UNED. Escuela Técnica Superior de Ingenieros Industriales

Departamento de Ingeniería Eléctrica, Electrónica y de Control

Curso de Doctorado: Protocolos de Comunicación entre sistemas de Información

Profesor del curso: Manuel Castro

Alumno: Jerónimo Quesada Castellano

Expediente: MD-10712

Curso: 1999/2000

Trabajo final

"Bus CAN: estado de buses industriales y aplicaciones"

Septiembre del 2000

(revisión 1.0)

Origen: J. Quesada **Destino:** D. Manuel Castro

j.quesada@computer.org

Departamento I+D. Corporación ZIGOR S.A.
http://www.zigor.com/

INDICE

1. Introducción

Este trabajo pretende

- Establer una perspectiva de las características, evolución y situación de las redes aplicables en sistemas distribuidos de control y supervisión industriales.

- Revisar los requerimientos de dichas redes en cada uno de los niveles jerárquicos, sin olvidar características no estrictamente técnicas que determinan su idoneidad.

- Estudiar el bus CAN (Controller Area Network), como base para arquitecturas de bus industrial especialmente idóneas para aplicaciones de tiempo real distribuido, sistemas de supervisión y control en el ámbito de célula de fabricación, y redes normalizadas para interconexión de sensores y actuadores inteligentes.

- Exponer, a través de una aplicación práctica sencilla, la problemática involucrada en la definición, desarrollo y puesta a punto de un nodo para bus de campo.

- Se presenta, finalmente, un ejemplo de aplicación de bus CAN en sistemas de control de tiempo real distribuidos y un método de sincronización de nodo captador y actuador.

2. Redes y buses de campo industriales.

2.1 Historia, evolución y perspectivas

Quién haya vivido la evolución en una planta industrial dinámica en el último cuarto de siglo ha tenido la oportunidad de seguir un proceso de cambios revolucionarios en la automatización de la fabricación. La automatización de los 70 se basaba en cuadros repletos de contactores electromecánicos con interconexión de cada contacto, cada sensor de posición, cada interruptor, cada indicador, por líneas de cable dedicada. Se utilizaban cableados de maniobra a 24 o 48 V. Los dispositivos de instrumentación analógica, sensores e indicadores, se enlazaban por la clásica línea de corriente analógica 4-20 mA. La informática corporativa, si existía, se basaba en computadores de generaciones previas a la aparición del IBM PC y se dedicaba casi en en exclusiva a tareas administrativas. Los cambios de configuración exigían recableado del sistema y la adquisición de datos se realizaba manualmente en la mayoría de los casos. Los formularios de datos de proceso habían de completarse a mano por los operadores.

La aparición del microprocesador y su evolución en relación precio/prestaciones hasta la situación actual (en la que se pueden conseguir dispositivos con importante capacidad de proceso con precios inferiores a 5 Euros) ha conducido a una nueva generación de dispositivos de control industrial inteligentes. Cada sensor y actuador pueden incluir su propia unidad de proceso digital, basada generalmente en microprocesador.

Inevitablemente el enlace entre estas unidades de proceso, dispersas geográficamente en una planta industrial, requiere de una red de comunicaciones. Estas redes, orientadas al enlace de dispositivos de control industrial, son conocidas genéricamente como redes de campo o buses de campo (“fieldbuses”). Desde la introducción del PLC (“Programmable Logic Controller” o autómata programable), como computador especializado en automatización, los cuadros de

maniobra han sido sustituidos por estos dispositivos. En paralelo el ordenador personal, que se ha extendido por las plantas de fabricación, en su versión industrial, compite con el PLC como procesador de adquisición de datos y control. Todos estos dispositivos inteligentes se enlazan entre sí y con la red informática corporativa por medio de buses de campo y redes de comunicaciones estratificadas en diversos niveles. Finalmente toda la red se puede enlazar a redes de area amplia o a la red universal (Internet) para compartición de datos, telegestión etc.

Pero no sólo las grandes plantas industriales siguen este proceso, las redes de campo se están introduciendo en entornos de menor amplitud. Una pequeña planta depuradora de aguas residuales, el ascensor de un edificio, y multitud de sistemas análogos pueden contar en la actualidad con varios procesadores enlazados por un bus de campo y finalmente conectados a una red externa para su telegestión.

Hacia el futuro, se vislumbra una evolución hacia una mayor cantidad de elementos inteligentes interconectados, sistemas muy simples como una lámpara o un interruptor pueden contar con un procesador digital. Por otra parte se extenderán nuevas normas de comunicación, en especial las inalámbricas (BlueTooth, Wireless Ethernet etc.). Los sistemas distribuidos incorporarán capacidad de autoreconocimiento y negociación entre nodos, funciones de autodiagnóstico, mantenimiento automático, base de datos de identificación etc.

Un motor de próxima generación incluirá un procesador no sólo para gestionar sus funciones básicas, sino para efectuar algoritmos de automantenimiento predictivo, mantener una base de datos con sus características que permita la gestión de repuestos automática, capacidad de inclusión automática en un sistema enlazado por bus de campo (“plug and play”). Puede acabar avisando directamente al mecánico de mantenimiento para que le cambie un rodamiento ruidoso y realizar él mismo el pedido de dicho rodamiento (aunque no conviene exagerar, para una divertida reflexión sobre la exageración en el uso de dispositivos inteligentes interconectados se recomienda el artículo “Reflections” de R.W.Lucky en IEEE Spectrum Marzo 1999).

2.1.1 Las reglas del juego

Se han establecido unas leyes fundamentales en la industria de las tecnologías de la información que han demostrado ser una buena referencia para estimar la evolución futura y explican la actual revolución basada en esas tecnologías.

- **Ley de Moore**: formulada por Gordon Moore de Intel a principios de los 70. La capacidad de proceso de los circuitos integrados digitales se dobla cada 18 meses. Corolario, los sistemas basados en microprocesadores aumentan su potencia, y el precio para un nivel de potencia de cálculo dado se reduce a la mitad cada 18 meses.

- **Ley de Gilder**. George Gilder, escritor y profeta de las nuevas tecnologías, estableció que el ancho de banda de los sistemas de comunicaciones se triplica cada doce meses.

- **Ley de Metcalfe**. Atribuida a Robert Metcalfe, involucrado en la creación de Ethernet y fundador de 3Com: El valor de una red de comunicaciones es proporcional al cuadrado del número de nodos. Es decir, al crecer la red, el valor de añadido de un nodo conectado a ella crece cuadráticamente, mientras el coste por nodo se mantiene o incluso se reduce.

Estas leyes explican la evolución de los procesos productivos hacia sistemas basados en multitud de nodos cada vez más inteligentes y potentes (Ley de Moore), con un aumento continuo de la transferencia de información entre ellos (Ley de Gilder) y cada vez más interconectados (Ley de Metcalfe). Un ejemplo, para un fabricante de ascensores es interesante evolucionar hacia elementos de control del ascensor cada vez más inteligentes, conectar entre sí esos elementos y con el exterior del edificio por líneas de comunicaciones, enlazar el ascensor a redes de area extensa (incluso Internet) para telegestión, telediagnóstico y telemantenimiento. El valor añadido crece como una función de orden superior a la de crecimiento del coste.

En todo caso la asombrosa capacidad de evolución de los sistemas inteligentes interconectados está modulada por la capacidad de evolución paralela de los procesos de negocio y la propia capacidad humana de absorción de nuevas tecnologías. En el ejemplo de la empresa de ascensores el cambio a nuevos sistemas, para que sea eficaz y se rentabilice, ha de contar con un entorno de gestión de servicio postventa que aproveche al máximo los nuevos potenciales. La implantación de esos sistemas puede suponer una revolución en personal técnico y en procesos de gestión.

Es ya irrebatible que los sistemas de información avanzados, entre ellos los buses de campo, contribuyen a la evolución hacia nuevas formas de organización y gestión empresariales. Los procesos productivos son más flexibles, los cambios de planificación son mucho más frecuentes. Hasta llegar al concepto de empresa manejada por eventos, en la que no hay un flujo contínuo de información y materiales desde compras a producto o cliente. Se producen eventos (demandas concretas de clientes, nichos de mercado) que generan nuevos eventos que se transmiten para su tratamiento a entes autónomos de la organización, algunos de estos entes recogen datos de los demás, los globalizan y coordinan acciones. No es un funcionamiento muy distinto al de un bus de campo moderno.

2.2 La pirámide CIM

El ideal de factoría completamente automatizada (Computer Integrated Manufacturing) se representa como una pirámide en la que en los niveles bajos se encuentran los sensores, motores y otros elementos de accionamiento. En los niveles intermedios se interconectan estos elementos para funcionar cooperativamente realizando funciones más o menos sincronizadas. Finalmente la información de estado, registros históricos, etc. se transmite a la red informática técnico-administrativa, de esta red se recogen también los datos de partida y consignas para los niveles inferiores. En el nivel inferior se situaría un humilde sensor de temperatura, en el superior el director de planta obtiene datos de la cantidad y calidad de la producción.

En la literatura se presentan diferentes modelos de “pirámide CIM”, con más o menos niveles, e incluso con contradicciones entre sí en cuanto a nomenclatura, se habla en unos casos de “field bus” como el bus que interconecta dispositivos de campo y en otros como el bus en que se basa la capa de control.

Un posible modelo de seis niveles es el siguiente:

1. Bus de sensores: El nivel más bajo, conecta sensores simples de bajo costo como interruptores on/off. Transmite información muy simple (bits) y requiere poca o nula capacidad de proceso en el sensor.

2. Bus de dispositivos: Es la categoría más amplia y proporciona servicios de comunicación para sensores, actuadores y terminales inteligentes.

3. Bus de campo: Un nivel hacia arriba del bus de dispositivos. Soporta la transmisión de mayor volumen de información a velocidad más reducida. Se requiere mayor capacidad de proceso en los dispositivos. En este nivel se gestionan automatismos al nivel de célula y se cierran bucles de regulación básicos.

4. Bus de control: orientado a la comunicación entre grandes dispositivos de control de alto nivel como PLCs u ordenadores de proceso. Gestiona de forma integral el proceso de fabricación.

5. Red ofimática: La tradicional red ofimática de empresa. Predomina TCP/IP sobre Ethernet.

6. Red de area Amplia- Enlace de la red de empresa con redes de área amplia privadas o públicas (Internet).

La tendencia actual de las redes de comunicaciones industriales es a la reducción de estos niveles, Ethernet y TCP/IP están penetrando en los niveles 4 (bus de control) e incluso en el 3 (bus de campo). En determinadas aplicaciones el bus de dispositivos (nivel 2) se enlaza directamente a la red de área amplia (Nivel 6). Como ejemplo considérese el de una estación de aguas residuales sencilla conectada vía modem a Internet para su supervisión remota.

Por todo ello en este trabajo se hará referencia, fundamentalmente, a un modelo simplificado de tres niveles, en el nivel inferior (bus de campo) se engloban los niveles 1,2 y 3 del modelo de 6 niveles, sobre él se sitúa el nivel de control o informática de fabricación y finalmente la informática corporativa o nivel 5 que enlaza con redes de área amplia

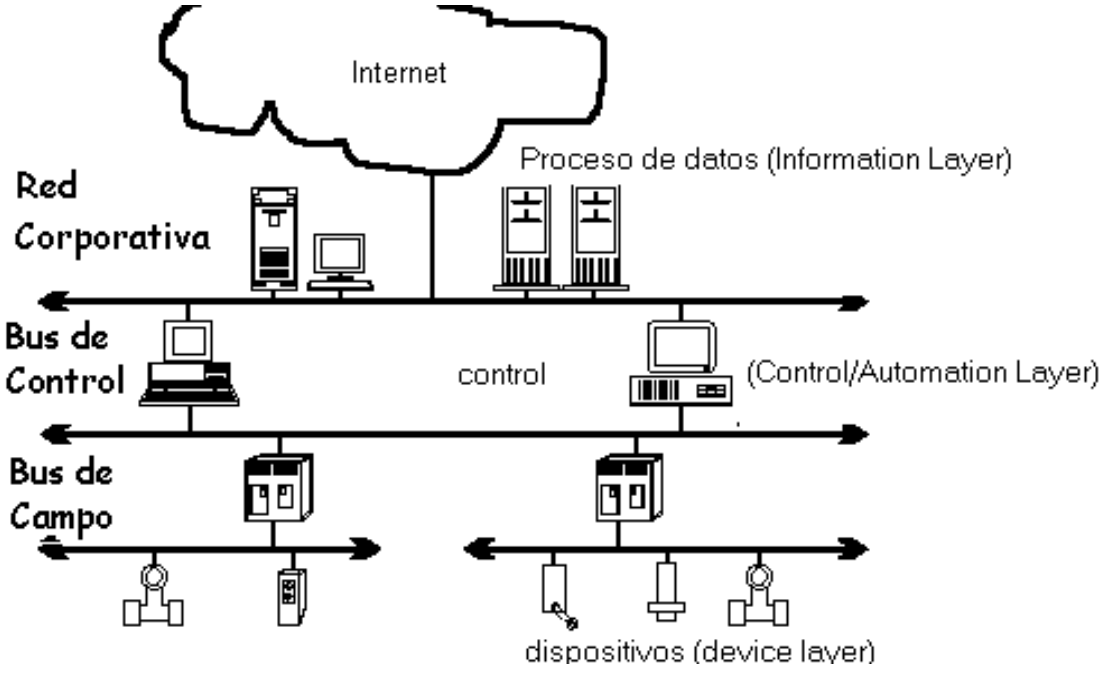


Figura 1. Niveles CIM (Inspirada en una figura similar en [21])

- Red de proceso de datos: En la capa superior se incluye la red informática corporativa que ,sobre todo, procesa datos administrativos, se enlaza a la capa de control o automatización para recoger datos de proceso y transmitir nuevas configuraciones o datos de partida para la red de automatización de planta. Enlaza con las redes de área amplia Se denominará ***red corporativa***

- Red de automatización y de control: La capa intermedia incluye los ordenadores, terminales, autómatas programables (PLCs), que establecen el sistema de control y automatización de planta. Entre ellos mueven información de tamaño medio y existen ciertos requisitos de tiempo real. Se denomirá ***bus de control***

- Bus de campo o célula: La capa inferior enlaza actuadores, sensores y dispositivos de control. Es una red que muchas veces tiene exigencias de tiempo real estricto, la información se transmite en cortos mensajes de alta frecuencia. Se denominará a este conjunto ***bus de campo.***

Esta jerarquía puede describir sistemas industriales, pero también otros en sectores como construcción o servicios. En un edificio inteligente cada ascensor puede constituir una célula de proceso con sensores (detectores de puerta abierta) y actuadores (motor, freno) enlazados por un bus de campo, los ascensores se enlazan entre sí por un bus de control que permite sincronizar y optimizar su funcionamiento cubriendo de forma conjunta y óptima las demandas de los usuarios, finalmente la red de control de ascensores se enlaza a la red de gestión del edificio inteligente de forma que puedan centralizarse los datos de funcionamiento, avisos de avería, emergencias etc.

Entre las redes de nivel superior cabe señalar la fuerte tendencia hacia la utilización de TCP/IP sobre Ethernet, propuestas como las redes especializadas MAP y TOP no se han extendido demasiado en la práctica, sobre todo en pequeña y mediana industria.

Este trabajo se centrará sobre todo en el nivel de bus de campo o célula, nivel en el que el bus CAN tiene su principal área de aplicación.

3. Conceptos fundamentales

3.1 El modelo ISO para interconexión de sistemas

No hay trabajo sobre redes de comunicaciones, que se precie, que no haga referencia al modelo ISO y éste no podía ser una excepción

En 1977 se estableció un comité ISO para desarrollar una arquitectura que estableciera un modelo de referencia para la "interconexión de sistemas abiertos" (Open Systems interconection, OSI). El estándar final, ISO 7498 no se publicó hasta 1984. CCITT (ITU) publicó una norma equivalente como X200.

La arquitectura se basa en la definición de una serie de capas que se superponen. Se siguen unos principios básicos: comunicación "virtual" extremo a extremo entre capas del mismo nivel, las capas superiores utilizan servicios de la capa inmediatamente inferior, cada capa incorpora a la información que proviene de la capa superior los datos adicionales para gestionar la comunicación con la capa simétrica del otro extremo.

3.1.1 El modelo

La esencia del modelo es que los datos se mueven entre dos procesos de aplicación, A y B, que pueden ejecutarse en nodos diferentes de la red de comunicaciones, por métodos que son desconocidos y transparentes para las propias aplicaciones. Las aplicaciones se enlazan al sistema de comunicaciones usando los servicios de la capa superior del modelo de referencia, la capa de Aplicación.

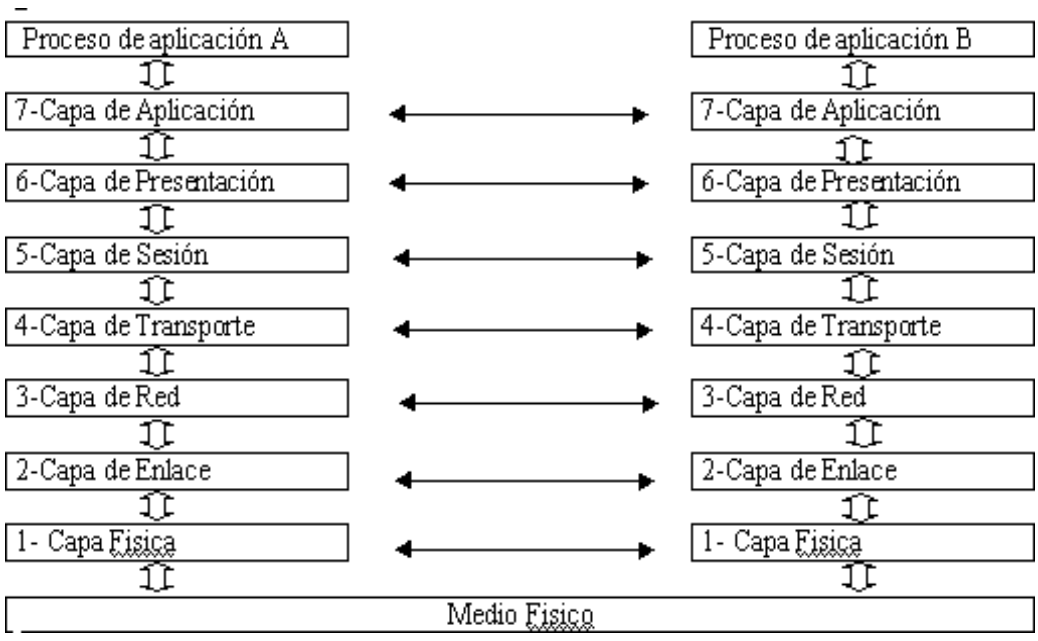


Figura 2. Modelo ISO

Cada capa usa los servicios de la capa inmediatamente inferior y ofrece servicios a la inmediatamente superior. Cada capa opera como si enlazara directamente con la capa gemela en el otro proceso. Los datos se mueven entre capas como Unidades de Datos (Data Units, DU), cada capa añade información adicional de control que utiliza para organizar la comunicación con la capa gemela en el proceso remoto.

La funcionalidad de cada tabla se resume en la siguiente tabla:

Modelo ISO	
Capa	**Función**
Aplicación	Servicios al programador de aplicaciones como transferencia de ficheros, variables compartidas en red, ejecución de procesos remotos, …
Presentación	Cambios de formato de datos, codificación, encriptación-desencriptación
Sesión	Negociación y establecimiento de conexiones
Transporte	Transmisión de paquetes con el otro nodo de modo transparente y fiable (detección de errores, reintento, multiplexación, fragmentación)
Red	Encaminamiento de los paquetes entre nodos a través de redes más o menos complejas
Enlace de datos	Transferencia de unidades de información (tramas), control de acceso al medio, control de flujo
Física	Conversión de bits a señales en medio físico y viceversa

Tabla 1 . Funciones de niveles ISO

En un bus de campo lo ideal para el programador de aplicaciones es acceder un objeto remoto (variable, procedimiento u objeto que combina ambos) de modo transparente y como si perteneciera a la propia aplicación. Este servicio ha de ser proporcionado por la capa de aplicación. Por debajo el software y hardware de protocolo han de:

1. Realizar las transformaciones necesarias en la presentación de datos (por ejemplo intercambio del orden de bytes si los procesadores usan diferente sistema, “big endian” o “little endian”). Lo hace la capa de presentación

2. Abrir y cerrar conexiones lógicas estables entre nodos. La capa de sesión se encarga de ésto.

3. Dividir el flujo de información intercambiado en paquetes, transmitir los paquetes, esperar reconocimientos, retransmitir en caso de error, reunir y ordenar los paquetes recibidos. Lo hace la capa de transporte.

4. Encaminar los paquetes a través del enlace más apropiado entre los disponibles teniendo en cuenta direcciones lógicas y su relación con las direcciones físicas. Capa de Red

5. Empaquetar los datos de usuario en tramas de tamaño adecuado incluyendo y extrayendo la información extra necesaria para direccionamiento y control de errores. Controlar el

acceso ordenado al medio de transmisión. Capa de enlace.

6. Convertir cada bit lógico a la correspondiente señal física (nivel eléctrico, luz, etc.) y viceversa. Capa física.

En el resto de este trabajo se utilizará este modelo como base para revisar las características de buses de campo y exponer normas e implementaciones concretas. Más detalles de modelo pueden obtenerse en las referencias ([1], [2])

Muchos buses industriales se ajustan a un modelo reducido en el que se definen sobre todo las capas 1 (física) y 2 (enlace) y sobre ellas se establece una capa de aplicación que cubre el resto de funciones.

3.2 Arquitectura de nodo

En los buses de campo industriales el medio físico por excelencia es el par trenzado. Aunque las especificaciones contemplan en muchos casos la utilización de otros medios, su utilización en la práctica es mucho más restringida. Quizás la fibra óptica es el medio alternativo más usual cuando las necesidades de inmunidad electromagnética y aislamiento lo aconsejan. En este trabajo se prestará atención sobre todo a la comunicación sobre cables conductores, fundamentalmente sobre par trenzado.

A pesar de usar el cobre como soporte físico final de la comunicación, no se ha de olvidar que la comunicación digital ha supuesto una reducción espectacular en la cantidad de cobre utilizada para la interconexión de dispositivos industriales. A cambio, se han de utilizar elementos de gestión de la comunicación inteligentes, basados en dispositivos lógicos digitales, es decir en chips de silicio. De aquí la frase que define la implantación de buses industriales como “sustitución de cobre por silicio”. Un sistema de control de una célula de proceso automatizada en cualquier sector de la industria, en los años 60-70, podía contar con decenas de contactores electromecánicos de maniobra interconectados con cable de 0,75 mm2 a 1,5 mm2 de sección (no olvidemos la utilización de cobre en los contactos de los contactores y en las bobinas de excitación). Al utilizar un bus digital la reducción de cable de conductores y de los propios contactores es significativa, a cambio, en cada nodo se habrá incluido al menos un dispositivo digital.

En la evaluación técnico-económica de un bus de campo influyen de forma importante los requerimientos en cuanto a plataforma hardware (silicio) y software, el hardware se traduce en coste variable, obviamente el coste de la plataforma hardware ha de multiplicarse por el número de nodos. El coste del software tiene características de coste fijo, es decir, se trata de una inversión en desarrollo que se amortiza sobre la base del número de unidades producidas.

De cara a contar con criterios que sirvan de base para la comparación de distintos buses de campo es conveniente contar con un modelo simple de la arquitectura hardware de un nodo. En la Figura 3 se representa un modelo.

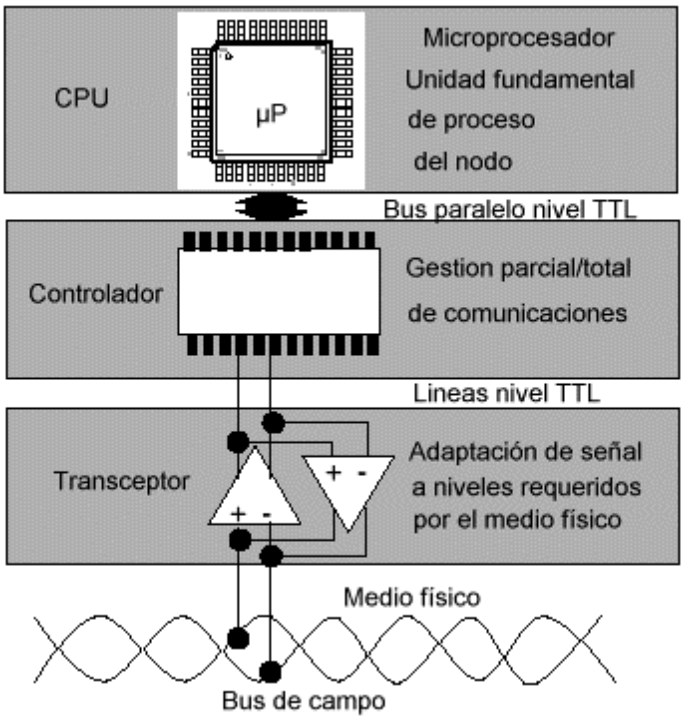


Figura 3 . Arquitectura de un nodo (inspirada en una figura similar en [31])

- El nodo cuenta con un microprocesador o microcontrolador que controla sus funciones específicas y puede, también, gestionar parte del protocolo de comunicaciones. Se denominará, a efectos del modelo, como ***CPU.***

- Puede existir un controlador de comunicaciones que descarga al procesador principal de parte (o toda) la gestión de las comunicaciones. De denomirá ***Controlador***. Este elemento se enlaza a la CPU por líneas digitales con niveles TTL, puede ser el bus del propio microprocesador, un bus serie de enlace de dispositivos (SPI, IIC,...) o una serie de líneas dedicadas.

- Finalmente puede existir un elemento adaptador de niveles de señal que transforma las señales digitales con niveles TTL a las requeridas por el medio físico que soporta las comunicaciones. Se denominará ***Transceptor***. En el caso más común, este medio físico suele ser un par trenzado, pero podría ser un enlace de fibra óptica, la red eléctrica, un enlace inalámbrico vía radio etc. Para algunos de estos tipos de enlace, el transceptor ha de incluir funciones de modulación-demodulación (modem) o similares. El transceptor se enlaza al controlador de comunicaciones por medio de una serie de líneas TTL dedicadas.

Como se verá en apartados posteriores en algunos casos puede no existir controlador de comunicaciones (toda la gestión se realiza por la CPU). Por otra parte, los microcontroladores actuales (microprocesador con periféricos) incluyen en muchos casos algún tipo de controlador de comunicaciones que, si es suficiente para soportar el protocolo concreto, elimina la necesidad de contar con un controlador independiente.

Si la comunicación se realizara utilizando los propios niveles de señal que proporciona el controlador (normalmente niveles eléctricos TTL), no es necesario un transceptor. No es una

solución muy extendida, ya que las líneas TTL de salida de dispositivos digitales no cuentan con la potencia necesaria para soportar líneas de cierta longitud sin deterioro de la señal, además de resultar muy susceptibles la interferencia electromagnética.

3.3 Topologías

Entre las topologías de conexión básicas utilizables en redes industriales se encuentra el enlace punto-punto, el bus, el anillo y la estrella (Figura 4). Como el propio nombre indica, la topología más utilizada en buses industriales es el bus, algunos (INTERBUS) utilizan el anillo y en muchos casos en pequeñas células de fabricación se tienen derivaciones hacia topología en estrella

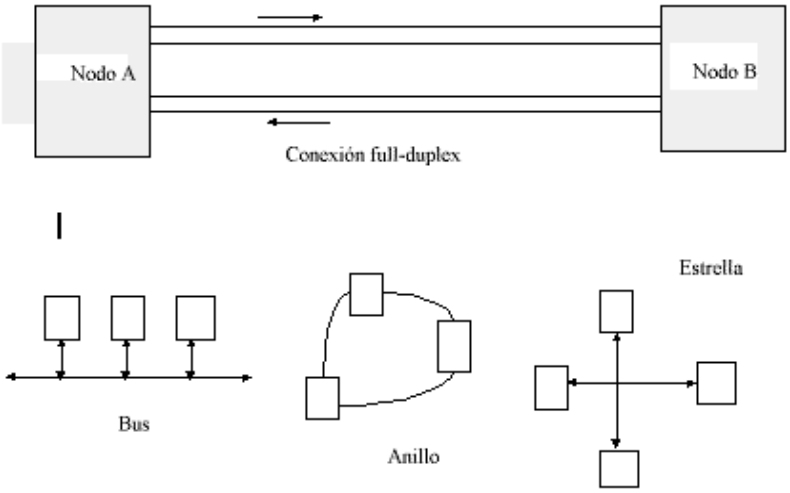


Figura 4 . Topologías de interconexión

3.4 La capa física.

3.4.1 Medio físico

En buses industriales la comunicación se realiza normalmente en banda base y transmisión digital de tipo NRZ (non zero return) o Manchester (estos conceptos pueden revisarse en [3]).

