCTION static INT16 fmsgdl_get_error_data_len
         (
          IN    USIGN8     service           /* Service  */
         )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to return the length of response-error-datas
of the following FMS-Services.

- INITIATE
- STATUS
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
- IDENTIFY
- READ
- WRITE
- GETOV
- READWITHTYPE
- WRITEWITHTYPE
- DEFINEVARIABLELIST
- DELETEVARIABLELIST
- INITIATEDOWNLOADSEQUENCE
- DOWNLOADSEGMENT
- TERMINATEDOWNLOADSEQUENCE
- INITIATEUPLOADSEQUENCE
- UPLOADSEGMENT
- TERMINATEUPLOADSEQUENCE
- REQUESTDOMAINDOWNLOAD
- REQUESTDOMAINUPLOAD
- CREATEPROGRAMINVOCATION
- DELETEPROGRAMINVOCATION
- START
- STOP
- RESUME
- RESET
- KILL
- ALTEREVENTCONDITIONMONITORING
- ACKNOWLEDGEEVENTNOTIFICATION
- PHYS_READ
- PHYS_WRITE
- INITIATE_PUT_OV
- PUT_OV
- TERMINATE_PUT_OV

)ossible return values:
- Data-length

-------------------------------------------------------------------------------*

LOCAL_VARIABLES

FUNCTION_BODY

switch(service)

   case  INITIATE:
         return(sizeof(T_CTXT_INIT_ERR_CNF));
         break;

   case  STATUS:
   case  IDENTIFY:
   case  READ:
   case  WRITE:
   case  GETOV:
   case  READWITHTYPE:
   case  WRITEWITHTYPE:
   case  DEFINEVARIABLELIST:
   case  DELETEVARIABLELIST:
   case  INITIATEDOWNLOADSEQUENCE:
   case  DOWNLOADSEGMENT:
   case  TERMINATEDOWNLOADSEQUENCE:
   case  INITIATEUPLOADSEQUENCE:
   case  UPLOADSEGMENT:
   case  TERMINATEUPLOADSEQUENCE:
   case  REQUESTDOMAINDOWNLOAD:
   case  REQUESTDOMAINUPLOAD:
   case  CREATEPROGRAMINVOCATION:
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
   case  DELETEPROGRAMINVOCATION:
   case  ALTEREVENTCONDITIONMONITORING:
   case  ACKNOWLEDGEEVENTNOTIFICATION:
   case  PHYS_READ:
   case  PHYS_WRITE:
   case  INITIATE_PUT_OV:
   case  PUT_OV:
         return(sizeof(T_ERROR));
         break;

   case  TERMINATE_PUT_OV:
         return(sizeof(T_OV_ERROR));
         break;

   case  START:
   case  STOP:
   case  RESUME:
   case  RESET:
   case  KILL:
         return(sizeof(T_PI_ERROR));
         break;

   default:
        return(0);
        break;

}
return(0);
}


FUNCTION extern INT16 fmsgdl_get_data_len
          (
            IN  INT16      result,          /* Service-Result       */
            IN  USIGN8     service,         /* Service              */
            IN  USIGN8     primitive,       /* Service-Primitive    */
            IN  USIGN8 FAR *data_ptr,       /* pointer to data      */
            OUT INT16      *data_len        /* length of data       */
          )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to return the data length of FMS-SERVICES

possible return values:
- Data-length

-------------------------------------------------------------------------*
{
LOCAL_VARIABLES


FUNCTION_BODY

*data_len = 0;