La señalización consiste en conmutar los valores de las variables físicas utilizadas para la comunicación entre una serie de estados. En el caso de comunicación serie sobre conductores eléctricos las variables utilizadas son diferencia de potencial o intensidad de corriente. Es habitual la utilización de dos únicos estados que corresponden a valores binarios ‘1’ o ‘0’ de la información a transmitir.

En buses industriales de campo el medio físico por excelencia es el par trenzado. La fibra óptica (en este caso la magnitud física es la luz) se utiliza para aplicaciones que requieren alta inmunidad a interferencia electromagnética. Como se indicará en apartados sucesivos la técnica de señalización de mayor implantación es la transmisión diferencial de tensión. Es decir, se utiliza como variable eléctrica la diferencia de potencial entre los dos conductores del par trenzado sobre el que se realiza la comunicación. La comunicación en modo común (diferencia de potencial entre conductores y masa de referencia) se utiliza en enlaces no

balanceados tipo RS-232 para cortas distancias, y resulta mucho más susceptible a interferencia. En la comunicación balanceada o diferencial los transceptores soportan unos valores máximos de tensión de modo común que están especificados en las normas.

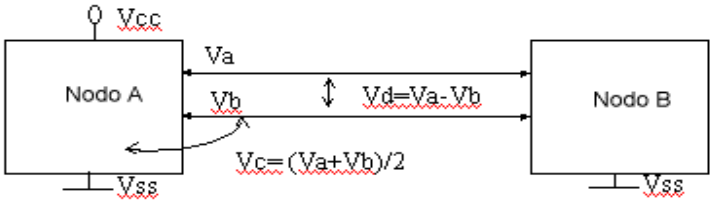


Figura 5. Niveles de tensión de transmisión balanceada

Se presentarán a continución algunos enlaces físicos básicos utilizados en buses industriales.

3.4.2 Conexión por líneas digitales a nivel TTL

La mayoría de los dispositivos digitales actuales se basan en tecnología CMOS, en el caso de microcontroladores es habitual que sus patillas puedan ser configuradas con entrada o salida indistintamente (Figura 6). Los niveles eléctricos habituales son los TTL

Para conexión multipunto los transceptores han de poder situarse en un tercer estado (alta impedancia), de modo que sólo uno de ellos controle la línea en cada instante.

Es posible realizar una comunicación serie enlazando directamente estas entradas-salidas digitales entre nodos. Los enlaces con lógica TTL o análogos no son viables para distancias superiores a unos pocos metros. Se utilizan sobre todo en buses de interconexión de periféricos en tarjetas electrónicas (Buses SPI, IIC,…)

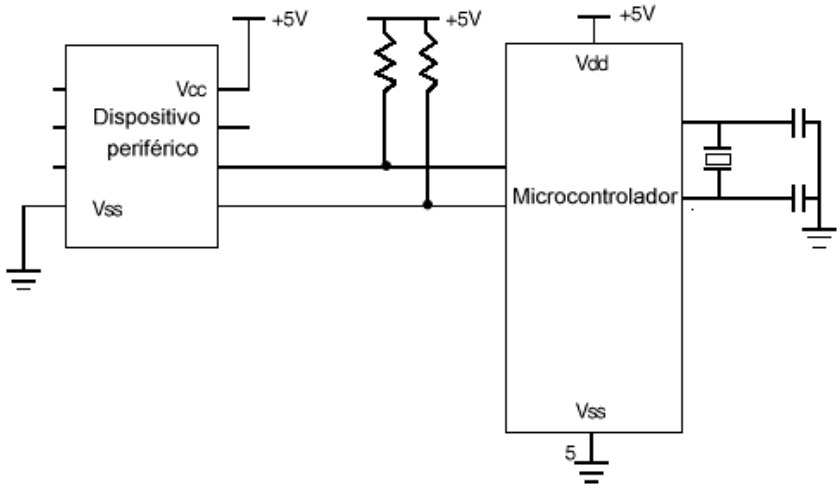


Figura 6. Enlace con niveles TTL

3.4.3 Bucle digital de corriente

Los niveles lógicos se indican por medio de la presencia de una intensidad dada (en el rango de 5-20 mA) o su ausencia (0 mA) en la línea. Existen diversas variantes, es muy usual su utilización en enlaces optoaislados en el que los transceptores básicamente son optoacopladores. Se pueden alcanzar distancias de varios cientos de metros a velocidades de 9600 bps.

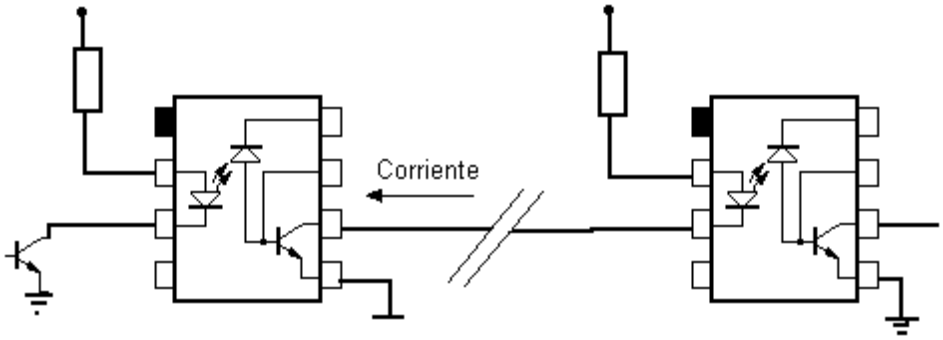


Figura 7. Enlace optoaislado en modo corriente

3.4.4 Tensión con niveles RS-232, no equilibrado

Se utiliza la norma RS-232 con niveles binarios indicados por valores de tensión positiva (del orden de +10V) o negativa (-10 V) respecto al punto común. La norma RS-232, orientada en principio a la conexión entre modems y dispositivos de proceso de datos, se ha utilizado en multitud redes industriales propietarias para enlazar sistemas de adquisición de datos, sensores o actuadores a dispositivos de proceso. La abundancia y bajo coste de transceptores RS-232 (algunos tan clásicos como el Maxim MAX-202) facilitan el diseño de circuitos de enlace utilizando esta norma. Sin embargo al tratarse de transceptores bi-estado (no cuentan con un tercer estado de alta impedancia que permita la conexión de varios transmisores) impide su utilización en bus o estrella. Básicamente su uso se restringe a enlaces punto a punto. Por otra parte la señalización es no equilibrada y es, por tanto, mucho más susceptible a interferencia electromagnética que las líneas equilibradas, su alcance queda reducido a decenas de metros para velocidades de 9600 bps.

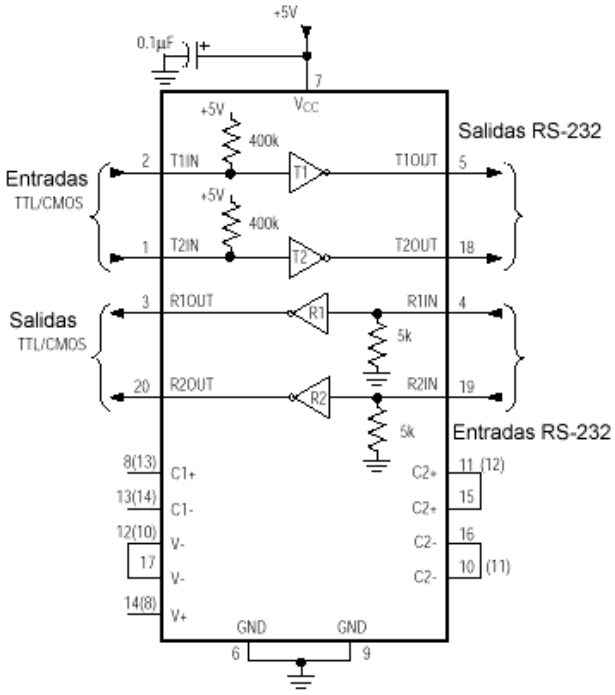


Figura 8. Transceptor RS-232 (cortesía MAXIM)

3.4.5 Tensión con niveles diferenciales según normas EIA RS 422 o RS-485 (con tercer estado).

Los bits se codifican como niveles diferenciales de tensión. La norma RS-485 ha sido de utilización generalizada como capa física de buses industriales normalizados y soluciones propietarias. Entre los primeros, PROFIBUS es un ejemplo relevante.

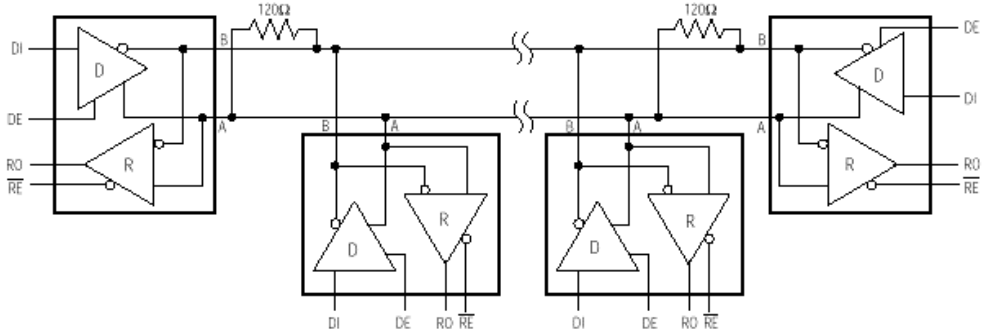


Figura 9. Bus RS-485 half-duplex (cortesía MAXIM)

Los transceptores RS-485 para conexión en bus son ampliamente disponibles y de bajo precio. Puesto que se cuenta con un tercer estado es posible la conexión en bus. Existen variantes optoaisladas. La señalización diferencial y las características de los transceptores permiten la utilización de RS-485 en distancias de varios kilómetros para baja velocidad (9,6 Kbps) o decenas de metros a alta velocidad (1 Mbps y superior).

3.4.6 Modulación sobre bus de alimentación (norma IEC 11158-2)

Se trata de la técnica contemplada en norma IEC 11158-2. Buses industriales orientados a industria de proceso con implicaciones de seguridad (incendio, explosión) como Profibus-PA, Foundation Fieldbus, WorldFIP etc. utilizan esta norma

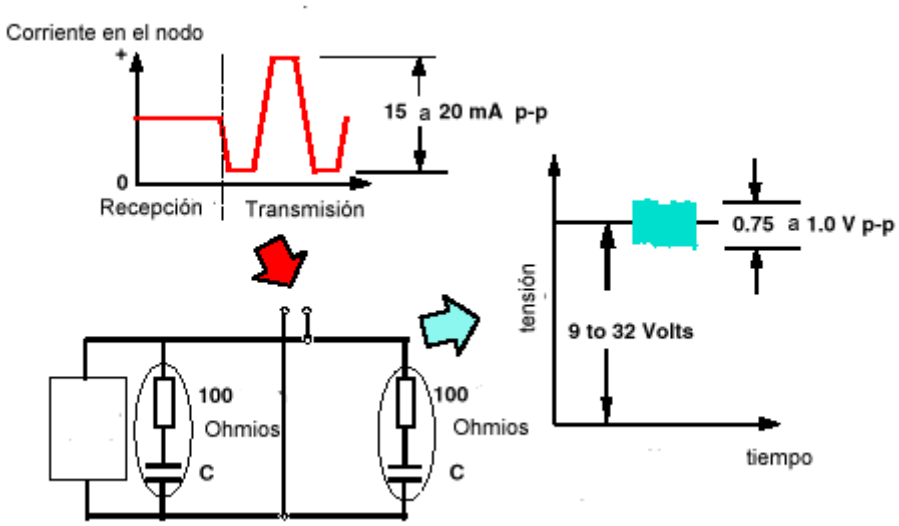


Figura 10. Comunicación según IEC 11158-2 ([18])

La línea de comunicación es también una línea de alimentación de energía para el nodo con nivele de tensión del orden de los 9 a 32 V.Cada uno de los nodos absorbe una corriente dada en reposo (del orden de 10 mA), la transmisión de realiza conmutando este consumo de corriente entre dos niveles, por ejemplo el dispositivo puede conmutar entre los 0 y 10 mA, esta conmutación genera una caida de tensión sobre las impedancias de cierre de línea que se superpone a la tensión de alimentación y que los transceptores de los nodos receptores detectan.

3.4.7 Fibra óptica

Las comunicaciones sobre fibra óptica no se utilizan de forma generalizada en redes de campo debido a: su mayor coste (trasnceptores y la propia fibra conductora) y a las dificultades de utilización sobre topología bus (sí, en cambio puede utilizarse fácilmente en topología en anillo). No se han de perder de vista, sin embargo, las indudables ventajas de la fibra óptica en cuanto a aislamiento entre nodos e inmunidad electromagnética. Muchas especificaciones de bus de campo contemplan la fibra óptica como una opción para entornos electromagnéticamente agresivos. En cuanto al coste, existen soluciones basadas en fibra óptica plástica multimodo que pueden competir con enlaces tipo RS-485 optoaislados (Ref [12]).

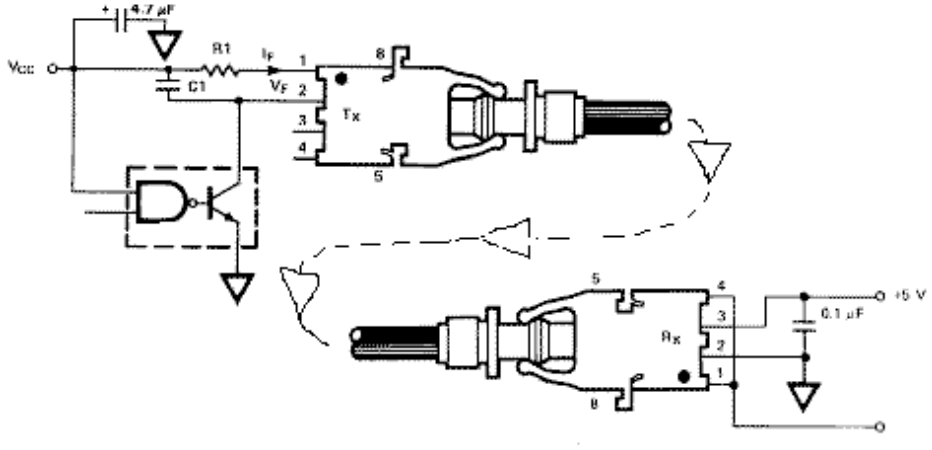


Figura 11 . Enlace de fibra óptica (cortesía Hewlet-Packard)

3.4.8 Comunicación serie síncrona o asíncrona

Cuando un transmisor señaliza una línea con las técnicas usuales en buses de campo (NRZ, Manchester), lo que hace es conmutar el nivel de una variable física entre dos estados. El receptor ha de muestrear el estado de esta variable física y traducirla a bits. La decodificación de esa información exige que el receptor se sincronice de forma adecuada con el transmisor para que el valor de bit que interpreta el receptor coincida con el que pretendía comunicar el transmisor. Se pueden considerar dos modos de operación fundamentales, en cuanto a sincronización de bit se refiere: modo asíncrono puro, y modo síncrono con reloj independiente (Figura 12).

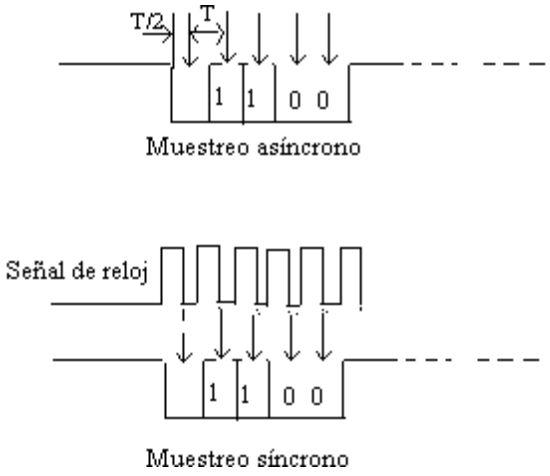


Figura 12 . Muestreo de bit

En el primer caso, modo asíncrono, el transmisor y el receptor han acordado una velocidad de comunicación, o un tiempo de bit T, y un número de bits en cada palabra transmitida. Ante una señal acordada (normalmente paso a nivel bajo tras un periodo de reposo en nivel alto o señal de “start”), el receptor muestrea la línea en intervalos de periodo T.

En el modo síncrono se ha de transmitir una señal de reloj adicional que marca al receptor los instantes en que debe muestrear la información.

Entre las ventajas del primer método sobresale el hecho de ahorrarse una línea adicional de transmisión. Sin embargo la temporización de muestreo debe ser precisa (lo que suele obligar a partir de una señal de reloj de frecuencia más elevada o, de otra forma, dividir la frecuencia base de reloj disponible) y constante.

En el segundo caso la ventaja es que no es necesario contar en el receptor con una frecuencia base de reloj precisa y constante, símplemente sigue los flancos de la señal de reloj que recibe, ésta señal de reloj incluso puede variar en frecuencia mientras los datos permanezcan sincronizados.

En los métodos de codificación Manchester se combinan ambas variantes, la señal transmitida incluye la frecuencia de reloj sobre la que se "modulan" los datos (véase [1], Cap 4).

3.5 La capa de enlace

3.5.1 Formato de tramas y control de errores

La capa de enlace gestiona el envío de paquetes de información de múltiples bits añadiendo la información redundante necesaria para que la alteración del contenido transmitido pueda ser detectada, y corregida en su caso, por el receptor. En buses de campo las tramas o paquetes siguen formatos derivados de los clásicos HDLC o similares ([1], Cap 6). Como método de detección y control de errores se utilizan los códigos de redundancia cíclica (CRC) o la simple suma de comprobación de la trama (Checksum). En muchos buses industriales se utiliza comunicación asíncrona, cada palabra (byte) puede incluir un método de control de errores (bit de paridad), además se incluye un CRC o checksum al nivel de trama.

3.5.2 Acceso al medio

La capa de transporte incluye también la gestión de acceso al medio. Puesto que la mayoría de los buses industriales siguen una topología en bus, como su propio nombre indica, el arbitraje en acceso al medio es un tema importante en la definición de un bus industrial. Importante porque, a los requisitos de cualquier red de proceso de datos, se añaden los requerimientos de tiempo real. Estos requisitos se traducen a una mayor necesidad de "determinismo". En redes de proceso de datos ha triunfado Ethernet con el método de arbitraje CSMA/CD, un método claramente no determinista. En buses de campo el determinismo obliga a consideraciones adicionales. Entre los métodos de arbitración en acceso al medio relevantes (véanse [13] y [14]), se consideran:

- La interrogación maestro-esclavo pura ("polling"), método normalmente adoptado en buses sencillos y soluciones propietarias.
- Las técnicas de paso de testigo (Profibus, por ejemplo)
- Las técnicas de CMSA/CD o variantes como CMSA/CA utilizada en el bus LonWorks de Echelon

- Las técnicas de división de tiempo (TDMA)
- El arbitraje no destructivo al nivel de bit (bus CAN), conocidos también como “Algoritmos de bit dominante”.

El método de acceso al medio es una característica muy importante de un bus industrial por una razón adicional: Métodos que vayan más allá del simple “polling” requieren normalmente un esfuerzo extra importante al nivel de software de protocolo, a no ser que se soporten sobre controladores hardware.

3.5.3 Gestión del enlace. Software o silicio

En el modelo de nodo presentado en 3.2 se divide la carga de proceso de información entre controlador y CPU. Esta división, sobre todo al nivel de capa de transporte, puede variar entre una gestión total en la CPU (el controlador no es necesario) o un importante soporte en el controlador.

- Cuando no existe controlador la CPU ha de gestionar la comunicación incluso al nivel de temporización de bit. Esta opción ahorra silicio en el nodo pero puede sobrecargar considerablemente la CPU. Una comunicación asíncrona, por ejemplo, gestionada al nivel de bit por la CPU, exige de ésta un muestreo de alta frecuencia en la recepción y el disparo de una interrupción temporizada a esa misma frecuencia en la transmisión. Este método, que era el adoptado en el pasado en microprocesadores venerables con el Intel 8085 para la gestión de comunicaciones asíncronas, ha vuelto a ponerse “de moda” con la aparición de microcontroladores de frecuencia de reloj alta (el fabricante Scenix, por ejemplo promociona esta opción de periféricos “software”).

- Una opción intermedia es la de tener un controlador (muchas veces incluido en el propio microprocesador o microcontrolador) que gestione la comunicación al nivel de palabra o byte, es decir el muestreo y transmisión de bits se realiza por el controlador (SCI o UART) y la CPU gestiona la comunicación de trama a partir de los bytes recibidos desde ese controlador. Es una de las opciones más extendidas, ya que es usual contar con controladores de comunicación de este tipo incluidos en los microprocesadores. La CPU gestiona la trama por software, el controlador informa de recepción de byte a la CPU por medio de un bit de estado concreto o generando una interrupción. En transmisión la CPU sólo ha de preparar la trama, la transmisión de cada uno de los bytes queda a cargo del controlador.

- Finalmente puede ser que el controlador gestione totalmente el envío y recepción de trama, entregando, o recibiendo de la CPU todo el contenido del mensaje. Este es el funcionamiento habitual de un controlador orientado a un protocolo concreto, por ejemplo un controlador de bus CAN.

La elección “más o menos” software contra “más o menos silicio” es un compromiso en el que están implicadas consideraciones de coste de nodo, disponibilidades, coste de desarrollo etc.

3.6 Otras capas ISO

3.6.1 Nivel de red. Encaminamiento, direccionamiento de nodos

En los buses de campo objeto de este trabajo no es habitual la necesidad de mantener una estructura de red extensa y ramificada. Es por ello que no se considera la capa de red un elemento esencial de la mayoría de protocolos de bus de campo. Sin embargo, esta afirmación debe reconsiderarse cuando el bus de campo está orientado a aplicaciones fuertemente distribuidas. Es el caso, por ejemplo, del bus Echelon (LONWORKS), orientado a aplicaciones de control y supervisión distribuida en edificios inteligentes y aplicaciones análogas. En este caso se establece una auténtica red con división en dominios y grupos. La arquitectura de la red se ha de basar, además, en la existencia de nodos con función de puente o enrutador. Estos elementos no son definidos de forma muy distinta a los elementos equivalentes en redes de proceso de datos. Se utilizan:

- Repetidores: elementos sin inteligencia que simplemente regeneran la señal en sus niveles físicos, es decir, retransmiten bits.
- Concentradores pasivos o activos.
- Puentes: elementos que operan al nivel de red redireccionando mensajes a otros segmentos. Esta función no se utiliza en todos los buses de campo, pero se incorpora en buses muy distribuidos.
- Pasarelas (“gateways”): traductores de protocolo que cubren prácticamente todas las capas de protocolo. Es el elemento imprescindible en interconexión de distintos buses de campo entre sí o con redes de nivel superior.

Quizás merece la pena prestar atención a un problema, a veces obviado, que en redes de proceso de datos se resuelve de forma estándar (direcciones físicas únicas en tarjetas de interfaz de red que se mapean a direcciones lógicas, en general) y en buses de campo tiene soluciones más variadas. Se trata de la asignación de dirección física concreta a nodos, sobre todo teniendo en cuenta que, en muchos casos, los nodos son instancias múltiples de una misma clase, un mismo tipo de nodo (considérese un bus de campo con decenas de sensores o actuadores análogos).

Entre los métodos habituales de asignación de dirección física se encuentran:

- Microinterruptores en la placa controladora de nodo. Método simple que asigna direcciones basadas en la codificación binaria de dichos interruptores. Es simple su configuración, el inconveniente es que ha de prestarse atención a la configuración correcta en operaciones de mantenimiento y cambio de placas.
- Microinterruptores en el conector de bus o conexión de la placa controladora de nodo. Similar al anterior pero con la ventaja de que la sustitución de una placa no exige reconfiguración ya que la dirección se lee por el conector, se configura en fabricación o al establecer el cableado. Exige mayor número de patillas en los conectores.
- Asignación de dirección por escritura en memoria no volatil (EEPROM). Práctico cuando existe una consola u ordenador para el mantenimiento de la red y personal entrenado en la

configuración. Poco práctico en entornos en los que dicha herramienta no es habitual, a diferencia de los sistemas de proceso de datos, los buses de campo se utilizan en entornos sociales en los que puede existir poca familiaridad con sistemas informáticos.

- Dirección física basada en código único en los microcontroladores o chips de la placa de nodo. Diversos fabricante ofrecen la posibilidad de incorporar un código único (de 48 o 64 bits) en cada microcontrolador. Si la aplicación incorpora un método de asignación de direcciones lógicas, automatizado, basado en dichos códigos, el direccionamiento y detección de nuevos nodos en la red es un proceso automático. El Neuron-Chip, en el que está basado el bus Echelon (LONWORKS) incorpora esta posibilidad. Existen microcontroladores de fabricantes como MicroChip que ofrecen esta opción.

3.6.2 Transporte, Aplicación, sesión, presentación. Paradigmas y modelos

Los protocolos de bus de campo se diferencian de los utilizados en redes de proceso de datos en algunas características relevantes:

- El volumen de datos a transmitir es menor y más atomizado. Frente a transporte de larga tramas en una red de proceso de datos, los buses de campo suelen transportar tramas cortas y muy frecuentes.

- Características de tiempo real. Es, muchas veces, más importante la actualización a tiempo o periódica de los datos que la seguridad de su recepción por el nodo de destino. Así las capas de transporte de proceso de datos reenvían cualquier trama no reconocida por el receptor hasta asegurar su envío, manteniendo la integridad de los datos. En un bus de campo, y ante una trama de tiempo real perdida, es más importante enviar la trama con datos actualizados que repetir una trama cuyo contenido es ya obsoleto.

- La capacidad de proceso de los nodos suele ser más reducida (pequeños microprocesadores o microcontroladores) y se pretende lograr más eficiencia en la implementación del protocolo.

- No se suelen crear estructuras de red complejas. Muchas veces se trata de un simple bus en el que no es necesario ningún encaminamiento especial.

Estas características han conducido a que muchas normas de bus de campo consideren como no relevantes o inexistentes los niveles ISO intermedios, desde red a presentación, de esta forma se pasa de la capa de enlace de datos a una capa de aplicación. No debe confundirse esta decisión con el hecho de que tareas normalmente encomendadas a esas capas intermedias no sean necesarias. Puede existir la necesidad de realizar algún tipo de encaminamiento de red, la gestión de envío de paquetes largos y consistentes de datos, el establecimiento de sesiones con un nodo o los cambios de presentación de datos. Sin embargo estas características suelen incluirse como un aspecto más de la especificación de la capa de aplicación y, en todo caso, se procura reducirlas al máximo.

Por tanto, un modelo ISO reducido para buses de campo sería el definido por las capas física, de enlace de datos y aplicación (Figura 13).

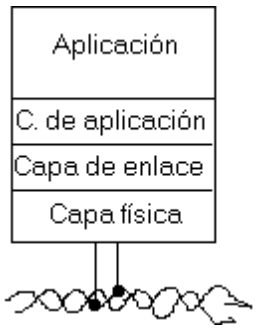


Figura 13 . Modelo ISO reducido

De acuerdo al modelo ISO la interacción de una aplicación con la capa de aplicación se realiza por medio de cuatro primitivas de servicio (27):

- Solicitud ("Request"). Ejecutada por la aplicación para solicitar un servicio.
- Indicación ("Indication"). Ejecutada por la capa de aplicación a la aplicación para indicar un evento de red o una solicitud de servicio transmitida desde otro nodo.
- Respuesta ("Response"). Ejecutada por la aplicación como respuesta a una indicación.
- Confirmación ("Confirmation"). Ejecutada por la capa de aplicación a la aplicación para informar del resultado de una solicitud previa.