switch (service)
{
    case INITIATE:
    case ABORT:
    case REJECT:
         if (result == POS) *data_len = fmsgdl_get_ctxt_data_len(service,primi
         else               *data_len = fmsgdl_get_error_data_len(service);
```

©Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
        break;

    case STATUS:
    case IDENTIFY:
    case UNSOLICITEDSTATUS:
    case CREATE_VFD_LOC:
    case VFD_SET_PHYS_STATUS_LOC:
        if (result == POS) *data_len = fmsgdl_get_vfd_data_len(service,primit
        else               *data_len = fmsgdl_get_error_data_len(service);
        break;


    case INITIATEDOWNLOADSEQUENCE:
    case DOWNLOADSEGMENT:
    case TERMINATEDOWNLOADSEQUENCE:
    case INITIATEUPLOADSEQUENCE:
    case UPLOADSEGMENT:
    case TERMINATEUPLOADSEQUENCE:
    case REQUESTDOMAINDOWNLOAD:
    case REQUESTDOMAINUPLOAD:
        if (result == POS) *data_len = fmsgdl_get_dom_data_len(service,primit
        else               *data_len = fmsgdl_get_error_data_len(service);
        break;


    case EVENTNOTIFICATION:
    case EVENTNOTIFICATIONWITHTYPE:
    case ALTEREVENTCONDITIONMONITORING:
    case ACKNOWLEDGEEVENTNOTIFICATION:
        if (result == POS) *data_len = fmsgdl_get_evn_data_len(service,primit
        else               *data_len = fmsgdl_get_error_data_len(service);
        break;


    case GETOV:
    case INITIATE_PUT_OV:
    case PUT_OV:
    case TERMINATE_PUT_OV:
    case OV_READ_LOC:
    case INITIATE_LOAD_OV_LOC:
    case LOAD_OV_LOC:
    case TERMINATE_LOAD_OV_LOC:
        if (result == POS) *data_len = fmsgdl_get_ov_data_len(service,primiti
        else               *data_len = fmsgdl_get_error_data_len(service);
        break;


    case CREATEPROGRAMINVOCATION:
    case DELETEPROGRAMINVOCATION:
    case START:
    case STOP:
    case RESUME:
    case RESET:
    case KILL:
    case PI_SET_STATE_LOC:
        if (result == POS) *data_len = fmsgdl_get_pi_data_len(service,primiti
        else               *data_len = fmsgdl_get_error_data_len(service);
        break;


    case READ:
    case READWITHTYPE:
    case WRITE:
    case WRITEWITHTYPE:
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
    case INFORMATIONREPORT:
    case INFORMATIONREPORTWITHTYPE:
    case DEFINEVARIABLELIST:
    case DELETEVARIABLELIST:
    case PHYS_READ:
    case PHYS_WRITE:
         if (result == POS) *data_len = fmsgdl_get_var_data_len(service,primit
         else               *data_len = fmsgdl_get_error_data_len(service);
         break;

    default:
         return(E_INVALID_SERVICE);
         break;

}
return(E_OK);
}
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*****************************************************************************
*                                                                           *
*                                  profibus                                 *
*                                                                           *
*****************************************************************************
*                                                                           *
*                                Softing GmbH                               *
*                              Dingolfinger Str. 2                          *
*                              D-8000 Muenchen 80                           *
*                                                                           *
*         Copyright (C) SOFTING GmbH 1992, 1993, 1994 All Rights Reserved   *
*                                                                           *
*****************************************************************************

FILE_NAME           PROFI.C

PROJECT_NAME            PROFIBUS  CP5480-A1

MODULE                      PROFI

COMPONENT_LIBRARY       PIFL.LIB or PIFM.LIB or PIFS.LIB

AUTHOR                      Matthias Boettcher

VERSION             4.00
                    4.01
                    4.01A

DATE                01.02.94

STATUS              finished

REVIEWER            Matthias Boettcher

TESTER              Matthias Boettcher

TERMINOLOGY

FUNCTIONAL_MODULE_DESCRIPTION

PROFIBUS Application-Communication Interface
--------------------------------------------

The communication between host and controller is performed
using some common memory. Softing has specified the data structures and
the synchronization mechanism to be used as "Common Memory Interface"
(CMI). The driver at the host should provide the functions described
below in order to offer a standardized procedural interface to the
communication.

FUNCTIONAL_SPECIFICATION

DESIGN_SPECIFICATION

CHANGE_NOTES

RELATED_DOCUMENTS

============================================================================*/
#include <keywords.h>                                   /* keyword defines   *

INCLUDES

#include <stdio.h>
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#include <conio.h>

#include <pb_type.h>
#include <pb_err.h>

#include <pb_if.h>

#include <pb_fms.h>
#include <pb_fma7.h>
#include <cmi.h>

GLOBAL_DEFINES

LOCAL_DEFINES


EXPORT_TYPEDEFS

LOCAL_TYPEDEFS



FUNCTION_DECLARATIONS


extern INT16    cmi_h_init  (USIGN32,USIGN16);
extern INT16    cmi_h_check_controller_exists (USIGN32);
extern INT16    cmi_h_read  (T_PROFI_SERVICE_DESCR FAR*,VOID FAR*,USIGN16 FAR*
extern INT16    cmi_h_write (T_PROFI_SERVICE_DESCR FAR*,VOID FAR*,USIGN16 );

extern INT16    fmsgdl_get_data_len (INT16,USIGN8,USIGN8,USIGN8 FAR*,USIGN16*)
extern INT16    fmagdl_get_data_len (INT16,USIGN8,USIGN8,USIGN8 FAR*,USIGN16*)


EXPORT_DATA

IMPORT_DATA

LOCAL_DATA


static  USIGN32     host_dpr_base_address;

static  BOOL        reset_controller;
static  USIGN8      reset_value;

/* -- copyright and version ------------------------------------------- *
#ifdef M_I86SM
   /* -- small memory model ------------------------------------------- *
   static char copyright[] = "@(#2) PIFS.LIB (c) Copyright 1993. SOFTING GmbH.
   static char comp_vers[] = "@(#2) PIFS.LIB  V4.01A 01.02.94";
#else
   #ifdef M_I86MM
   /* -- medium memory model ------------------------------------------ *
   static char copyright[] = "@(#2) PIFM.LIB (c) Copyright 1993. SOFTING GmbH.
   static char comp_vers[] = "@(#2) PIFM.LIB  V4.01A  01.02.04";
   #endif
#endif

FUNCTION static VOID profi_reset_cp5480(VOID)

/*-----------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
This function is used to reset the PROFIBUS controller

Note: only used for CP5480A1 PROFIBUS controller

Possible return values:
- NONE

------------------------------------------------------------------------*
{
LOCAL_VARIABLES

USIGN8 FAR *reset_addr;

FUNCTION_BODY

reset_addr  = (USIGN8 FAR*) (host_dpr_base_address + HW_RESET_OFFSET);
reset_value = *reset_addr;

return;
}

FUNCTION extern INT16 init_profibus
          (
          IN  USIGN32 h_dpr_base_address, /* host base address of DPR       */
          IN  USIGN16 reserved,           /* for internal use               */
          IN  BOOL    hw_reset            /* TRUE reset PROFIBUS controller */
            )

/*----------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to reset the CP5480 PROFIBUS controller board and to
initialize the PROFIBUS-Host-Interface

Possible return values:
- E_OK                          profibus interface is initialized
- E_NO_CNTRL_RES                controller does not respond
- E_INVALID_CNTRL_TYPE_VERSION  invalid controller type
------------------------------------------------------------------------*
{
LOCAL_VARIABLES

INT16 ret_val = E_OK;

FUNCTION_BODY

if ((ret_val = cmi_h_check_controller_exists(h_dpr_base_address)) != E_OK)
    return(ret_val);


host_dpr_base_address  = h_dpr_base_address;
reset_controller       = hw_reset;

/* -- reset the PROFIBUS controller ------------------------------------- *
if (hw_reset) profi_reset_cp5480();

/* -- initialize the PROFIBUS host driver (CMI) ------------------------- *
return(cmi_h_init(h_dpr_base_address,CONTROLLER_TYPE));
}


FUNCTION extern INT16 profi_snd_req_res
            (
             IN T_PROFI_SERVICE_DESCR FAR *psd_ptr,        /* pointer to prof
         IN VOID                  FAR *data_ptr,       /* pointer to service
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
        IN BOOL                     dummy           /* dummy
           )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to send a PROFIBUS service request or a service response
to the communication.


INPUT:  psd_ptr             -> pointer to PROFI-SERVICE-DESCRIPTION-BLOCK
        data_ptr            -> pointer to service specific data

Possible return values:

- E_OK                      -> no error occured

- E_FATAL_CMI_ERROR         -> unrecoverable error in CMI
- E_FATAL_LAYER2_ERROR      -> unrecoverable error in LAYER2
- E_FATAL_LAYER7_ERROR      -> unrecoverable error in LAYER7
- E_FATAL_OS_ERROR          -> unrecoverable error in OS

- E_INVALID_LAYER           -> invalid layer
- E_INVALID_SERVICE         -> invalid service identifier
- E_INVALID_PRIMITIVE       -> invalid service primitive
- E_INVALID_COMM_REF        -> invalid communication reference
- E_RESOURCE_UNAVAILABLE    -> no resource available
- E_NO_PARALLEL_SERVICES    -> no parallel services allowed
- E_SERVICE_CONSTR_CONFLICT -> service temporarily not executable
- E_SERVICE_NOT_SUPPORTED   -> service not supported in subset

- E_NO_CNTRL_RES            -> controller does not respond  (CMI_TIMEOUT)
- E_INVALID_DATA_SIZE       -> not enough cmi data block memory
- E_INVALID_CMI_CALL        -> invalid CMI call
- E_CMI_ERROR               -> fatal error in CMI
-------------------------------------------------------------------------*/
  {
LOCAL_VARIABLES

INT16     ret_val;              /* return value   */
USIGN16   data_len;             /* size of datas  */

FUNCTION_BODY

ret_val  = E_OK;
data_len = 0;

/* --- check communication reference ---------------------------------- *
if (psd_ptr->comm_ref > MAX_COMREF) return(E_INVALID_COMM_REF);

/* --- check Service-Primitive ---------------------------------------- */
if ((psd_ptr->primitive == IND) || (psd_ptr->primitive == CON))
{
  return (E_INVALID_PRIMITIVE);
}

/* --- set the result parameter to POS if a request is send ----------- *
if (psd_ptr->primitive == REQ) psd_ptr->result = POS;


/* --- get service specific data length ------------------------------- *
switch(psd_ptr->layer)
{
  case FMS:
       ret_val = fmsgdl_get_data_len(psd_ptr->result,
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
                                        psd_ptr->service,
                                        psd_ptr->primitive,
                                        data_ptr,
                                        &data_len
                                       );
        break;

  case FMA7:
        ret_val = fmagdl_get_data_len(psd_ptr->result,
                                        psd_ptr->service,
                                        psd_ptr->primitive,
                                        data_ptr,
                                        &data_len
                                       );
        break;

  default:
        ret_val = E_INVALID_LAYER;
        break;
}


/* --- transfer service description block and data block to CMI ---------- */
if (ret_val == E_OK)
{
  return(cmi_h_write (psd_ptr,data_ptr,data_len));
}

return(ret_val);
}


FUNCTION extern INT16 profi_rcv_con_ind
            (
             IN T_PROFI_SERVICE_DESCR  FAR *psd_ptr,
          IN VOID                      FAR *data_ptr,
          IN USIGN16                   FAR *data_len
                )

/*-----------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function is used to receive a PROFIBUS service indication or a
service confirmation from the communication.


IN:         psd_ptr     -> pointer to PROFI-SERVICE-DESCRIPTION-BLOCK
INOUT:      data_ptr    -> pointer to service specific data
INOUT:      data_len    -> length of service specific data

Possible return values:

- CON_IND_RECEIVED      -> a confirmation or indication has been received
- NO_CON_IND_RECEIVED   -> no confirmation or indication has been received

- E_FATAL_CMI_ERROR     -> unrecoverable error in CMI
- E_FATAL_LAYER2_ERROR  -> unrecoverable error in LAYER2
- E_FATAL_LAYER7_ERROR  -> unrecoverable error in LAYER7
- E_FATAL_OS_ERROR      -> unrecoverable error in OS

- E_INVALID_DATA_SIZE   -> size of data block provided not sufficient

- E_CMI_ERROR           -> error occured in CMI
- E_INVALID_CMI_CALL    -> invalid CMI call
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
- E_CMI_ERROR           -> fatal error in CMI

------------------------------------------------------------------------------*/
{
LOCAL_VARIABLES

INT16    ret_val = E_OK;           /* return value */


FUNCTION_BODY

/* --- read service description block and data block from CMI -------------- *
if ((ret_val = cmi_h_read (psd_ptr,
                           data_ptr,
                           data_len
                          )) != CON_IND_RECEIVED) return(ret_val);

/* - if service-identifier equal FMA7_RESET, reset the Controller-Firmware - *
if ((psd_ptr->layer     == FMA7_USR  ) &&
    (psd_ptr->primitive == CON       ) &&
    (psd_ptr->service   == FMA7_RESET)    )
{
  /* --- initialize the communication interface ---------------------------- */
  if ((ret_val = init_profibus(host_dpr_base_address,0,reset_controller)) != E
     return(ret_val);
}

return(CON_IND_RECEIVED);
}
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

ARCHIVOS INCLUIDOS EN EL PROGRAMA

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*****************************************************************************
/*      Copyright (C) SOFTING GmbH 1992,1993,1994 All Rights Reserved        *
/*                                                                           *
/*                          COMMON MEMORY INTERFACE                          *
/*                                                                           *
/*  Filename             :   cmi.h                                           *
/*  Version              :   4.01A                                           *
/*  Date                 :   01.02.94                                        *
/*                                                                           *
/*  Description          :   This file defines the "Common Memory Interface" *
/*                           (CMI) which performs the data transfer and      *
/*                           synchronization between host and PROFIBUS       *
/*                           controller via a common memory area like a      *
/*                           dual ported RAM.                                *
/*                                                                           *
/*  date      name      change                                               *
/*  ------------------------------------------------------------------------ *
/*  15.12.93  Boe       change data structure T_CMI_DESCRIPTOR attributes     *
/*                      - h_address_mode to h_address_swap_mode              *
/*                      - c_address_mode to c_address_swap_mode              *
/*                                                                           *
/*  15.12.93  Boe       - constants SWAPP_OFF, SWAPP_ON, C_SWAP_16_32        *
/*                                  C_NO_SWAP_16_32                          *
/*                                                                           *
/*  15.12.93 Boe        - change ADRESS_MODE to ADRESS_SWAP_MODE             *
/*                                                                           *
/*  15.12.93 Boe        - change host type PC_AT_DOS to CMI_V300             *
/*                                                                           *
*****************************************************************************/


/* ========================================================================= *
/* global DEFINES                                                            *
/* ========================================================================= *

/* ------------------------------------------------------------------------- *
/* defines for CMI DESCRIPTOR BLOCK                                          *
/* ------------------------------------------------------------------------- *
/* --- communication mode ------------------------------------------------- *
#define INIT_MODE             0x00        /* initializing  mode              *
#define CONFIG_MODE           0x05        /* configuration mode              *
#define COMM_MODE             0x08        /* communication mode              *


/* --- check controller mask ---------------------------------------------- *
#define CHECK_CNTRL_MASK      0xF0E00E0F  /* check controller mask           *

/* --- controller common memory address ----------------------------------- *
#define C_BASE_ADDRESS        0x28000000  /* DPR address controller   2KB    *

/* --- interrupt cell offset ---------------------------------------------- *
#define H_INT_OFFSET          0x07FF       /* host to CP                     *
#define C_INT_OFFSET          0x07FE       /* CP to host                     *

/* --- hardware reset offset ---------------------------------------------- *
#define HW_RESET_OFFSET       0x0807       /* reset CP                       *

/* --- timeout to wait for controller response ---------------------------- *
#define CMI_TIMEOUT           0x03        /* timeout after 03 seconds        *

/* --- common memory ready masks ------------------------------------------ *
#define H_READY_MASK          0x1E2D4B87  /* host ready mask                 *
#define C_READY_MASK          0xE1D2B478  /* controller ready mask           *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* --- host identifiers ------------------------------------------------ *
#define CMI_V300              0xA0        /* host driver with CMI V3.0      *

#define HOST_TYPE             CMI_V300


/* --- controller identifier ------------------------------------------- *
#define TYPE_PROFI_SFT        0xA0        /* Softing   controller board     *
#define TYPE_CP5412A1         0xA2        /* CP5412A1  controller board     *
#define TYPE_CP5480A1         0xA4        /* CP5480A1  controller board     *
#define TYPE_PROFI_MEC        0xA6        /* PROFI_MEC controller board     *

#define CONTROLLER_TYPE       TYPE_CP5480A1

/* --- polling or interrupt mode --------------------------------------- *
#define REQ_ACK_BY_POLLING              0  /* request and ackn. by polling   *
#define REQ_BY_IR_ACK_BY_POLLING        1  /* request by IR, ackn. by polling *
#define REQ_BY_POLLING_ACK_BY_IR        2  /* request by polling, ackn. by IR *
#define REQ_ACK_BY_IR                   3  /* request and acknowledge by IR  *

#define POLL_INT_MODE                   REQ_ACK_BY_POLLING

/* --- swap mode -(important for MOTOROLA hosts)----------------------- *
#define SWAP_OFF                     0x00 /* controller can not swapp       *
                                          /* words and long words           *
#define SWAP_ON                      0x80 /* controller can swapp           *
                                          /* words and long words           *

#define C_NO_SWAP_16_32              0x00 /* controller has not to swap     *
                                          /* words and long words           *
#define C_SWAP_16_32                 0x40 /* controller has to swap words   *
                                          /* and long words if swapping is  *
                                          /* supported                      *

/* --- address mode ---------------------------------------------------- *
#define ABS_32BIT_MOTOROLA              0  /* linear absolute 32 bit Motorola *
#define ABS_16BIT_MOTOROLA              1  /* linear absolute 16 bit Motorola *
#define REL_32BIT_MOTOROLA              2  /* linear relative 32 bit Motorola *
#define REL_16BIT_MOTOROLA              3  /* linear relative 16 bit Motorola *

#define ABS_32BIT_INTEL                 8  /*  linear absolute 32 bit Intel   *
#define ABS_16BIT_INTEL                 9  /*  linear absolute 16 bit Intel   *
#define REL_32BIT_INTEL                10  /*  linear relative 32 bit Intel   *
#define REL_16BIT_INTEL                11  /*  linear relative 32 bit Intel   *
#define SEGMENT_OFFSET_INTEL           12  /*  SEGEMENT/OFFSET Intel          *

#define ADDRESS_SWAP_MODE     SEGMENT_OFFSET_INTEL + C_NO_SWAP_16_32


/* --- semaphore ------------------------------------------------------- *
#define _IDLE                        0x0  /* semaphor idle                  *
#define _BUSY                        0x1  /* semaphor busy                  *


/* ================================================================== *
/* global TYPEDEFS                                                    *
/* ================================================================== *

/* --- CMI descriptor block -------------------------------------------- *
typedef struct T_CMI_DESCRIPTOR
{
  USIGN32 h_ready_mask;       /* ready mask                             *
  USIGN32 h_base_address;     /* base address of DPR                    *
  USIGN8  h_id;               /* type and version of host software      *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
  USIGN8  h_int_enable;       /* poll / interrupt mode                        *
  USIGN8  h_address_swap_mode;/* address mode                                 *
  USIGN8  h_state;            /* communication state                          *
  USIGN32 h_param_addr;       /* parameter block address (FIELDBUS_SDB)       *
  USIGN32 h_data_addr;        /* data block address                           *
  USIGN16 h_param_size;       /* parameter block size                         *
  USIGN16 h_data_size;        /* data block size                              *
  USIGN8  h_sema;             /* semaphore                                    *
  USIGN8  h_ret_val;          /* return value                                 *
  USIGN8  h_head;             /* reserved for a circular request block buffer*
  USIGN8  h_tail;             /* reserved for a circular request block buffer*
  USIGN32 h_data_descr_addr;  /* data description address (DATA_DESCR)        *

  USIGN32 c_ready_mask;       /* ready mask                                   *
  USIGN32 c_base_address;     /* base address of DPR                          *
  USIGN8  c_id;               /* type and version of host software            *
  USIGN8  c_int_enable;       /* poll / interrupt mode                        *
  USIGN8  c_address_swap_mode;/* address mode                                 *
  USIGN8  c_state;            /* communication state                          *
  USIGN32 c_param_addr;       /* parameter block address (FIELDBUS_SDB)       *
  USIGN32 c_data_addr;        /* data block address                           *
  USIGN16 c_param_size;       /* parameter block size                         *
  USIGN16 c_data_size;        /* data block size                              *
  USIGN8  c_sema;             /* semaphore                                    *
  USIGN8  c_ret_val;          /* return value                                 *
  USIGN8  c_head;             /* reserved for a circular request block buffer*
  USIGN8  c_tail;             /* reserved for a circular request block buffer*
  USIGN32 c_data_descr_addr;  /* data description address (DATA_DESCR)        *
} T_CMI_DESCRIPTOR;


/* --- DATA description block ------------------------------------------- *
typedef struct T_DATA_DESCR
{
  USIGN8  d_id;               /* identifier to describe the data description *
  USIGN8  d_sema;             /* semaphore                                   *
  USIGN16 d_data_size;        /* data block size                             *
  USIGN32 d_data_addr;        /* data block address                          *
} T_DATA_DESCR;

/*****************************************************************************/
/*      Copyright (C) SOFTING GmbH 1992,1993,1994 All Rights Reserved        */
/*                                                                           */
/*               SOFTING SOURCE CODE KEYWORDS                                */
/*                                                                           */
/*  Filename            :   keywords.h                                       */
/*  Version             :   4.01A                                            */
/*  Date                :   01.02.94                                         */
/*  Author              :   Matthias Boettcher                               */
/*                                                                           */
/*  Description : This include file defines some keywords which are used     */
/*  for structuring and formatting the PROFIBUS       source files. These    */
/*  keywords are ignored by the the compiler.                                */
/*                                                                           */
/*****************************************************************************/

#define INCLUDES
#define GLOBAL_DEFINES
#define LOCAL_DEFINES
#define EXPORT_TYPEDEFS
#define LOCAL_TYPEDEFS
#define EXPORT_DATA
#define IMPORT_DATA
#define LOCAL_DATA
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#define FUNCTION
#define IN
#define OUT
#define INOUT
#define FUNCTIONAL_DESCRIPTION
#define ASSERTIONS
#define LOCAL_VARIABLES
#define FUNCTION_BODY
#define FUNCTION_DECLARATIONS

/*****************************************************************************
/*      Copyright (C) SOFTING GmbH 1992,1993,1994 All Rights Reserved        *
/*                                                                           *
/*              PROFIBUS ERROR DATA-TYPES and DATA-STRUCTERS                 *
/*                                                                           *
/*  Filename            :   pb_err.h                                         *
/*  Version             :   4.01A                                            *
/*  Date                :   01.02.94                                         *
/*  Author              :   Matthias Boettcher                               *
/*                                                                           *
/*  Description : This file contains the exported PROFIBUS ABORT, REJECT     *
/*                and ERROR data types and data structures and the           *
/*                according reason codes.                                    *
/*                                                                           *
/*****************************************************************************

/*****************************************************************************
/*************   ABORT REASON CODES                    *********************
/*****************************************************************************

/* --- USER abort reasons ------------------------------------------------- *
#define USR_ABT_RC1      0    /* disconnected by user                        *
#define USR_ABT_RC2      1    /* version object dictionary incompatible      *
#define USR_ABT_RC3      2    /* password error                              *
#define USR_ABT_RC4      3    /* profile number incompatible                 *
#define USR_ABT_RC5      4    /* limited services permitted                  *
#define USR_ABT_RC6      5    /* ov loading interacting (ov is beeing loaded)*

/* --- FMS abort reasons -------------------------------------------------- *
#define FMS_ABT_RC1      0   /* FMS-KBL error (KBL entry invalid)            *
#define FMS_ABT_RC2      1   /* user error (protocol violation by user)      *
#define FMS_ABT_RC3      2   /* FMS-PDU error (invalid FMS PDU received)     *
#define FMS_ABT_RC4      3   /* connection state conflict LLI                *
#define FMS_ABT_RC5      4   /* LLI error                                    *
#define FMS_ABT_RC6      5   /* PDU size exceeds maximum PDU size            *
#define FMS_ABT_RC7      6   /* feature not supported                        *
#define FMS_ABT_RC8      7   /* invoke id error in service response          *
#define FMS_ABT_RC9      8   /* max services overflow                        *
#define FMS_ABT_RC10     9   /* connection state conflict FMS (Initiate.req) *
#define FMS_ABT_RC11    10   /* service error (res != ind or con != req)     *
#define FMS_ABT_RC12    11   /* invoke id error in service request           *
#define FMS_ABT_RC13    12   /* FMS is disabled                              *

/* --- FMA7 abort reasons ------------------------------------------------- *
#define FMA_ABT_RC1      0   /* FMA7-KBL error                               *
#define FMA_ABT_RC2      1   /* user error                                   *
#define FMA_ABT_RC3      2   /* FMA7-PDU error                               *
#define FMA_ABT_RC4      3   /* connection state conflict LLI                *
#define FMA_ABT_RC5      4   /* LLI error                                    *
#define FMA_ABT_RC6      5   /* PDU size                                     *
#define FMA_ABT_RC7      6   /* feature not supported                        *
#define FMA_ABT_RC8      7   /* response error                               *
#define FMA_ABT_RC9      8   /* max services overflow                        *
#define FMA_ABT_RC10     9   /* connection state conflict FMA                *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#define FMA_ABT_RC11     10   /* service error                                  *

/* --- LLI abort reasons ------------------------------------------------- *
#define LLI_ABT_RC1       0   /* LLI context check neg., remote context in AD *
#define LLI_ABT_RC2       1   /* invalid LLI PDU during associate or abort     *
#define LLI_ABT_RC3       2   /* invalid LLI PDU during data transfer phase    *
#define LLI_ABT_RC4       3   /* unknown or invalid LLI PDU received           *
#define LLI_ABT_RC5       4   /* DTA_ACK_PDU received and SAC = 0              *
#define LLI_ABT_RC6       5   /* max no of parallel services exceeded (by PDU)*
#define LLI_ABT_RC7       6   /* unkown invoke id                              *
#define LLI_ABT_RC8       7   /* priority error                                *
#define LLI_ABT_RC9       8   /* local error at remote station                 *
#define LLI_ABT_RC10      9   /* timer 1 expired (associate)                   *
#define LLI_ABT_RC11     10   /* timer 3 expired (supervision of connection)   *
#define LLI_ABT_RC12     11   /* receive timer expired                         *
#define LLI_ABT_RC13     12   /* error while activating LSAP (state in AD)     *
#define LLI_ABT_RC14     13   /* illegal FDL prim. during ASS or ABT (see AD)  *
#define LLI_ABT_RC15     14   /* illegal FDL prim. in data transfer  (see AD)  *
#define LLI_ABT_RC16     15   /* unkown FDL primitive                          *
#define LLI_ABT_RC17     16   /* unkown LLI primitive                          *
#define LLI_ABT_RC18     17   /* illegal LLI prim. during ASS or ABT (see AD)  *
#define LLI_ABT_RC19     18   /* illegal LLI prim. in data transfer  (see AD)  *
#define LLI_ABT_RC20     19   /* invalid KBL entry                             *
#define LLI_ABT_RC21     20   /* conflict during associate                     *
#define LLI_ABT_RC22     21   /* procedural error on cyclic connection         *
#define LLI_ABT_RC23     22   /* max no of parallel services exceeded (by FMS)*
#define LLI_ABT_RC24     23   /* KBL beeing loaded, LLI disabled               *
#define LLI_ABT_RC25     24   /* confirm / indication mode error               *
#define LLI_ABT_RC26     25   /* illegal FMA1/2 primitive received             *
#define LLI_ABT_RC27     26   /* illegal FMS service on cyclic connection      *
#define LLI_ABT_RC28     27   /* FMS PDU too large on cyclic connection        *
#define LLI_ABT_RC29     28   /* resource error during associate               *
#define LLI_ABT_RC30     29   /* resource error in data transfer phase         *
#define LLI_ABT_RC31     30   /* resource error during abort                   *
#define LLI_ABT_RC32     31   /* LLI state error                               *
#define LLI_ABT_RC33     32   /* timer error                                   *
#define LLI_ABT_RC34     33   /* resource transfer to FDL failed               *

/* --- LLI abort details ------------------------------------------------- *
#define LLI_ABT_AD1       0   /* error while loading update buffer             *
#define LLI_ABT_AD2       1   /* error while activating poll list entry        *
#define LLI_ABT_AD3       2   /* error while deactivating poll list entry      *
#define LLI_ABT_AD4       3   /* transmit error (SDA.con)                      *
#define LLI_ABT_AD5       4   /* transmit error (CSRD.con)                     *
#define LLI_ABT_AD6       5   /* transmit error (SRD.con)                      *
#define LLI_ABT_AD7       6   /* receive error (CSRD.con)                      *


/*****************************************************************************
/*************   REJECT PDU TYPES and REASON CODES       *********************
/*****************************************************************************

/* --- PDU types --------------------------------------------------------- *
#define CONFIRMED_REQUEST_PDU       1
#define CONFIRMED_RESPONSE_PDU      2
#define UNCONFIRMED_PDU             3
#define UNKNOWN_PDU_TYPE            4

/* --- reason codes ------------------------------------------------------ *
#define REJ_RC0  0           /* other than RC1 to RC6                        *
#define REJ_RC1  1           /* invoke id exists already                     *
#define REJ_RC2  2           /* max services overflow (max. SCC exceeded)    *
#define REJ_RC3  3           /* feature not supported - connection oriented  *
#define REJ_RC4  4           /* feature not supported - connectionless       *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#define REJ_RC5  5              /* PDU size exceeds maximum PDU size allowed   *
#define REJ_RC6  6              /* user error on connectionless relation       *


/*****************************************************************************
/*************   FMS ERROR CLASSES and ERROR CODES       *********************
/*****************************************************************************

/* The error class is encoded in the high byte of the 16-bit-result,         *
/* the error code in the low byte.                                           *

#define E_INIT                              0x0000
#define E_INIT_OTHER                        0x0000
#define E_INIT_MAX_PDU_SIZE_INSUFF          0x0001
#define E_INIT_FEAT_NOT_SUPPORTED           0x0002
#define E_INIT_OV_VERSION_INCOMP            0x0003
#define E_INIT_USER_DENIED                  0x0004
#define E_INIT_PASSWORD_ERROR               0x0005
#define E_INIT_PROFILE_NUMB_INCOMP          0x0006


#define E_VFD_STATE_OTHER                   0x0100

#define E_APPLICATION_OTHER                 0x0200
#define E_APPLICATION_UNREACHABLE           0x0201

#define E_DEF_OTHER                         0x0300
#define E_DEF_OBJ_UNDEF                     0x0301
#define E_DEF_OBJ_ATTR_INCONSIST            0x0302
#define E_DEF_OBJECT_ALREADY_EXISTS         0x0303

#define E_RESOURCE_OTHER                    0x0400
#define E_RESOURCE_MEM_UNAVAILABLE          0x0401

#define E_SERV_OTHER                        0x0500
#define E_SERV_OBJ_STATE_CONFLICT           0x0501
#define E_SERV_PDU_SIZE                     0x0502
#define E_SERV_OBJ_CONSTR_CONFLICT          0x0503
#define E_SERV_PARAM_INCONSIST              0x0504
#define E_SERV_ILLEGAL_PARAM                0x0505

#define E_ACCESS_OTHER                      0x0600
#define E_ACCESS_OBJ_INVALIDATED            0x0601
#define E_ACCESS_HARDWARE_FAULT             0x0602
#define E_ACCESS_OBJ_ACCESS_DENIED          0x0603
#define E_ACCESS_ADDR_INVALID               0x0604
#define E_ACCESS_OBJ_ATTR_INCONST           0x0605
#define E_ACCESS_OBJ_ACCESS_UNSUPP          0x0606
#define E_ACCESS_OBJ_NON_EXIST              0x0607
#define E_ACCESS_TYPE_CONFLICT              0x0608
#define E_ACCESS_NAME_ACCESS_UNSUP          0x0609


#define E_OV_OTHER                          0x0700
#define E_OV_NAME_LEN_OVERFLOW              0x0701
#define E_OV_OVERFLOW                       0x0702
#define E_OV_WRITE_PROTECT                  0x0703
#define E_OV_EXTENSION_LEN_OVERFLOW         0x0704
#define E_OV_OBJ_DESCR_OVERFLOW             0x0705
#define E_OV_OPERAT_PROBLEM                 0x0706

#define E_OTHER                             0x0800
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*****************************************************************************
/*************   FMA7 ERROR CLASSES and ERROR CODES      *********************
/*****************************************************************************

/* The error class is encoded in the high byte of the 16-bit-result,         *
/* the error code in the low byte.                                           *

#define E_FMA7_INIT_OTHER                0x0000
#define E_FMA7_INIT_MAX_PDU_SIZE_INSUFF  0x0001
#define E_FMA7_INIT_FEAT_NOT_SUPPORTED   0x0002
#define E_FMA7_INIT_USER_DENIED          0x0003

#define E_FMA7_APPLICATION_OTHER         0x0100
#define E_FMA7_APPLICATION_UNREACHABLE   0x0101

#define E_FMA7_RESOURCE_OTHER            0x0200
#define E_FMA7_RESOURCE_MEM_UNAVAILABLE  0x0201

#define E_FMA7_SERV_OTHER                0x0300
#define E_FMA7_SERV_OBJ_STATE_CONFLICT   0x0301
#define E_FMA7_SERV_OBJ_CONSTR_CONFLICT  0x0302
#define E_FMA7_SERV_PARAM_INCONSIST      0x0303
#define E_FMA7_SERV_ILLEGAL_PARAM        0x0304
#define E_FMA7_SERV_PERM_INTERN_FAULT    0x0305

#define E_FMA7_USR_OTHER                 0x0400
#define E_FMA7_USR_DONT_WORRY_BE_HAPPY   0x0401
#define E_FMA7_USR_MEM_UNAVAILABLE       0x0402

#define E_FMA7_ACCESS_OTHER              0x0500
#define E_FMA7_ACCESS_OBJ_ACC_UNSUP      0x0501
#define E_FMA7_ACCESS_OBJ_NON_EXIST      0x0502
#define E_FMA7_ACCESS_OBJ_ACCESS_DENIED  0x0503
#define E_FMA7_ACCESS_HARDWARE_FAULT     0x0504
#define E_FMA7_ACCESS_TYPE_CONFLICT      0x0505

#define E_FMA7_KBL_OTHER                 0x0600
#define E_FMA7_KBL_INVALID_ENTRY         0x0601
#define E_FMA7_KBL_NO_KBL_ENTRY          0x0602
#define E_FMA7_KBL_INVALID_KBL           0x0603
#define E_FMA7_KBL_NO_KBL                0x0604
#define E_FMA7_KBL_WRITE_PROTECTED       0x0605

#define E_FMA7_OTHER                     0x0700

/*****************************************************************************
/************* ADDITIONAL ERROR CODES          ********************************
/*****************************************************************************

#define NO_ADD_DETAIL                      0x00

#define AD_LLI_UNEXP_FDL_OR_TIMER_EVT      0x01
#define AD_LLI_LSAP_ACT_FAILED             0x02
#define AD_LLI_POLL_LIST_LOAD_FAILED       0x03
#define AD_LLI_PUT_RESRC_FAILED            0x04
#define AD_LLI_FDL_RESET_FAILED            0x05

#define AD_FMA7_TO_MANY_KBL_ENTRIES        0x10
#define AD_FMA7_COMM_REF_NOT_ALLOWED       0x11
#define AD_FMA7_TO_MANY_PARALLEL_SERV      0x12
#define AD_FMA7_ILLEGAL_FMS_PDU_SIZE       0x13


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

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/************** PROFIBUS LLI FAULT INDICATIONS  ********************************
/************** PROFIBUS FMA7 EVENTS            ********************************
/************** PROFIBUS FMA2 EVENTS            ********************************
/*****************************************************************************

#define LLI_FMA7_RC1      1   /* error while activating SAP (result in AD)     *
#define LLI_FMA7_RC2      2   /* error while deactivating SAP (result in AD)   *
#define LLI_FMA7_RC3      3   /* error during FDL_XXX_UPDATE (result in AD)    *
#define LLI_FMA7_RC4      4   /* error while activ. poll list entry (see AD)   *
#define LLI_FMA7_RC5      5   /* error while deactiv. poll list entry (see AD)*
#define LLI_FMA7_RC6      6   /* illegal FDL prim. during ASS or ABT (see AD)  *
#define LLI_FMA7_RC7      7   /* illegal FDL prim. in data transfer (see AD)   *
#define LLI_FMA7_RC8      8   /* unknown FDL primitive                         *
#define LLI_FMA7_RC9      9   /* unknown LLI primitive                         *
#define LLI_FMA7_RC10    10   /* illegal LLI prim. during ASS or ABT (see AD)  *
#define LLI_FMA7_RC11    11   /* illegal LLI prim. in data transfer (see AD)   *
#define LLI_FMA7_RC12    12   /* SDA failed (result in AD)                     *
#define LLI_FMA7_RC13    13   /* CSRD transmission failed (result in AD)       *
#define LLI_FMA7_RC14    14   /* SRD failed (result in AD)                     *
#define LLI_FMA7_RC15    15   /* SDN failed (result in AD)                     *
#define LLI_FMA7_RC16    16   /* CSRD reception failed (result in AD)          *
#define LLI_FMA7_RC17    17   /* poll list loading failed (result in AD)       *
#define LLI_FMA7_RC18    18   /* timer 1 expired (associate)                   *
#define LLI_FMA7_RC19    19   /* timer 2 expired (abort)                       *
#define LLI_FMA7_RC20    20   /* poll list deactivation failed (result in AD)  *
#define LLI_FMA7_RC21    21   /* no matching comm. reference found             *
#define LLI_FMA7_RC22    22   /* illegal FMA1/2 prim. (see AD)                 *
#define LLI_FMA7_RC23    23   /* illegal FDL prim. during LLI start (see AD)   *
#define LLI_FMA7_RC24    24   /* confirm / indication mode error               *
#define LLI_FMA7_RC25    25   /* timer error                                   *
#define LLI_FMA7_RC26    26   /* resource transfer to FDL failed               *
#define LLI_FMA7_RC27    27   /* resource error during associate               *
#define LLI_FMA7_RC28    28   /* resource error in data transfer phase         *
#define LLI_FMA7_RC29    29   /* resource error during abort                   *
#define LLI_FMA7_RC30    30   /* LLI state error                               *


#define FMA2_FAULT_ADDRESS          0x01     /* duplicate address recognized *
#define FMA2_FAULT_TRANSCEIVER      0x02     /* transceiver error occured    *
#define FMA2_FAULT_TTO              0x03     /* time out on bus detected     *
#define FMA2_FAULT_SYN              0x04     /* no receiver synchronization  *
#define FMA2_FAULT_OUT_OF_RING      0x05     /* station out of ring          *
#define FMA2_GAP_EVENT              0x06     /* new station in ring          *


/*****************************************************************************
/*************   PROFIBUS INTERFACE ERRORS                 *******************
/*****************************************************************************

#define   E_FATAL_CMI_ERROR              5 /* unrecoverable error in layer2
#define   E_FATAL_LAYER2_ERROR           6 /* unrecoverable error in layer2
#define   E_FATAL_LAYER7_ERROR           7 /* unrecoverable error in layer7
#define   E_FATAL_OS_ERROR               8 /* unrecoverable error in OS

#define   E_LOADER_ERROR                 9 /* download error

#define   E_NO_CNTRL_RES                10 /* controller does not respond
#define   E_INVALID_CNTRL_TYPE_VERSION  11 /* inv. controller type or SW vers.
#define   E_INVALID_LAYER               12 /* invalid layer
#define   E_INVALID_SERVICE             13 /* invalid service identifier
#define   E_INVALID_PRIMITIVE           14 /* invalid service primitive
#define   E_INVALID_DATA_SIZE           15 /* not enough cmi data block memory
#define   E_INVALID_COMM_REF            16 /* invalid communication reference
#define   E_INVALID_FMS_COMM_REF        17 /* invalid FMS comm. reference
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#define   E_INVALID_FMA_COMM_REF        18 /* invalid FMA7 comm. reference
#define   E_INVALID_CMI_CALL            19 /* invalid CMI call
#define   E_CMI_ERROR                   20 /* error occured in CMI
#define   E_RESOURCE_UNAVAILABLE        21 /* no resource available
#define   E_NO_PARALLEL_SERVICES        22 /* no parallel services allowed
#define   E_SERVICE_CONSTR_CONFLICT     23 /* serv. tempor. not executable
#define   E_SERVICE_NOT_SUPPORTED       24 /* service not supported
#define   E_SERVICE_NOT_EXECUTABLE      25 /* service not executable



/*****************************************************************************
/*************   PROFIBUS ERROR DATA STRUCTURES  *****************************
/*****************************************************************************

#define MAX_ERROR_DESCR_LENGTH    _NAME_LENGTH(ERROR_DESCR_LENGTH)

/* --- standard error data structure ------------------------------------- *
typedef struct T_ERROR
{
  USIGN16 class_code;                                /* class and code        *
  INT16   add_detail;                                /* additional detail     *
  STRINGV add_description[MAX_ERROR_DESCR_LENGTH]; /* additional description *
} T_ERROR;


/* --- PI error data structure ------------------------------------------- *
typedef  struct T_PI_ERROR
{
   T_ERROR   error;                                  /*  standard error type   *
   USIGN8    pi_state;                               /*  pi state              *
   USIGN8    dummy;                                  /*  alignment             *
} T_PI_ERROR;


/* --- PI-LOC error data structure --------------------------------------- *
typedef  struct T_PI_LOC_ERROR
{
  T_ERROR    error;                                 /*  standard error type    */
  USIGN8     pi_state;                              /*  pi state               */
  USIGN8     dummy;                                 /*  alignment              */
  USIGN32    vfd_number;                            /*  vfd number             */
} T_PI_LOC_ERROR;


/* --- OV error data structure ------------------------------------------- *
typedef struct T_OV_ERROR
{
   T_ERROR  error;                                   /* standard error type    *
   USIGN16  index;                                   /* error index            *
} T_OV_ERROR;


/* --- source OV error data structure ------------------------------------ *
typedef struct T_SRC_OV_ERROR
{
   T_ERROR  error;                                   /* standard error type    *
   USIGN32  vfd_number;                              /* vfd number             *
   USIGN16  index;                                   /* error index            *
} T_SRC_OV_ERROR;


/* --- VFD error data structure ------------------------------------------ *
typedef struct T_VFD_ERROR
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
{
   T_ERROR  error;                                  /* standard error type     *
   USIGN32  vfd_number;                             /* vfd number              *
} T_VFD_ERROR;


/* --- KBL error data structure ------------------------------------------- *
typedef struct T_KBL_ERROR
{
   T_ERROR  error;                                  /* standard error type     *
   USIGN16  error_cr;                               /* error cr                *
} T_KBL_ERROR;


/*****************************************************************************/
/*  Copyright (C) SOFTING GmbH 1992, 1993, 1994 All Rights Reserved          */
/*                                                                           */
/*           PROFIBUS FMA7 - DATA-TYPES and DATA-STRUCTURES                  */
/*                                                                           */
/*  Filename            :   pb_fma7.h                                        */
/*  Version             :   4.01A                                            */
/*  Date                :   01.02.94                                         */
/*  Author              :   Matthias Boettcher                               */
/*                                                                           */
/*  Description :                                                            */
/*                                                                           */