De acuerdo con estas primitivas de servicio los servicios de capa de aplicación pueden ser clasificados como (Figura 14):

- Servicio Local: la aplicación transmite una solicitud que ejecuta un evento en las capas de protocolo y/o en la red.
- Servicio iniciado por proveedor: la aplicación recibe una indicación originada por un evento de red.
- Servicio no confirmado: la aplicación de un nodo A emite una solicitud que produce una indicación en otro nodo B.
- Servicio confirmado: la aplicación de un nodo A emite una solicitud que produce una indicación en otro nodo B. Éste nodo emite una respuesta que produce una confirmación en el nodo A.

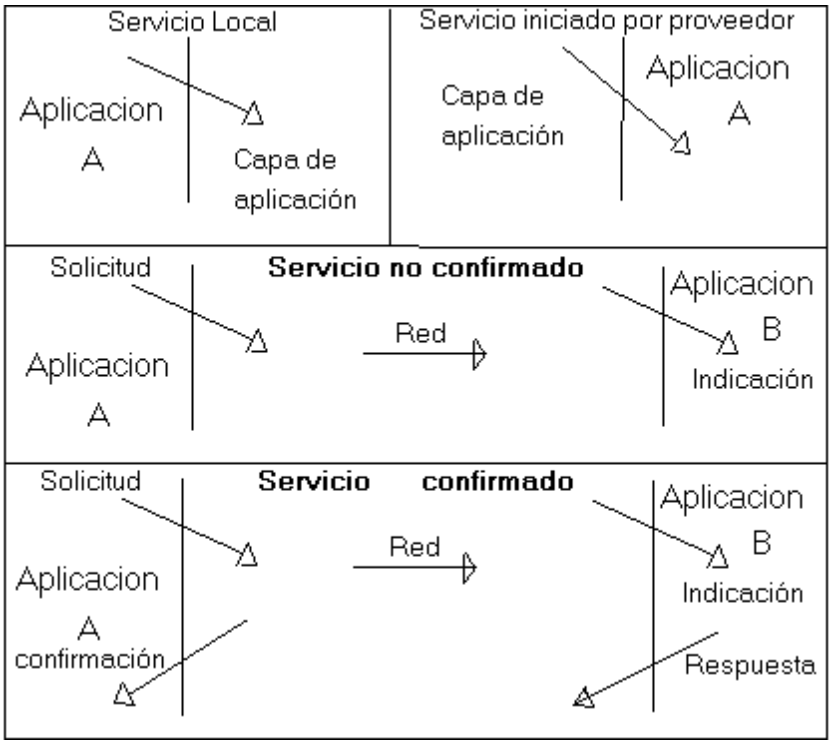


Figura 14. Tipos de servicios (Ref 27)

3.7 Relaciones o paradigmas de comunicación

Existen una serie de modelos básicos relacionados con la forma en la que se relacionan entre sí los nodos de una red en el proceso de comunicación.

- **Relación maestro-esclavo**[1]: un solo nodo tiene capacidad para iniciar servicios. Puede iniciar servicios no confirmados emitiendo una solicitud dirigida a uno o varios nodos esclavo o servicios confirmados emitiendo solicitud a un esclavo que ha de responder. Todo el proceso de comunicación está dirigido por el maestro, esta opción tiene la ventaja de simplificar el arbitraje en general y la propia capa de aplicación en los nodos esclavo. No es, sin embargo, apropiada para una comunicación flexible,eficiente y dirigida por eventos.

- **Relación cliente-servidor**: Se realiza entre dos nodos cualesquiera en la red por medio de servicios confirmados. El cliente emite una solicitud que el servidor responde. Pueden existir varias relaciones cliente-servidor simultáneas en la red.

- **Relación Productor-Consumidor**: Se realiza mediante servicios no confirmados, cualquier nodo productor emite información cuando lo considera oportuno y como mensaje global a la red ("broadcast"), los demas nodos, consumidores de esa información concreta, deciden sí la reciben y procesan o no. Existen dos variantes: el modelo "push" se basa en la transmisión expontánea por el o los nodos transmisores, el modelo "pull" se inicia ante solicitud de una determinada información por un nodo.

[1] Amo-esclavo sería una traducción más apropiada del inglés. Pero en el mundo de la electrónica se han mantenido una serie de barbarismos de este tipo.

Estos paradigmas pueden convivir en un mismo bus. En un sistema puede ser conveniente mantener una comunicación de tiempo real que mantenga el sistema en plena capacidad de respuesta a eventos, control de variables etc. y, en paralelo, establecer relaciones cliente-servidor entre pares de nodos para trasvase de información de gestión e incluso hacer que un nodo supervise el sistema y transmita cambios de modo de funcionamiento, configuración etc. con relación maestro-esclavo.

Sin embargo, la implementación de las relaciones productor-consumidor, especialmente, y la existencia de relaciones simultáneas, se complica enormemente cuando se ha de basar en una capa de enlace de datos en la que el arbitraje en acceso al medio no se ha resuelto totalmente o impone restricciones a las capas superiores. El bus CAN, objeto de este trabajo, resuelve de manera efectiva y elegante dicho arbitraje y por tanto ha permitido la definición de capas de aplicación para buses de campo sofisticados y en las que pueden convivir todos estos paradigmas.

4. Las normas y estándares

Existen una serie de estándares recogidos y aprobados por organismos de normalización internacionales que se deben considerar en el análisis de los buses de campo industrial

4.1 EIA-RS-232

En un intento de asegurar la compatibilidad entre terminales de datos y dispositivos de comunicaciones de diversos fabricantes la Asociación de Industrias Electrónicas de Estados Unidos (EIA) publicó en 1969 la recomendación número 232 en su versión C o RS232-C. Esta norma fue adoptada por el CCITT (ITU) en el estándar casi idéntico V.24, complementado con el estándar V.28 para las características eléctricas. La norma RS232 define al interconexión serie entre un dispositivo terminal de datos (DTE) y un equipo transmisor de datos (DCE) o más específicamente entre un ordenador o terminal y un módem. Sin embargo, la inclusión de la interfaz RS-232 en el ordenador personal de IBM (IBM PC) en 1984 y su adopción por fabricantes de dispositivos de control industrial como medio de interconexión ha dado lugar a la proliferación de aplicaciones no normalizadas de este estándar en la interconexión de sistemas industriales. Todo ello pese a sus evidentes limitaciones: enlace punto-punto, señalización no equilibrada, susceptibilidad y corto alcance etc.

4.2 EIA-RS-485

La norma de transmisión balanceada EIA RS-485 (Figura 9) fue desarrollada en 1983 como una versión más flexible de la RS 422 La norma contempla tan solo la capa física. En una red RS-485 estándar pueden conectarse hasta 32 nodos para comunicación half-duplex. El transmisor activa el bus con una tensión diferencial del orden de +- 2 V sobre impedancias de cierre de 120 Ω en los extremos. Un detector toma como valor lógico ‘1’ un valor diferencial de tensión superior a +200 mV y como ‘0’ uno inferior a –200mV. Puede utilizarse en distancias de más de 1 Km para velocidades de varios kbps y en distancias del orden de 40 m la velocidad puede subirse hasta los Mbps.

La norma RS-485 es uno de los métodos preferidos para la implantación de buses de campo no normalizados. La combinación de un transceptor RS-485 y el canal de comunicaciones

asíncrono de un microcontrolador es la opción más común de enlace en bus industrial. Protocolos normalizados muy importantes (PROFIBUS, MODBUS, etc.) la han adoptado.

4.3 IEC 11158-2

Se ha presentado en 3.4.6 . Es la norma adoptada para interconexión de instrumentación de proceso y se extiende para cubrir normativas especiales para entornos con riesgo de incendio y explosión.

4.4 ISO 11898- CAN

Norma de capa física común para bus CAN. Se analizará en apartados posteriores.

4.5 Normas IEC-870

Normas para sistemas de telecontrol que tiene, sobre todo, implantación en la industria eléctrica para la interconexión de dispositivos de medida y protección ([11]).

4.6 Ethernet

La norma IEEE 802.3 basada en la red Ethernet de Xerox (ver [1], [2]) se ha convertido en el método más extendido para interconexión de ordenadores personales en redes de proceso de datos. En la actualidad se vive una auténtica revolución en cuanto a su desplazamiento hacia las redes industriales. Ha puesto fuera de perspectiva cualquier otra opción en los niveles intermedios y está empezando a cubrir el nivel de control. Buses de campo como PROFIBUS la han adoptado como solución de nivel superior. Con variantes más orientadas a tiempo real, Ethernet conmutada, hay fabricantes que incluso la ofrecen como solución universal. Este avance se ve favorecido por varios factores (en el fondo otra vez las leyes de Moore, Gilder y Metcalfe):

- Disponibilidad y bajo precio de controladores y transceptores Ethernet.
- Proliferación de los protocolos TCP/IP soportados, en gran parte, sobre Ethernet.
- Introducción en el proceso industrial de ordenadores industriales compactos de bajo precio que incluyen, entre otras características heredadas de sus parientes de sobremesa, un controlador Ethernet.

5. Algunos buses estandarizados

5.1 Algunas opciones disponibles

Una primera aproximación a los buses o redes de comunicaciones industriales puede ser la enumeración de algunas de las alternativas disponibles. Aunque poco esclarecedora puede dar una idea de la confusión que ha existido, y existe, en este campo. A esta reducida selección se le podrían añadir otras opciones más o menos estandarizadas y las innumerables soluciones propietarias no estándar que se han adoptado. Téngase en cuenta que la competencia de fabricantes en este campo es muy fuerte y muchas veces el envoltorio comercial interfiere en la elección basada en características reales. Actualmente:

- Todos los fabricantes de buses dicen que su estándar es abierto y múltiples fabricantes de dispositivos lo soportan.
- Todos los fabricantes de buses dicen que cubren todo el espectro de necesidades de red industrial, aunque en la parte superior se apoyen en simples tarjetas Ethernet y TCP/IP.
- Todos los fabricantes de buses apoyan el estándar universal que resolverá los problemas de interconexión de planta, de hecho lo llevan apoyando en las últimas décadas, nadie se explica como no ha sido posible llegar a ese estándar. Naturalmente y dado que su bus propietario es el mejor, el estándar debería basarse en su bus.

5.1.1 Elección de un bus de campo

Los siguientes criterios, entre otros, pueden considerarse al elegir un bus:

- Coste del hardware por nodo
- Coste de desarrollo
- Influencia en el coste de desarrollo de aplicaciones para los nodos
- Tiempo de respuesta y flujo máximo. Características de tiempo real.
- Fiabilidad
- Recuperación ante fallos. Funcionamiento en condiciones degradadas
- Medios físicos que permite: par trenzado, fibra óptica, infrarrojos, …
- Capacidad para funcionamiento multimaster, relaciones cliente-servidor o productor-consumidor.
- Topologías permitidas (bus, estrella, anillo, ..)
- Servicios de gestión de red, flexibilidad para la configuración de dominios y grupos
- Interfaces de programación (API) disponibles para desarrollo de aplicaciones.
- Normalización. Soporte por organizaciones de usuarios y/o fabricantes

En todo caso, éstos y otros posibles criterios han de ser ponderados de acuerdo a la aplicación. Un integrador de sistemas soportará aquellos buses más extendidos en el mercado. Un fabricante de maquinaria puede requerir un bus de interconexión interna de sus dispositivos que no ha de ser normalizado y sin embargo permitir prestaciones especiales para su producto. Un fabricante de dispositivos interconectables ha de prestar atención a los buses más utilizados por sus potenciales clientes, etc.

5.2 PROFIBUS

Profibus se desarrolló bajo un proyecto financiado por el gobierno alemán. Está normalizado en Alemania por DIN E 19245 y en Europa por EN 50170. El desarrollo y posterior comercialización ha contado con el apoyo de importantes fabricantes com ABB, AEG, Siemens, Klöckner-Moeller, … Está controlado por la PNO (Profibus User Organisation) y la PTO (Profibus Trade Organisation).

Existen tres perfiles:

- Profibus DP. Orientado a sensores/actuadores enlazados a procesadores (PLCs) o terminales.
- Profibus PA. Para control de proceso y cumpliendo normas especiales de seguridad para la industria química (IEC 11158-2, seguridad intrínseca).
- Profibús FMS. Para comunicación entre células de proceso o equipos de automatización. La evolución de Profibús hacia la utilización de protocolos TCP/IP para enlace al nivel de proceso hace que este perfil esté perdiendo importancia.

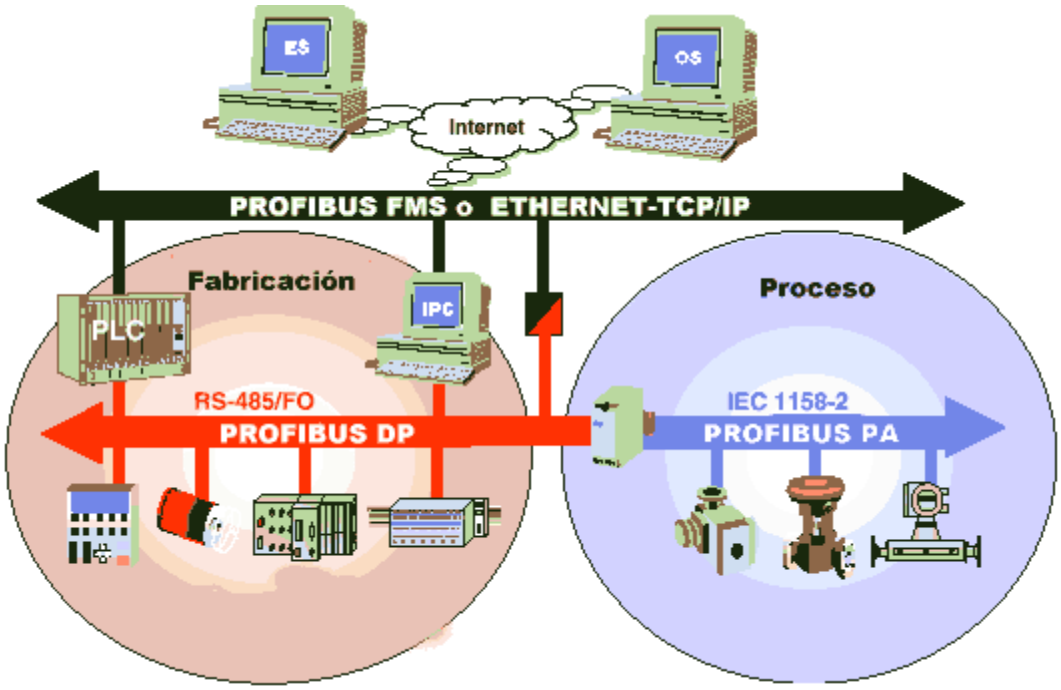


Figura 15. Perfiles PROFIBUS (tomado de [15])

Utiliza diferentes capas físicas. La más importante, en PROFIBUS DP, está basada en EIA RS-485. Profibús PA utiliza la norma IEC 11158-2 (norma de comunicación síncrona entre sensores de campo que utiliza modulación sobre la propia línea de alimentación de los dispositivos y puede utilizar los antiguos cableados de instrumentación 4-20 mA, véase 3.4.6) y para el nivel de proceso se tiende a la utilización de Ethernet. También se contempla la utilización de enlaces de fibra óptica. Existen puentes para enlace entre diferentes medios, además de gateways que permiten el enlace entre perfiles y con otros protocolos.

Se distingue entre dispositivos tipo maestro y dispositivos esclavo. El acceso al medio entre maestros se arbitra por paso de testigo, el acceso a los esclavos desde un maestro es un proceso de interrogación cíclico (pollig). Se pueden configurar sistemas multimaestro o sistemas más simples maestro-esclavo.
En Profibus DP se distingue entre: maestro clase 1 (estaciones de monitorización y diagnóstico), maestro clase 2 (elementos centralizadores de información como PLCs, PCs, etc.), esclavo (sensores, actuadores).

El transporte en Profibus-DP se realiza por medio de tramas según IEC 870-5-1. La comunicación se realiza por medio de datagramas en modo broadcast o multicast. Se utiliza comunicación serie asíncrona por lo que es utilizable una UART genérica.

Profibus DP prescinde de los niveles ISO 3 a 6 y la capa de aplicación ofrece una amplia gama de servicios de diagnóstico, seguridad, protecciones etc. Es una capa de aplicación relativamente compleja debido a la necesidad de mantener la integridad en el proceso de paso de testigo (un y sólo un testigo)

Profibús FMS es una compleja capa de aplicación que permite la gestión distribuida de procesos al nivel de relación entre células con posibilidad de acceso a objetos, ejecución remota de procesos etc. Los dispositivos de definen como dispositivos de campo virtuales (VFDs), cada uno incluye un diccionario de objetos (OD) que enumera los objetos de comunicación (ODs). Los servicios disponibles son un subconjunto de los definidos en MMS (Manufacturing Message Specification ISO 9506)

Las plataformas hardware utilizadas para soportar Profibus se basan en microprocesadores de 16 bits más procesadores de comunicaciones especializados o circuitos ASIC como el LSPM2 de Siemens. La PNO se encarga de comprobar y certificar el cumplimiento de las especificaciones PROFIBUS.

Entre sus perspectivas de futuro se encuentra la integración sobre la base de redes Ethernet al nivel de planta y la utilización de conceptos de tiempo real y filosofía productor-consumidor en la comunicación entre dispositivos de campo.

Las distancias potenciales de bus van de 100 m a 24 Km (con repetidores y fibra óptica). La velocidad de comunicación puede ir de 9600 bps a 12 Mbps. Utiliza mensajes de hasta 244 bytes de datos.

Profibus se ha difundido ampliamente en Europa y también tiene un mercado importante en América y Asia. El conjunto Profibus DP- Profibus PA cubre la automatización de plantas de proceso discontínuo y proceso contínuo cubriendo normas de seguridad intrínseca.

5.3 INTERBUS

Protocolo propietario, inicialmente, de la empresa Phoenix Conctact GmbH, aunque posteriormente ha sido abierta su especificación. Normalizado bajo DIN 19258, norma europea EN 50 254. Fue introducido en el año 1984.

Utiliza una topología en anillo y comunicación mediante un registro de desplazamiento en cada nodo. Se pueden enlazar buses periféricos al principal.

Capa física basada en RS-485. Cada dispositivo actúa como repetidor. Así se puede alcanzar una distancia entre nodos de 400 m para 500Kbps y una distancia total de 12 KM. Es posible utilizar también enlaces de fibra óptica.

Capa de transporte basada en una trama única que circula por el anillo (trama de suma)

La información de direccionamiento no se incluye en los mensajes, los datos se hacen circular por la red. Alta eficiencia. Para aplicaciones de pocos nodos y un pequeño conjunto de entradas/salidas por nodo, pocos buses pueden ser tan rápidos y eficientes como INTERBUS.

Físicamente tiene da la impresión de seguir una topología en estrella, pero realmente cada nodo tiene un punto de entrada y otro de salida hacia el siguiente nodo.

Es muy sensible a corte completo de comunicación al abrirse el anillo en cualquiera de los nodos. Por otra parte, la estructura en anillo permite una fácil localización de fallos y autodiagnóstico.

Es muy apropiado para comunicación determinista a alta velocidad, es muy difícil una filosofía de comunicación orientada a eventos.

5.4 DeviceNet

Bus basado en CAN. Su capa física y capa de transporte se basan, por tanto, en ISO 11898 sobre par trenzado, y en la especificación de Bosh 2.0. Sobre esta capa de transporte DeviceNet define una de las más sofisticadas capas de transporte industriales sobre bus CAN ([17]).

DeviceNet fue desarrollado por Allen-Bradley a mediados de los noventa, posteriormente pasó a ser una especificación abierta soportada en la ODVA (Open DeviceNet Vendor Association). Cualquier fabricante puede asociarse a esta organización y obtener especificaciones, homologar productos etc.

Es posible la conexión de hasta 64 nodos con velocidades de 125 Kbps a 500 Kbps en distancias de 100 a 500 m.

Utiliza una definición basada en orientación a objetos para modelar los servicios de comunicación y el comportamiento externo de los nodos. Define mensajes y conexiones para funcionamiento maestro-esclavo, interrogación cíclica, “strobing” o lanzamiento de interrogación general de dispositivos, mensajes espontáneos de cambio de estado, comunicación uno-uno, modelo productor-consumidor, carga y descarga de bloques de datos y ficheros etc.

DeviceNet ha conseguido una significativa cuota de mercado. Existen más de 300 productos homologados y se indica que el número de nodos instalados superaba los 300.000 en 1998. Está soportado por numerosos fabricantes: Allen-Bradley, ABB, Danfoss, Crouzet, Bosh, Control Techniques, Festo, Omron, …

5.5 Foundation Fieldbus

Un bus orientado sobre todo a la interconexión de dispositivos en industrias de proceso contínuo. Su desarrollo ha sido apoyado por importantes fabricantes de instrumentación (Fisher-Rosemount, Foxboro,…). En la actualidad existe una asociación de fabricantes que utilizan este bus, que gestiona el esfuerzo normalizador, la Fieldbus Foundation. Normalizado como ISA SP50, IEC-ISO 61158 (ISA es la asociación internacional de fabricantes de dispositivos de instrumentación de proceso).

En su nivel H1 la capa física sigue la norma IEC 11158-2 para comunicación a 31,25 Kbps, es por tanto, compatible con Profibús PA, su principal contendiente. Presta especial atención a las versiones que cumplen normas de seguridad intrínseca para industrias de proceso en ambientes combustibles o explosivos. Se soporta sobre par trenzado y es posible la reutilización de los antiguos cableados de instrumentación analógica 4-20 mA. Se utiliza comunicación síncrona con codificación Manchester Bifase-L.

La capa de transporte utiliza un protocolo sofisticado, orientado a objetos con múltiples formatos de mensaje. Distingue entre dispositivos con capacidad de arbitración (Link Master) y normales. En cada momento un solo Link Master arbitra el bus, puede ser sustituido por otro en caso de fallo. Utiliza diversos mensajes para gestionar comunicación por paso de testigo, comunicación cliente-servidor, modelo productor-consumidor etc. Existen servicios para configuración, gestión de diccionario de objetos en nodos, acceso a variables, eventos, carga-descarga de ficheros y aplicaciones, ejecución de aplicaciones etc. La codificación de mensajes se define según ASN.1

El nivel H2 está basado en Ethernet de alta velocidad (100 Mbps) y orientado al nivel de control de la red industrial.

5.6 FIP- WorldFIP

Desarrollado en Francia a finales de los ochenta y normalizado por EN 50170, que también cubre Profibus. Sus capas física y de transporte son análogas a las de Foundation Fieldbus H1 y Profibus PA. La división norteamerica de WorldFIP se unió a mediados de los noventa a la Fieldbus Foundation en el esfuerzo por la normalización de un bus industrial común.

Utiliza un modelo productor-consumidor con gestión de variables cíclicas, eventos y mensajes genéricos.

5.7 LonWorks

La empresa Echelon, localizada en Palo Alto (California), fue fundada en 1988. Comercializa el bus de campo LonWorks basado en el protocolo LonTalk y soportado sobre el NeuronChip. Alrededor de estas marcas ha construido toda una estructura de productos y servicios, hábilmente comercializados, dirigidos al mercado del control distribuido en domótica, edificios inteligentes, control industrial etc. Asegura que varios miles de empresas trabajan con LonWorks, que cientos de empresas comercializan productos basados en su bus y que se han instalado millones de nodos.

El protocolo LonTalk cubre todas las capas OSI. El protocolo se soporta en hardware y

firmware sobre el NeuronChip. Se trata de un microcontrolador que incluye el controlador de comunicaciones y toda una capa de firmware que, además de implementar el protocolo, ofrece una serie de servicios que permiten el desarrollo de aplicaciones en el lenguaje Neuron C, una variante de ANSI C. Motorola y Toshiba fabrican el NeuronChip, además Echelon ofrece la posibilidad de abrir la implementación de LonWorks a otros procesadores.

La red Lonworks ofrece una variada selección de medios físicos y topologías de red: par trenzado en bus, anillo y topología libre, fibra óptica, radio, transmisión sobre red eléctrica etc. El soporte más usual es par trenzado a 38 o 78 Kbs. Se ofrece una amplia gama de servicios de red que permiten la construcción de extensas arquitecturas con multitud de nodos, dominios y grupos, típicas de grandes edificios inteligentes.

El método de compartición de medio es acceso CSMA predictivo e incluye servicios de prioridad de mensajes.

Echelon ofrece herramientas de desarrollo, formación, documentación y soporte técnico. Echelon basa su negocio en la comercialización del bus, medios, herramientas y soporte.

5.8 SDS

SDS ("Smart Distributed System") es, junto con DeviceNet y CANOpen, uno de los buses de campo basados en CAN más extendidos. Fue desarrollado por el fabricante de sensores industriales Honeywell en 1989.

Se ha utilizado sobre todo en aplicaciones de sistemas de almacenamiento, empaquetado y clasificación automática. Se define una capa física que incluye alimentación de dispositivos en las conexiones. La capa de aplicación define autodiagnóstico de nodos, comunicación por eventos y prioridades de alta velocidad.

5.9 CANOpen

Bus de campo basado en CAN. Fue el resultado de un proyecto de investigación financiado por la Comunidad Europea y se está extendiendo de forma importante entre fabricantes de maquinaria e integradores de célula de proceso. Está soportado por la organización CiA (CAN In Automation), organización de fabricantes y usuarios de CAN que también apoya DeviceNet, SDS etc. Al final de este trabajo se describirá con más detalle este bus, como ejemplo de bus de campo normalizado soportado sobre CAN.

5.10 Modbus

En su definición inicial Modbus era una especificación de tramas, mensajes y funciones utilizada para la comunicación con los PLCs Modicon. Modbus puede implementarse sobre cualquier línea de comunicación serie y permite la comunicación por medio de tramas binarias o ASCII con un proceso interrogación-respuesta simple. Debido a que fue incluido en los PLCs de la prestigiosa firma Modicon en 1979, ha resultado un estándar de facto para el enlace serie entre dispositivos industriales.

Modbus Plus define un completo bus de campo basado en técnica de paso de testigo. Se utiliza como soporte físico el par-trenzado o la fibra óptica.

En la actualidad Modbus es soportado por el grupo de automatización Schneider (Telemechanique, Modicon,…).

5.11 Industrial Ethernet

En 4.6 se ha presentado una breve reseña de la situación de Ethernet en su avance desde las redes de proceso de datos hacia las redes industriales. Es indudable esa penetración. Diversos buses de campo establecidos como Profibus, Modbus etc. han adoptado Ethernet como la red apropiada para los niveles superiores. En todo caso se buscan soluciones a los principales inconvenientes de Ethernet como soporte para comunicaciones industriales:

- El intrínseco indeterminismo de Ethernet se aborda por medio de topologías basadas en conmutadores. En todo caso esas opciones no son gratuitas.
- Se han de aplicar normas especiales para conectores, blindajes, rangos de temperatura etc. La "barata" tarjeta adaptadora Ethernet empieza a encarecerse cuando se la dota de robustez para un entorno industrial
- Parece difícil que Ethernet tenga futuro a nivel de sensor, aunque puede aplicarse en nodos que engloban conexiones múltiples de entrada-salida.

Como conclusión Ethernet está ocupando un area importante entre las opciones para redes industriales, pero parece aventurado afirmar, como se ha llegado a hacer, que pueda llegar a penetrar en los niveles bajos de la pirámide CIM.

5.12 ASI

Desarrollado por un consorcio de empresas de automatización en los noventa. Orientado a la comunicación de sensores y actuadores con el bajo coste por nodo como objetivo. Es un sistema maestro-esclavo en el que se pueden enlazar a un maestro hasta 31 esclavos. El cable de interconexión incluye alimentación para los esclavos. La conexión es sencilla, basada en un clip de presión. Los mensajes son de 4 bits de datos y se transmiten a 167 Kbps. Se utiliza codificación Manchester II con modulación por pulsos alternados, un par de hilos se usa simultáneamente para la comunicación y la alimentación de nodos. Es normalmente necesario un chip específico como controlador de bus.

Fabricantes como Siemens utilizan ASI como bus en el nivel más bajo para interconexión de sensores y actuadores a buses de campo (PROFIBUS).

5.13 BitBus

Introducido por Intel a principios de los 80. Es un bus maestro-esclavo soportado sobre RS-485 y normalizado en IEEE-1118. Debido a su sencillez ha sido adoptado en redes de pequeños fabricantes o integradores. En su capa de aplicación se contempla la gestión de tareas distribuidas, es decir es, en cierto modo, un sistema multitarea distribuido. Existe una organización europea de soporte (Bitbus European User's Group).