/*  This headerfile contains the exported Data-Types and Data-Structures     */
/*  of the PROFIBUS FMA7-SERVICES and FMA7 Objects                           */
/*                                                                           */
/*  CHANGE_NOTES                                                             */
/*                                                                           */
/*  date      name      change                                               */
/*  ------------------------------------------------------------------------ */
/*  14.05.93  Boe       insert copyright                                     */
/*  14.05.93  Boe       remove literally NUM_FMA7_SERVICES                   */
/*  15.09.93  Boe       change data structure T_SET_CONFIGURATION_REQ        */
/*  22.10.93  Boe       change MAX_KBL_EXTENSION_LENGTH calculation          */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/

/*****************************************************************************/
/*************   FMA7-STRING-LENGTH-CONSTANTS            *******************/
/*****************************************************************************/

#define MAX_KBL_EXTENSION_LENGTH        _NAME_LENGTH(KBL_EXTENSION_LENGTH+1)
#define MAX_KBL_SYMBOL_LENGTH           _NAME_LENGTH(KBL_SYMBOL_LENGTH)
#define MAX_IDENT_STRING_LENGTH         _NAME_LENGTH(IDENT_STRING_LENGTH)


/*****************************************************************************/
/*************   FMA7-SERVICE-IDENTIFIERS                *******************/
/*****************************************************************************/

/*---FMA7-Service-Identifiers ----------------------------------------------  */
#define FMA7_INITIATE               0
#define FMA7_READ_KBL_REM           1
#define FMA7_INIT_LOAD_KBL_REM      2
#define FMA7_LOAD_KBL_REM           3
#define FMA7_TERM_LOAD_KBL_REM      4
#define FMA7_SET_VALUE_REM          5
#define FMA7_READ_VALUE_REM         6
#define FMA7_LSAP_STATUS_REM        7
#define FMA7_IDENT_REM              8
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#define FMA7_ABORT                38

#define FMA7_READ_KBL_LOC         11
#define FMA7_INIT_LOAD_KBL_LOC    12
#define FMA7_LOAD_KBL_LOC         13
#define FMA7_TERM_LOAD_KBL_LOC    14
#define FMA7_SET_VALUE_LOC        15
#define FMA7_READ_VALUE_LOC       16
#define FMA7_LSAP_STATUS_LOC      17
#define FMA7_IDENT_LOC            18

#define FMA7_EVENT                19
#define FMA7_RESET                20

#define FMA7_PROFIBUS_EXIT        21
#define FMA7_SET_BUSPARAMETER     22
#define FMA7_SET_STATISTIC_CTR    23
#define FMA7_READ_BUSPARAMETER    24
#define FMA7_READ_STATISTIC_CTR   25
#define FMA7_GET_LIVE_LIST        26
#define FMA7_SET_CONFIGURATION    27



/*****************************************************************************/
/*************   CONSTANTS IN KBL ENTRIES                 *******************/
/*****************************************************************************/

/* --- Special values of segment address ---------------------------------- */
#define NO_SEGMENT              255

/* --- Special values of station address ---------------------------------- */
#define GLOBAL_ADDRESS          127
#define ALL                     255

/* --- Special LSAP values ------------------------------------------------ */
#define BRCT_SAP                 63
#define DEFAULT_SAP             128

/* --- Connection-Type ---------------------------------------------------- */
#define MMAZ                   0x00
#define MSAZ                   0x01
#define MSAZ_SI                0x05
#define MSZY                   0x03
#define MSZY_SI                0x07
#define BRCT                   0x08
#define MULT                   0x0A

/* --- LLI-SAP ------------------------------------------------------------ */
#define FMS_SAP                  0
#define FMA7_SAP                 1

/* --- Connection Attribute ----------------------------------------------- */
#define D_CONN                   0
#define I_CONN                   1
#define O_CONN                   2



/*****************************************************************************/
/*************   STATES OF THE LLI STATE MACHINE        *******************/
/*****************************************************************************/

#define CLOSED                   0
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* --- Associate ------------------------------------------------------ */
#define ASS_REQ_ACT_SAP              1
#define ASS_SEND_UPDATE              2
#define ASS_POLL_LIST_ON             3
#define ASS_REQ_WAIT_FOR_CON         4
#define ASS_REQ_WAIT_FOR_RES         5
#define ASS_POLL_LIST_OFF            6

#define ASS_RES_DEACT_SAP            7
#define ASS_RES_ACT_SAP              8
#define ASS_WAIT_LOC_RES             9
#define ASS_SEND_RES_PDU            10
#define ASS_REPLY_UPDATE            11

/* --- Abort ---------------------------------------------------------- */
#define ABT_UPDATE                  12
#define ABT_POLL_LIST_ON            13
#define ABT_SEND_PDU                14
#define ABT_POLL_LIST_OFF           15
#define ABT_ACT_SAP                 16
#define ABT_DEACT_SAP               17

/* --- Open Connection ------------------------------------------------ */
#define OPEN                        18
#define OP_POLL_LIST_ON             19
#define OP_POLL_LIST_OFF            20

/* --- DTC Requester (acyclic) ---------------------------------------- */
#define DTC_WAIT_FOR_REQ             0
#define DTC_SEND_UPDATE              1
#define DTC_WAIT_FOR_CON             2
#define DTC_WAIT_FOR_RES             3

/* --- DTC Responder  (acyclic) --------------------------------------- */
#define DTC_WAIT_FOR_REQ_PDU         0
#define DTC_WAIT_FOR_LOC_RES         1
#define DTC_REPLY_UPDATE             2
#define DTC_SEND_RES                 3

/* --- DTC Requester (cyclic) ----------------------------------------- */
#define DTZ_WAIT_FOR_REQ             0
#define DTZ_SEND_UPDATE              1
#define DTZ_WAIT_FOR_CON             2
#define DTZ_WAIT_FOR_RES             3
#define DTZ_WAIT_FOR_CYC_RES         4

/* --- DTC Responder  (cyclic) ---------------------------------------- */
#define DTZ_WAIT_FOR_REQ_PDU         0
#define DTZ_WAIT_FOR_FMS_RES         1
#define DTZ_REPLY_UPDATE             2
#define DTZ_SEND_RES                 3

 * --- DTA Requester ------------------------------------------------- */
#define DTA_WAIT_FOR_REQM            0
#define DTA_SEND_UPDATE              1
 define DTA_WAIT_FOR_CON             2

#define DTA_WAIT_FOR_REQS            3
 define DTA_REPLY_UPDATE             4
 define DTA_WAIT_FOR_IND             5

 * --- DTA Responder ------------------------------------------------- */
 define DTA_WAIT_FOR_REQ_PDU         0
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#define DTA_H_WAIT_FOR_BUFFER_FREE  1
#define DTA_H_REPLY_UPDATE          2
#define DTA_H_SEND_ACK              3

#define DTA_L_WAIT_FOR_BUFFER_FREE  4
#define DTA_L_UPDATE                5
#define DTA_L_SEND_ACK              6

/* --- IDLE Requester ------------------------------------------------------ */
#define IDLE_WAIT_FOR_REQM          0
#define IDLE_WAIT_FOR_CON           1
#define IDLE_SEND_UPDATE            2

#define IDLE_WAIT_FOR_REQS          3
#define IDLE_REPLY_UPDATE           4
#define IDLE_WAIT_FOR_IND           5

/* --- DTU Requester ------------------------------------------------------- */
#define CONLS_REQ                   0
#define CONLS_WAIT_FOR_CON          1

/* --- DTU Receiver -------------------------------------------------------- */
#define CONLS_SERVER                0
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*****************************************************************************/
/*************   IDENTIFIERS FOR SET/READ VALUE SERVICE   ********************/
/*****************************************************************************/

/* --- FDL-Variable-Identifiers ------------------------------------------ */
#define ID_TS                       1
#define ID_BAUD_RATE                2
#define ID_MEDIUM_RED               3
#define ID_HW_RELEASE               4
#define ID_SW_RELEASE               5
#define ID_TSL                      6
#define ID_MIN_TSDR                 7
#define ID_MAX_TSDR                 8
#define ID_TQUI                     9
#define ID_TSET                    10
#define ID_TTR                     11
#define ID_G                       12
#define ID_IN_RING_DESIRED         13
#define ID_HSA                     14
#define ID_MAX_RETRY_LIMIT         15
#define ID_TRR                     16
#define ID_LAS                     17
#define ID_GAPL                    18

/* --- Statistic-Counter ------------------------------------------------- */
#define ID_FRAME_SENT_COUNT        20
#define ID_RETRY_COUNT             21
#define ID_SD_COUNT                22
#define ID_SD_ERROR_COUNT          23

/* --- PHY variables ----------------------------------------------------- */
#define ID_TRANSMITTER_OUTPUT      30
#define ID_RECEIVED_SIGNAL_SOURCE  31
#define ID_LOOP                    32
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*-------------------------------------------------------------*/
/*      defined constants for FDL variables                    */
/*-------------------------------------------------------------*/

/* baud_rate */
#define C_KBAUD_9                   0
#define C_KBAUD_19                  1
#define C_KBAUD_93                  2
#define C_KBAUD_187                 3
#define C_KBAUD_500                 4
#define C_MBAUD_1_5                 7

/* medium_red */
#define C_NO_REDUNDANCY             0
#define C_BUS_A_HIGHPRIOR           1
#define C_BUS_B_HIGHPRIOR           2

/* in_ring_desired */
#define C_IN_RING_DESIRED           TRUE
#define C_NOT_IN_RING_DESIRED       FALSE


/*-------------------------------------------------------------*/
/*      defined constants for PHY variables                    */
/*-------------------------------------------------------------*/

/* transmitter_output and loop */
#define C_ENABLED                   TRUE
#define C_DISABLED                  FALSE

/* received_signal_source */
#define C_PRIMARY                   0
#define C_ALTERNATE                 1
#define C_RANDOM                    2


/*-------------------------------------------------------------*/
/*      instance identier for the ident / event service        */
/*-------------------------------------------------------------*/

#define ID_FMA7                     0
#define ID_FMS                      1
#define ID_LLI                      2
#define ID_FDL                      3
#define ID_STATION                  4

#define ID_PHY                      5
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*-------------------------------------------------------------*/
/*     defined constants for the functions-supported bit-string */
/*-------------------------------------------------------------*/

#define BIT_MASTER_IN_MM_KB          0x000001
#define BIT_MASTER_IN_MS_KB          0x000002
#define BIT_SLAVE_IN_MS_KB           0x000004
#define BIT_CONNECTION_FOR_CYCLIC    0x000008
#define BIT_CONNECTIONLESS           0x000010
#define BIT_UNCONFIRMED_HIGH_PRIOR   0x000020
#define BIT_DOMAIN_NAME_ADDR         0x000040
#define BIT_PI_NAME_ADDR             0x000080
#define BIT_VAR_NAME_ADDR            0x000100
#define BIT_VAR_LIST_NAME_ADDR       0x000200
#define BIT_EVN_NAME_ADDR            0x000400
#define BIT_PRESERVED_PARAM          0x000800
#define BIT_KBAUD_9                  0x001000
#define BIT_KBAUD_19                 0x002000
#define BIT_KBAUD_93                 0x004000
#define BIT_KBAUD_187                0x008000
#define BIT_KBAUD_500                0x010000
#define BIT_KBAUD_38                 0x020000
#define BIT_MBAUD_1_5                0x040000
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* ------------------------------------------------------------------------ */
/*       List of Additional Details for Negative FMA7 Confirmations         */
/* ------------------------------------------------------------------------ */

#define NO_ADD_DETAIL                    0x00
#define AD_LLI_UNEXP_FDL_OR_TIMER_EVT    0x01
#define AD_LLI_LSAP_ACT_FAILED           0x02
#define AD_LLI_POLL_LIST_LOAD_FAILED     0x03
#define AD_LLI_PUT_RESRC_FAILED          0x04
#define AD_LLI_FDL_RESET_FAILED          0x05



/*****************************************************************************/
/* FMA7-Context-Management                                                   */
/*****************************************************************************/

#define DETAIL_LENGTH 16      /* length of abort detail                      */
#define FEAT_SUPP_LEN  6      /* length of supported feature field           */


/*****************************************************************************/
/* Initiate Service                                                          */
/*****************************************************************************/

typedef struct T_FMA_INIT_REQ
{
  USIGN8  snd_len_low;                          /* max FMA7 PDU size to send wit
  USIGN8  rcv_len_low;                          /* max FMA7 PDU size to receive
  USIGN8  supported_services[FEAT_SUPP_LEN];    /* supported FMA7 services
} T_FMA_INIT_REQ;


typedef struct  T_FMA_INIT_ERR_CNF
{
  USIGN16 class_code;                           /* error code and class
  USIGN8  snd_len_low;                          /* max FMA7 PDU size to send wit
  USIGN8  rcv_len_low;                          /* max FMA7 PDU size to receive
  USIGN8  supported_services[FEAT_SUPP_LEN];    /* supported FMA7 services
} T_FMA_INIT_ERR_CNF;



/*****************************************************************************/
/* Abort Service                                                             */
/*****************************************************************************/

typedef struct T_FMA_ABORT_REQ
{
  BOOL     local;                     /* local or remote detected            */
  USIGN8   abort_id;                  /* identifier (USR,LLI_USR,LLI,FDL)    */
  USIGN8   reason;                    /* abort reason code                   */
  USIGN8   detail_length;             /* length of detail information        */
  USIGN8   detail[DETAIL_LENGTH];     /* detail information about the reason  */
} T_FMA_ABORT_REQ;



/*****************************************************************************/
/* FMA7-Configuration-Management                                             */
/*****************************************************************************/

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

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* Set-Configuration Service (only local)                                  *
/*************************************************************************/

typedef struct T_SET_CONFIGURATION_REQ
{
  USIGN16 max_nr_of_fal_msg_buffers;    /* max # of Layer7-Message-Buffers
  USIGN16 max_nr_of_fdl_msg_buffers;    /* max # of Layer2-Message-Buffers
  USIGN16 max_nr_of_data_buffers;       /* max # of PDU-Buffers
  USIGN16 max_nr_of_api_data_buffers;   /* max # of abort/poll/idle PDU-Buffer
  USIGN16 max_nr_of_sap_buffers;        /* max # of SAP-Buffers
  USIGN16 max_nr_of_poll_list_entries;  /* max # of Poll-List-Entries
  USIGN16 max_data_buffer_size;         /* max size of FMS-/FMA7-PDU-Buffer
} T_SET_CONFIGURATION_REQ;



/*************************************************************************/
/* Set-Busparameter Service (only local)                                 */
/*************************************************************************/

typedef struct T_SET_BUSPARAMETER_REQ
{
  USIGN8         loc_add;             /* local station                 */
  USIGN8         loc_segm;            /* local segment                 */
  USIGN8         baud_rate;           /* baud rate                     */
  USIGN8         medium_red;          /* medium redundancy             */
  USIGN16        tsl;                 /* slot time                     */
  USIGN16        min_tsdr;            /* min. station delay time resp. */
  USIGN16        max_tsdr;            /* max. station delay time resp. */
  USIGN8         tqui;                /* quiet time                    */
  USIGN8         tset;                /* setup time                    */
  USIGN32        ttr;                 /* target token rotation time    */
  USIGN8         g;                   /* gap update factor             */
  BOOL           in_ring_desired;     /* active or passive station     */
  USIGN8         hsa;                 /* highest station address       */
  USIGN8         max_retry_limit;     /* max. retry limit              */
  USIGN16        reserved;            /* not used                      */
  USIGN8         ident[202];          /* FDL-Ident-String              */
} T_SET_BUSPARAMETER_REQ;



/*************************************************************************/
/* KBL-Management                                                        */
/*************************************************************************/

/*--- KBL-Header ------------------------------------------------------*/
typedef struct T_KBL_HDR
{
  INT16          nr_of_entries;           /* number of KBL enries       */
  USIGN8         poll_sap;                /* poll list SAP              */
  USIGN8         symbol_length;           /* max symbol lenght in KBL   */
  USIGN32        ass_abt_ci;              /* ASS / ABT controll intervall */
  BOOL           vfd_pointer_supported;   /* VFD pointer supported      */
  USIGN8         dummy;                   /* alignment byte             */
} T_KBL_HDR;


/*--- KBL-Static-Entry ------------------------------------------------
typedef struct T_KBL_STATIC
{
   USIGN8    loc_lsap;                    /* local LSAP
   USIGN8    rem_add;                     /* remote address
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
  USIGN8    rem_segm;                          /* remote segment
  USIGN8    rem_lsap;                          /* remote LSAP
  USIGN8    conn_type;                         /* connection type
  USIGN8    lli_sap;                           /* LLI SAP
  USIGN8    multiplier;                        /* multiplier in cyclic connectic
  USIGN8    conn_attr;                         /* connection attribute
  USIGN8    max_scc;                           /* max. of send confirmed counter
  USIGN8    max_rcc;                           /* max. of receive confirmed coun
  USIGN8    max_sac;                           /* max. of send acknowledged coun
  USIGN8    max_rac;                           /* max. of receive acknowledged c
  USIGN32   ci;                                /* controll intervall
  USIGN8    max_pdu_snd_high;                  /* max. length of FMS-PDU send hi
  USIGN8    max_pdu_snd_low;                   /* max. length of FMS-PDU send lo
  USIGN8    max_pdu_rcv_high;                  /* max. length of FMS-PDU rcv hig
  USIGN8    max_pdu_rcv_low;                   /* max. length of FMS-PDU rcv low
  USIGN8    feature_supp[FEAT_SUPP_LEN];       /* FMS features supported
  STRINGV   symbol[MAX_KBL_SYMBOL_LENGTH];     /* symbolic name
  USIGN32   vfd_pointer;                       /* vfd number
  USIGN8    extension[MAX_KBL_EXTENSION_LENGTH]; /* KBL-Extension
} T_KBL_STATIC;



/*--- KBL-Dynamic-Entry ------------------------------------------------*/
typedef struct T_KBL_DYNAMIC
{
  USIGN8         rem_add;                    /* current remote address       */
  USIGN8         rem_segm;                   /* current remote segment       */
  USIGN8         rem_lsap;                   /* current remote LSAP          */
  USIGN8         scc;                        /* send confirmed counter       */
  USIGN8         rcc;                        /* receive confirmed counter    */
  USIGN8         sac;                        /* send acknowledged counter    */
  USIGN8         rac;                        /* receive acknowledged counter */
  BOOL           poll_entry_enabled;         /* poll entry flag              */
} T_KBL_DYNAMIC;



/*****************************************************************************/
/*  Load-KBL Services    (Local and Remote)                                  */
/*****************************************************************************/


/*--- Load-KBL-Service -------------------------------------------------*/
typedef struct T_LOAD_KBL_REQ
{
  USIGN16        desired_cr;           /* desired communication reference */
  union
  {
    T_KBL_HDR    kbl_hdr;              /* KBL-Header-Entry                */
    T_KBL_STATIC kbl_static;           /* KBL-Static-Entry                */
  } id;
} T_LOAD_KBL_REQ;



/*****************************************************************************/
/* Read-KBL Service  (Local and Remote)                                      */
/*****************************************************************************/

typedef struct T_READ_KBL_REQ
{
  USIGN16        desired_cr;           /* desired communication reference  */
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
} T_READ_KBL_REQ;


/*--- KBL-Entry Data-Type ------------------------------------------------*/
typedef struct T_KBL_ENTRY
{
   T_KBL_STATIC   kbl_static;                    /* KBL-Static-Entry        */
   T_KBL_DYNAMIC  kbl_dynamic;                   /* KBL-Dynamic-Entry       */
   USIGN8         dummy;                         /* alignment               */
   USIGN8         kbl_status_len;                /* length of KBL-Status    */
/* USIGN8         kbl_status[kbl_status_len];       KBL-Status              */
} T_KBL_ENTRY;


typedef struct T_READ_KBL_CNF
{
  USIGN16          desired_cr;          /* desired communication reference */
  union
  {
    T_KBL_HDR      kbl_hdr;             /* KBL-header                      */
    T_KBL_ENTRY    kbl_entry;           /* KBL-Entry                       */
  } id;
} T_READ_KBL_CNF;




/*****************************************************************************/
/* Set-Value Service  (Local and Remote)                                     */
/*****************************************************************************/

typedef struct T_STATISTICS_BLOCK
{
  USIGN32       frame_send_count;       /* frame sent counter              */
  USIGN32       sd_count;               /* valid start delimiter counter   */
  USIGN16       retry_count;            /* retry frame sent counter        */
  USIGN16       sd_error_count;         /* invalid start delimiter counter */
} T_STATISTICS_BLOCK;


typedef struct T_SET_VALUE_REQ
{
   USIGN8   id;                                  /* value identifier    */
   USIGN8   length;                              /* # of values in byte */
/* USIGN8   value[length];                          list of values      */
} T_SET_VALUE_REQ;



/*****************************************************************************/
/* Read-Value Service  (Local and Remote)                                    */
/*****************************************************************************/

typedef struct T_READ_BUSPARAMETER_CNF
{
  USIGN8          loc_add;                /* local station                 */
  USIGN8          loc_segm;               /* local segment                 */
  USIGN8          baud_rate;              /* baud rate                     */
  USIGN8          medium_red;             /* medium redundancy             */
  USIGN16         tsl;                    /* slot time                     */
  USIGN16         min_tsdr;               /* min. station delay time resp. */
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
  USIGN16        max_tsdr;                  /* max. station delay time resp. */
  USIGN8         tqui;                      /* quiet time                    */
  USIGN8         tset;                      /* setup time                    */
  USIGN32        ttr;                       /* target token rotation time    */