5.14 ARCNet

Originalmente desarrollada como red para proceso de datos en los años 70 ARCNet ha encontrado aplicación en el mundo industrial. Su técnica de paso de testigo hace que sea predecible, determinista y robusta. Está normalizada como ANSI/ATA 878.1. La velocidad de comunicación es de 2,5 Mbps con paquetes del 0 a 512 bytes. Soporta topología en bus y estrella y diversos medios físicos (cable coaxial, par trenzado, fibra óptica).

Es una red muy apropiada para un nivel intermedio en la jerarquía CIM. Algunos fabricatnes (Contemporary Controls, por ejemplo, véase [21]) proponen como jerarquía ideal para control industrial una basada en Ethernet en el nivel superior, ArcNET en el intermedio y CAN al nivel de célula de fabricación.

5.15 ControlNet

Bus de alta velocidad (5 Mbps) y distancia (hasta 5000 m), muy seguro y robusto promovido por Allen-Bradley. Utiliza cable RG6/U (utilizado en televisión por cable) y se basa en un controlador ASIC de Rockwell.

No es soportado por muchos fabricantes y resulta de elevado precio por nodo. Se ha utilizado para interconexión de redes de PLCs y ordenadores industriales en aplicaciones de alta velocidad y cierta criticidad.

5.16 HART

Es un protocolo para bus de campo soportado por la HART Communication Foundation y la Fieldbus Foundation. Su campo de aplicación básico es la comunicación digital sobre las líneas analógicas clásicas de los sistemas de instrumentación, manteniendo éstas en servicio. Sus prestaciones como bus de campo son reducidas.

Utiliza el bus analógico estándar 4-20 mA sobre el que transmite una señal digital modulada en frecuencia (modulación FSK 1200-2200 Hz). Transmite a 1200 bps manteniendo compatibilidad con la aplicación analógica inicial y sobre distancias de hasta 3 Km.

Normalmente funciona en modo maestro-esclavo.

5.17 La guerra de los buses.

Ante la variedad de opciones existente, en realidad sólo se han enumerado unas pocas, parece razonable pensar que fabricantes y usuarios hicieran un esfuerzo en la búsqueda de normativas comunes para la interconexión de sistemas industriales.

Lo que ha venido llamándose “la guerra de los buses” tiene que ver con la permanente confusión reinante en los entornos normalizadores en los que se debate la especificación del supuesto “bus de campo universal”. Desde mediados de los 80 la Comisión Electrotécnica Internacional (IEC-CEI) y la Sociedad de Intrumentación Americana (ISA) ha sido escenario del supuesto esfuerzo de los fabricantes para lograr el establecimiento de una norma única de bus de campo de uso general. En 1992 surgieron dos grupos, el ISP (Interoperable Systems Project) y WorldFIP cada uno promoviendo su propia versión del bus de campo. En el primer

grupo estaban fabricantes como Siemens, Fisher-Rosemount, Foxboro y Yokogawa. En el segundo Allen-Bradley, HoneyWell, Square D y diversas empresas francesa. En 1994 ambos grupos se unieron en la Fieldbus Foundation. El debate se trasladó luego, y continua en la actualidad, a la conjunción de Fieldbus y el mundo Profibus. Los años pasan, la norma del supuesto bus universal nunca se acaba de generar y en el camino aparecen nuevas opciones como CAN, LonWorks, Ethernet. Incluso el debate es confuso y totalmente incomprensible (empresas como Siemens han llegado a participar en los dos grupos contendientes), otras empresas participantes en el debate generaban en paralelo soluciones propias, es el caso de Allen-Bradley con DeviceNet y HoneyWell con SDS. La realidad es que sólo los usuarios están realmente interesados en la obtención de normas de uso general. Los fabricantes luchan por su cuota de mercado y, en general, sólo están a favor de una norma cuando ésta recoge las características de su propia opción, lo cual es comprensible dadas las fuertes inversiones necesarias para el desarrollo de un bus industrial normalizado. El debate sigue abierto (véanse [22] [23]).

6. Introducción al bus CAN

6.1 Origen

La continua incorporación de dispositivos eléctronicos en el automóvil, y la necesidad de interconexión entre ellos, alcanzó a inicios de la década de los 80 un nivel de complejidad tal, que hacía antieconómica la clásica solución de conexión punto a punto entre cada dispositivo (Figura 16).

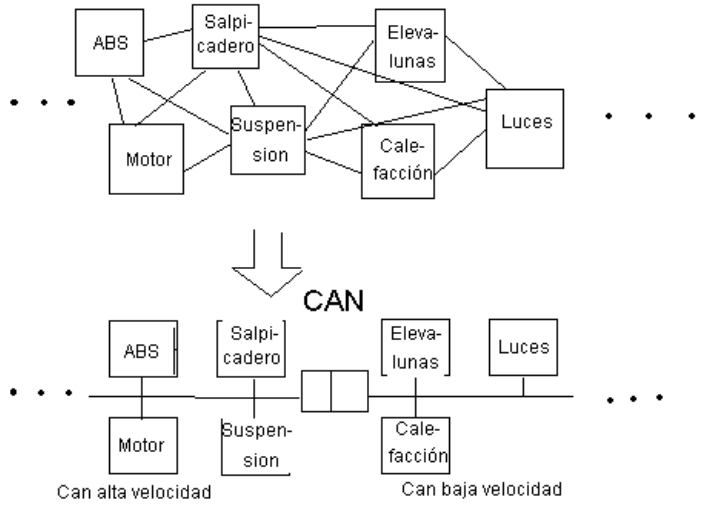


Figura 16 . Necesidad del bus CAN

Se tendía hacia cableados de cientos de metros y decenas de kilos de peso, si a estos factores se añaden los relacionados con la variedad de mazos de cable a generar, dificultades en la normalización de interconexiones, etc. se comprenderá que los principales fabricantes de automóviles invirtieran importantes recursos en el desarrollo de una red digital que permitiera la interconexión de esos dispositivos. En paralelo las organizaciones de normalización trabajaron en la formalización de niveles de aplicación y en la adopción de estándares. Así la S.A.E. (Sociedad normativa del automóvil en Norteamérica) estableció las aplicaciones de tipo A (para dispositivos auxiliares, cerraduras, luz interior etc.), B (aire acondicionado..) , C (Información de tiempo real, frenos, suspensión, etc.) y D (para elementos multimedia y de información, radio, teléfono, etc.). Fueron especificados y desarrollados diversos buses para interconexión de elementos internos del automóvil: CAN, VAN, J1850, …

CAN fue propuesto y desarrollado por el fabricante alemán de componentes para automóvil Robert BosH GmbH, que colaboró con uno de los líderes mundiales en semiconductores, Intel, para el desarrollo de un dispositivo controlador. De aquí nació el primer controlador CAN, el Intel 82256, que ha sido luego sustituido por el 80257. La especificación de Bosh no predeterminaba una capa física concreta. Posteriormente fueron normalizadas las correspondientes a las normas ISO 11898 (aplicaciones de alta velocidad, hasta 1Mbps) e ISO 11519-2 para aplicaciones de baja velocidad. Los vehículos Mercedes Clase S, lanzados en 1992, incorporaron el bus CAN por primera vez.

El par formado por las especificaciones de Bosh, “CAN Specificacion 2.0” en sus variantes A (“Standard” CAN, identificador de 11 bits) y B (CAN extendido, identificador de 29 bits) y la norma ISO 11898 para capa física es lo que conceptualmente puede definirse como “bus

CAN". Estas especificaciones se concretan en multitud de dispositivos controladores (autónomos o incluidos como periféricos en microcontroladores) que soportan CAN 2.0 A o 2.0 B y transceptores que soportan la norma ISO 11898 (transmisión diferencial sobre par trenzado).

A principios de los noventa, las empresas de automatización detectaron en CAN un gran potencial como soporte para buses de campo industrial. El gran volumen de consumo que genera la industria del automóvil conduce naturalmente a bajos precios para dispositivos controladores y transceptores. Por otra parte las características de CAN en cuanto a robustez, filosofía multimaestro, acceso al medio múltiple no destructivo, facilidad para la comunicación por eventos y productor-consumidor etc. lo hacía muy atractivo como base para redes de dispositivos industriales. Importantes fabricantes normalizaron redes basadas en CAN: DeviceNet de Allen-Bradley, SDS de Honeywell (por cierto, mientras tanto seguían participando en los sesudos comités de normalización del bus industrial universal). En 1992 se fundó CiA ("CAN in Automation), una organización de usuarios y fabricantes de dispositivos industriales basados en CAN. CiA ha trabajado en la armonización de CAN hacia un protocolo abierto para uso industrial, complementando especificaciones hacia abajo (capa física al nivel de conectores, cableado etc.) y hacia arriba (capas de aplicación).

CAN ha alcanzado en el año 2000 un nivel extraordinario de madurez e implantación, se habla de cientos de millones de nodos, los fabricantes de microcontroladores y procesadores digitales de señal (Texas TMS320C24, por ejemplo) están incorporando controladores CAN de forma bastante generalizada. Los modelos VHDL de controladores CAN se pueden incorporar en ASICs y dispositivos de lógica programable (FPGAs). CAN resulta una opción a tener en cuenta en sistemas distribuidos de tiempo real.

6.2 Resumen de características

6.2.1 Niveles OSI

Como ya se ha indicado la especificación de Bosh para CAN cubre la capa de enlace de datos contemplando las subcapas LLC (Control de enlace lógico) y MAC (Control de acceso al medio). La capa física habitual en buses industriales es la establecida en la norma 11898, comunicación sobre par trenzado. Para los niveles superiores existen diversas opciones, desde soluciones propietarias a protocolos normalizados como CANOpen, DeviceNet, SDS, etc.

<table>
<tr><td colspan="2">Aplicación</td><td rowspan="5">Cubierto por diversas especificaciones para capa de aplicación: CANOpen, DeviceNet, SDS, CAN-Kingdom etc.

Filtrado de mensajes y buzones en controladores</td></tr>
<tr><td colspan="2">Presentación</td></tr>
<tr><td colspan="2">Sesión</td></tr>
<tr><td colspan="2">Transporte</td></tr>
<tr><td colspan="2">Red</td></tr>
<tr><td rowspan="2">Enlace</td><td>LLC/MAC</td><td>Cubierto por especificación CAN de Bosh. Implementado en Controladores</td></tr>
<tr><td colspan="2">Físico</td><td>ISO 11898 sobre par trenzado. Implementado en transceptores. Existen otras variantes de capa física: optoaislada, infrarrojos, radio, diversos niveles de robustez ante fallo etc.</td></tr>
</table>

Tabla 2 Niveles ISO en CAN

Al establecer su especificación ([24]) Bosh se centró en la definición de la capa de enlace (LLC+MAC) pero la filosofía CAN se extiende hacia la capa física (arbitraje bit a bit, sincronización de bit, son aspectos normalmente asignados a esta capa) y hacia arriba en lo que Bosh define como "Capa de Objetos", es decir el tratamiento que las capas superiores dan a los mensajes, ese tratamiento se basa en una filosofía de variables distribuidas y muchos controladores CAN gestionan parte del tratamiento de esa capa (filtrado de mensajes, buzones)

6.2.2 Capa física

Las técnicas de señalización, temporización de bit y sincronización (la subcapa PLS de la capa física) se definen en la especificación CAN de Bosh. El método de señalización es asíncrono NRZ ("non return to zero"). Cada bit se señala en dos níveles físicos diferenciados: ***dominante*** (bit '0') y ***recesivo*** (bit '1'). Para garantizar cambios de nivel que permitan la sincronización se sigue la técnica del ***relleno de bits*** ("bit stuffing") intercalando un bit opuesto para evitar que aparezcan más de 5 bits seguidos de la misma polaridad. Los valores de tiempo de velocidad, tiempo de bit y longitud máxima (sin repetidores) contemplados en ISO 11898 son los detallados en la Tabla 3

Velocidad	Tiempo de bit	Longitud máxima
1 Mbps	1 µS	30 m
800 Kbps	1,25 µS	50 m
500 Kbps	2 µS	100 m
250 Kbps	4 µS	250 m
125 Kbps	8 µS	500 m
50 Kbps	20 µS	1000 m
20 Kbps	50 µS	2500 m
10 Kbps	100 µS	5000 m

Tabla 3. Velocidad-Distancia en CAN

Estos son valores orientativos que varían dependiendo de la tolerancia de los osciladores de los nodos, impedancias y retardos en la línea etc.

La topología es bus con derivaciones de corta longitud. Con pérdida de prestaciones en cuanto a velocidad o longitud máxima se pueden adoptar estructuras en estrella. El bus se cierra en los extremos con impedancias de carga.

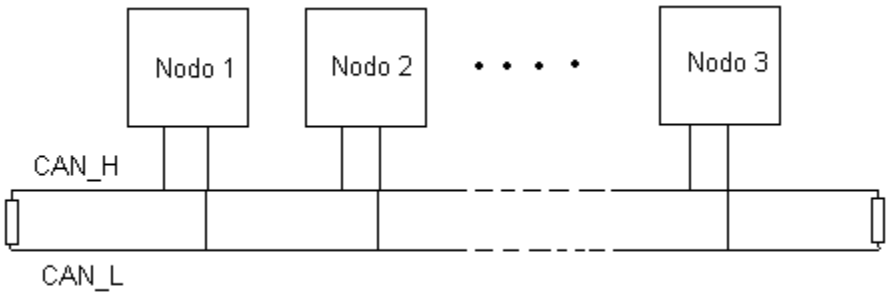


Figura 17 . Topología en bus CAN

El número máximo de nodos no está limitado por la especificación básica y depende de las características de los transceptores, las especificaciones de buses de campo lo limitan a 32 o 64 en una red sin repetidores.

En un bus según ISO 11898 se considera bit dominante aquel en que la tensión diferencial entre las líneas del bus V CAN_H- V CAN_L supera los +0,9 V y un bit recesivo cuando la diferencia es inferior a +0,5 V. Las tensiones nominales en estado dominante son 3,5 V para CAN_H y 1,5 V para CAN_L respecto a la masa de referencia, en estado recesivo son de 2,5 V en ambas líneas.

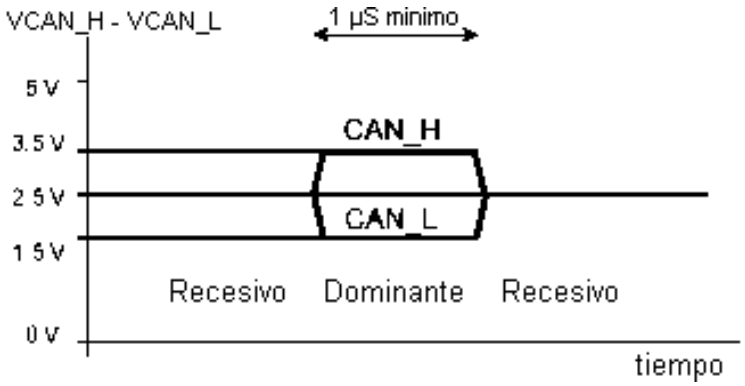


Figura 18. Niveles eléctricos en bus CAN

En los buses CAN normalizados los conectores usualmente utilizados son el SubD-9, el Mini de 5 patillas o el tipo regleta.

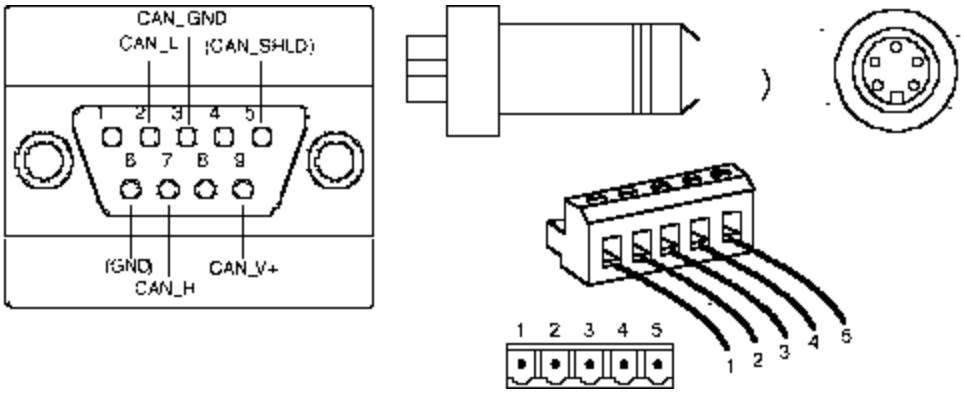


Figura 19. Conectores típicos para bus CAN (tomado de especificaciones de CANOpen)

6.2.3 Acceso al medio

Quizás unas de las características que distingue a CAN, y otros protocolos de automoción, respecto a otras normas, es su técnica de acceso al medio denominada como CSMA/CD+CR o

"Carrier Sense, Multiple Access/Colission Detection+ Collision Resolution" (Acceso múltiple con detección de portadora , detección de colisión más resolución de colisión).

El acceso al medio por medio de técnicas de acceso múltiple y detección de colisión evolucionaron desde el método ALOHA inicial hasta su consagración como método de acceso al medio de las redes Ethernet, con técnica de acceso múltiple con detección de portadora y detección de colisión ("Carrier Sense Multiple Access/Colision Detection" o CSMA/CD). El método de acceso al medio utilizado en bus CAN añade una característica adicional: la resolución de colisión. En la técnica CSMA/CD utilizada en redes Ethernet ante colisión de varias tramas, todas se pierden, CAN resuelve la colisión con la supervivencia de una de las tramas que chocan en el bus. Además la trama superviviente es aquella a la que se ha identificado como de mayor prioridad.

La resolución de colisión se basa en una topología eléctrica que aplica una función lógica determinista a cada bit, que se resuelve con la prioridad del nivel definido como bit de tipo dominante. Definiendo el bit ***dominante*** como equivalente al valor lógico '0' y bit ***recesivo*** al nivel lógico '1' se trata de una función AND de todos los bits transmitidos simultáneamente. Cada transmisor escucha contínuamente el valor presente en el bus, y se retira cuando ese valor no coincide con el que dicho transmisor ha forzado. Mientras hay coincidencia la transmisión continua, finalmente el mensaje con identificador de máxima prioridad sobrevive. Los demás nodos reintentarán la transmisión lo antes posible.

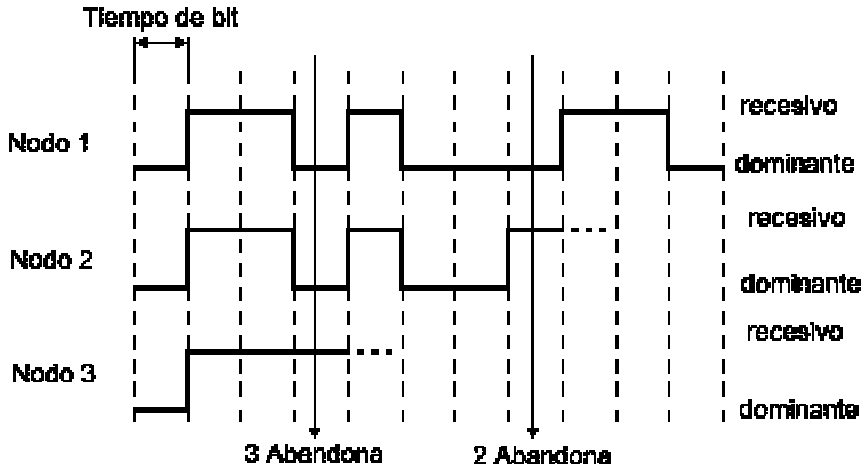


Figura 20. Resolución de colisión en bus CAN (de [42])

Se ha de tener en cuenta que la especificación CAN de Bosh no establece cómo se ha de traducir cada nivel de bit (dominante o recesivo) a variable física. Cuando se utiliza par trenzado según ISO 11898 el nivel dominante es una tensión diferencial positiva en el bus, el nivel recesivo es ausencia de tensión, o cierto valor negativo, (los transceptores no generan corriente sobre las resistencias de carga del bus).

Esta técnica aporta la combinación de dos factores muy deseados en aplicaciones industriales distribuidas: la posibilidad de fijar con determinismo la latencia en la transmisión de mensajes entre nodos y el funcionamiento en modo multimaestro sin necesidad de gestión del arbitraje, es decir control de acceso al medio, desde las capas de software de protocolo.

La prioridad queda así determinada por el contenido del mensaje, en CAN es un campo determinado, el identificador de mensaje, el que determina la prioridad.

6.2.4 Consistencia de información en el bus

Cualquier mensaje transmitido en un bus CAN ha de ser aceptado por todos los nodos como válido o rechazado por todos ellos. Ante detección de error por parte de cualquier nodo, éste transmite una trama de error que invalida el mensaje para todos los demás. El resultado es una consistencia total de la información en el sistema distribuido: todos los nodos reciben la misma información al mismo tiempo o no reciben información válida.

En los mensajes CAN no se utiliza ningún campo de dirección de nodo (aunque en capas de aplicación puede seguirse alguna regla). CAN sólo distingue entre distintos identificadores de mensaje, sin importar qué nodo lo transmite o qué nodos pueden estar interesados en su recepción. El contenido el mensaje se identifica por el campo *Identificador*, cualquier nodo en la red puede decidir, sobre la base de su configuración de filtrado de mensajes, qué mensajes desea recibir. Como ya se ha indicado, el identificador de mensaje tiene un papel fundamental en el proceso de arbitraje, y por tanto determina la prioridad del mensaje y su probabilidad de supervivencia ante colisión. Por otra parte los identificadores han de asignarse unívocamente a todos los potenciales mensajes que se utilicen en la red

6.2.5 Mensajes y tipos de tramas

CAN utiliza mensajes de estructura predefinida, tramas, para la gestión de la comunicación.

Se distinguen entre dos variantes de CAN, el definido en CAN 2.A o “CAN Standard” y el definido en CAN 2.B o “CAN Extendido”, los formatos de trama son análogos diferenciándose básicamente en el número de bits que se utiliza para el identificador de mensaje: 11 bits (2032 identificadores) diferentes en CAN Standard y 29 bits (536.870.912 identificadores) en CAN Extendido.

Las tramas CAN son de longitud reducida, la trama más larga es de 130 bits en CAN Estándar y 154 bits en CAN Extendido.

Los tipos de trama, y estados de bus, utilizados son:

- ***Trama de datos***: la que un nodo utiliza normalemente para poner información en el bus (siempre es un “broadcast” a todos los demás nodos). Puede incluir entre 0 y 8 Bytes de información útil.
- Trama de interrogación remota (en lo que sigue se denominará como ***trama remota*** (“remote frame”): puede ser utilizada por un nodo para solitar la transmisión de una trama de datos con la información asociada a un identificador dado. El nodo que disponga de la información definida por el identificador la transmitirá en una trama de datos.
- ***Tramas de error***: usadas para señalar al resto de nodos la detección de un error, invalidando el mensaje erróneo normalmente (un caso especial es un nodo en estado de “error pasivo”)

- ***Trama de sobrecarga***: permite que un nodo fuerce a los demás a alargar el tiempo entre transmisión de tramas sucesivas
- ***Espaciado inter-tramas***: Las tramas de datos (y de interrogación remota) se separan entre sí por una secuencia predefinida que se denomina espaciado inter-trama.
- ***Bus en reposo***: En los intervalos de inactividad se mantiene constantemente el nivel recesivo del bus.

En un bus CAN los nodos transmiten la información espontáneamente con tramas de datos, bien sea por un proceso cíclico o activado ante eventos en el nodo. La trama de interrogación remota sólo se suele utilizar para detección de presencia de nodos o para puesta al día de información en un nodo recién incorporado a la red. Los mensajes pueden entrar en colisión en el bus, el de identificador de mayor prioridad sobrevivirá y los demás son retransmitidos lo antes posible.

6.2.6 Variables de red

CAN se basa en una filosofía en la que la información no se identifica con el nodo al que está asociada, sino con el significado del mensaje. Cada mensaje CAN transmite una información que está identificada por uno de los campos de trama (el identificador), esta información es consistente y única en toda la red, el identificador determina la prioridad del mensaje en la red. Es en el proceso de configuración o programación inicial de la aplicación, cuando se ha de decidir qué identificadores se asignan a cada variable y qué variables son generadas en cada nodo. Esta asignación y gestión de identificadores de forma estructurada y con organización de identificadores apropiada, y por tanto prioridad en el bus, es una de las tareas más importantes en el desarrollo de una aplicación distribuida basada en CAN.

6.2.7 Robustez. Detección y gestión de errores

CAN fue concebido como un protocolo de alta seguridad (está orientado a comunicar el sistema de frenos de un vehículo, por ejemplo). Para ello se han adoptado medidas adecuadas en cada una de las capas de protocolo:

Capa física

Niveles diferenciales de alta inmunidad electromagnética. Disponibilidad de transceptores con capacidad de funcionamiento en condiciones degradadas (fallo de uno de los dos conductures, por ejemplo).

Reconocimiento de mensajes

Todo mensaje transmitido al bus ha de ser reconocido de forma consistente por los receptores forzando a dominante un bit concreto de la trama (bit ACK) que se transmite como recesivo

Relleno de trama

En las tramas de datos e interrogación remota se aplica la regla de relleno de bits que evita una secuencia sucesiva de más de 5 bits del mismo signo, para ello se inserta un sexto bit de signo

contrario, el receptor ha de eliminar este bit adicional siguiendo la misma regla.

Detección de errores

Cada trama incluye un código CRC con distancia Hamming 6, la tasa de error no detectado es menor que (tasa de error en mensajes)*4,7.10^{-11}

Señalización de errores y recuperación

Cualquier nodo que detecta un error emite una trama que señala el error a los demás nodos, si el nodo detector es un nodo totalmente activo (no se encuentra en nivel pasivo de error) el mensaje queda invalidado para toda la red y se retransmitirá lo antes posible. El tiempo de recuperación es de, como máximo, 29 veces el tiempo de bit.

Aislamiento de nodos con fallo

Se sigue un sofisticado proceso de autodiagnóstico en los nodos, cuando un nodo acumula errores pasa inicialmente a una situación de funcionamiento pasivo y si la degradación continúa el nodo queda excluido de la comunicación evitando perturbar al resto de nodos de la red. Es decir el estado de un nodo puede ser: Activo, Pasivo o Anulado. Un nodo anulado ha de deshabilitar su transceptor y no participa en la comunicación.

6.2.8 Gestión de mensajes en los nodos. Filtrado y buzones

En cuanto al tratamiento de mensajes en los controladores, se ha de distinguir entre las implementaciones Basic-CAN y Full-CAN (no se debe confundir esta división con la de CAN Estándar y CAN Extendido).

En un controlador de tipo Basic-CAN éste mantiene una cantidad reducida de “buffers”, “mailboxes” o buzones para los mensajes recibidos. Se pueden programar máscaras y filtros de mensajes que limitan el rango de identificadores de mensajes admitidos por el controlador. Esta configuración es muy importante para evitar la sobrecarga de la CPU por interrupciones de recepción inútiles provocadas por mensajes en los que el nodo no está interesado.

En un controlador de tipo Full-CAN el número de buzones es mucho mayor y cada uno de ellos se puede programar para la recepción de un mensaje concreto. En su versión más sofisticada el controlador se convierte en una especie de memoria compartida entre el nodo y la red, y direccionable (por medio del identificador CAN) desde ésta última.

Naturalmente los controladores concretos pueden variar entre estos dos tipos, pueden ser Basic-CAN con varios buzones, Full-CAN y Basic-CAN combinados etc.

En los apartados que siguen se profundizará en cada uno de estos aspectos recorriendo los aspectos relacionados con CAN desde capa física a capa de aplicación.

7. La capa física. Señalización. Transceptores

7.1 Introducción

La definición de CAN exige que la capa física sea capaz de representar los estados dominante y recesivo en el medio de transmisión y soportar el arbitraje no destructivo bit a bit. Por tanto cualquier medio físico en el que se cumplan las siguientes características:

- Posibilidad de dos estados diferenciados: dominante y recesivo.
- Si ningún nodo transmite el estado debe ser el recesivo
- Si uno o más nodos transmiten el estado será dominante si alguno impone esta condición y recesivo si todos imponen esta condición (representando dominante como valor binario '0' y recesivo como '1' se ha de efectuar una operación AND de los bits transmitidos.

Existen, por tanto, infinitas posibilidades de cara a materializar una capa física para CAN. Sin embargo la forma dominante es la transmisión diferencial sobre par trenzado según ISO 11898. Es la capa elegida para los más importantes buses de campo basados en CAN, DeviceNet, CANOpen, SDS,... y también mayoritaria en el automóvil, si bien en este caso existen variantes y otras normas (SAE/J1939 por ejemplo es la norma americana para vehículos pesados, ISO11519-2 para aplicaciones de baja velocidad,…). En aplicaciones industriales se ha utilizado CAN sobre RS-485, fibra óptica, infrarrojos, … En lo que sigue se hará referencia fundamentalmente a la opción ISO 11898.

7.2 Capa física según ISO 11898. Transceptores

7.2.1 Características básicas

La norma ISO 11898 establece la comunicación sobre par trenzado entre hasta 32 nodos a velocidades de hasta 1Mbps y longitudes dependientes de la velocidad (40 m para 1Mbps, hasta 5000 m sin repetidores para baja velocidad, véase Tabla 3).

El cable debe incluir conductores para las dos señales de bus CAN_H y CAN_L con una impedancia característica de línea de 120 Ω, para el cierre de línea se recomiendan resistencias de 124 Ω, 1%, 200 mW, en cada extremo. Los nodos han de detectar una condición recesiva si la diferencia de tensión entre CAN_H y CAN_L no es mayor de 0,5 V, y una condición dominante para diferencias superiores a 0,9 V. Los transceptores han de ser capaces de generar una caida de tensión diferencial de entre 1,5 y 3,0 V sobre una carga resistiva de 60 Ω.

Aunque es posible diseñar un circuito electrónico basado en componentes discretos (algunos transistores, diodos y resistencias) que actúe como transceptor CAN de acuerdo con estos requisitos, es ventajosa la utilización de transceptores integrados comerciales.

7.2.2 Transceptores

Un transceptor CAN es, básicamente, una combinación de un amplificador de línea y un comparador de recepción (Figura 21). La etapa transmisora cierra el propio circuito de alimentación del transceptor sobre la línea en estado dominante y se desconecta de ella en

estado recesivo. La etapa receptora detecta la tensión diferencial CAN_H-CAN_L, esa tensión pasa a ser prácticamente nula, o, en otras palabras, ambas líneas adquieren un nivel de tensión de aproximadamente 2,5 V respecto a la masa de referencia, cuando no hay transmisores. Esto último se asegura por redes resistivas incluidas en el circuito receptor. La etapa transmisora es controlada por una entrada digital de nivel TTL y la receptora proporciona en la salida del comparador esos mismos niveles en función del valor diferencial detectado. La lógica de conversión traduce un nivel dominante en la línea a un nivel TTL-bajo en las líneas digitales y un nivel recesivo en el nivel contrario. Esa misma lógica se sigue en la etapa transmisora.

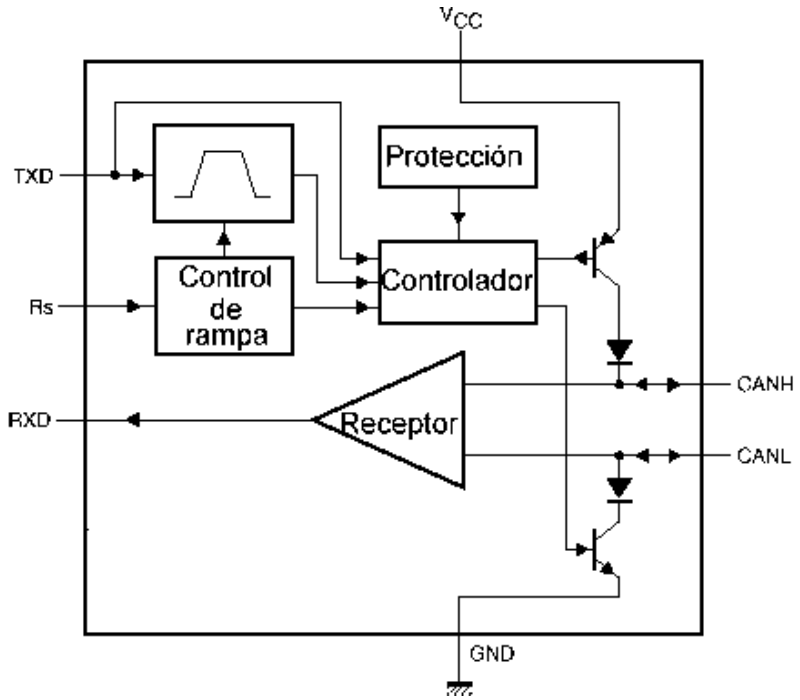


Figura 21. Transceptor CAN (inspirado en hoja de datos del 82C250 de Philips)

Otras características adicionales que puede incluir un transceptor CAN son :

- Control de rampa en las conmutaciones de nivel dominante-recesivo. Esta característica permite reducir la energía electromagnética radiada de acuerdo a las normas aplicables.

- Modo común extendido. El rango de funcionamiento en modo común determina los niveles extremos de tensión de modo común que el transceptor soporta.

- Modo de funcionamiento en bajo consumo ante falta de actividad en el bus.

- "Fan out" o capacidad del transmisor para actuar sobre un número dado de entradas receptoras. Partiendo del "fan out" habitual (32 nodos) existen transceptores con capacidad para 64 o 128 nodos.

- Protección térmica.

- Retardo entrada-salida. Medido como el retardo entre cambio de nivel en la entrada transmisora y en la salida receptora, es un parámetro importante en aplicaciones de alta

velocidad y ha de ser medido como dato de partida para la configuración de los parámetros de temporización y sincronización de bit (se tratará en 7.3)

7.3 Temporización y sincronización de bit

7.3.1 Temporización de bit

Las señales eléctricas reales en un bus CAN sufren las alteraciones propias de la distorsión que se produce por las características físicas de la línea (resistencia, capacidad e inductancia distribuida), los retardos de propagación, y los producidos en los propios controladores y las posibles fuentes externas de interferencia electromágnetica. Por otra parte, cada controlador CAN en un bus depende, normalmente, de un oscilador distinto, entre estos osciladores existen diferencias de frecuencia que pueden dar lugar a desfases en el muestreo de tramas en los distintos nodos. Los controladores CAN siguen un proceso de muestreo y resincronización orientado a evitar, en la medida de lo posible, los desajustes debidos a estos factores.

Estos desajustes se manifiestan especialmente en el arbitraje. Varios transmisores que inicien transmisión que se considera simultánea, no estarán tan perfectamente sincronizados en el inicio, además su frecuencia de oscilador puede ser ligeramente distinta. Los receptores que se han sincronizado con el primer flanco de bajada detectado, deberán irse resincronizando sucesivamente a los flancos producidos por los o el transmisor que resulte vencedor en la contienda.

El bus CAN utiliza señalización asíncrona con codificación NRZ, es necesaria, por tanto, una operación de muestreo en los receptores y el muestreo se ha de resincronizar periódicamente.

Para que el controlador, que es un sistema digital síncrono con frecuencia de reloj generada por un oscilador, pueda muestrear la señal asíncrona recibida con seguridad y precisión ha de utilizar una frecuencia de muestreo superior a la de la señal transmitida en el bus.

En la especificación CAN se contemplan las siguientes definiciones:

Frecuencia nominal de bit (***Nbr***, "nominal bit rate"): se define como el número de bits por segundo transmitidos en ausencia de resincronizaciones por un transmisor ideal. Su inverso es el ***tiempo nominal de bit*** (***Nbt***, "nominal bit time")

El controlador CAN ha de utilizar una frecuencia de reloj múltiplo de la frecuencia nominal de bit, esa frecuencia se obtiene como cociente de la frecuencia del oscilador. Se define como ***cuanto de tiempo mínimo*** (***Mtq*** o "minimum time quantum") el obtenido a partir de la frecuencia de oscilador (suele ser de la misma frecuencia o mitad), y como ***cuanto de tiempo básico (Tq)*** (su inverso es la frecuencia básica del controlador) el obtenido al dividir la frecuencia de esa señal por un ***divisor de frecuencia (Brp*** o "baud rate prescaler") que es normalmente configurable

$$Tq = Brp.\ Mtq$$

En la especificación CAN se establece que el Brp ha de poder configurarse al menos en el rango de valores 1 a 32 (suele ser un registro, BRP, en el que se pueden asignar los valores 0..31, por tanto la división es por BRP+1) .La frecuencia básica del controlador es así varias veces superior a la de comunicación en el bus, el número de Tq en un bit, ***NTq*** ha de poder programarse en un rango de, al menos, 8 a 25.

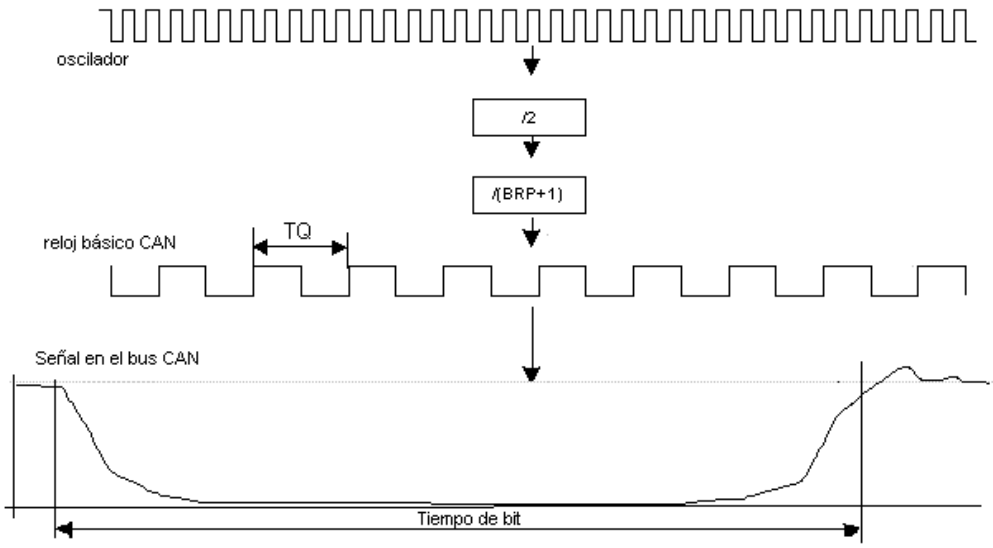


Figura 22. Muestreo de bit

El bit se muestrea en un punto dado del tiempo de bit (en algunos controladores y para bajas velocidades es posible muestreo múltiple con decisión por mayoría). La velocidad de comunicación a la que queda configurado el controlador, para un oscilador de frecuencia dada, queda definida por el valor del BRP, y por el número de Tq por bit. Este último valor se obtiene como suma de varios valores parciales, configurables también, que tienen que ver con la sincronización de bit y que se detallarán a continuación.

7.3.2 Sincronización de bit

En CAN se siguen dos métodos de sincronización del muestreo con la secuencia de bits de la trama:

- Sincronización de trama (“Hard Synchronization”): Es la reinicialización de la lógica de muestreo de bit que ocurre ante el flanco de bajada (paso de recesivo a dominante) producido por un bit de inicio de trama

- Sincronización intra-trama (“Soft Synchronization”): Es un proceso de sincronización que ocurre en cada flanco de bajada que se produce durante la transmisión de la trama.

El primer método no requiere excesiva explicación. El segundo es complejo, y de hecho su correcta configuración influye en las prestaciones finales (velocidad, distancia máxima, porcentaje de errores) que se alcanzan en una aplicación.

Parámetros

Conceptualmente el conjunto total de Tqs que corresponden con el tiempo nominal de bit se

dividen en los siguientes subconjuntos, en este orden:

- Segmento de sincronización (Sync_Seg): es de un Tq. En este intervalo se espera que se produzca cualquier cambio recesivo a dominante en el bus.
- Segmento de propagación (Prop_Seg): puede configurarse como de 1 a 8 Tq. Está orientado a la compensación de retardo entre transmisor y receptor. Este retardo es la suma del retardo de línea y el retardo debido a la electrónica de interface de los controladores.
- Segmento de fase 1 (Phase_Seg1): puede configurarse como de 1 a 8 Tq. Como se verá está orientado a la detección y reajuste de desfases entre nodos. Puede alargarse en operaciones de resincronización inter-trama
- Segmento de fase 2 (Phase_Seg2): su valor es el máximo de lo que se define como "Tiempo de proceso de información" y un valor igual a Phase_Seg1. El tiempo de proceso de información es igual o menor de 2 Tq. Puede acortarse (nunca menos del tiempo de proceso de información) en las operaciones de resincronización inter-trama.

El muestreo del bit se realiza en el momento final de Phase_Seg1 e inicial de Phase_Seg2. Por otra parte se define un parámetro adicional: el salto de resincronización (Sjw o "Resynchronization jump width") con valores de entre 1 y 4 Tq, pero no mayores que Phase_Seg2.

En muchos controladores se programan en realidad tres valores: TSEG1, TSEG2 Y SJW. TSEG1 determina el número de TQs en Prop_Seg + Phase_Seg 1 menos 1 (para que el rango vaya de 0 a n-1 y se ahorren bits en la representación) TSEG2 equivale a Phase_Seg2-1 y SJW a Sjw-1. Resumiendo, en la Tabla 4 se presentan los parámetros más importantes de configuración de resincronización y velocidad de comunicación (o tiempo de bit) y en Tabla 5 los valores programables normalmente en un controlador, aunque en ciertos controladores pueden existir registros de configuración que guarden una relación distinta con los parámetros de resincronización.

Parámetro	Rango	Descripción
brp	1..32	divisor de frecuencia
Sync_Seg	1 TQ	cuanto en que se espera cambio de bit sin desfase
Prop_Seg	1..8 TQ	segmento de propagación, compensa retardos
Phase_Seg1	1..8 TQ	segmento de fase 1
Phase_Seg2	1..8 TQ	segmento de fase 2
Sjw	1..4 TQ	salto de resincronización

Tabla 4 . Parámetros de sincronización de bit

Registro	Rango	Descripción
BRP	0..31	BRP=brp-1
TSEG1	0..15	TSEG1=Prop_Seg+Phase_Seg1-1
TSEG2	0..7	TSEG2= Phase_Seg2-1
SJW	0..3	SJW= Sjw-1

Tabla 5. Registros de sincronización de bit

Proceso de resincronización inter-trama

En condiciones normales los flancos de cambio recesivo-dominante se esperan durante el Sync_Seg (téngase en cuenta que pueden no ocurrir cambios en cada periodo nominal de bit), el bit se muestreará tras los segmentos Prop_Seg y Phase_Seg1.

La resincronización consiste en el acortamiento o alargamiento del tiempo de bit (o número de Tq por bit) para que el momento de muestreo se desplaze con relación al último flanco recesivo-dominante detectado. La decisión de resincronización la toma el controlador tras detectar el flanco y evaluar el ***error de fase e.*** Se tiene:

e = 0 si el flanco aparece en el Sync_Seg

e>0 si el flanco aparece después del Sync_Seg y antes del punto de muestreo

e<0 si el flanco ha aparecido después del punto de muestreo del bit previo (en Phase_Seg2 de bit previo)

Cuando el error de fase no es nulo se efectúa una resincronización consistente en (Figura 23):

Si e>0 se alarga el Phase_Seg1 del bit en proceso en un número de Tq igual a máximo entre el error e y Sjw para atrasar el punto de muestreo

Si e<0 se acorta el Phase_Seg2 del bit previo en un número de Tq igual a máximo entre el error e y Sjw para atrasar el punto de muestreo

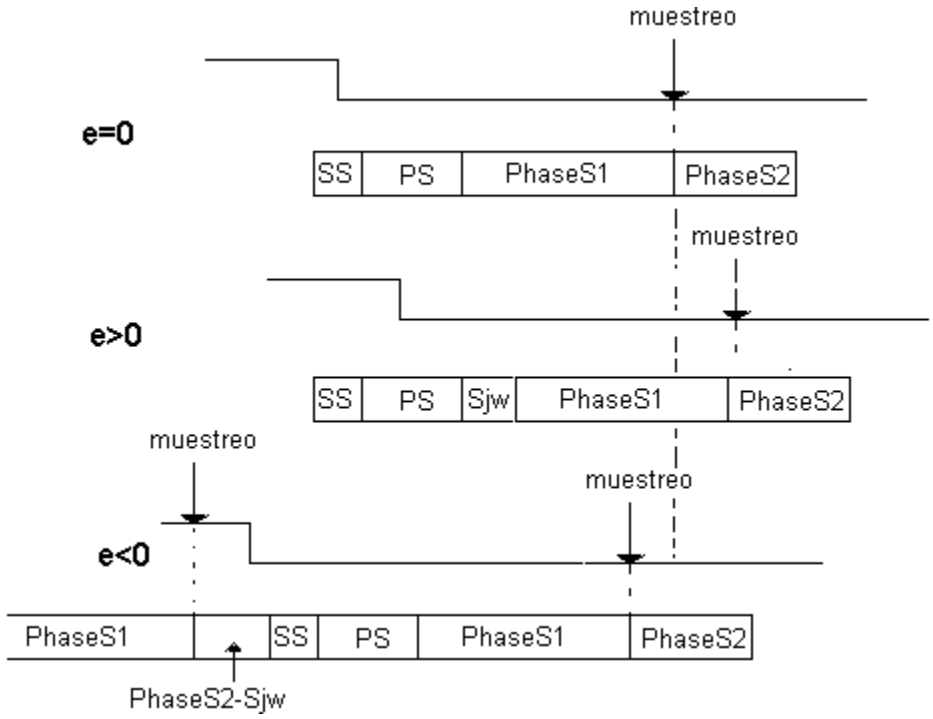


Figura 23 . Proceso de sincronización de bit

El proceso de resincronización sólo se ejecuta una vez por bit

7.3.3 Configuración de parámetros de temporización y resincronismo

Como ya se ha indicado, la correcta configuración de los parámetros de resincronización es un aspecto clave en el funcionamiento correcto de una aplicación CAN. Sin entrar en detalles, el proceso se gestiona de acuerdo a las siguientes reglas básicas:

- Desde luego la suma Sinc_Seg+Prop_Seg+Phase_Seg1+Phase_Seg2 tiene que ser igual al número de Tq por bit Ntq. Además el Brp ha de dar un valor de Tq tal que el producto Ntq.Tq sea igual al tiempo nominal de bit Nbt, es decir partiendo de un reloj de sistema de frecuencia fc (que en muchos casos es la mitad de la de oscilador) y para una velocidad nominal de comunicación Nbr se tiene:

$$Tq = \frac{Brp}{fc}$$

$$Ntq = 1 + Pr\ op\_Seg + Phase\_Seg1 + Phase\_Seg2 \quad (Sinc\_Seg = 1 \text{ siempre})$$

$$Nbr = \frac{1}{Ntq * Tq}$$

En términos de configuración de registros se tendrá:

Ntq= (TSEG1+1)+(TSEG2+1)+1= TSEG1+TSEG2+3

Tq= (BRP+ 1) / fc

Como la velocidad de comunicación Nbr es una especificación de partida del sistema se han de configurar el resto de parámetros con la restricción de que se tenga un valor final del producto Ntq*Tq igual al tiempo nominal de bit o con el mínimo error.

- En aplicaciones de alta velocidad es conveniente que la tolerancia de osciladores sea pequeña, el parámetro más importante a tener en cuenta es el retardo entre transmisor y receptores, el punto de muestreo ha de situarse de forma que esos retardos se tengan en cuenta. Por tanto se deberá alargar el Prop_Seg y muestrear hacia la última parte del tiempo nominal de bit (60-80%). Normalmente se tomará un Sjw mas bien reducido

- En aplicaciones en las que se quiera conseguir máxima robustez ante diferencias en frecuencia de osciladores Sjw tomará un valor alto, por lo tanto ha de tomar mayor valor Phase_Seg2 (recuérdese que Sjw ha de ser menor o igual que Phase_Seg2), el muestreo se centra más en el tiempo de bit y se pierde capacidad de compensación de retardos.

Ejemplo de cálculo: se desea una velocidad nominal de 125 Kbps y se ha supuesto en base a medidas experimentales un retardo máximo de bit de 1µs. Se cuenta con un oscilador con cristal de 20 Mhz, la frecuencia de oscilador interna es de 10 Mhz.

Se fija el preescaler a BRP=4 (Brp=5), con lo que Tq= 5 /10 Mhz = 500 ns.

Para conseguir la velocidad nominal requerida se necesitan 1/(0,5 µs. 0,125 Mhz)=16 Tq

Se fija Prop_Seg=2 (de acuerdo con el retardo máximo esperado) y con Phase_Seg1=7 quedan 16-1-2-7= 6 Tq para Phase_Seg2. El muestreo se hará al final del 10º Tq. El Sjw se puede fijar en cualquier valor de su rango 1..4, para máxima robustez ante diferencias de osciladores puede fijarse en 4, un valor tan alto sólo es necesario si se utilizan resonadores cerámicos de baja precisión.

En todos los aspectos, pero especialmente en los referentes a configuración y funcionamiento del proceso de resincronización, es conveniente analizar con detenimiento la documentación del controlador concreto utilizado. Bosh ha publicado adendas y modificaciones a la especificación desde su primera versión para permitir mayor tolerancia en osciladores y deshacer ambigüedades, por otra parte los fabricantes de controladores pueden incluir variaciones y mejoras en el muestreo y resincronización. Existen herramientas informáticas para el cálculo de los parámetros de temporización de diferentes controladores. En aplicaciones críticas de alta velocidad conviene medir experimentalmente los retardos de línea y los retardos de controlador (existen técnicas basadas en respuesta a un bit dominante intercalado en una secuencia larga de recesivos).

8. Capa de transporte. Controladores

8.1 Formatos de trama

8.1.1 Trama de datos

Una trama de datos es generada por un nodo CAN cuando transmite información. Los campos incluidos en una trama de datos son para CAN Estandar. (Figura 24)

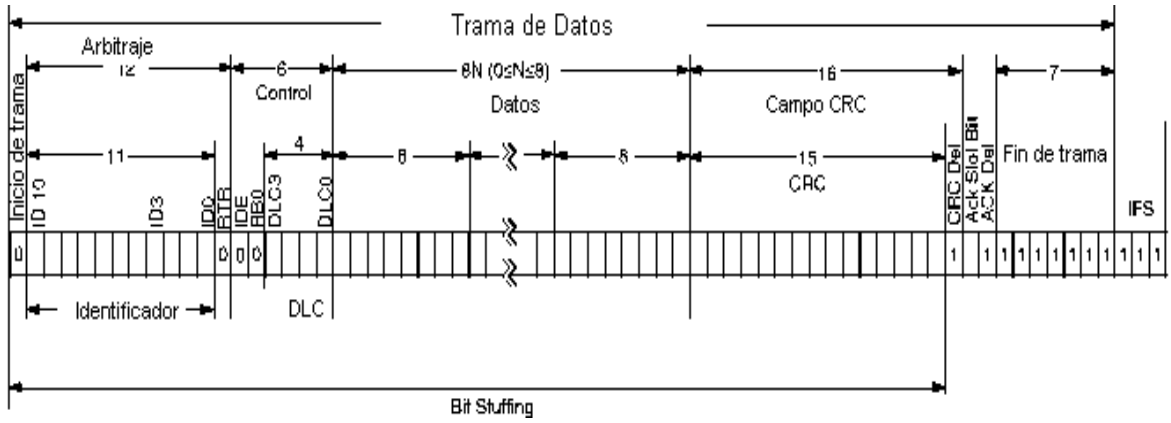


Figura 24. Trama de datos CAN (figura tomada de hoja de datos de MCP2510 de Microchip)

- ***Inicio de trama*** (SOF): El inicio de trama es un campo de un solo bit siempre dominante que indica el inicio de la transmisión. Los nodos receptores se sincronizan con el flanco de bajada de este bit.

- ***Arbitraje***: El campo de identificación está formado por el identificador de mensaje (11 bits) más el bit RTR. En una trama de datos el bit RTR es dominante. En una trama remota es

recesivo. Los bits de identificador se transmiten en orden de más significativo a menos significativo.

- ***Control***: El campo de control está formado por dos bits reservados para uso futuro y cuatro bits adicionales que indican el número de bytes de datos. En realidad el primero de estos bits (IDE) se utiliza para indicar si la trama es de CAN Estandar (IDE dominante) o Extendido (IDE recesivo). El segundo bit (RB0) es siempre recesivo. Los cuatro bits de código de longitud (DLC) indican en binario el número de bytes de datos en el mensaje (0 a 8)

- ***Datos***: Es un campo formado por 0 a 8 bytes de datos , es decir 0 a 64 bits en saltos de 8. Cada byte se transmite con bit más significativo primero.

- ***CRC***: Código de redundancia cíclica que genera el transmisor por la división módulo 2 de todos los bits precedentes del mensaje, incluyendo los de relleno si existen, por el polinomio generador: $X^{15}+ X^{14}+ X^{8}+ X^{7}+ X^{4}+ X^{3}+ X^{1}+1$, el resto de esta división es el código CRC transmitido. Los receptores comprueban este código. Tras el código CRC se incluye un bit recesivo (delimitador de CRC)

- ***Campo de reconocimiento*** (ACK): es un campo de dos bits que el transmisor pone como recesivos. El primero de estos bits se sobreescribe por un bit dominante de reconocimiento transmitido por los nodos que han recibido el mensaje correctamente. El bit de ACK queda así insertado entre dos bits dominantes de delimitación.

- ***Fin de trama*** (EOF). Cierra la trama, consiste en 7 bits recesivos sucesivos.

- ***Espaciado entre tramas*** (IFS). Consta de un mínimo de 3 bits recesivos.

La trama de datos de CAN Extendido se diferencia de la de CAN Estandar en que un bit dominante fijo (SRR) aparece en la posición del bit RTR de CAN Estandar, se fija el bit IDE como recesivo, siguen luego los 18 bits adicionales del identificador, el campo de control con RTR, dos bits reservados y la longitud de datos y el resto de la trama es análogo.

En un bus CAN pueden convivir nodos CAN Estandar y CAN Extendido, para ello los nodos CAN Estandar han de ser del tipo CAN 2.0B Pasivo, estos nodos reaccionan ignorando tramas CAN Extendido en lugar de señalarlas como erróneas. Los nodos que cumplen CAN 2.0B pueden funcionar en modo Estandar o Extendido indistintamente.

Durante este trabajo se hará referencia sobre todo a CAN Estandar, en todo caso las diferencias con CAN Extendido son mínimas, excepto la posibilidad de contar con un número mucho mayor de identificadores disponibles.

8.1.2 Trama remota

El formato es análogo a la trama de datos pero con el bit RTR recesivo[2]. Por otra parte una trama remota no incluye nunca datos. El identificador es el del mensaje que se solicita, el campo longitud corresponde a la longitud de ese mensaje

[2] Un ejemplo más de atención al detalle en la especificación CAN. Si hay colisión entre una petición de datos, con un identificador dado, y el mensaje transmitido espontáneamente por el nodo propietario de la información solicitada, la colisión se resolverá con la supervivencia del mensaje transmitido espontáneamente, ya que el bit RTR forma parte del campo de arbitraje ¡ el nodo que ha solicitado la información la recibirá inmediatamente!

8.1.3 Trama de error

Las tramas de error son generadas por cualquier nodo que detecta un error. Consiste en dos campos: Indicador de error ("Error Flag") y Delimitador de error. El delimitador de error consta de 8 bits recesivos consecutivos y permite a los nodos reiniciar la comunicación limpiamente tras el error. El Indicador de error es distinto según el estado de error (los estados de error de nodo se describirán en páginas sucesivas) del nodo que detecta el error:

- Si un nodo en estado de error "Activo" detecta un error en el bus interrumpe la comunicación del mensaje en proceso generando un "Indicador de error activo" que consiste en una secuencia de 6 bits dominantes sucesivos. Esta secuencia rompe la regla de relleno de bits y provocará la generación de tramas de error en otros nodos. Por tanto el Indicador de error puede extenderse entre 6 y 12 bits dominantes sucesivos. Finalmente se espera el campo de delimitación de error formado por los 8 bits recesivos. Entonces la comunicación se reinicia y el nodo que había sido interrumpido reintenta la transmisión del mensaje.