  USIGN8         g;                         /* gap update factor             */
  BOOL           in_ring_desired;           /* active or passive station     */
  USIGN8         hsa;                       /* highest station address       */
  USIGN8         max_retry_limit;           /* max. retry limit              */
  USIGN16        reserved;                  /* not used                      */
  USIGN8         ident[202];                /* FDL-Ident-String              */
} T_READ_BUSPARAMETER_CNF;



typedef struct T_READ_STATISTIC_CTR_CNF
{
  USIGN32      frame_send_count;        /* frame sent counter                */
  USIGN32      sd_count;                /* valid start delimiter counter     */
  USIGN16      retry_count;             /* retry frame sent counter          */
  USIGN16      sd_error_count;          /* invalid start delimiter counter   */
} T_READ_STATISTIC_CTR_CNF;



typedef struct T_READ_VALUE_REQ
{
  USIGN8   id;                                  /* value identifier   */
  USIGN8   dummy;                               /* alignment          */
} T_READ_VALUE_REQ;



typedef struct T_READ_VALUE_CNF
{
   USIGN8   id;                                 /* value identifier    */
   USIGN8   length;                             /* # of values in byte */
/* USIGN8   value[length];                        list of values      */
} T_READ_VALUE_CNF;



/*****************************************************************************/
/* Ident Service  (Local and Remote)                                         */
/*****************************************************************************/

typedef struct T_CHARACTERISTICS
{
  USIGN8     profile_number     [2];       /* profile number                */
  USIGN8     functions_supp     [3];       /* functions supported           */
  USIGN8     dummy1;                          /* alignment byte
  USIGN8     max_pdu_len;                   /* max. PDU length               */
  USIGN8     dummy2;                        /* alignment byte                */
  USIGN8     fms_features_supp  [6];       /* FMS features supported        */
  USIGN8     fma7_services_supp [6];       /* FMA7 features supported       */
  USIGN8     max_sap_value;                 /* max. LSAP number              */
  USIGN8     max_no_of_saps;                /* max. number of LSAPs          */
  USIGN16    max_comref;                    /* max communication reference   */
  USIGN16    max_kbl_len;                   /* max count of KBL-Entries      */
  USIGN32    total_len_of_pdu;              /* total length of PDUs          */
  USIGN16    no_of_parallel_serv;           /* number of parallel serv.      */
  USIGN16    max_ov_index;                  /* max. OV-Index                 */
  USIGN16    max_ov_entries;                /* max. OV-Entries               */
  USIGN8     max_vfd;                       /* max. VFDs                     */
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
  USIGN8      max_las_entries;                  /* max. no. of LAS entries     */
  USIGN8      min_tsdr;                         /* min. station delay time     */
  USIGN8      trdy;                             /* ready time                  */
  USIGN8      tsdi;                             /* station delay initiatior    */
  USIGN8      max_tsdr;                         /* station delay responder     */
} T_CHARACTERISTICS;


typedef struct T_IDENT_REQ
{
  USIGN8   instance_id;                         /* instance identifier         */
  USIGN8   dummy;                               /* alignment byte              */
} T_IDENT_REQ;


typedef struct T_IDENT_CNF
{
  USIGN8            instance_id;                                  /* instance ide
  USIGN8            dummy;                                        /* alignment
  STRINGV           vendor_name[MAX_IDENT_STRING_LENGTH];         /* vendor name
  STRINGV           controller_type[MAX_IDENT_STRING_LENGTH];     /* controller t
  STRINGV           hw_release[MAX_IDENT_STRING_LENGTH];          /* HW release
  STRINGV           sw_release[MAX_IDENT_STRING_LENGTH];          /* SW release
  T_CHARACTERISTICS characteristics;
} T_IDENT_CNF;



/*****************************************************************************/
/* LSAP-Status Service  (Local and Remote)                                   */
/*****************************************************************************/

typedef struct T_LSAP_STATUS_REQ
{
  USIGN8    lsap;                               /* desired LSAP                */
  USIGN8    dummy;                              /* alignment byte              */
} T_LSAP_STATUS_REQ;



typedef struct T_LSAP_STATUS_CNF
{
  USIGN8    access;                             /* station address or all      */
  USIGN8    addr_extension;                     /* segment number              */
  USIGN8    sda;                                /* SDA                         */
  USIGN8    sdn;                                /* SDN                         */
  USIGN8    srd;                                /* SRD                         */
  USIGN8    csrd;                               /* CSRD                        */
} T_LSAP_STATUS_CNF;



/*****************************************************************************/
/* Get-Live-List Service  (Local and Remote)                                 */
/*****************************************************************************/

typedef struct T_LIVE_LIST
{
  USIGN8      station;                          /* station number              */
  USIGN8      status;                           /* current station of station  */
} T_LIVE_LIST;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
typedef struct T_GET_LIVE_LIST_CNF
{
   USIGN8       dummy;                          /* alignment                    */
   USIGN8       no_of_elements;                 /* # of live list elements      */
/* T_LIVE_LIST live_list[no_of_elements];          list of live list elements   */
} T_GET_LIVE_LIST_CNF;


/*****************************************************************************/
/* FMA7 Fault-Management                                                     */
/*****************************************************************************/


/*****************************************************************************/
/* FMA7-Event-Service (FMA2-Event and LLI-Fault-Indications)                 */
/*****************************************************************************/

typedef struct T_FMA7_EVENT_IND
{
  USIGN16    comm_ref;                          /* communication reference   */
  USIGN8     instance_id;                       /* LLI, FDL, PHY             */
  USIGN8     reason;                            /* reason code               */
  USIGN8     add_detail;                        /* additional detail         */
  USIGN8     dummy;                             /* alignment                 */
} T_FMA7_EVENT_IND;


/*****************************************************************************
/*   Copyright (C) SOFTING GmbH 1992, 1993, 1994 All Rights Reserved         *
/*                                                                           *
/*               PROFIBUS FMS DATA TYPES and DATA STRUCTURES                 *
/*                                                                           *
/*  Filename            :   pb_fms.h                                         *
/*  Version             :   4.01A                                            *
/*  Date                :   01.02.93                                         *
/*  Author              :   Matthias Boettcher                               *
/*                                                                           *
/*  Description :                                                            *
/*                                                                           *
/*  This headerfile contains the exported data types and data structures     *
/*  of the PROFIBUS FMS-Services and FMS-Objects.                            *
/*                                                                           *
/*  CHANGE_NOTES                                                             *
/*                                                                           *
/*  date      name      change                                               *
/*  ------------------------------------------------------------------------ *
/*  14.05.93  Boe       insert copyright                                     *
/*  14.05.93  Boe       remove literallies: NUM_FMS_SERVICES and             *
/*                                          NUM_FMS_FEATURES                 *
/*  25.10.93  Boe       change value of domain-state: EXISTENT to 0x01       *
/*  25.10.93  Boe       insert new local service identifiere 'PI_SET_STATE'  *
/*  25.10.93  Boe       insert T_PI_SET_STATE_REQ,T_PI_SET_STATE_CNF         *
/*                          data structure                                   *
/*****************************************************************************


/*****************************************************************************
/*                                                                           *
/*                            C O N S T A N T S                              *
/*                                                                           *
/*****************************************************************************
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*****************************************************************************
/*************   FMS STRING LENGTH CONSTANTS                  ****************
/*****************************************************************************

#define MAX_VFD_STRING_LENGTH           _NAME_LENGTH(VFD_STRING_LENGTH)

#define MAX_ACCESS_NAME_LENGTH          _NAME_LENGTH(ACCESS_NAME_LENGTH)
#define MAX_OBJECT_NAME_LENGTH          _NAME_LENGTH(OBJECT_NAME_LENGTH)
#define MAX_EXTENSION_LENGTH            _NAME_LENGTH(EXTENSION_LENGTH)
#define MAX_EXECUTION_ARGUMENT_LENGTH   _NAME_LENGTH(EXECUTION_ARGUMENT_LENGTH


/*****************************************************************************
/*************   FMS SERVICE IDENTIFIERS                      ****************
/*****************************************************************************

/* --- remote FMS services ------------------------------------------------ *
#define INITIATE                        0

#define STATUS                          2
#define IDENTIFY                        3
#define READ                            4
#define WRITE                           5
#define GETOV                           6
#define READWITHTYPE                    7
#define WRITEWITHTYPE                   8
#define DEFINEVARIABLELIST              9
#define DELETEVARIABLELIST             10
#define INITIATEDOWNLOADSEQUENCE       11
#define DOWNLOADSEGMENT                12
#define TERMINATEDOWNLOADSEQUENCE      13
#define INITIATEUPLOADSEQUENCE         14
#define UPLOADSEGMENT                  15
#define TERMINATEUPLOADSEQUENCE        16
#define REQUESTDOMAINDOWNLOAD          17
#define REQUESTDOMAINUPLOAD            18
#define CREATEPROGRAMINVOCATION        19
#define DELETEPROGRAMINVOCATION        20
#define START                          21
#define STOP                           22
#define RESUME                         23
#define RESET                          24
#define KILL                           25
#define ALTEREVENTCONDITIONMONITORING  26
#define ACKNOWLEDGEEVENTNOTIFICATION   27
#define PHYS_READ                      28
#define PHYS_WRITE                     29
#define INITIATE_PUT_OV                30
#define PUT_OV                         31
#define TERMINATE_PUT_OV               32

#define INFORMATIONREPORT              33
#define UNSOLICITEDSTATUS              34
#define EVENTNOTIFICATION              35
#define INFORMATIONREPORTWITHTYPE      36
#define EVENTNOTIFICATIONWITHTYPE      37

#define ABORT                          38
#define REJECT                         39

/* --- FMS features ------------------------------------------------------- *
#define GET_OV_LONG                    40
#define NAME_ADDRESSING                41
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/* --- local FMS services ------------------------------------------------ *
#define OV_READ_LOC                    42
#define INITIATE_LOAD_OV_LOC           43
#define LOAD_OV_LOC                    44
#define TERMINATE_LOAD_OV_LOC          45
#define CREATE_VFD_LOC                 46
#define VFD_SET_PHYS_STATUS_LOC        47
#define PI_SET_STATE_LOC               48


/*****************************************************************************
/*************   FMS CONTEXT MANAGEMENT                        ***************
/*****************************************************************************

#define DETAIL_LENGTH                  16      /* length of abort detail      *
#define FEAT_SUPP_LEN                   6      /* length of supp. feat. field *


/*****************************************************************************
/*************   VFD SUPPORT MANAGEMENT                        ***************
/*****************************************************************************

/* --- logical status of the communcation -------------------------------- *
#define STATE_CHANGES_ALLOWED           0
#define LIMITED_SERVICES_PERMITTED      2

/* --- physical status of the application device ------------------------- *
#define OPERATIONAL                     0
#define PARTIALLY_OPERATIONAL           1
#define INOPERABLE                      2
#define NEEDS_COMMISSIONING             3


/*****************************************************************************
/*************   OV-MANAGEMENT                                 ***************
/*****************************************************************************

/* --- ov states --------------------------------------------------------- *
#define OV_LOADABLE                     2
#define OV_READY                        3
#define OV_LOADING_NON_INTERACTING      4
#define OV_LOADING_INTERACTING          5
#define OV_NOT_EXISTENT                 6

/* --- indices in ov type description for PROFIBUS standard types ---------- *
#define BOOLEAN                         1
#define INTEGER8                        2
#define INTEGER16                       3
#define INTEGER32                       4
#define UNSIGNED8                       5
#define UNSIGNED16                      6
#define UNSIGNED32                      7
#define FLOATING_POINT                  8
#define VISIBLE_STRING                  9
#define OCTET_STRING                   10
#define DATE_TYPE                      11
#define TIME_OF_DAY                    12
#define TIME_DIFFERENCE                13
#define BIT_STRING                     14

 * --- codes of PROFIBUS objects ------------------------------------------ *
#define  NULL_OBJECT                    0    /* Null                         *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#define  OV_OBJECT                    1    /* object description object     *
#define  DOMAIN_OBJECT                2    /* domain object                 *
#define  INVOCATION_OBJECT            3    /* program invocation object     *
#define  EVENT_OBJECT                 4    /* alarm object                  *
#define  TYPE_OBJECT                  5    /* basic data type description   *
#define  TYPE_STRUCT_OBJECT           6    /* structured data type descript.*
#define  SIMPLE_VAR_OBJECT            7    /* simple variable object        *
#define  ARRAY_OBJECT                 8    /* array object                  *
#define  RECORD_OBJECT                9    /* record object                 *
#define  VAR_LIST_OBJECT             10    /* variable list object          *

#define  VAR_OBJECT                  11    /* class of all variable objects *
#define  ALL_OBJECT                  12    /* super class of all objects    *


/* --- codes for object access mode, to be used as tag in T_ACC_SPEC-------- *
#define ACCESS_INDEX                  0    /* access by index               *
#define ACCESS_NAME                   1    /* acess by name                 *
#define ACCESS_NAME_LIST              2    /* acess by name list            *

/* --- codes for named access specification, to be used for ov services only *
#define  INDEX_ACCESS                 1    /* access by index               *
#define  VAR_NAME_ACCESS              2    /* access by variable name       *
#define  VAR_LIST_NAME_ACCESS         3    /* access by variable list name  *
#define  DOMAIN_NAME_ACCESS           4    /* access by domain name         *
#define  PI_NAME_ACCESS               5    /* access by pi name             *
#define  EVENT_NAME_ACCESS            6    /* access by event name          *
#define  START_INDEX_ACCESS           7    /* access by start index         *


/* --- ov loading selection ----------------------------------------------*
#define CONSEQUENCE_0                 0
#define CONSEQUENCE_1                 1
#define CONSEQUENCE_2                 2


/*************************************************************************
/*************   VARIABLE ACCESS                    **********************
/*************************************************************************

/* --- tag values for type description (T_TYPE_DESCR)---------------------- *
#define SIMPLE_TYPE                   1
#define ARRAY_TYPE                    2
#define RECORD_TYPE                   3


/*************************************************************************
/*************   DOMAIN-MANAGEMENT                  **********************
/*************************************************************************

/* --- domain states ------------------------------------------------------ *
#define  EXISTENT                  0x01
#define  LOADING                   0x02
#define  INCOMPLETE                0x03
#define  COMPLETE                  0x04
#define  READY                     0x05
#define  IN_USE                    0x06

/* --- upload states ------------------------------------------------------ *
#define  NON_EXISTENT              0x00
#define  UPLOADING                 0x01
#define  UPLOADED                  0x02
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*****************************************************************************
/*************   PROGRAM INVOCATION MANAGEMENT          **********************
/*****************************************************************************

/* --- program invocation states ------------------------------------------ *
#define   NON_EXISTENT              0x00
#define   UNRUNNABLE                0x01
#define   IDLE                      0x02
#define   RUNNING                   0x03
#define   STOPPED                   0x04

#define   STARTING                  0x05
#define   STOPPING                  0x06
#define   RESUMING                  0x07
#define   RESETTING                 0x08


/*****************************************************************************
/*                                                                           *
/*                        D A T A   S T R U C T U R E S                      *
/*                                                                           *
/*****************************************************************************



/*****************************************************************************
/*************   ACCESS CONTROL                         **********************
/*****************************************************************************

typedef struct T_ACCESS
{
  USIGN8     pass_word;                 /* password                          *
  USIGN8     acc_groups;                /* access groups                     *
  USIGN16    acc_right;                 /* access rights                     *
} T_ACCESS;


/*****************************************************************************
/*************   ACCESS SPECIFICATION                   **********************
/*****************************************************************************

typedef struct T_ACC_SPEC
{
  USIGN8     tag;                       /* id of the access specification    *
  USIGN8     dummy;                     /* alignment byte                    *
  union
  {
    USIGN16  index;                          /* access by index             *
    STRINGV  name[MAX_ACCESS_NAME_LENGTH];   /* access by symbolic name     *
  } id;
} T_ACC_SPEC;


/*****************************************************************************
/*************   FMS CONTEXT MANAGEMENT                 **********************
/*****************************************************************************

/* --- Initiate-Service --------------------------------------------------*
typedef struct T_CTXT_INIT_REQ
{
  USIGN8     profile_number[2];         /* profile number                    *
  BOOL       protection;                /* access protection                 *
  USIGN8     password;                  /* password                          *
  USIGN8     access_groups;             /* access groups                     *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
  USIGN8    dummy;                          /* alignment byte                    *
  INT16     ov_version;                     /* ov version                        *
  USIGN8    snd_len_h;                      /* max pdu size to send (high prio)  *
  USIGN8    snd_len_l;                      /* max pdu size to send (low prio)   *
  USIGN8    rcv_len_h;                      /* max pdu size to receive (high)    *
  USIGN8    rcv_len_l;                      /* max pdu size to receive (low)     *
  USIGN8    supported_features[FEAT_SUPP_LEN];   /* supported features           *
} T_CTXT_INIT_REQ;


typedef struct  T_CTXT_INIT_CNF
{
  USIGN8    profile_number[2];              /* profile number                    *
  INT16     ov_version;                     /* ov version                        *
  BOOL      protection;                     /* access protection                 *
  USIGN8    password;                       /* password                          *
  USIGN8    access_groups;                  /* access groups                     *
  USIGN8    dummy;                          /* alignment byte                    *
} T_CTXT_INIT_CNF;


typedef struct  T_CTXT_INIT_ERR_CNF
{
  USIGN16 class_code;                       /*  error class, error code          *
  USIGN8  snd_len_h;                        /*  max pdu size to send (high prio) *
  USIGN8  snd_len_l;                        /*  max pdu size to send (low prio)  *
  USIGN8  rcv_len_h;                        /*  max pdu size to receive (high)   *
  USIGN8  rcv_len_l;                        /*  max pdu size to receive (low)    *
  USIGN8  supported_features[FEAT_SUPP_LEN];   /*  supported features           *
} T_CTXT_INIT_ERR_CNF;



/*--- abort service ------------------------------------------------------*
typedef struct T_CTXT_ABORT_REQ
{
  BOOL      local;                          /* local or remote detected          *
  USIGN8    abort_id;                       /* abort identifier USR, FMS, ...    *
  USIGN8    reason;                         /* abort reason code                 *
  USIGN8    detail_length;                  /* length of abort detail            *
  USIGN8    detail[DETAIL_LENGTH];          /* detail information                *
} T_CTXT_ABORT_REQ;


/*--- reject service -----------------------------------------------------*
typedef struct T_CTXT_REJECT_IND
{
  BOOL      detected_here;                  /* local or remote detected          *
  USIGN8    orig_invoke_id;                 /* original invoke ID                *
  USIGN8    pdu_type;                       /* reject PDU types                  *
  USIGN8    reject_code;                    /* reject code                       *
} T_CTXT_REJECT_IND;



/*****************************************************************************
/*************   VARIABLE ACCESS MANAGEMENT                 *******************