- Si un nodo en estado de error "Pasivo" detecta un error, el nodo transmite un "Indicador de error pasivo" seguido, de nuevo, por el campo delimitador de error. El indicador de error de tipo pasivo consiste en 6 bits recesivos seguidos y, por tanto, la trama de error para un nodo pasivo es una secuencia de 14 bits recesivos. De aquí se deduce que la transmisión de una trama de error de tipo pasivo no afectará a ningún nodo en la red, excepto cuando el error es detectado por el propio nodo que está transmitiendo. En ese caso los demás nodos detectarán una violación de las reglas de relleno y transmitirán a su vez tramas de error.

Tras señalar un error por medio de la trama de error apropiada cada nodo transmite bits recesivos hasta que recibe un bit también recesivo, luego transmite 7 bits recesivos consecutivos antes de finalizar el tratamiento de error.

8.1.4 Espacio entre tramas

El espacio entre tramas separa una trama (de cualquier tipo) de la siguiente trama de datos o interrogación remota. El espacio entre tramas ha de constar de, al menos, 3 bits recesivos. Esta secuencia de bits se denomina "intermission". Una vez transcurrida esta secuencia un nodo en estado de error activo puede iniciar una nueva transmisión o el bus permanecerá en reposo. Para un nodo en estado error pasivo la situación es diferente, deberá espera una secuencia adicional de 8 bits recesivos antes de poder iniciar una transmisión. De esta forma se asegura una ventaja en inicio de transmisión a los nodos en estado activo frente a los nodos en estado pasivo.

8.1.5 Trama de sobrecarga

Una trama de sobrecarga tiene el mismo formato que una trama de error activo. Sin embargo, la trama de sobrecarga sólo puede generarse durante el espacio entre tramas. De esta forma se diferencia de una trama de error, que sólo puede ser transmitida durante la transmisión de un mensaje. La trama de sobrecarga consta de dos campos, el Indicador de Sobrecarga, y el delimitador. El indicador de sobrecarga consta de 6 bits dominantes que pueden ser seguidos por los generados por otros nodos, dando lugar a un máximo de 12 bits dominantes. El delimitador es de 8 bits recesivos.

Una trama de sobrecarga puede ser generada por cualquier nodo que debido a sus condiciones internas no está en condiciones de iniciar la recepción de un nuevo mensaje. De esta forma retrasa el inicio de transmisión de un nuevo mensaje. Un nodo puede generar como máximo 2 tramas de sobrecarga consecutivas para retrasar un mensaje. Otra razón para iniciar la transmisión de una trama de sobrecarga es la detección por cualquier nodo de un bit dominante en los 3 bits de "intermission". Por todo ello una trama de sobrecarga de generada por un nodo dará normalmente lugar a la generación de tramas de sobrecarga por los demás nodos dando lugar, como se ha indicado, a un máximo de 12 bits dominantes de indicador de sobrecarga.

8.1.6 Arbitraje

Un nodo transmisor monitoriza constantemente el estado del bus. Durante la transmisión del campo Arbitraje la detección de un bit dominante, cuando el bit transmitido ha sido recesivo, hace que el nodo detenga la transmisión y pase a recepción de la trama. De esta forma no hay pérdida de información y no se destruye por colisión ninguna trama de datos o remota. La especificación Bosh admite para CAN Standard los identificadores en el rango 0x000 a 0x7EF[3]. En dicha especificación se indica que los 7 bits más significativos no han de ser todos recesivos. Sin embargo muchos controladores admiten el rango 0x000 a 0x7FF. Un mensaje de máxima prioridad utilizará, por tanto, el identificador 0x000.

En un bus único un identificador de mensaje ha de ser asignado a un solo nodo concreto, es decir, se ha de evitar que dos nodos puedan iniciar la transmisión simultánea de mensajes con el mismo identificador y datos diferentes. La filosofía CAN es de que un mensaje es único en el sistema. Las tramas remotas con identificador concreto que puedan ser generadas por cualquier nodo han de coincidir en cuanto al campo longitud. Resumiendo, definiendo un mensaje como el conjunto identificador + longitud de campo de datos + semántica de estos datos, el mensaje ha de ser único en el sistema y estar asignado a un nodo concreto. Así, por ejemplo, si en un automóvil existe la variable "presión de aceite" esta variable ha de ser transmitida por un nodo concreto, con un identificador concreto, con longitud fija y consistente con la codificación de la información en el campo de datos.

8.2 Detección y gestión de errores

8.2.1 Relleno de bits

Los campos Inicio de Trama, Arbitraje, Control, Datos y CRC se codifican con relleno de bits, cuando el transmisor detecta una secuencia de 5 bits del mismo signo añade uno de relleno de signo contrario. El resto de campos de las tramas de datos y remota y las tramas de error o sobrecarga no siguen la regla de relleno de bits.

8.2.2 Detección de errores

Se distinguen los siguientes tipos de error de trama:

- ***Error de bit***: Un nodo que transmite monitoriza el bus simultáneamente. Cualquier bit que reciba con polaridad inversa a la que ha transmitido se considera un error de bit, excepto cuando

[3] Par a designar valores numéricos codificados en hexadecimal se seguirá la sintaxis del lenguaje C precediendo el valor con 0x

se recibe durante el campo de arbitraje o en el bit de reconocimiento. Tampoco se considera error de bit la detección de bit dominante por un nodo en estado de error pasivo que transmite un señalizador pasivo de error.

- ***Error de relleno***: Se considera error de relleno la detección de 6 bits consecutivos del mismo signo en cualquier campo que haya de seguir la regla de relleno.
- ***Error de CRC***: Cuando el cálculo de CRC por un receptor no coincide con el recibido en la trama.
- ***Error de forma***: cuando un campo de formato fijo se recibe alterado en algún bit
- ***Error de reconocimiento***: cuando ningún nodo cambia a dominante el bit de reconocimiento

Ante detección de uno de estos errores el nodo detector inicia la transmisión de una trama de error.

8.2.3 Validación de mensajes

Para un transmisor el mensaje es válido si no detecta ningún error hasta el final del campo Fin de Trama. Si se detecta error el nodo lo intentará retransmitir automáticamente tan pronto como el bus queda en reposo. Un mensaje es válido para un receptor si no detecta ningún error hasta el anteúltimo bit del campo Fin de Trama.

8.2.4 Aislamiento de nodos defectuosos

Se se produce una perturbación local en un nodo de la red (para ilustrar esta situacion, imáginese un nodo situado varios cientos de metros alejado de los demás en una factoría y sometido a condiciones ambientales agresivas) la lógica de marcado de errores hará que este nodo envíe tramas de error que, a su vez, provocarán errores en los demás nodos. Para evitar que un nodo en problemas condicione el funcionamiento del resto de la red se han incorporado en la especificación de CAN, y son gestionadas por los controladores, medidas de aislamiento de nodos defectuosos. Un nodo puede encontrarse en uno de los tres estados siguientes en relación con la gestión de errores:

- ***Error activo***: Es el estado normal de un nodo. Participa en la comunicación normalmente y en caso de detección de error envía una trama de error activa
- ***Error pasivo***: Un nodo en estado de error pasivo participa en la comunicación, sin embargo ha de esperar una secuencia adicional de bits recesivos antes de intentar transmitir. Sólo puede señalar errores con una trama de error pasiva
- ***Anulado***: En este estado deshabilitará su transceptor y no participa en la comunicación.

La evolución entre estos estados se basa en dos contadores incluidos en el controlador de comunicaciones: Contador de errores de transmisión (TEC) y Contador de errores de recepción (REC) Los contadores se actulizan de acuerdo a las reglas expuestas en las siguientes tablas:

Condición	TEC
Un nodo transmisor detecta un error de bit al enviar un mensaje activo de error[4]	incrementa 8
Al detectar el 14º bit sucesivo dominante tras un indicador de error activo o de sobrecarga o detectar el 8º bit dominante consecutivo tras un indicador pasivo de error. Después de cada secuencia sucesiva de 8 bits dominantes sucesivos	incrementa 8
Cualquier otro tipo de error (relleno, trama, CRC, ACK)	incrementa 8
Ante una transmisión de trama válida	Decrementa en 1 si >0

Condición	REC
Un nodo receptor detecta un error de bit al enviar un mensaje activo de error	incrementa 8
Un nodo receptor detecta un bit dominante como primer bit tras el envío de un indicador de error	incrementa 8
Al detectar el 14º bit sucesivo dominante tras un indicador de error activo o de sobrecarga o detectar el 8º bit dominante consecutivo tras un indicador pasivo de error. Después de cada secuencia sucesiva de 8 bits dominantes sucesivos	incrementa 8
Cualquier otro tipo de error (relleno, trama, CRC, ACK)	incrementa 1
Ante una recepción de trama válida	Decrementa en 1 si >0 y <=127 Si es mayor de 127 se fija en un valor entre 119 y 127

Existen algunas excepciones y tratamientos especiales del TEC en las siguientes situaciones:

- Error de relleno durante arbitraje, cuando se recibe un bit transmitido dominante y ha sido transmitido como recesivo. No se incrementa el TEC
- Se detecta un error de reconocimiento (ACK) y no se detectan bits dominantes al transmitir el indicador de error pasivo. No se incrementa el TEC. Así, si sólo hay un nodo en el bus tras transmitir un mensaje no recibe reconocimiento. Esto se tomará como un error y el mensaje se repite. Cuando el nodo llega a la situación de error pasivo el TEC no se incrementará más. De esta forma no pasará a estado Anulado.

Un nodo pasa de estado de Activo a Pasivo cuando su TEC supera 127 o cuando el REC supera 127. Cuando un nodo pasa de Activo a Pasivo envía una trama de error activo.

[4] Cuando un nodo detecta un error debido a una perturbación local, enviará un indicador de error. Los demás nodos detectan un error tras el sexto bit del Identificador de nodo y enviarán entonces tramas de error. El nodo perturbado detectará un bit dominante tras su trama de error e incrementará su contador de errores en 8, además del incremento en 1 de REC debido a la detección de error, los demás nodos sólo incrementan en 1. Por tanto el nodo localmente perturbado pasará a condición de error pasivo antes que los demás.

Un nodo pasa de Pasivo a Anulado cuando TEC supera 255

Un nodo casa de Pasivo a Activo cuando tanto TEC como REC bajan de 128

Un nodo puede pasar de Anulado a Activo (reiniciando TEC y REC) tras 128 secuencias de 11 bits recesivos sucesivos[5]

Cuando un contador de error alcanza el valor 96 se fijará un bit de estado en los controladores y se puede generar una interrupción al procesador. También se produce una interrupción ante paso a estado de nodo anulado. Algunos controladores permiten el acceso desde la aplicación a los contadores de error. En todo caso, el controlador descarga a la CPU principal de la gestión de errores de trama informando sólo de cambios de estado.

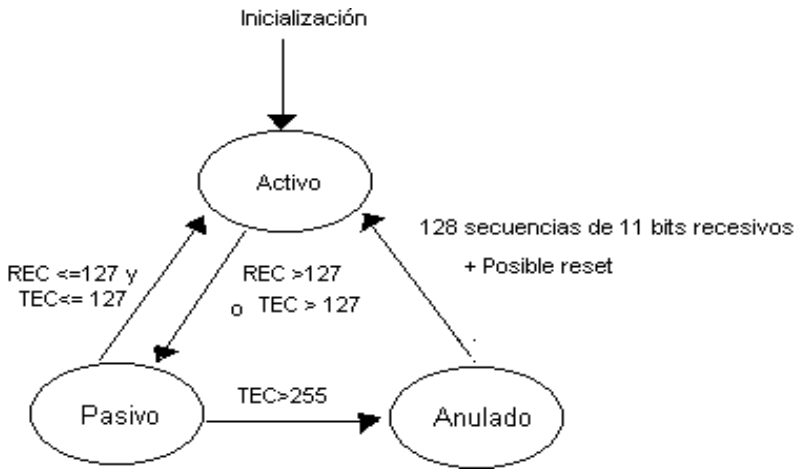


Figura 25 . Evolución entre estados de error

Depende de la aplicación la política de acciones concretas que se toman ante paso de un nodo a estados de error. En aplicaciones críticas puede llegar a detenerse el funcionamiento total del sistema, en otro tipo de aplicaciones se sigue el funcionamiento incluso en condiciones muy degradadas. Obsérvese que el estado de error pasivo permite la comunicación con el nodo en ese estado de forma que el resto del sistema puede informarse de la situación y tomar medidas concretas.

8.3 Tratamiento de mensajes en el nodo. Buzones y filtrado.

En un controlador CAN pueden incluirse varios registros de recepción, conocidos como buzones, la CPU recoge los mensajes desde estos buzones. Existen dos niveles fundamentales

[5] Esta regla aparece en la especificación de Bosh. Sin embargo algunos controladores pueden configurarse para exigir condiciones más estrictas para que un nodo pase de Anulado a Activo, además en muchos casos la aplicación puede exigir un reset manual del nodo. Una regla en algunos controladores es la posibilidad de paso de Anulado a Activo tras detectar un número dado de mensajes válidos en el bus, así un nodo defectuoso que conmute entre Activo y Anulado y viceversa permite, entre conmutaciones, el funcionamiento del bus durante 128 mensajes.

de controlador CAN en cuanto a sofisticación en la gestión de buzones, los conocidos con Basic-CAN y Full-CAN (nada que ver con CAN estándar y CAN extendido).

En un controlador Basic-CAN se tendrán muy pocos buzones, quizás sólo uno y la CPU ha de recoger cada mensaje recibido antes de que sea sobreescrito por otro. Si se recibieran todos los mensajes que circulan por la red se podría saturar facilmente la CPU, dedicando mucha carga de proceso a la recogida de mensajes inútiles. Por ello todos los controladores Basic-CAN incorporan uno registros de máscara y filtro que permiten establecer una barrera selectiva sobre los mensajes a los que se permite paso hacia el buzón. El nodo puede configurar la máscara y filtro para que la CPU sólo sea interrumpida por los mensajes de interés.

En un controlador Full-CAN se dispone de un amplio conjunto de buzones, cada uno de los cuales puede ser programado para que atienda un mensaje concreto. De esta forma las variables de red en las que está interesado el nodo se recogen y se mantienen en esos buzones. Se tiene así una memoria compartida con la red que flexibiliza mucho el acceso desde la CPU.

Hay controladores que combinan las características Basic-CAN y Full-CAN incluyendo buzones con asignación a mensajes concretos y buzones con máscara y filtro.

8.4 Controladores CAN

Los controladores CAN están disponibles en distintas opciones:

a) controladores independientes, conectables a cualquier CPU por bus serie o paralelo.

b) controladores integrados en microcontrolades como un periférico más

c) modelos VHDL o Verilog de controladores implementables sobre dispositivos lógicos programables (FPGAs o ASICs)

Ha perdido interés una forma adicional de controlador, los conocidos como SLIOs, dispositivos de entrada-salida enlazables a bus CAN y sin CPU (periféricos CAN). En la actualidad son sustituidos con ventaja por microcontroladores de bajo precio, con bus CAN.

Tan sólo se mencionará al controlador más clásico, el AN82527 de Intel. El sustituto del primer controlador CAN el 82526, de tipo Full-CAN con enlace por bus serie o paralelo y quizás al PCA82C200 de Philips, actualmente sustituido por el SJ1000, por haber sido muy utilizados en diseños de cierta antigüedad. En la actualidad la variedad de controladores independientes y, sobre todo, de microprocesadores y microcontroladores con CAN incorporado es muy amplia. Para una lista exhaustiva y puesta al día se puede acudir a las páginas de la organización CiA (www.can-cia.de)

9. Las capas de aplicación

9.1 Perspectiva

CAN se ha utilizado y se utiliza, muchas veces, en aplicaciones en las que no se establece una capa estructurada entre el nivel de enlace que proporciona el propio controlador y la aplicación en el nodo. En muchos sistemas de control industrial se adopta CAN como un medio de compartición de variables de red en el que los nodos tienen variables e identificadores predefinidos y permanentes.

Sin embargo, es habitual que surja la necesidad de funcionalidades adicionales clásicamente asociadas a los niveles superiores del modelo ISO. Aunque en el mundo de los protocolos asociados a CAN, y en el de los protocolos de buses de campo industrial en general, muchas veces se dice que las capas intermedias (red, transporte, sesión, presentación) no existen, en realidad se quiere decir que no existen como normalización concreta. Cuando el modelo ISO se interpreta como eso, como modelo, algunas de esas capas sí están presentes. En todo caso ha sido normal considerarlas como integradas en la capa de aplicación. La justificación es ([27]):

- No es habitual la necesidad de gestión de red en buses CAN.
- No se considera necesaria una capa de transporte estándar ya que en sistemas industriales el contenido de la información es normalmente de tiempo real, si una transferencia falla es normalmente más importante enviar los datos actualizados que repetir los obsoletos. Los servicios necesarios para la transferencia de paquetes de información de más de 8 bytes se incluyen en las capas de aplicación.
- No se crean normalmente sesiones.
- La codificación de los datos distribuidos en los nodos se establece por medio de reglas de codificación en las capas de aplicación, lo que hace innecesaria una capa de presentación.

Los estándares que cubren estos aspectos de CAN son conocidos como capas de aplicación. La necesidad de capas de aplicación normalizadas ha surgido sobre todo en el sector de los buses de campo industrial, en el que la intercambiabilidad entre dispositivos es una característica básica.

Entre las capas de aplicación más extendidas cabe señalar:

- CANOpen: de ámbito europeo, sobre todo. Promocionada por CiA (Can in Automation).
- DeviceNet: la asociada al bus de campo DeviceNet desarrollado por Allen-Bradley y actualmente abierto y gestionado por la ODVA.
- SDS: la asociada al bus de campo SDS de Honeywell.
- OSEK: Orientada a sistemas distribuidos de tiempo real en vehículos.

- CAN Kingdom. De la empresa sueca Kvaser. Orientada a automatización de maquinaria.

Se prestará atención en este trabajo a CANOpen como ejemplo de las funcionalidades, y requisitos de implementación, relacionados con una capa de aplicación CAN.

9.2 CAL y CANOpen

Philips (en su división de electromedicina) definió una capa de aplicación para CAN denominada CAL (CAN Application Layer). La organización CiA la adoptó y desarrolló publicándola en una serie de normas [29]. CAL define los servicios básicos de una capa de aplicación en cuanto a metodología, pero no concreta la estructura de la información. Para que resultara un estándar más concreto, utilizable en normalización industrial, fueron necesarias las especificaciones adicionales que dieron lugar, en el contexto de un proyecto europeo de investigación, a CANOpen [28]. En las primeras versiones de su especificación CANOpen hacía referencia a CAL de forma contínua. En la actualidad CANOpen puede considerarse como una norma de bus de campo, basada en CAN, con entidad propia y en competencia con DeviceNet, SDS, CANKingdom etc. En su especificación, recogida en diversos documentos publicados por la organización CiA, se contemplan todas las capas de protocolo, desde nivel físico (conectores + capa física ISO 11898), a capa de enlace de datos basada en la especificación de Bosh, hasta la propia capa de aplicación, que incorpora un subconjunto de las características de CAL, a la que se añade la definición de perfiles específica de CANOpen.

En este apartado se analizará CANOpen como norma íntegra para interconexión de dispositivos al nivel de célula y centrándose sobre todo en la capa de aplicación ya que la capa física y de transporte ha sido tratada en los apartados previos en relación con lo que se puede denominar como “bus CAN” genérico.

Una descripción detallada de CANOpen, como de cualquier capa de aplicación no trivial, da lugar a decenas de páginas. De ese manual, una parte importante, en volumen, es simplemente una recopilación de datos y detalles de implementación. Los conceptos básicos ocupan mucho menos volumen. Se tratarán de exponer estos conceptos no extendiéndose demasiado pero tratando a la vez de establecer una perspectiva clara de su estructura. Se ha de tener en cuenta, también, que CANOpen está en evolución, la organización CiA está haciendo un esfuerzo importante en su normalización y las últimas revisiones (en especial DS301 V4.0) amplian y aclaran la especificación.

9.3 Estructura básica de CANOpen

CANOpen hereda de CAL cuatro areas de servicio básicas.

1) CMS (“CAN Based Message Specification”): establece la forma en la funcionalidad de un dispositivo se hace accesible en el sistema distribuido soportado en CAN. Ofrece objetos de tipo Variable, Evento y Dominio.

2) NMT (“Network Management”): ofrece servicios para gestionar la red, es decir, inicializar, arrancar y parar nodos, detectar fallos de nodo, etc.

3) DBT ("Distributor"). Ofrece un servicio de distribución de los identificadores de mensaje CAN a los nodos de la red.

4) LMT ("Layer Management"). Ofrece servicios de configuración de las capas de comunicación CAN, por ejemplo, velocidad en bps, temporización de bit, dirección de un nodo en la red. etc.

Los tres últimos servicios, NMT, DBT y LMT son servicios de configuración y gestión del bus de campo en sí mismo. Se gestionan de acuerdo a un protocolo maestro-esclavo y sólo se implementan, en la parte de maestro, en nodos con capacidad de configuración y gestión. En cambio el servicio CMS es el servicio básico de proceso contínuo.

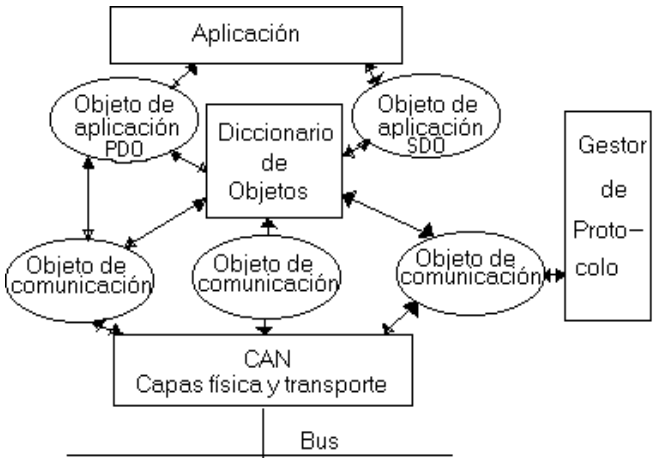
Un elemento adicional básico en CANOpen es el Diccionario de Objetos (Object Dictionary). De hecho, en CANOpen, las areas de servicio definidas anteriormente son traducidas a gestión del diccionario de objetos. En dicha estructura de información se establecen, desde los tipos básicos de representación de los datos, hasta el mapeado de la información a intercambiar en tramas CAN. Es el enlace básico entre los objetos de comunicación y las variables y objetos relacionados con la aplicación.

Finalmente se establecen los denominados perfiles, pueden clasificarse en dos tipos:

- Perfil de communicación. Establece el comportamiento del nodo y su funcionalidad como elemento de la red.

- Perfil de dispositivo. Establece su funcionalidad propia en cuanto a elemento de proceso, se han definido perfiles para dispositivos concretos: Nodos de entrada/salida digital, nodos de control de motores etc.

-



-

Figura 26 . Modelo CANOpen

CANOpen sigue una definición basada en orientación a objetos. En la Figura 26 se representa este modelo. Se distinguen los siguientes tipos de objetos fundamentales:

- Objetos de comunicación para gestión de red: este tipo de objetos se acceden desde un nodo maestro en una relación maestro-esclavo y permiten la verificación y configuración del nodo

en la red, es decir gestión de red. Este tipo de objeto actúa sobre el gestor de protocolo (máquina de estados que procesa la comunicación).

- Objetos de comunicación para transferencia de datos de servicio ("Service data objects", SDOs). Tienen acceso a todos los objetos en el dispositivo a través del diccionario de objetos y con una relación de tipo cliente-servidor. Por medio de este tipo de objetos se pueden acceder todas las funcionalidades del dispositivo, fundamentalmente para configuración. Sin embargo no son muy apropiados para acceso en tiempo real debido a la sobrecarga que supone la indexación a través del diccionario de objetos.

- Objetos de comunicación para transferencia de datos de proceso ("Process data objects", PDOs). Este tipo de objetos establece una relación directa con un objeto de aplicación para el intercambio de datos de forma síncrona o asíncrona, en tiempo real y mediante una relación productor-consumidor.

- Objetos de aplicación de proceso. Como ya se ha indicado son los objetos de tiempo real, representan las variables de red en tiempo real del nodo.

- Objetos de aplicación de servicio. Representan la información de configuración, calibración, programas o cualquier otra estructura de datos de mayor volumen y sin necesidades de gestión en tiempo real.

Los objetos de comunicación de CANOpen incluyen además algunos tipos específicos: objeto de sincronización, objeto de reloj, objeto de emergencia. Las diferencias básicas entre SDOs y PDOs se presentan en Tabla 6.

SDO	PDO
• Información de mayor volumen (se pueden realizar transferencias de más de 8 bytes)	• Información de menor volumen (8 bytes máximo)
• Transferencia de baja prioridad	• Transferencia de alta prioridad
• Se direccionan en diccionario de objetos (indice + subíndice)	• Diversos objetos se pueden mapear en una trama CAN
• Transferencia asíncrona	Transferencia síncrona, asíncrona y cíclica.

Tabla 6 . Diferencia entre SDOs y PDOs

Los servicios de red (NMT) se gestionan desde un nodo maestro hacia el resto de nodos esclavo. Estos servicios proporcionan funciones de iniciación de nodos, supervisión del estado de los nodos, estado de la comunicación, escritura y lectura de datos de configuración.

Se parte de que cualquier nodo ha de contener en origen un conjunto de objetos predefinidos mínimo (aunque puede ser que en origen se tenga un perfil de nodo más específico). A partir de ese conjunto el nodo puede adquirir una configuración más concreta desde un nodo remoto que actúe como configurador. Esa configuración puede ser almacenada en forma no volátil en muchos casos y determina el perfil del nodo.

La configuración de un nodo queda definida en dos documentos electrónicos escritos con

codificación ASCII y que pueden ser datos de entrada para sistemas de configuración y prueba de conformidad de un nodo: la hoja de datos electrónica (EDS), que establece el mapa de objetos, y el fichero de descripción de dispositivo (DCF) que determina valores para esos objetos.

9.4 Componentes fundamentales de CANOpen.

9.4.1 SDOs

Los objetos de servicio (SDOs) se utilizan para que un nodo cliente tenga acceso a un nodo servidor para transferencia de datos de baja prioridad y volumen medio-alto. En concreto el propio acceso al diccionario de objetos ha de hacerse a través de SDOs. Un SDO debe definirse como servidor o cliente, según su función. En todo nodo CANOpen ha de existir al menos un SDO servidor que permita el acceso al diccionario de objetos.

Un SDO se soporta sobre dos mensajes CAN (dos identificadores), uno destinado al envío de mensajes de cliente a servidor y el otro para el flujo en sentido opuesto. Estos mensajes siempre son de 8 bytes, aunque hayan de ser completados con datos inútiles.

En un SDO se incluye un índice y subíndice para direccionamiento del diccionario de objetos. Por medio de esa combinación de índice y subíndice es posible el acceso a estructuras complejas de datos.

La transferencia de datos de un SDO se realiza por medio de alguno de los procedimientos permitidos en la norma:

- Transmisión segmentada: el modo más genérico
- Transmisión Inmediata: método optimizado para volumen reducido de datos
- Transmisión de bloque: método optimizado para volumen alto de datos (ficheros, aplicaciones)

La transferencia la inicia un cliente, pero puede ser abortada tanto por el cliente como por el servidor.

La descripción del formato exacto de las tramas y secuencias que siguen cada uno de estos procedimientos es una de esas partes de la especificación CANOpen que dan lugar a varias páginas de información de detalle. En una red CANOpen se pueden tener hasta 256 canales SDO (dos identificadores por canal).

9.4.2 PDOs

Los objetos de proceso están destinados a proporcionar acceso de tiempo real y directo a los objetos de aplicación de un dispositivo. La información que intercambian es reducida en tamaño, máximo 8 bytes, y no incluye sobrecarga de control incluida en el campo de datos. El significado de los datos habrá sido definido sin ambigüedad, previamente, a través del diccionario de objetos. El intecambio de PDOs sigue el modelo productor-consumidor y se utilizan tanto los métodos “push” como “pull” (3.7).

Se consideran dos servicios básicos para el intercambio de PDOs entre un productor y, al menos, un consumidor: escritura de PDO que sigue el método “push” y lectura de PDO que sigue el método “pull”.

Los PDOs pueden ser organizados con amplia flexibilidad en CANOpen y en la capa de aplicación se han contemplado tipos y modos de iniciación de PDO que permiten configurar sistemas de adquisión de datos y control en tiempo real. En concreto:

El lanzamiento de un PDO desde un nodo puede ser originado por:

- Un evento en el propio nodo.
- Por agotamiento de un periodo de tiempo asignado al PDO
- Por demanda desde un nodo remoto

La transmisión puede organizarse según diferentes categorías:

- Transmisión síncrona. La transmisión se inicia sólo ante recepción de un mensaje de sincronización periódico, el objeto de sincronización, transmitido por un dispositivo que dirige ese proceso de sincronización. Los nodos que reciben un objeto de transmisión síncrona sólo harán uso de la información ante recepción del próximo objeto de sincronización. De esta forma se pueden establecer mecanismos de actuación coordinada en un sistema distribuido. Dentro de la transmisión síncrona se pueden establecer, a su vez, dos tipos de criterios: transmisión síncrona periódica en cada instante de sincronización, y transmisión síncrona aperiódica, en la que la transmisión, ante recepción de mensaje de sincronización, sólo se inicia si se ha producido un determinado evento en el nodo.
- Transmisión asíncrona. En este caso la transmisión puede iniciarse de forma asíncrona por un evento que tenga su origen en la aplicación del nodo (objeto de aplicación) o por un evento relacionado con un objeto de comunicación, bien sea agotamiento de una temporización asíncrona o disparo por una solicitud remota.
- Se pueden configurar tiempos de inhibición y generación de PDOs de forma que se creen ventanas virtuales de tiempo asignadas a cada PDO, así se consigue que la información se actualice en los instantes correctos sin sobrecargar el bus más de lo necesario.

Estos conceptos son coherentes con las necesidades de un sistema de tiempo real distribuido

Los PDOs se configuran a través del diccionario de objetos. La configuración de un PDO supone:

1) Establecer el mapeo de datos en el mensaje que soporta el PDO, es decir establecer qué datos se envían en cada uno de los bytes del mensaje. Esos datos reflejarán uno o varios objetos de aplicación. En el diccionario de objetos se define en el parámetro de comunicación del PDO.

2) Fijar los parámetros del PDO: Identificador del mensaje, tipo de transmisión etc. Se define en el diccionario de objetos en el parámetro de mapeo del PDO.

La configuración de PDOs es la operación fundamental que determina las características de tiempo real, eficiencia, carga del bus y prioridades en la aplicación distribuida.

En una red CANOpen puede haber hasta 512 PDOs de transmisión (“push”) y otros 512 de recepción (“pull”). Está prevista la configuración de un tiempo mínimo de inhibición entre transmisión de PDOs para evitar la saturación de la red.

9.4.3 Objetos especiales

Existen en una red CANOpen, y se recogen en el diccionario de objetos, algunos objetos singulares, entre ellos:

- Objeto de sincronización (0x1005). Elemento básico de sincronización de PDOs, se transmite según un concepto maestro-esclavo desde un nodo originador único. Es un mensaje de alta prioridad sin datos.
- Temporización absoluta (0x1013). Un mensaje que contiene el tiempo absoluto en milisegundos con origen en el 1 de Enero de 1984 y contenido en una secuencia de 48 bits (6 bytes). Existe también una variante de alta resolución (microsegundos).
- Objeto de emergencia. Emitido ante un error interno del nodo. En la especificación se definen los códigos de error aplicables.
- Objeto de vigilancia de nodo. Se tratará al describir los servicios de red.
- Tipo de dispositivo (0x1000).
- Registro de error (0x1001)
- Objeto identificador de entidad (0x1018)

9.4.4 El Diccionario de Objetos

El diccionario de objetos de un dispositivo CANOpen es el más importante elemento de cara a definir la funcionalidad del nodo. El diccionario de objetos es, en esencia, un conjunto de objetos accesibles desde la red siguiendo unas reglas predefinidas. Cada objeto se direcciona por medio de un índice de 16 bits y un subíndice de 8 bits.

INDICE	**Objeto**
0x0000	Reservado
0x0001-0x001F	Tipos de datos estáticos
0x0020-0x003F	Tipos de datos complejos (PDOCommPar, SDOParam etc.)
0x0040-0x005F	Tipos de datos específicos de fabricante
0x0060-0x007F	Tipos de datos estáticos específicos de perfil de dispositivo
0x0080-0x009F	Tipos de datos complejos específicos de perfil de dispositivo
0x00A0-0x0FFF	Reservados
0x1000-0x1FFF	Zona de definición de perfil de comunicación
0x2000-0x5FFF	Zona de definición de perfil específico de fabricante
0x6000-0x9FFF	Zona de definición de perfiles de dispositivo normalizado
0xA000-0xFFFF	Reservados

En la primera parte del diccionario de objetos se colocan definiciones de tipos de datos básicos. No es necesario traducir esta parte a implementación de ningún tipo en el nodo. Se

trata tan sólo de la codificación de tipos de datos.

La descripción de un objeto con un índice dado incluye:

- El nombre de clase (DEFTYPE, VAR, ARRAY etc.) que determina un código numérico de objeto que establece su funcionalidad.
- El nombre del objeto con una descripción textual
- El tipo de la entrada de diccionario. Puede estar predefinida
- Los atributos de acceso (lectura, escritura, lectura-escritura, constante, lectura-escritura para entrada de proceso, lectura-escritura para salida de proceso etc.)
- Atributo de implementación (M/O, obligatoria u opcional)
- Rango de valores
- Valor por defecto en inicialización
- Si hay mapeo de PDO

Nombre simbólico	**Descripción**	**Código**
NULL	Entrada vacía	0
DOMAIN	Segmento de datos extenso y sin estructura (por ejemplo un programa cargable)	2
DEFTYPE	Definiciones de tipos simples	5
DEFSTRUCT	Definicion de estructura tipo registro	6
VAR	Variable de tipo simple	7
ARRAY	Array de variables del mismo tipo	8
RECORD	Registro de variables de distinto tipo	9

Tabla 7. Clases de objeto

Una entrada de diccionario de objetos es traducible a una estructura (o registro) en un lenguaje de programación, en el caso más general.

Los PDOs se configuran en el diccionario de objetos por medio de entradas en el rango 0x1400-0x1BFF en los que se fijan dos entradas de diccionario por PDO: el parámetro de configuración y el mapeo. El parámetro de configuración fija las variables que determinan el tipo de transmisión: identificador de mensaje, tipo de transmisión, valores de tiempos de inhibición o disparo de evento si procede etc. El mapeo establece qué variables de aplicación, a su vez definidas como objetos de aplicación en el diccionario, se mapean en el campo de datos del PDO. El formato del objeto de mapeo es el indicado en la siguiente:

Indice	Subíndice	Descripción	Tipo de datos
1XXX	0x00	Número de objetos mapeados	Unsigned8
	0x01	Objeto 1	Unsigned32
	0x02	Objeto 2	Unsigned32
	…	….	Unsigned32
	n	Objeto n	Unsigned32

Formato de campo de objeto:

31 16	15 8	7 0
Indice de 16 bits	Subindice 8 bits	Longitud

Es decir para mapear objetos a un PDO se ha de crear un objeto de comunicación que contiene el número de objetos mapeados y una sucesión de campos que indican qué objetos se mapean en el PDO. Cada campo codifica la información del objeto por medio de su índice, subíndice y longitud empaquetados en una palabra de 32 bits. Como resultado de este mapeo se tendrá un mensaje CAN con identificador concreto que contendrá en su campo de datos uno o varios objetos claramente empaquetados. Una vez los nodos interesados en dicho PDO (el originario y los potenciales consumidores) tengan esta información asumida, la comunicación se realiza de manera eficiente y sin necesidad de consultas posteriores al diccionario.

Con los SDOs, en cambio, siempre acceden a datos de aplicación a través del diccionario de objetos. En el primer segmento transferido se indica el índice y subíndice del objeto a leer o escribir. La transferencia se gestiona por medio de dos SDOs. La combinación de índice y subíndice se denomina multiplexor, ya que permite la transferencia de estructuras de datos largas y complejas a través de dos enlaces de comunicación únicos

9.4.5 gestión de red

La gestión de red en CANOpen (NMT) sigue una relación maestro-esclavo y se basa en gestión por nodos. Requiere la existencia, permanente o temporal, de un dispositivo en la red que ejerza la función maestro (NMT-maestro). Cualquier otro nodo tiene funcionalidad NMT-esclavo, esta funcionalidad NMT queda definida por su identificador de nodo. Se consideran los siguientes grupos funcionales:

- Servicios de iniciación de nodos
- Servicios de supervisión
- Servicios de configuración

Iniciación y conmutación de estado

El servicio NMT-esclavo consiste, fundamentalmente, en la gestión de una máquina de estados que determina el comportamiento del nodo. Los estados fundamentales son:

- Iniciación. Proceso de iniciación interna
- Pre-operacional. Se tiene acceso al nodo por medio de SDOs para configuración. No se permite el acceso por medio de PDOs.
- Operacional. El nodo funciona con acceso completo por medio de SDOs y PDOs, aunque algunas funcionalidades permitidas en modo pre-operacional pueden quedar inhibidas (carga de programas de aplicación, por ejemplo)
- Parada. El nodo interrumpe su comunicación por PDOs o SDOs y puede entrar en un estado de funcionamiento que depende de la aplicación.

El NMT-Maestro emite órdenes de cambio de estado del tipo: arranque de nodo, parada de nodo paso a pre-operacional, reinicialización de nodo y reinicialización de comunicación. El NMT-Esclavo puede emitir mensajes de indicación de arranque de nodo ("boot-up protocol")

Supervisión

La supervisión del estado de nodos se basa en un doble proceso: El NMT-Maestro detecta dispositivos ausentes manteniendo una base de datos en la que se registra el estado esperado de cada nodo. Monitoriza los PDOs que emite cada nodo de cara a registrar su estado. Los NMT-esclavos, a su vez, comprueban constantemente la existencia de un NMT-Maestro en la red. Se ha definido también una función alternativa de latido basada en relaciones productor-consumidor, el productor de latido emite un mensaje cíciclo que puede ser tenido en cuenta por los consumidores y dar lugar a la generación de un evento.

Configuración

La configuración por defecto de una red CANOpen se basa en la asignación de identificadores de mensajes a nodos siguiendo un esquema basado en códigos de función y dirección de nodo. En el identificador los 4 bits más significativos, a efectos de arbitraje, determinan la función, los 7 inferiores la dirección de nodo. La dirección de nodo puede basarse en microinterruptores, configuración en memoria no volátil etc. Está definido un diccionario de objetos por defecto que establece los objetos mínimos requeridos por un nodo CANOpen antes de configuración. Se incluye un objeto de emergencia, un SDO, un máximo de 4 PDOs de recepción y 4 de transmisión, objeto de sincronización, objeto de temporización etc.

El enlace de PDOs por defecto se realiza de forma que se tiene una relación maestro-esclavo, a partir de esa relación el NMT-Maestro puede reconfigurar los nodos remapeando PDOs, definiendo nuevos objetos en el diccionario etc.

9.4.6 Los perfiles

Las especificaciones CANOpen se extienden a la definición de perfiles de nodo apropiados para aplicaciones concretas. DS-401 para módulos de entrada-salida, DS-402 para control de motores, perfiles para vehículos de transporte público, ferrocarriles etc. Estas especificaciones están en contínua ampliación y evolución y pueden obtenerse de la organización CiA. Un fabricante interesado en la certificación de sus productos bajo un perfil CANOpen ha de acudir a CiA para la validación, pruebas y certificación final.

10. Ejemplo de desarrollo de un nodo CAN.

10.1 Introducción

En este apartado se esquematizará el proceso de desarrollo de un nodo CAN. Se ha escogido un controlador de reciente aparición en el mercado, por su disponibilidad, no excesiva complejidad y, sobre todo, flexibilidad en el enlace a cualquier CPU debido a su interfaz serie. Se trata del MCP2510 de Microchip. La elección de un dispositivo CAN es siempre un compromiso entre las ventajas y desventajas que ofrece para una aplicación concreta [32], y en este caso ha primado la sencillez de programación e interfaz genérica, que permite su enlace a cualquier procesador, sobre otras consideraciones. Además, Microchip ha anunciado ya la disponibilidad de microcontroladores, una gama de los populares PIC, que incluirán este controlador como periférico integrado.

Puesto que la conexión del controlador a una CPU genérica se ha realizar, entre otras vías, por un bus TTL serie según estándar SPI, se describirá este bus en primer lugar.

En cuanto a la aplicación software del nodo se presentarán algunas de las rutinas básicas de acceso escritas en lenguaje ANSI C. Se pretende tan sólo dar una perspectiva del proceso de desarrollo de un nodo CAN en sus aspectos hardware y software, sin profundizar en detalles y teniendo en cuenta que cualquier aplicación no trivial generará una carga de desarrollo y pruebas muy superior a la que aquí se esboza.

10.2 El bus SPI

El bus SPI es uno de los buses serie, en niveles TTL, más extendidos para la interconexión de periféricos a microcontroladores (junto con el I^2C, desarrollado por Philips). Se trata de un bus síncrono. Un master puede comunicarse con uno o varios esclavos por medio de una línea de datos de salida, otra de datos de entrada y una señal de reloj, además los dispositivos esclavos se habilitan por una señal adicional, para cada uno de ellos (/CS)

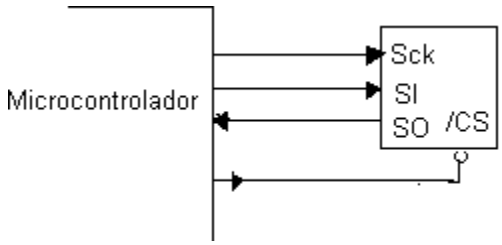


Figura 27 . Conexión SPI

El maestro genera la señal de reloj y controla la línea de salida que se conecta a la entrada SI del periférico. Éste genera sus datos de salida por la línea SO que es leida por una entrada del maestro. El maestro ha de habilitar el acceso al periférico bajando la línea /CS (Figura 27)

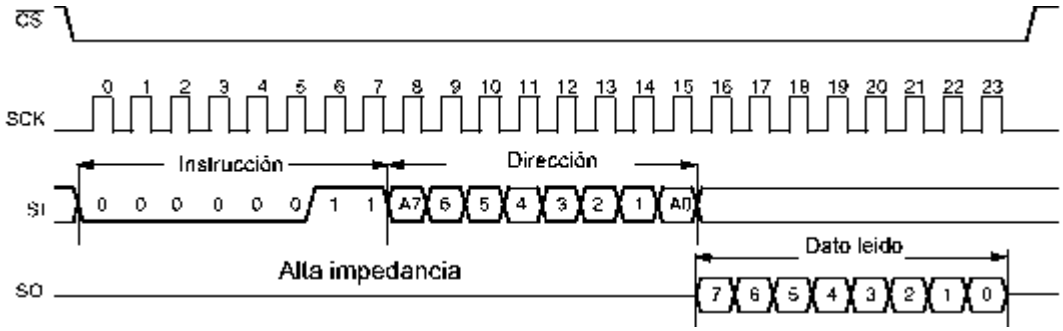


Figura 28. Lectura por bus SPI (tomada de hojas de datos de MCP2510)

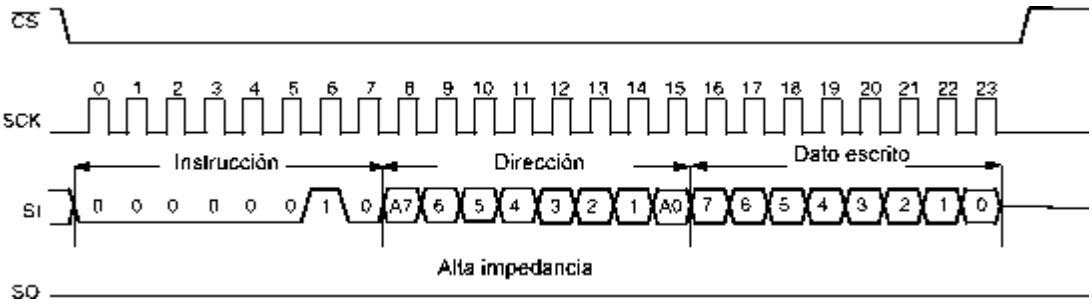


Figura 29. Escritura por bus SPI (tomada de hojas de datos de MCP2510)

En la Figura 28 y Figura 29 se representan, respectivamente, ejemplos de operaciones de lectura y escritura del periférico por el bus SPI. Normalmente ejerce de maestro un microcontrolador. Existen microcontroladores que cuentan con canales SPI con controlador interno. Pero es también posible, y muy habitual, generar las señales de bus SPI por software utilizando patillas de entrada/salida de uso general. Ésta es la solución que se adoptará en la aplicación. De esta forma se puede realizar el enlace con el controlador CAN desde cualquier microcontrolador, sin que sea imprescindible que cuente con controlador SPI. Incluso es posible la utilización de gran parte de software para acceso al controlador desde el puerto paralelo de un PC compatible.

10.3 Controlador MCP2510 de Microchip. Transceptor.

Se describirá en forma resumida el controlador MCP2510, para más detalles, como siempre, la fuente de información más detallada es la hoja de características del componente [33]. Existe también una hoja adicional de errata [34]. No se detallará aquí la arquitectura de registros (ver hoja de características), describiendo únicamente la funcionalidad genérica de éstos. En los listados se indicará con comentarios la utilización de cada uno de los registros. La nomenclatura utilizada será la que utiliza Microchip en la hoja de características.

10.3.1 Transmisión de mensajes

El MCP2510 cuenta con tres buffers de transmisión, cada uno de ellos ocupa 14 bytes de memoria del controlador. Además intervienen en la transmisión alguno de los registros de control y estado generales del controlador. El primer byte TXBiCTRL es un byte de control y estado asociado al buffer de mensaje. Cinco bytes se usan para el identificador, estándar o extendido, y otras opciones de prioridad y arbitraje. Los últimos ocho bytes son los datos a

transmitir.

Entre las opciones configurables en el registro de control se encuentra la prioridad de transmisión del mensaje (no confundir con la prioridad determinada por el identificador), esta prioridad se aplica en la elección del mensaje a transmitir, entre los pendientes de transmisión, de los tres posibles.

La transmisión se puede iniciar desde la CPU por acceso al registro de control (bit de solicitud de transmisión) a través del bus SPI o por actuación sobre una de las líneas /TxiRTS. Si estas líneas no se utilizan para esa función pueden ser utilizadas como entradas genéricas. El fin de transmisión o error se señala en bits del registro de control y estado y además se puede configurar el dispositivo para que provoque una interrupción ante estos eventos. La pérdida de bus por arbitraje se señala en uno de los bits del registro de control y estado. El controlador intentará automáticamente la retransmisión de mensajes que pierden arbitraje o ante error. Un mensaje que no iniciado transmisión puede ser abortado fijando bits en el registro de control.

10.3.2 Recepción de mensajes

Los registros básicos de recepción son dos RBX0 y RBX1, cada uno de ellos tiene asociados registros para el establecimiento de máscaras y filtros, los mensajes se reciben inicialmente en un registro general MAB del que se mueven a RBX0 o RBX1 si son aceptados (Figura 30). La recepción de un mensaje activa un bit en uno de los registros de estado. Además el controlador se puede configurar para que ante recepción se active la patilla de interrupción /INT. RBX0 es el registro de recepción de mayor prioridad, RBX1 de menos prioridad. Se puede configurar el controlador para que RBX1 actúe como buffer secundario de RBX0, de modo que ante recepción de un mensaje en RBX0 antes de que sea recogido el anterior, en lugar de generar un error de sobreescritura se mueve el nuevo mensaje a RBX1 independientemente de los criterios de recepción programados en los filtros para este registro. Es posible programar el controlador, también, para que acepte mensajes de CAN estándar, CAN extendido o ambos.

Las salidas /RX0BF y /RX1BF pueden ser configuradas para que señalen la recepción de un mensaje, o utilizadas simplemente como salidas genéricas.

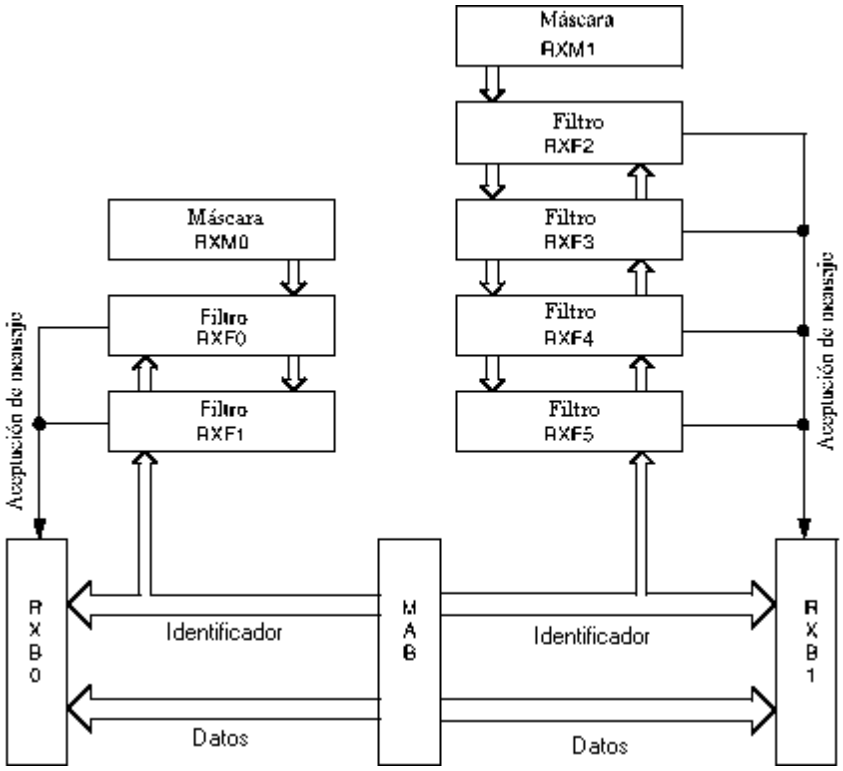


Figura 30. Registros de recepción de MCP2510 (de hoja de datos de MCP2510)

10.3.3 Filtrado de mensajes

Para cada uno de los dos registros de recepción existe un registro de máscara y varios registros de filtro, dos para RBX0 y cuatro para RBX1. Un mensaje es aceptado por un registro cuando los bits de su identificador señalados en la máscara coinciden con los bits de alguno de los filtros. En registros de estado del controlador se registrará que filtro permitió la aceptación del mensaje. Supóngase que una máscara se ha programado como '00001110000' y uno de los filtros asociados como '00000100000', entonces cualquier mensaje de la forma 'XXXX010XXXX' será aceptado, siendo indeferente el valor de los bits X del identificador.

10.3.4 Configuración de temporización y sincronización de bit. Errores

En tres registros de configuración, CNF1 a CNF3, es posible configurar los parámetros de temporización y sincronización de acuerdo a lo expuesto en 7.3.3. Es posible programar muestreo múltiple de bit.

Los errores se señalan en bits del registro EFLG, además es posible el acceso a los contadores de error (véase 8.2) en los registros TEC y REC.

10.3.5 Interrupciones

El controlador puede generar una señal de interrupción en la patilla /INT ante 8 eventos potenciales, cada uno de ellos puede habilitarse independientemente, son:

- Interrupciones por fin de transmisión (3). Señala que un buffer de transmisión ha quedado vacío por transmisión del mensaje que se había escrito en él.
- Interrupción por recepción (2). Señala que se ha recibido un mensaje en uno de los registros RBXi
- Interrupción por error en mensaje.
- Interrupción por salida de modo "sleep" ante actividad en el bus.
- Interrupción de error. Indica diversas condiciones ante cambios singulares en el control de errores (REC, TEC)

10.3.6 Modos de funcionamiento

El 2510 puede encontrarse en cinco modos de funcionamiento diferentes, seleccionables en bits de registros de configuración. Estos modos son:

- Modo de configuración. Estado en el que se permiten operaciones de configuración que no son posibles en modo normal.
- Modo normal. Es el estado normal de funcionamiento. El único en que es posible la transmisión.
- Modo sleep. Modo de bajo consumo en que el controlado queda desactivado, del que puede salir por control desde la CPU o actividad en el bus.
- Modo de sólo escucha.
- Modo de bucle cerrado. Para pruebas.

10.4 Hardware de base

El esquema de la interfaz física CAN se presenta en la Figura 31. Como se ha indicado se utiliza el controlar MCP2510 y el transceptor 82C250 de Philips, uno de los más extendidos. Se incluyen los componentes adicionales necesarios de alimentación (se supone una fuente de 12 V. con filtrado pero sin necesidad de estabilización previa, ya que se ha incluido un circuito estabilizador 7805), oscilador etc. El circuito se supone enlazado a cualquier CPU por medio de patillas de entrada salida genéricas, algunas de ellas con capacidad para generar interrupción por flanco de bajada, para las siguientes señales:

- **Rst** :. Salida de CPU que permite dar reinicializar el controlador
- **P1.0**: Salida de CPU que permite controlador la señal /CS del bus SPI
- **P1.1**: Entrada de CPU que corresponderá a la recepción de datos de bus SPI
- **P1.2**: Salida de CPU que corresponderá a la transmisión de datos de bus SPI
- **P1.3**: Salida de CPU que corresponderá a la generación de reloj de bus SPI, la CPU actuará

como dispositivo maestro.

- **P2.0** a **P2.2**: Salidas de CPU que se conectan a las señales /TXiRTS del controlador.

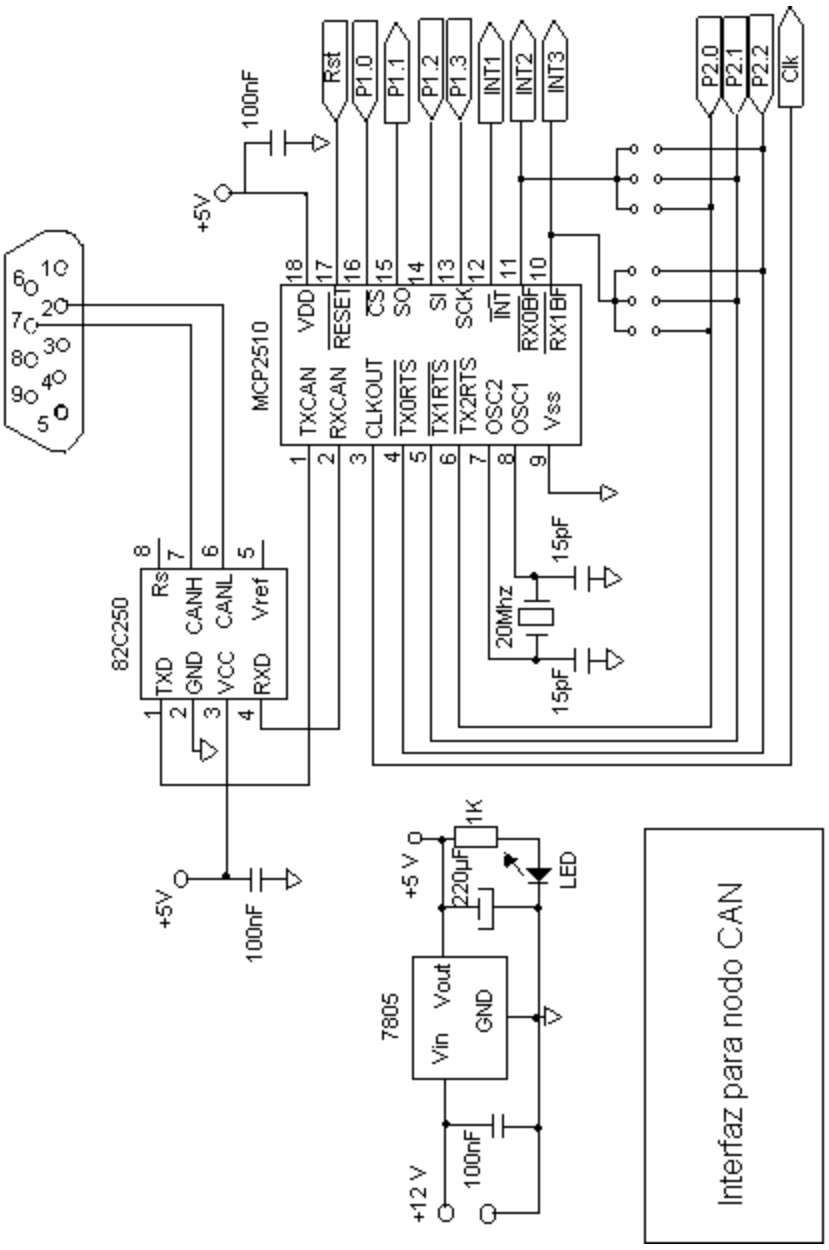
- **INT1** a **INT2**: Señales de entrada de CPU, capaces de provocar interrupción que se conectarán, respectivamente, a /INT, /RX0BF y /RX1BF del controlador. En algunos microcontroladores puede no contarse con varias entradas de interrupción, en este caso será necesario un compromiso, limitando la atención por interrupción, enlazando las señales en un línea OR de colector abierto etc.