/*****************************************************************************

typedef struct T_SIMPLE_TYPE
{
   USIGN16    data_type_index;             /* index of data type                *
   USIGN8     length;                      /* size of data type                 *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
   USIGN8      dummy;                      /* alignment byte                  *
}  T_SIMPLE_TYPE;

typedef struct T_ARRAY_TYPE
{
   USIGN16     data_type_index;            /* index of data type              *
   USIGN8      length;                     /* size of data type               *
   USIGN8      no_of_elements;             /* number of data types            *
}  T_ARRAY_TYPE;


typedef struct T_RECORD_TYPE
{
   USIGN8         no_of_elements;                  /* number of record elem.  *
   USIGN8         dummy;                           /* alignment byte          *
   T_SIMPLE_TYPE simple[MAX_VAR_RECORD_ELEMENTS]; /* list of simple types    *
}  T_RECORD_TYPE;


typedef struct T_TYPE_DESCR
{
   USIGN8      tag;                        /* type description identifier     *
   USIGN8      dummy;                      /* alignment byte                  *
   union
   {
     T_SIMPLE_TYPE  simple;                /* simple type                     *
     T_ARRAY_TYPE   array;                 /* array type                      *
     T_RECORD_TYPE  record;                /* record type                     *
   } id;
}  T_TYPE_DESCR;



/*--- VARIABLE-OBJECT-DESCRIPTIONS ----------------------------------------


/*--- Simple-Variable-Object-Description ---------------------------------
typedef struct T_SIMPLE_VAR_OBJECT
{
   USIGN16     index;                                /* logical address of the obj
   USIGN8      obj_code;                             /* object code
   USIGN8      length;                               /* length of object in octets
   USIGN16     index_of_type;                        /* logical address of the typ
   T_ACCESS    access;                               /* access right structure
   USIGN32     local_address;                        /* local address
   STRINGV     name[MAX_OBJECT_NAME_LENGTH];         /* name
   USIGN8      extension[MAX_EXTENSION_LENGTH];      /* extension
}  T_SIMPLE_VAR_OBJECT;


/*--- Array-Variable-Object-Description ----------------------------------
typedef struct T_ARRAY_OBJECT
{
   USIGN16     index;                                /* logical address of the obj
   USIGN8      obj_code;                             /* object code
   USIGN8      length;                               /* length of an element in oc
   USIGN16     index_of_type;                        /* logical address of the typ
   USIGN8      nof_elements;                         /* number of array elements
   USIGN8      dummy;                                /* alignment byte
   T_ACCESS    access;                               /* access right structure
   USIGN32     local_address;                        /* local address
   STRINGV     name[MAX_OBJECT_NAME_LENGTH];         /* name
   USIGN8      extension[MAX_EXTENSION_LENGTH];      /* extension
}  T_ARRAY_OBJECT;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*--- Record-Variable-Object-Description ----------------------------------
typedef struct T_RECORD_OBJECT
{
   USIGN16     index;                                  /* index
   USIGN8      obj_code;                               /* object code
   USIGN8      no_of_address;                          /* number of local address
   USIGN16     index_of_type;                          /* logical address of the typ
   T_ACCESS    access;                                 /* access right structure
   STRINGV     name[MAX_OBJECT_NAME_LENGTH];           /* name
   USIGN8      extension[MAX_EXTENSION_LENGTH];        /* extension
   USIGN32     reserved;                               /* for internal use
/* USIGN32     local_address_list[no_of_address];      local address list
} T_RECORD_OBJECT;


/*--- Variable-List-Object-Description ------------------------------------
typedef struct T_VAR_LIST_OBJECT
{
   USIGN16     index;                                   /* logical address of the ob
   USIGN8      obj_code;                                /* object code
   USIGN8      no_of_var;                               /* number of variables
   T_ACCESS    access;                                  /* access right
   BOOL        deletable;                               /* TRUE - deletable; else FA
   USIGN8      dummy;                                   /* alignment-byte
   STRINGV     name[MAX_OBJECT_NAME_LENGTH];            /* name
   USIGN8      extension[MAX_EXTENSION_LENGTH];         /* extension
   USIGN32     reserved;                                /* for internal use
/* USIGN16     var_list[no_of_var];                        list of variables
} T_VAR_LIST_OBJECT;



/* --- Read-Service ------------------------------------------------------
typedef struct T_VAR_READ_REQ
{
   T_ACC_SPEC acc_spec;                          /* access specification
   INT8       subindex;                          /* subindex
   USIGN8     dummy;                             /* alignment byte
} T_VAR_READ_REQ;


typedef struct  T_VAR_READ_CNF
{
   USIGN8  dummy;                                /* alignment byte
   USIGN8  length;                               /* length of values in bytes
/* USIGN8  value[lenght];                          list of data  */
} T_VAR_READ_CNF;



/*--- Read-With-Type-Service ----------------------------------------------
typedef struct T_VAR_READ_WITH_TYPE_REQ
{
   T_ACC_SPEC acc_spec;                          /* access specification
   INT8       subindex;                          /* subindex
   USIGN8     dummy;                             /* alignment byte
} T_VAR_READ_WITH_TYPE_REQ;



typedef struct  T_VAR_READ_WITH_TYPE_CNF
{
   USIGN8       no_of_type_descr;                     /* number of typedescrip
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
   USIGN8       length;                                  /* # of values in bytes
/* T_TYPE_DESCR type_descr_list[no_of_type_descr];       list of type descript
/* USIGN8       value[length];                           list of data
} T_VAR_READ_WITH_TYPE_CNF;



/*--- Write-Service ------------------------------------------------------
typedef struct T_VAR_WRITE_REQ
{
   T_ACC_SPEC acc_spec;                                  /* access specification
   INT8       subindex;                                  /* subindex
   USIGN8     length;                                    /* # number of values in b
/* USIGN8     value[length];                             list of values
} T_VAR_WRITE_REQ;



/*--- Write-With-Type-Service --------------------------------------------
typedef struct T_VAR_WRITE_WITH_TYPE_REQ
{
   T_ACC_SPEC   acc_spec;                                /* access specification
   INT8         subindex;                                /* subindex
   USIGN8       dummy;                                   /* alignment byte
   USIGN8       no_of_type_descr;                        /* number of type descript
   USIGN8       length;                                  /* # of values in bytes
/* T_TYPE_DESCR type_descr_list[no_of_type_descr];       list of type descriptio
/* USIGN8       value[length];                           list of datas
} T_VAR_WRITE_WITH_TYPE_REQ;



/*--- Information-Report-Service -----------------------------------------
typedef struct T_VAR_INFO_RPT_REQ
{
   USIGN8     priority;                                  /* priority
   INT8       subindex;                                  /* subindex
   T_ACC_SPEC acc_spec;                                  /* access specification
   USIGN8     dummy;                                     /* alignment-byte
   USIGN8     length;                                    /* # of values in bytes
/* USIGN8     value[length];                             list of data
} T_VAR_INFO_RPT_REQ;

/*--- Information-Report-With-Type-Service -------------------------------
typedef struct T_VAR_INFO_RPT_WITH_TYPE_REQ
{
   USIGN8       priority;                                /* priority
   INT8         subindex;                                /* subindex
   T_ACC_SPEC   acc_spec;                                /* access specification
   USIGN8       no_of_type_descr;                        /* number of typedescrip
   USIGN8       length;                                  /* # of values in bytes
/* T_TYPE_DESCR type_descr_list[no_of_type_descr];       list of type descript
/* USIGN8       value[length];                           list of data
} T_VAR_INFO_RPT_WITH_TYPE_REQ;


/*--- Define-Variable-List-Service ---------------------------------------
typedef struct T_VAR_DEFINE_VAR_LIST_REQ
{
   T_ACCESS   access;                                    /* access rights
   STRINGV    name[MAX_OBJECT_NAME_LENGTH];              /* variable list name
   USIGN8     extension[MAX_EXTENSION_LENGTH];           /* extension
   USIGN16    index;                                     /* index of variable lis
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
   USIGN8      dummy;                                  /* alignment byte
   USIGN8      no_of_var;                              /* number of variables
/* T_ACC_SPEC var_list[no_of_var];                     list of variables
} T_VAR_DEFINE_VAR_LIST_REQ;



typedef struct  T_VAR_DEFINE_VAR_LIST_CNF
{
   USIGN16     index;                                  /* index of variable lis
} T_VAR_DEFINE_VAR_LIST_CNF;



/*--- Delete-Variable-List-Service ----------------------------------------
typedef struct T_VAR_DELETE_VAR_LIST_REQ
{
   T_ACC_SPEC  acc_spec;                               /* access specification
} T_VAR_DELETE_VAR_LIST_REQ;



/*--- Physical-Read-Service -----------------------------------------------
typedef struct T_VAR_PHYS_READ_REQ
{
   USIGN32       int_addr;                         /* physical address to be rea
   USIGN8        length;                           /* length in octets
   USIGN8        dummy;                            /* alignment byte
} T_VAR_PHYS_READ_REQ;


typedef struct T_VAR_PHYS_READ_CNF
{
    USIGN8   dummy;                                 /* alignment byte
    USIGN8   length;                                /* length of values in bytes
/*  USIGN8   data[length];                             list of data
} T_VAR_PHYS_READ_CNF;


/*--- Physical-Write-Service ----------------------------------------------
typedef struct T_VAR_PHYS_WRITE_REQ
{
   USIGN32  int_addr;                               /* physical address to be wri
   USIGN8   dummy;                                  /* length in octets
   USIGN8   length;                                 /* length in octets
/* USIGN8   data[length];                             list of datas
} T_VAR_PHYS_WRITE_REQ;




/*****************************************************************************
/*************   EVENT-MANAGEMENT                        *********************
/*****************************************************************************


/*--- Event-Object-Description --------------------------------------------*
typedef struct T_EVENT_OBJECT
{
  USIGN16     index_event;                           /* index                *
  USIGN8      obj_code;                              /* object code          *
  USIGN8      data_length;                           /* size of event data   *
  USIGN16     index_event_data;                      /* index of event-data  *
  T_ACCESS    access;                                /* access protection    *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
  BOOL        enabled;                          /* =>TRUE event is enabled *
  USIGN8      dummy;                            /* alignment byte          *
  STRINGV     name[MAX_OBJECT_NAME_LENGTH];     /* symbolic name           *
  USIGN8      extension[MAX_EXTENSION_LENGTH];  /* extension               *
} T_EVENT_OBJECT;


/*--- Event-Notification-Service ----------------------------------------*
typedef  struct  T_EVENT_NOTIFY_REQ
{
   USIGN8      priority;                  /* priority                     *
   USIGN8      event_number;              /* event number                 *
   T_ACC_SPEC  acc_spec;                  /* access specification         *
   USIGN8      dummy;                     /* alignment                    *
   USIGN8      data_length;               /* # of event datas in byte     *
/* USIGN8      event_data[data_length];      list of event_datas           *
} T_EVENT_NOTIFY_REQ;


/*--- Event-Notification-With-Type-Service ------------------------------*
typedef  struct  T_EVENT_NOTIFY_WITH_TYPE_REQ
{
   USIGN8         priority;               /* priority                     *
   USIGN8         event_number;           /* event number                 *
   T_ACC_SPEC     acc_spec;               /* access specification         *
   T_TYPE_DESCR   type_descr;             /* type description             *
   USIGN8         dummy;                  /* alignment                    *
   USIGN8         data_length;            /* length of event data         *
/* USIGN8         event_data[data_length];   list of event_datas          *
} T_EVENT_NOTIFY_WITH_TYPE_REQ;



/*--- Alter-Event-Condition-Monitoring ----------------------------------*
typedef  struct  T_ALT_EVN_CND_MNT_REQ
{
   T_ACC_SPEC  acc_spec;                  /* access specification         *
   BOOL        enabled;                   /* enable or disable the event  *
   USIGN8      dummy;                     /* alignment byte               *
} T_ALT_EVN_CND_MNT_REQ;



/*--- Acknowledge-Event-Notification ------------------------------------*
typedef  struct  T_ACK_EVN_NOTIFY_REQ
{
   T_ACC_SPEC  acc_spec;                  /* access specification    *
   USIGN8      event_number;              /* event count number      *
   USIGN8      dummy;                     /* alignment byte          *
} T_ACK_EVN_NOTIFY_REQ;



/*****************************************************************************
/*************   DOMAIN - MANAGEMENT                    *********************
/*****************************************************************************

/*--- Domain-Object-Description -----------------------------------------*
typedef struct T_DOM_OBJECT
{
   USIGN16     index;                     /* index                   *
   USIGN8      obj_code;                  /* object code             *
   USIGN8      state;                     /* domain state            *
   USIGN8      upload_state;              /* upload state            *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
   INT8        counter;                                /* in use counter          *
   USIGN16     max_octets;                             /* max domain length       *
   T_ACCESS    access;                                 /* access protection       *
   USIGN32     local_address;                          /* local address           *
   STRINGV     name[MAX_OBJECT_NAME_LENGTH];           /* symbolic name           *
   USIGN8      extension[MAX_EXTENSION_LENGTH];        /* extension               *
} T_DOM_OBJECT;


/*--- Domain-Download-Services ------------------------------------------------*
/*--- Domain-Upload-Services   ------------------------------------------------*

typedef struct T_DOM_REQ
{
   T_ACC_SPEC  acc_spec;                  /* access specification              *
} T_DOM_REQ;


typedef struct T_DNL_UPL_SEG_CNF
{
   BOOL     more_follows;                 /* more_follows                      *
   USIGN8   data_len;                     /* data length                       *
/* USIGN8   data[data_len];                  list of data                       *
} T_DNL_UPL_SEG_CNF;


typedef struct T_TERM_DNL_REQ
{
   T_ACC_SPEC  acc_spec;                     /* access specification
   BOOL        final_result;                 /* final result
   USIGN8      dummy;                        /* alignment
} T_TERM_DNL_REQ;


typedef struct T_REQUEST_DOM_REQ
{
   T_ACC_SPEC  acc_spec;                     /* access specification
   USIGN8      dummy;                        /* alignment
   USIGN8      add_info_length;              /* length of add. information
/* STRINGV     add_info[add_info_length];       additional information
} T_REQUEST_DOM_REQ;



/*****************************************************************************
/*************   PROGRAM-INVOCATION-MANAGEMENT          *********************
/*****************************************************************************


/*--- Program-Invocation-Object -----------------------------------------------*
typedef  struct T_PI_OBJECT
{
   USIGN16     index;                                  /* pi_index in ov      *
   USIGN8      obj_code;                               /* object code for OV  *
   USIGN8      cnt_dom;                                /* # domains           *
   T_ACCESS    access;                                 /* access              *
   BOOL        deletable;                              /* deletable           *
   BOOL        reusable;                               /* reusable            *
   USIGN8      pi_state;                               /* state of pi         *
   USIGN8      dummy;                                  /* alignment byte      *
   STRINGV     name[MAX_OBJECT_NAME_LENGTH];           /* symbolic name of pi *
   USIGN8      extension[MAX_EXTENSION_LENGTH];        /* extension           *
   USIGN32     reserved;                               /* for internal use    *
/* USIGN16     dom_list[cnt_dom];                        domain index list    *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
} T_PI_OBJECT;



/*--- Create-PI-Service ------------------------------------------------*
typedef struct T_PI_CR8_REQ
{
   T_ACCESS    access;                                /* access rights          *
   USIGN8      cnt_dom;                               /* number of domains      *
   BOOL        reusable;                              /* => TRUE pi is reusable *
   USIGN16     index;                                 /* PI-index               *
   STRINGV     name[MAX_OBJECT_NAME_LENGTH];          /* symbolic name          *
   USIGN8      extension[MAX_EXTENSION_LENGTH];       /* extension              *
/* T_ACC_SPEC  dom_list[cnt_dom];                        list of domains        *
} T_PI_CR8_REQ;


typedef struct T_PI_CR8_CNF
{
  USIGN16   index;                              /* index of PI       */
} T_PI_CR8_CNF;



/*--- Delete-PI-Service ------------------------------------------------*
typedef  struct  T_PI_DEL_REQ
{
   T_ACC_SPEC   acc_spec;              /* access specification */
} T_PI_DEL_REQ;



/*--- Start-PI-Service -------------------------------------------------*
typedef  struct T_PI_START_REQ
{
   T_ACC_SPEC   acc_spec;                                  /* access specific. *
   USIGN8       exec_arg[MAX_EXECUTION_ARGUMENT_LENGTH]; /* execution arg.   *
} T_PI_START_REQ;


/*--- Stop-PI-Service --------------------------------------------------*
typedef  struct  T_PI_STOP_REQ
{
   T_ACC_SPEC   acc_spec;                                  /* access specific. *
} T_PI_STOP_REQ;



/*--- Resume-PI-Service ------------------------------------------------*
typedef  struct T_PI_RESUME_REQ
{
   T_ACC_SPEC   acc_spec;                                  /* access specific. *
   USIGN8       exec_arg[MAX_EXECUTION_ARGUMENT_LENGTH]; /* execution arg.   *
} T_PI_RESUME_REQ;


/*--- Reset-PI-Service -------------------------------------------------*
typedef  struct  T_PI_RESET_REQ
{
   T_ACC_SPEC   acc_spec;                                  /* access specific. *
} T_PI_RESET_REQ;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*--- Kill-PI-Service ------------------------------------------------------*
typedef  struct  T_PI_KILL_REQ
{
   T_ACC_SPEC   acc_spec;                                  /* access specific. *
} T_PI_KILL_REQ;


/*--- PI-SET-STATE-Service ( only local Service ) --------------------------*
typedef struct  T_PI_SET_STATE_REQ
{
  USIGN32       vfd_number;                               /* vfd number           */
  T_ACC_SPEC    acc_spec;                                 /* access specification */
  USIGN8        state;                                    /* new PI state         */
  USIGN8        dummy;                                    /* alignment byte       */
} T_PI_SET_STATE_REQ;


typedef struct  T_PI_SET_STATE_CNF
{
  USIGN32       vfd_number;                               /* vfd number           */
} T_PI_SET_STATE_CNF;


/*****************************************************************************
/*************   OV-MANAGEMENT                          ********************
/*****************************************************************************

/* --- OV-Object-Description ------------------------------------------------
typedef struct T_OV_OBJ_DESCR_HDR
{
   USIGN16  index;              /* index = 0
   USIGN8   obj_code;           /* object-code = 1
   BOOL     flag;               /* => TRUE  write protected
   USIGN8   length;             /* size of names (0-32)
   BOOL     protection;         /* access protection supported
   INT16    version;            /* version */
   INT16    len_st_ov;          /* length of the static type description
   USIGN16  first_index_s_ov;   /* start index of the static object description
   INT16    len_s_ov;           /* length of the static object description
   USIGN16  first_index_dv_ov;  /* start index of the dyn. var. list descriptio
   INT16    len_dv_ov;          /* length of the  dyn. variable list descriptio
   USIGN16  first_index_dp_ov;  /* start index of the dyn. pi description
   INT16    len_dp_ov;          /* length of the dyn. pi description
   USIGN32  int_addr;           /* internal address of ov description
   USIGN32  int_addr_st_ov;     /* internal address of the static type descr.
   USIGN32  int_addr_s_ov;      /* internal address of the static object descr.
   USIGN32  int_addr_dv_ov;     /* internal address of the dyn. var. list descr
   USIGN32  int_addr_dp_ov;     /* internal address of the dyn. pi description
} T_OV_OBJ_DESCR_HDR;


/* --- OV-Null-Object-Description -------------------------------------------*
typedef struct T_OV_NULL_OBJECT
{
   USIGN16  index;                          /* index                        *
   USIGN8   obj_code;                       /* object code                  *
   USIGN8   dummy;                          /* alignment reasons            *
} T_OV_NULL_OBJECT;


/* --- OV-Static-Type-Object-Description ------------------------------------*
typedef struct T_OV_ST_DT_DESCR
{
   USIGN16  index;                           /* index                       *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
   USIGN8   obj_code;                                  /* object code              *
   USIGN8   dummy;                                     /* alignment byte           *
   STRINGV  meaning[MAX_OBJECT_NAME_LENGTH];           /* meaning of the type      *
                                                       /* for information only     *
} T_OV_ST_DT_DESCR;



/* --- OV-Static-Structure-Object-Description ------------------------------*
typedef struct T_OV_DT_LIST
{
   USIGN16   index_of_type;                /* logical address of the type      *
   USIGN8    length;                       /* length of the element in octets  *
   USIGN8    dummy;                        /* alignment byte                   *
} T_OV_DT_LIST;


typedef struct T_OV_ST_DS_DESCR
{
   USIGN16       index;                    /* index                            *
   USIGN8        obj_code;                 /* object code                      *
   USIGN8        no_of_elements;           /* number of record elements        *
   USIGN32       reserved;                 /* for internal use                 *
/* T_OV_DT_LIST  dt_list[no_of_elements];    data type list                     *
} T_OV_ST_DS_DESCR;



/* --- OV-OBJECT-DESCRIPTION ------------------------------------------------ *
typedef struct T_OBJECT_DESCR
{
   union
   {
     T_OV_OBJ_DESCR_HDR    ov_obj_descr;
     T_OV_NULL_OBJECT      null_obj_descr;
     T_OV_ST_DT_DESCR      dt_obj_descr;
     T_OV_ST_DS_DESCR      ds_obj_descr;
     T_SIMPLE_VAR_OBJECT   s_var_obj_descr;
     T_ARRAY_OBJECT        a_var_obj_descr;
     T_RECORD_OBJECT       r_var_obj_descr;
     T_VAR_LIST_OBJECT     vlist_obj_descr;
     T_DOM_OBJECT          dom_obj_descr;
     T_EVENT_OBJECT        evn_obj_descr;
     T_PI_OBJECT           pi_obj_descr;
   } id;
} T_OBJECT_DESCR;


/* --- OV-PACKED-OBJECT-DESCRIPTION ----------------------------------------- *
typedef struct T_PACKED_OBJECT_DESCR
{
   USIGN8    length;                    /* length of packed object description*
/* USIGN8    packed_obj_descr[length];    packed object description          *
} T_PACKED_OBJECT_DESCR;


/* --- Get-OV-Service ------------------------------------------------------*
typedef struct T_GET_OV_REQ
{
    BOOL       format;                 /* TRUE = long format / FALSE = short  *
   USIGN8     dummy;                   /* alignment byte                      *
   T_ACC_SPEC acc_spec;                /* access specification                *
} T_GET_OV_REQ;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
typedef struct T_GET_OV_CNF
{
   BOOL                  more_follows;                  /* further object de
   USIGN8                no_of_ov_descr;                /* # of object descr
/* T_PACKED_OBJECT_DESCR  obj_descr_list[no_of_ov_descr];    list of object de
} T_GET_OV_CNF;



/* --- Put-OV-Services ------------------------------------------------------*
typedef struct T_INIT_PUT_OV_REQ
{
  INT8   consequence;              /* Loading interactive/non-interactive   *
} T_INIT_PUT_OV_REQ;


typedef struct T_PUT_OV_REQ
{
   USIGN8                dummy;                         /* alignment
   USIGN8                no_of_ov_descr;                /* # of object descr
/* T_PACKED_OBJECT_DESCR  obj_descr_list[no_of_ov_descr];    list of object de
} T_PUT_OV_REQ;


/* --- Load-OV-Local-Service ------------------------------------------------*
typedef struct T_INIT_LOAD_OV_REQ
{
   USIGN32       vfd_number;
   INT8          consequence;
   USIGN8        dummy;
} T_INIT_LOAD_OV_REQ;

typedef struct T_INIT_LOAD_OV_CNF
{
   USIGN32       vfd_number;
} T_INIT_LOAD_OV_CNF;


typedef struct T_LOAD_OV_REQ
{
   USIGN32        vfd_number;
   T_OBJECT_DESCR obj_descr;
} T_LOAD_OV_REQ;


typedef struct T_LOAD_OV_CNF
{
   USIGN32        vfd_number;
} T_LOAD_OV_CNF;


typedef struct T_TERM_LOAD_OV_REQ
{
   USIGN32       vfd_number;
} T_TERM_LOAD_OV_REQ;


typedef struct T_TERM_LOAD_OV_CNF
{
   USIGN32       vfd_number;
} T_TERM_LOAD_OV_CNF;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
typedef struct T_GET_OV_CNF
{
   BOOL                    more_follows;                    /* further object de
   USIGN8                  no_of_ov_descr;                  /* # of object descr
/* T_PACKED_OBJECT_DESCR  obj_descr_list[no_of_ov_descr];     list of object de
} T_GET_OV_CNF;



/* --- Put-OV-Services ------------------------------------------------------*
typedef struct T_INIT_PUT_OV_REQ
{
  INT8    consequence;              /* Loading interactive/non-interactive   *
} T_INIT_PUT_OV_REQ;


typedef struct T_PUT_OV_REQ
{
   USIGN8                  dummy;                           /* alignment
   USIGN8                  no_of_ov_descr;                  /* # of object descr
/* T_PACKED_OBJECT_DESCR  obj_descr_list[no_of_ov_descr];     list of object de
} T_PUT_OV_REQ;


/* --- Load-OV-Local-Service ------------------------------------------------*
typedef struct T_INIT_LOAD_OV_REQ
{
   USIGN32       vfd_number;
   INT8          consequence;
   USIGN8        dummy;
} T_INIT_LOAD_OV_REQ;

typedef struct T_INIT_LOAD_OV_CNF
{
   USIGN32       vfd_number;
} T_INIT_LOAD_OV_CNF;


typedef struct T_LOAD_OV_REQ
{
   USIGN32        vfd_number;
   T_OBJECT_DESCR obj_descr;
} T_LOAD_OV_REQ;


typedef struct T_LOAD_OV_CNF
{
   USIGN32        vfd_number;
} T_LOAD_OV_CNF;


typedef struct T_TERM_LOAD_OV_REQ
{
   USIGN32       vfd_number;
} T_TERM_LOAD_OV_REQ;


typedef struct T_TERM_LOAD_OV_CNF
{
   USIGN32       vfd_number;
} T_TERM_LOAD_OV_CNF;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*--- Read-OV-Local-Service ------------------------------------------------*
typedef struct T_OV_READ_LOC_REQ
{
  USIGN32    vfd_number;
  USIGN8     obj_code;
  USIGN8     dummy;
  T_ACC_SPEC acc_spec;
} T_OV_READ_LOC_REQ;


typedef struct T_OV_READ_LOC_CNF
{
  USIGN32            vfd_number;          /* vfd number                    *
  T_OBJECT_DESCR     obj_descr;           /* object description            *
} T_OV_READ_LOC_CNF;


/*****************************************************************************
/*************   VFD-SUPPORT-MANAGEMENT                  *********************
/*****************************************************************************

/*--- Create-VFD-Service ---------------------------------------------------*
typedef struct T_VFD_CREATE_REQ
{
   USIGN32      vfd_number;                          /* vfd number           *
   STRINGV      vendor_name[MAX_VFD_STRING_LENGTH];  /* vendor name          *
   STRINGV      model_name[MAX_VFD_STRING_LENGTH];   /* model-name           *
   STRINGV      revision[MAX_VFD_STRING_LENGTH];     /* revision no          *
   USIGN8       profile_number[2];                   /* profile number       *
} T_VFD_CREATE_REQ;

typedef struct T_VFD_CREATE_CNF
{
   USIGN32      vfd_number;                          /* vfd number           *
} T_VFD_CREATE_CNF;


/*--- VFD-Set-Physical-Status ----------------------------------------------*
typedef struct T_VFD_SET_PHYS_STATUS_REQ
{
   USIGN32      vfd_number;                  /* vfd number                   *
   USIGN8       physical_status;             /* physical status              *
   USIGN8       dummy;                       /* alignment byte               *
} T_VFD_SET_PHYS_STATUS_REQ;

typedef struct T_VFD_SET_PHYS_STATUS_CNF
{
   USIGN32      vfd_number;                  /* vfd number                   *
} T_VFD_SET_PHYS_STATUS_CNF;


/*--- Status-Service -------------------------------------------------------*
typedef struct T_VFD_STATUS_CNF
{
   USIGN8       logical_status;              /* logical status               *
   USIGN8       physical_status;             /* physical status              *
   USIGN8       local_detail[3];             /* local detail                 *
   USIGN8       dummy;                       /* alignment byte               *
} T_VFD_STATUS_CNF;


/*--- Unsolicited-Status-Service -------------------------------------------*
typedef struct T_VFD_UNSOL_STATUS_REQ
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
{
   USIGN8       priority;                    /* priority                  *
   USIGN8       logical_status;              /* logical status            *
   USIGN8       physical_status;             /* physical status           *
   USIGN8       dummy1;                      /* alignment byte            *
   USIGN8       local_detail[3];             /* local detail              *
   USIGN8       dummy2;                      /* alignment byte            *
} T_VFD_UNSOL_STATUS_REQ;


/*--- Identify-Service ------------------------------------------------*
typedef struct T_VFD_IDENTIFY_CNF
{
   STRINGV   vendor_name[MAX_VFD_STRING_LENGTH]; /* producer of the device   *
   STRINGV   model_name[MAX_VFD_STRING_LENGTH];  /* model-name of the device *
   STRINGV   revision[MAX_VFD_STRING_LENGTH];    /* revision of the device   *
} T_VFD_IDENTIFY_CNF;



/*****************************************************************************
/*      Copyright (C) SOFTING GmbH 1992,1993,1994 All Rights Reserved        *
/*                                                                           *
/*           PROFIBUS INTERFACE DATA TYPES and DATA STRUCTURES               *
/*                                                                           *
/*  Filename            :   pb_if.h                                          *
/*  Version             :   4.01A                                            *
/*  Date                :   01.02.94                                         *
/*  Author              :   Matthias Boettcher                               *
/*                                                                           *
/*  Description :                                                            *
/*                                                                           *
/*  This headerfile contains the external data types and data structures of  *
/*  the PROFIBUS COMMUNICATION INTERFACE                                     *
/*                                                                           *
/*****************************************************************************



/*****************************************************************************
/*************   PROFIBUS INTERFACE DEFINITIONS          *********************
/*****************************************************************************


#define CON_IND_RECEIVED            1   /* indication or confirmation        *
                                        /* has been received                 *
#define NO_CON_IND_RECEIVED         0   /* nothing has been received         *



/*****************************************************************************
/*************   PROFIBUS SERVICE DESCRIPTOR  (PSD)      *********************
/*****************************************************************************

typedef struct T_PROFI_SERVICE_DESCR

  USIGN16      comm_ref;                  /* communication reference          *
  USIGN8       layer;                     /* layer                            *
  USIGN8       service;                   /* service identifier               *
  USIGN8       primitive;                 /* service primitive                *
  INT8         invoke_id;                 /* invoke id                        *
  INT16        result;                    /* service result POS or NEG        *
  T_PROFI_SERVICE_DESCR;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*****************************************************************************
/*************    external functions from the file "profi.c"   ***************
/*****************************************************************************

/*
*       The communication between host and controller is performed
*       using some common memory. Softing has specified the data structures and
*       the synchronization mechanism to be used as "Common Memory Interface"
*       (CMI). The driver at the host should provide the functions described
*       below in order to offer a standardized procedural interface to the
*       communication.
*/


FUNCTION extern INT16 init_profibus
            (
          IN  USIGN32 h_dpr_base_address, /* host base address of DPR       */
          IN  USIGN16 reserved,           /* for internal use               */
          IN  BOOL    hw_reset            /* TRUE reset PROFIBUS controller */
              );

/*---------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to reset the CP5480 PROFIBUS controller board and to
initialize the PROFIBUS-Host-Interface

Possible return values:
- E_OK                          profibus interface is initialized
- E_NO_CNTRL_RES                controller does not respond
- E_INVALID_CNTRL_TYPE_VERSION  invalid controller type
---------------------------------------------------------------------------*



FUNCTION extern INT16 profi_snd_req_res
          (
          IN T_PROFI_SERVICE_DESCR FAR *psd_ptr,        /* pointer to PSD  */
          IN VOID                  FAR *data_ptr,       /* pointer to data */
          IN BOOL                      dummy            /* dummy           */
                );

/*---------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION
This function is used to send a PROFIBUS service request or a service response
to the communication.


INPUT:  psd_ptr              -> pointer to PROFI-SERVICE-DESCRIPTION-BLOCK
        data_ptr             -> pointer to service specific data

Possible return values:

- E_OK                       -> no error occured

- E_FATAL_CMI_ERROR          -> unrecoverable error in CMI
- E_FATAL_LAYER2_ERROR       -> unrecoverable error in LAYER2
- E_FATAL_LAYER7_ERROR       -> unrecoverable error in LAYER7
  E_FATAL_OS_ERROR           -> unrecoverable error in OS

- E_INVALID_LAYER            -> invalid layer
- E_INVALID_SERVICE          -> invalid service identifier
  E_INVALID_PRIMITIVE        -> invalid service primitive
- E_INVALID_COMM_REF         -> invalid communication reference
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
- E_INVALID_FMS_COMM_REF      -> invalid FMS comm. reference
- E_INVALID_FMA_COMM_REF      -> invalid FMA7 comm. reference
- E_RESOURCE_UNAVAILABLE      -> no resource available
- E_NO_PARALLEL_SERVICES      -> no parallel services allowed
- E_SERVICE_CONSTR_CONFLICT   -> service temporarily not executable
- E_SERVICE_NOT_SUPPORTED     -> service not supported in subset
- E_SERVICE_NOT_EXECUTABLE    -> service not executable
- E_NO_CNTRL_RES              -> controller does not respond  (CMI_TIMEOUT)
- E_INVALID_DATA_SIZE         -> not enough cmi data block memory
- E_INVALID_CMI_CALL          -> invalid CMI call
- E_CMI_ERROR                 -> fatal error in CMI

-------------------------------------------------------------------------*



FUNCTION extern INT16 profi_rcv_con_ind
         (
         IN T_PROFI_SERVICE_DESCR FAR *psd_ptr,        /* pointer to psd  */
         IN VOID                  FAR *buffer_ptr,     /* pointer to data */
         IN USIGN16               FAR *buffer_len      /* length of data  */
         );

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

This function is used to receive a PROFIBUS service indication or a
service confirmation from the communication.


INPUT:  psd_ptr          -> pointer to PROFI-SERVICE-DESCRIPTION-BLOCK
        buffer_ptr       -> pointer to service specific data
        buffer_len       -> length of service specific data

Possible return values:

- CON_IND_RECEIVED      -> a confirmation or indication has been received
- NO_CON_IND_RECEIVED   -> no confirmation or indication has been received

- E_FATAL_CMI_ERROR     -> unrecoverable error in CMI
- E_FATAL_LAYER2_ERROR  -> unrecoverable error in LAYER2
- E_FATAL_LAYER7_ERROR  -> unrecoverable error in LAYER7
- E_FATAL_OS_ERROR      -> unrecoverable error in OS

- E_INVALID_DATA_SIZE   -> size of data block provided not sufficient

- E_CMI_ERROR           -> error occured in CMI
- E_INVALID_CMI_CALL    -> invalid CMI call
- E_CMI_ERROR           -> fatal error in CMI

-------------------------------------------------------------------------*/

/*****************************************************************************/
/*      Copyright (C) SOFTING GmbH 1992,1993,1994 All Rights Reserved        */
/*                                                                           */
/*              PROFIBUS BASIC DATA TYPES                                    */
/*                                                                           */
/*  Filename            :   pb_type.h                                        */
/*  Version             :   4.01A                                            */
/*  Date                :   01.02.94                                         */
/*  Author              :   Matthias Boettcher                               */
/*                                                                           */
/*  Description : This file contains the PROFIBUS Basic-Data-Types           */
/*                and the 'PORTIERUNGS-PARAMETER'                            */
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
/*                                                                          */
/*  date      name      change                                              */
/*  ------------------------------------------------------------------------ */
/*  22.10.93  Boe       remove the following typedefs                       */
/*                      - typedef     unsigned char       BYTE              */
/*                      - typedef     unsigned short      WORD              */
/*                      - typedef     unsigned long       DWRD              */
/*                      - typedef     unsigned char       ORD08             */
/*                      - typedef     unsigned short      ORD16             */
/*                      - typedef     unsigned long       ORD32             */
/*                      - typedef     signed char         INT08             */
/*                                                                          */
/****************************************************************************/

#define     FAR                     far     /* for application in medium model */


/* --- PROFIBUS Basic-Data-Types ------------------------------------------ */
#define     VOID                    void

typedef     unsigned char           bool;
#define     BOOL                    bool

typedef     unsigned char           USIGN8;
typedef     unsigned short          USIGN16;
typedef     unsigned long           USIGN32;

typedef     signed char             INT8;
typedef     signed short            INT16;
typedef     signed long             INT32;

typedef     char                    STRINGV;
typedef     float                   FLOAT;


/* --- define  escape jump ------------------------------------------------ *
/* --- not necessary for the application ---------------------------------- *
#define escape goto endlbl

/* --- ok return value ---------------------------------------------------- *
#define E_OK                    0               /* no errors */

/* --- boolean type ------------------------------------------------------- *
#define TRUE                    0xFF
#define FALSE                   0x00

/* --- service result ----------------------------------------------------- *
#define NEG                     1       /* result for negative response      *
#define POS                     0       /* result for positive response      *

/* --- service priority --------------------------------------------------- *
#define LOW                     0       /* high priority                     *
#define HIGH                    1       /* low priority                      *


/* --- service primitives ------------------------------------------------- *
#define REQ                     0       /* request          */
#define CON                     1       /* confirmation     */
#define IND                     2       /* indication       */
#define RES                     3       /* response         */


/* --- layer and abort identifiers ---------------------------------------- *
#define USR                     0       /* identifier USER                  */
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
#define FMS                     1       /* identifier FMS                       */
#define LLI_USR                 1       /* identifier LLI_USER (FMS / FMA7)     */
#define LLI                     2       /* identifier LLI                       */
#define FDL                     3       /* identifier FDL                       */
#define FMA7                    4       /* identifier FMA7                      */
#define FMA2                    5       /* identifier FMA2                      */
#define FMS_USR                 6       /* identifier FMS-USER                  */
#define FMA7_USR                7       /* identifier FMA7-USER                 */


/*****************************************************************************
/*                          Implementation Constants                         *
/*                                                                           *
/* The constants given below define the sizes of various data structures in  *
/* the PROFIBUS protocol software and thus influence memory consumption.     *
/*                                                                           *
/* NOTE: Do not change the following constants without recompiling the       *
/*       the PROFIBUS protocol software on the communication controller      *
/*****************************************************************************


/* --- macro to calculate MAX_xxxx_NAME_LENGTH ---------------------------- *

/* This macro calculates the internal sizes of byte arrays in a way that the *
/* desired alignment on byte, word or long word boundaries is achieved.      *
/* The alignment is specified by the constant ALIGNMENT (e. g. longword = 4) *

#define ALIGNMENT                       2   /* alignment on word boundary      *

#define _NAME_LENGTH(length) (length + ALIGNMENT - (length % ALIGNMENT))

/*     NOTE: According to PROFIBUS the maximum sizes of the character strings*
/*           given below may vary between 0 and 32 bytes.                    *

#define VFD_STRING_LENGTH               32
#define IDENT_STRING_LENGTH             32

#define ACCESS_NAME_LENGTH              32
#define OBJECT_NAME_LENGTH              32
#define EXTENSION_LENGTH                32
#define EXECUTION_ARGUMENT_LENGTH       32
#define ERROR_DESCR_LENGTH              32
#define KBL_SYMBOL_LENGTH               32
#define KBL_EXTENSION_LENGTH             2


#define MAX_FMS_PDU_LENGTH      241     /* max size of the FMS-PDU-Buffer       *
#define MAX_VAR_LIST_ELEMENTS    10     /* max count of variable list elements *
#define MAX_DOM_LIST_ELEMENTS    10     /* max count of domain list elements   *
#define MAX_VAR_RECORD_ELEMENTS  10     /* max count of record elements        *

#define MAX_VFD                  5    /* max supported VFDs                     *

#define MAX_COMREF              31    /* max supported communication references *
#define MAX_KBL_LEN             30    /* max entries in KBL                     *

#define MAX_PARA_LOC_SERVICES    5    /* max parallel local FMS-Services        *
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
- FMA7_SET_VALUE_LOC
- FMA7_READ_VALUE_LOC
- FMA7_LSAP_STATUS_LOC
- FMA7_IDENT_LOC
- FMA7_EVENT
- FMA7_RESET
- FMA7_PROFIBUS_EXIT
- FMA7_SET_BUSPARAMETER
- FMA7_SET_STATISTIC_CTR
- FMA7_READ_BUSPARAMETER
- FMA7_READ_STATISTIC_CTR
- FMA7_GET_LIVE_LIST

possible return values:
- Data-length

------------------------------------------------------------------------------*
{
LOCAL_VARIABLES

FUNCTION_BODY

switch(service)
{
   case FMA7_INITIATE:
        return(sizeof(T_FMA_INIT_ERR_CNF));
        break;

   case FMA7_READ_KBL_REM:
   case FMA7_INIT_LOAD_KBL_REM:
   case FMA7_LOAD_KBL_REM:
   case FMA7_SET_VALUE_REM:
   case FMA7_READ_VALUE_REM:
   case FMA7_LSAP_STATUS_REM:
   case FMA7_IDENT_REM:
   case FMA7_READ_KBL_LOC:
   case FMA7_INIT_LOAD_KBL_LOC:
   case FMA7_LOAD_KBL_LOC:
   case FMA7_SET_VALUE_LOC:
   case FMA7_READ_VALUE_LOC:
   case FMA7_LSAP_STATUS_LOC:
   case FMA7_IDENT_LOC:

   case FMA7_EVENT:
   case FMA7_RESET:
   case FMA7_PROFIBUS_EXIT:
   case FMA7_SET_BUSPARAMETER:
   case FMA7_SET_STATISTIC_CTR:
   case FMA7_READ_BUSPARAMETER:
   case FMA7_READ_STATISTIC_CTR:
   case FMA7_GET_LIVE_LIST:
   case FMA7_SET_CONFIGURATION:
        return(sizeof(T_ERROR));
        break;


   case FMA7_TERM_LOAD_KBL_LOC:
   case FMA7_TERM_LOAD_KBL_REM:
        return(sizeof(T_KBL_ERROR));
        break;

   default:
        return(0);
        break;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

BIBLIOGRAFIA

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

BIBLIOGRAFIA

* **PROFIBUS COMMUNICATION INTERFACE LAYER 7 FOR CP5480-A1 CONTROLLER.** Manual de referencia de SOFTING GmbH.

* **PROFIBUS STANDARD.** DIN 19 245.

* **PROFIBUS: The Fieldbus for Industrial Automation.** Kaus Bender

* **COMPUTER NETWORKS.** Andrew S. Tanenbaum

* **APUNTES DE SISTEMAS DISTRIBUIDOS Y REDES OSI.** Domingo Marrero Marrero.

* **C: A REFERENCE MANUAL.** Samuel P. Harbison-Guy L.Steele JR.

* **APUNTES SOBRE EL EQUIPO W_90** de la empresa ISOLUX-WAT.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006

```
}

return(0);
}


FUNCTION extern INT16 fmagdl_get_data_len
          (
            IN  INT16      result,          /* Service-Result       */
            IN  USIGN8     service,         /* Service              */
            IN  USIGN8     primitive,       /* Service-Primitive    */
            IN  USIGN8 FAR *data_ptr,       /* pointer to data      */
            OUT INT16      *data_len        /* length of data       */
          )

/*-------------------------------------------------------------------------
FUNCTIONAL_DESCRIPTION

this function is used to return the data length of FM7-SERVICES

possible return values:
- Data-length

-------------------------------------------------------------------------*/
{
LOCAL_VARIABLES


FUNCTION_BODY

*data_len = 0;

switch (service)
{
   case FMA7_READ_KBL_LOC:
   case FMA7_INIT_LOAD_KBL_LOC:
   case FMA7_LOAD_KBL_LOC:
   case FMA7_TERM_LOAD_KBL_LOC:
   case FMA7_READ_KBL_REM:
   case FMA7_INIT_LOAD_KBL_REM:
   case FMA7_LOAD_KBL_REM:
   case FMA7_TERM_LOAD_KBL_REM:
        if (result == POS) *data_len = fmagdl_get_kbl_data_len(service,primiti
        else               *data_len = fmagdl_get_error_data_len(service);
        break;

   case FMA7_SET_VALUE_LOC:
   case FMA7_READ_VALUE_LOC:
   case FMA7_SET_BUSPARAMETER:
   case FMA7_SET_STATISTIC_CTR:
   case FMA7_READ_BUSPARAMETER:
   case FMA7_READ_STATISTIC_CTR:
   case FMA7_SET_VALUE_REM:
   case FMA7_READ_VALUE_REM:
        if (result == POS) *data_len = fmagdl_get_s_r_value_data_len(service,p
        else               *data_len = fmagdl_get_error_data_len(service);
        break;

   case FMA7_LSAP_STATUS_LOC:
   case FMA7_LSAP_STATUS_REM:
        if (result == POS) *data_len = fmagdl_get_lsap_status_data_len(service
        else               *data_len = fmagdl_get_error_data_len(service);
        break;
```

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2006