- **Clk**: es una salida de reloj emitida por el controlador. Puede ser utilizada como reloj del microcontrolador, ahorrándose un oscilador o cristal adicional.

Las señales P1.0 a P1.4 se suponen asociadas a un puerto de entrada-salida del controlador y las señales P2.0 a P2.2 a otro de esos puertos.

Es interesante dejar prevista en la placa la posibilidad de enlace directo entre patillas /RXiBF y /TXiRTX (basta incluir puentes configurables) ya que en algunas aplicaciones puede automatizarse la respuesta a una recepción indicada en la patilla /RxiBF con la activiación de una /TXiRTX que inicie una transmisión de respuesta.

Figura 31 . Interfaz de nodo CAN



10.5 Software de base

Se describirá de forma resumida la capa de software de base que establezca una capa de abastracción de hardware que ha de servir de soporte a la capa de aplicación. Se presentan listados en lenguaje C de fragmentos seleccionados.

10.5.1 Enlace con el controlador

Para el enlace con el controlador se utilizan patillas de entrada-salida. En un microcontrolador genérico estas patillas se controlan normalmente desde registros de configuración y actuación sobre puertos de entrada-salida que agrupan las patillas. Es habitual que un puerto controle 8 patillas y que se controle por medio de un registro de configuración, en el que cada bit determina si la patilla se usa para entrada o salida, y un registro de dato en el que se puede leer el estado de la patilla, en el caso de una entrada, y escribir el valor de salida, en el caso de una salida. Se supondrán que los puertos de entrada salida utilizados para la interconexión con el controlador CAN están controlados en base a las definiciones y macros incluidas en el listado 1. Partiendo de esta definición es posible el desarrollo de rutinas de lectura y escritura por bus SPI que utilizan patillas de entrada-salida genérica, las señales se generan conmutando estados con inclusión de un tiempo de retardo entre conmutaciones, ese retardo puede ser muy reducido, dependiendo del microcontrolador se basará en un bucle de cuenta o una simple sucesión de instrucciones NOP (no operación). En Listado 2 se presenta una rutina de escritura por bus SPI y en el listado 3 una de lectura. El resto del enlace con el controlador consiste en los potenciales controladores de interrupción para las interrupciones de INT1 ,INT2 e INT3 y el posible uso de las patillas de salida que se conectan a las patillas /TXiRTS del controlador.

10.5.2 Instrucciones de acceso al controlador

Se actúa sobre el controlador a través del bus SPI por medio de un conjunto de instrucciones básicas (Tabla 8). Cada una de estas instrucciones ha de comenzar con la escritura, a través del SPI, del código de instrucción, luego puede ser necesaria la escritura de byte de dirección (en instrucciones READ, WRITE, Modificación de bit), y posteriormente se escriben o leen datos y parámetros.

Instrucción	Código	Descripción
RESET	11000000	Reinicializa el controlador, pasa a modo configuración
READ	00000011	Lectura de registros
WRITE	00000010	Escritura de registros
RTS	10000nnn	Fija bit de inicio de transmisión en byte de control de registro de transmisión
Lectura de estado	10100000	Lectura de bits de estado más importantes
Modificación de bit	00000101	Modificación de bits en registro

Tabla 8. Instrucciones básicas del MCP2510

Las instrucciones de lectura y escritura tienen la particularidad de que, una vez iniciada la operación en una dirección o registro dado, si se contínua generando señal de reloj en el bus SPI se seguirán leyendo, o escribiendo, datos de registros sucesivos del controlador. De esta forma es posible tener configuraciones almacenadas como tablas en el microcontrolador y escribir estas configuraciones en el controlador por medio de una operación continuada de escritura. Del mismo modo es posible una lectura continuada de conjuntos de registros del controlador. Los 128 registros de 1 byte cada uno del controlador, están organizados de forma que se pueda realizar una lectura secuencial óptima de registros relacionados. Los registros de estado y control generales son accesibles en direcciones múltiples, para optimizar este modo de acceso secuencial.

La instrucción de modificación de bits consiste en la escritura, tras la instrucción, de un byte de dirección que determina el registro a modificar, un byte de máscara que aisla los bits concretos a modificar y un byte de dato con los nuevos valores de los bits.

En el listado 4 se presentan las cabeceras de rutinas apropiadas para escritura de registros a partir de una dirección dada, lectura de registros a partir de una dirección dada y modificación de bits, además de las más sencillas para escritura de un solo registro, reset y activación de transmisión. Estas rutinas son fácilmente implementables basándose en las básicas de escritura y lectura de byte. El bus SPI de interconexión se puede hacer trabajar con velocidad máxima de reloj de 5 Mhz. Con acceso de velocidad cercana al máximo la lectura o escritura de un mensaje CAN + registros de estado y control ocupará un tiempo del orden de 10-20 µs.

Además de estas rutinas se han de desarrollar rutinas de cambio de modo, inicialización de velocidad etc. (listado 5) y otras de más alto nivel para montar mensajes a transmitir y escribirlos en registros de transmisión y de recogida de mensajes recibidos y aislamiento de información desde éstos. De cara a procesar los mensajes en colas en las capas intermedias de protocolo es conveniente crear estructuras de datos apropiadas (listado 6). En la definición de tipo para un mensaje CAN se ha incluido un campo que permite fijar una estampa de tiempo con el momento de recepción a partir de un tick de reloj del microcontrolador.

Por otra parte, la gestión de la comunicación puede basarse en, por lo menos, dos rutinas de interrupción del microcontrolador:

1) Interrupción por recepción de mensaje. En esta interrupción puede hacerse un proceso inmediato, si es necesario, como por ejemplo una actuación relacionada con la aplicación concreta, o una respuesta inmediata al mensaje recibido. Si no es imprescindible la actuación inmediata, el proceso lógico consistiría en el almacenamiento del mensaje recibido en una cola de mensajes.

2) Interrupción temporizada (basada en alguno de los temporizadores normalmente disponibles en un microcontrolador). En esta interrupción se ejecutará la máquina principal del protocolo gestionando: a) la transmisión periódica de mensajes del nodo, b) recogida de mensajes recibidos desde cola de recepción, tratada en el apartado anterior, y reacción asociada (respuesta, actualización de variables de aplicación, cambios de estado etc.)

Además pueden existir interrupciones o gestiones de tiempo real del nodo que den lugar a la transmisión inmediata de un mensaje.

Se ha de tener en cuenta que el acceso por bus SPI al controlador ha de ser una operación atómica, protegida, para operación de lectura o escritura completa. Por lo tanto, las rutinas del listado 5 han de deshabilitar interrupciones al comienzo y restaurar a la salida. Muchos compiladores C para microcontroladores facilitan esta operación por medio de atributos de funciones como “monitor” o “protected”.

10.6 Ejemplo de Aplicación

Como ejemplo concreto de aplicación, este controlador puede ser utilizado en una sistema de supervisión de un convertidor de energía basado en módulos convertidores paralelables. En estos sistemas una serie de convertidores de potencia (DC/DC o DC/AC) se hacen funcionar de forma sincronizada (Figura 32). Simplificando mucho la descripcion, cada uno de los convertidores ha de atender un mensaje de órdenes que produce una unidad de gestión central, un mensaje de medidas que es producido por una unidad de medida de variables eléctricas globales etc. Además ha de atender en baja prioridad un flujo contínuo de actualización de parámetros de configuración que proviene de la unidad central y está orientado a mantener la configuración homogénea en todos los convertidores (los convertidores pueden ser sustituidos en caliente en cualquier momento). Por otra parte el convertidor emite a intervalos periódicos información sobre su propio estado. Pueden incluirse nodos adicionales en configuraciones especiales: gestor de baterías en aplicaciones de carga de baterías, controlador de by-pass estático en aplicaciones corriente alterna,...

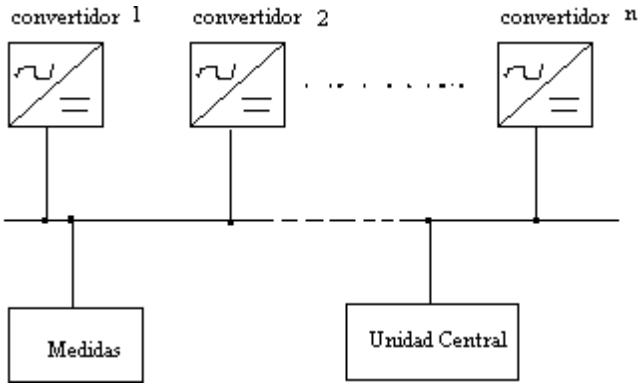


Figura 32 . Sistema de supervisión de convertidores de energía

Listado 1. Definiciones para acceso a patillas del MCP2510

```
typedef unsigned char byte;

//puerto P1, se supone registro de control en dirección 0xffb0
//y registro de datos en 0xffb1

#define R_CFG_P1       ((byte *)( 0xffb0 ))
#define R_DAT_P1      ((byte *)( 0xffb1 ))

//puerto P1, se supone registro de control en dirección 0xffc0
//y registro de datos en 0xffc1

#define R_CFG_P2       ((byte *)( 0xffc0 ))
#define R_DAT_P2      ((byte *)( 0xffc1 ))

//macro para lectura del estado de un puerto
#define IO_Lee_Entradas(puerto) (*(puerto))

//macro que activa un bit
#define IO_Bit_a_1 (puerto,mascara) ((*(puerto))|=(mascara)))

//macro que desactiva un bit
#define IO_Bit_a_0(puerto,mascara)(*(puerto))&=(~(mascara)))


//definiciones de puertos y máscaras para acceso SPI
#define P_CFG_CS R_CFG_P1
#define P_CS  R_DAT_P1
#define M_CS  0x01   //P1.0 es /CS

#define P_CFG_SI R_CFG_P1
#define P_SI  R_DAT_P1
#define M_SI  0x02   //P1.1 es SI (de CPU)

#define P_CFG_SO R_CFG_P1
#define P_SO  R_DAT_P1
#define M_SO  0x04   //P1.2 es SO (de CPU)

#define P_CFG_SCK R_CFG_P1
#define P_SCK R_DAT_P1
#define M_SCK 0x08   //P1.3 es SCK
```

Listado 2. Rutina de escritura de byte por bus SPI

```
/*
FUNCION: void SPI_Escribe_Byte(byte b)
OBJETO: Transmite un byte por bus serie SPI, con cronograma ( cada _ un
t_retardo):

           b0  b1  b2  b3  b4  b5  b6  b7
      DA <==><==><==><==><==><==><==><==>
      SO __--__--__--__--__--__--__--__--

ENTRADA: b: byte a escribir

SALIDA:  void

NOTAS:
    Se supone /CS habilitado
    Deja CLK alto tras ultimo bit */

static void SPI_Escribe_Byte(byte b)
{

  byte i;

  // transmite el byte

  Io_Bit_a_1 (P_CFG_SO,M_SO); //bit SO como salida
  for(i=0;i<8;i++) {
      IO_Bit_a_0(P_SCK,M_SCK); //baja SCK
      if(b&0x01)               //Pone en SO bit
        IO_Bit_a_1(P_SO,M_SO)
      else
        IO_Bit_a_0(P_SO,M_SO)
      retardo();               //retardo
      retardo();
      IO_Bit_a_1(P_SCK,M_SCK); //sube SCK
      retardo();               //retardo
      retardo();
      b>>=1;
  }
  return;
}
```

Listado 3. Rutina de lectura de byte por bus SPI

```
/*
FUNCION: byte SPI_Lee_Byte(void)
OBJETO:  Lee  un  byte  desde  bus  serie  SPI  con  cronograma  (  cada  _  un
t_retardo):
           b0  b1  b2  b3  b4  b5  b6  b7
      DA <==><==><==><==><==><==><==><==>
      CK __--__--__--__--__--__--__--__--
ENTRADA: void
SALIDA:  byte leido

NOTAS:
    Se supone CS habilitado
    Deja SCLK alto y SDI en lectura
*/

static byte SPI_Lee_Byte(void)
{
  byte i,b;

  IO_Bit_a_0(P_CFG_SI,M_SI);  // Pasa pin a lectura
  b=0;
  for(i=0;i<8;i++) {
      b>>=1;
      IO_Bit_a_0(P_SCK,M_SCK);   //baja SCK
      retardo();              //retardo
      retardo();
      if(IO_Lee_Entradas(P_DI)& M_DI) //lee bit
        b|=0x80;
      else
        b&=0x7F;
      IO_Bit_a_1(P_SCK,M_SCK);   //sube SCK
      retardo();              //retardo
      retardo();
  }
  return b;
}
```

Listado 4. Rutinas de acceso a registros

```
/*
FUNCION:void MCP2510_Escribe_Registro(byte dir,byte valor)
OBJETO: escribe en un registro del MCP2510
ENTRADA:  dir: direcci¢n de registro
        valor: valor a escribir en registro
SALIDA: void
*/
void MCP2510_Escribe_Registro(byte dir,byte valor);

/*
FUNCION:byte MCP2510_Lee_Registro(byte dir)
OBJETO: lee un registro del MCP2510
ENTRADA:  dir: direcci¢n de registro
SALIDA: byte leido de registro
*/
byte MCP2510_Lee_Registro(byte dir);

/*
FUNCION: void MCP2510_Escribe_Registros(byte dir,byte *datos,int n_bytes)
OBJETO: escribe una serie de datos en registros sucesivos del MCP2510
ENTRADA:  dir: direcci¢n inicial de escritura
        datos: puntero a buffer con datos a escribir
        n_bytes: n£mero de bytes de datos a escribir (<=16)
SALIDA: void
NOTAS:
    La rutina no admite n_bytes mayor a 16, si n_bytes>16 la operaci¢n
    de escritura no se realiza
*/
void MCP2510_Escribe_Registros(byte dir,byte *datos,int n_bytes);

/*
FUNCION:  MCP2510_Lee_Registros(byte dir, byte *datos,int n_bytes)
OBJETO:  lee una serie de datos desde registros sucesivos del MCP2510
ENTRADA:  dir: direcci¢n inicial de lectura
        datos: puntero a buffer para datos leidos
        n_bytes: n£mero de bytes a leer (<=16)
SALIDA:  void
NOTAS:  La rutina no admite n_bytes mayor que 16, si n_bytes>16 la operaci¢n
      de lectura no se realiza
*/
void MCP2510_Lee_Registros(byte dir,byte *datos,int n_bytes);
/*
FUNCION: void MCP2510_Modifica_Bits(byte dir,byte mascara,byte valor);
OBJETO : Modifica bits en registro de direcci¢n dir con mascara y
       datos dados
ENTRADA: dir : direcci¢n de registro a modificar
       mascara: mascara que aisla bits a modificar
       valor: byte con valores de los bits
SALIDA:  void
NOTA:  La rutina no comprueba si el registro es v lido como registro al que
se pueda aplicar  la operaci¢n de modificaci¢n de bits. Es responsabilidad
del usuario.
*/

void MCP2510_Modifica_Bits(byte dir,byte mascara,byte valor);
```

Listado 5. Rutinas de inicialización

```
void MCP2510_Inicia(void)
{
  MCP2510_Reset();
  //asegura interrupciones deshabilitadas
  MCP2510_Escribe_Registro(CANINTE, 0x00);

MCP2510_Modifica_Bits(CANCTRL,   0x03,   0x00);      //   El   divisor   de
frecuencia a 1
/* Fija par metros de temporizaci¢n y sincronizaci¢n de bit
     para comunicaci¢n a 125 Kbps
     f_osc = 20MHz
     BRP        = 4
     Sync_Seg   = 1TQ
     Prop_Seg   = 2TQ
     Phase_Seg1 = 7TQ
     Phase_Seg2 = 6TQ
     TQ = 2 * (1/f_osc) * (BRP+1)= (2*5/20e6)=0,5e-6
     Nbr = 1/((Sync_Seg+Prop_Seg+Phase_Seg1+Phase_Seg2) * TQ)
     =1/16/0,5e-6=125000 bps
 */

MCP2510_Escribe_Registro(CNF1,0x04);  //SJW==1 BRP==4

MCP2510_Escribe_Registro(CNF2,0xB1); // 1 0 110 001 Phase_Seg2 dado por CNF3
                              // 1 Muestra
                              // Phase_Seg1=7
                              // Prop_Seg=2

MCP2510_Escribe_Registro(CNF3,0x05); //Phase_Seg2=6

MCP2510_Configura_Filtros();

MCP2510_Pon_Modo_Normal();
}


void MCP2510_Pon_Modo_Configuracion()
{
    MCP2510_Modifica_Bits(0x0F, 0xE0, 0x80);
}


void MCP2510_Pon_Modo_Normal()
{
    /* Segun la hoja de errata del MCP2510 se han de mantener reintentos de
paso a modo normal con retorno a modo configuraci¢n hasta comprobar paso a
dicho modo */
    do
    {
      MCP2510_Modifica_Bits(0x0F, 0xE0, 0x80);    //Modo Config.
      MCP2510_Modifica_Bits(0x0F, 0xE0, 0x00);    //Modo Normal
    } while (MCP2510_Lee_Registro(0x0E) & 0xE0);
}
```

Listado 6. Cola de Mensajes

```
//tipo para mensajes CAN

typedef struct {
      uint ID;  //identificador , CAN Estándar 11 bits menos sig.
      byte Long;     //longitud
      byte Datos[8]; //datos
      uint t;        //tiempo de recepción
} t_Mensaje_CAN;


//cola de mensajes de entrada

#define N_MENSAJES_MAX  6

static t_Mensaje_CAN cola_mensajes_entrada[N_MENSAJES_MAX];
static int ie_cm_e, is_cm_e;
static int n_cm_e;

//inicia cola de mensajes de entrada

void inicia_cola_mensajes_entrada(void)
{
   ie_cm_e=is_cm_e=0;
   n_cm_e=0;
}


//lee un mensaje de cola de mensajes de entrada

int lee_cola_mensajes_entrada(t_mensaje_CAN *mensaje)
{
 if(n_cm_e>0) {
      *mensaje=cola_mensajes_entrada[is_cm_e];
      is_cm_e++;
      if(is_cm_e>=N_MENSAJES_MAX)
            is_cm_e=0;
      n_cm_e--;
      return 1;
   }
   else {
       write_exr(exr);
       return 0;
   }
}

//tipo para mensajes CAN

typedef struct {
      uint ID;  //identificador , CAN Estándar 11 bits menos sig.
      byte Long;     //longitud
      byte Datos[8]; //datos
      uint t;        //tiempo de recepción
} t_Mensaje_CAN;
```

11. Sistemas distribuidos de control en tiempo real y CAN.

En este apartado se trata de establecer una mínima perspectiva de uno de los campos de aplicación del bus CAN más interesantes para el autor: los sistemas de control de tiempo real distribuido. En éste campo,CAN aporta características especialmente atractivas, sobre todo cuando se trata de incorporarlas en productos y sistemas industrializables y de bajo costo.

Cualquier solución de tiempo real distribuido sobre buses clásicos (realmente para ser distribuido y de bajo coste, se trataría de un sistema basado en microcontroladores con controlador asíncrono de comunicaciones y transceptores RS-485) exigía mutitud de “trucos” desestructurados y poco generalizables, las únicas reglas estructuradas de organización de la información de tiempo real consistían en variantes de los métodos de paso de testigo más o menos ingeniosas.

Se presenta también un método de sincronización de relojes en nodos relacionados por un bucle de control.

11.1 Sistemas de tiempo real distribuido.

Refiriéndose a sistemas digitales un sistema de tiempo real es aquél, en el que no sólo es relevante el resultado del proceso que ejecuta, sino también el instante o instantes temporales en que produce los resultados. De forma más precisa se puede definir un sistema de tiempo real como aquél al que se pueden asignar a algunas de sus salidas y eventos internos funciones de peso que determinan “la calidad” del resultado en relación con el instante temporal en que se produce. En la Figura 33 se representan algunas funciones genéricas de este tipo. En el primer caso el resultado sólo es positivo si se produce entre dos instantes precisos de tiempo, en el segundo caso hay una ponderación suave en función del tiempo en que se produzca el resultado, en el tercer caso se puede producir incluso una ponderación negativa. El primer caso determina un sistema de tiempo real más o menos estricto, el segundo es típido de sistemas de tiempo real suave, entre los que se puede incluir cualquier programa de computador (en un programa de cálculo de nómina no sería aceptable un retardo en la salida de más de un mes), el último caso determina un sistema de tiempo real con fuertes implicaciones de seguridad, ya que una salida en instante no apropiado puede producir un beneficio negativo, es decir un daño más o menos grave (considérese el instante de disparo del airbag de un automóvil).

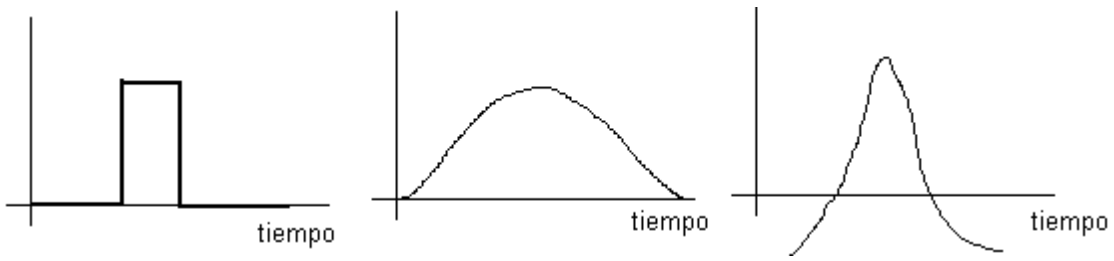


Figura 33 . Ponderación de salida de un proceso

El estudio de los sistemas de tiempo real se puede dividir en dos niveles, por una parte el estudio de la corrección lógica de la ejecución concurrente de tareas diversas: sistemas concurrentes (véase, por ejemplo [35]), por otra el estudio de la corrección en cuanto a cumplimiento de los requisitos temporales, en este último campo se han desarrollado una en los últimos años una serie de técnicas de aplicación sistemática conocidas como técnicas RMA, "rate monotonic analysis" (una referencia orientada a la aplicación práctica de estas técnicas es [36])

Cuando el sistema de tiempo real consta de varios procesadores enlazados por una red de comunicaciones, se tiene un nivel adicional de complejidad, es el área de los sistemas distribuidos de control en tiempo real. En este apartado se pretende realizar una perspectiva de la problemática asociada al desarrollo de sistemas distribuidos de control en tiempo real utilizando CAN como red de comunicaciones.

Antes de CAN, los protocolos orientados a aplicaciones de tiempo real, en los que se requiere una estimación determinista de los tiempos y retardos, estaban basados en la asignación de "slots" de tiempo a cada nodo (TDMA o paso de testigo). El comportamiento, en cuanto a características de tiempo real, de estos métodos no es especialmente bueno. Un nodo con datos urgentes a enviar ha de esperar su turno, "slot" de tiempo o testigo, antes de poder transmitir. El resultado es el fenómeno conocido como inversión de prioridades, un mensaje prioritario ha de esperar la transmisión de mensajes de menor prioridad. En CAN la asignación de prioridades a los mensajes y el arbitraje no destructivo permite un mejor comportamiento en aplicaciones de tiempo real distribuido.

11.2 CAN en aplicaciones de tiempo real

CAN ha recibido la atención de diversos grupos de investigación como bus serie apropiado para aplicaciones de tiempo real distribuido ([37],[38],[39]).

11.2.1 Análisis básico de tiempos de respuesta en CAN

En un mensaje CAN Estándar, sin considerar los bits de relleno, hay 47 bits de sobrecarga y hasta 64 bits de datos (8.1.1). Se han de determinar los bits de relleno para el caso peor, se obtiene

$nb = \left\lfloor \frac{34 + 8n + 1}{4} \right\rfloor$, siendo $\lfloor x \rfloor$ la función que da el entero menor más cercano al racional x, ($\left\lfloor \frac{5}{4} \right\rfloor = 1$, por ejemplo)

Luego el número máximo de bits en un mensaje es:

$$Nbmax = 8n + 47 + \left\lfloor \frac{34 + 8n - 1}{4} \right\rfloor$$

Para el mensaje más largo, 8 bytes de datos, la ecuación anterior da un valor de 135. El tiempo de transmisión de este mensaje será **135.tbit**, siendo tbit el tiempo de transmisión de un bit. A

la máxima velocidad de bus CAN (1Mbps), el mensaje más largo se transmite en **135 µs**.

Cuando se tiene en cuenta la concurrencia entre mensajes, el análisis de tiempos de respuesta en CAN se puede generalizar para cualquier prioridad y no difiere mucho del aplicable al análisis de sistemas monoprocesador multitarea. Se concluye también que la mejor estrategia de planificación es la asignación "deadline monotonic" de prioridades, es decir, mayor prioridad a mensajes con información que requiera tiempos de respuesta más cortos ([37][36]). Para el ejemplo de aplicación que se analizará a continuación se hace una simplificación. Se restringirá el análisis a un solo mensaje de tiempo real, que tendrá máxima prioridad, los demás mensajes se considerarán sin requisitos de tiempo real.

Un mensaje puede ver retardada su transmisión, independientemente de su prioridad, por cualquier otro mensaje que haya iniciado transmisión. Por lo tanto el tiempo de retardo máximo entre orden de transmisión a un controlador de un nodo dado y fin de recepción en otro nodo para un mensaje de máxima prioridad será **2.135.tbit=270.tbit**. Para la máxima velocidad de bus se tendrá un retardo de 270 µs.

11.2.2 La influencia del controlador

En el análisis anterior no se ha tenido en cuenta el hecho de que dentro del propio controlador se pueden tener varios mensajes pendientes de transmisión. Si el controlador transmite basándose en el orden de llegada de mensajes (FIFO, primero en llegar, primero en salir), un mensaje de alta prioridad puede ser detenido, a su vez, por mensajes de baja prioridad en la cola interna del controlador. Lo ideal es que el controlador transmita los mensajes ya almacenados en sus buffers de transmisión de acuerdo a una indicación de prioridad adicional. Éste es el caso de MCP2510 en el que es posible fijar la prioridad del mensaje en relación con la "salida del controlador". En aplicaciones de tiempo real otra característica importante es la posibilidad de abortar la transmisión de cualquier mensaje que haya pasado al controlador, de cara a sustituirlos por mensajes de mayor prioridad. También ésta posibilidad existe en el MCP2510.

11.3 Bucles distribuidos de control sobre CAN

Considerése un ejemplo sencillo,en una red CAN un nodo A actúa como captador y otro B como actuador de un bucle de regulación, el regulador, digital, se ejecuta en el nodo B, el valor de medida de la variable a regular lo transmite A. En la Figura 34 se presenta el ejemplo de regulación de presión en una tubería actuando sobre la velocidad de una bomba.

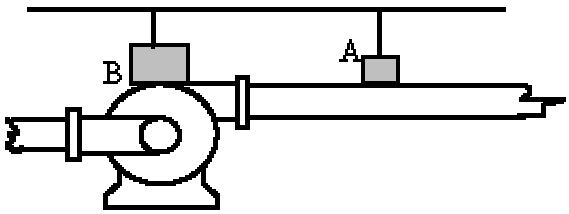


Figura 34. Ejemplo de bucle de regulación

El nodo captador muestrea la variable, en el ejemplo la presión, con periodo ***Tm.*** El nodo correspondiente al variador de velocidad de la bomba ha de funcionar en sincronismo con el

muestreo de las variables a regular. Tras la captación del valor de presión concreto el primer nodo transmitirá un mensaje en el que se incluye el valor de presión, Si ***Tm*** es el periodo de muestreo, la variable ha de estar disponible en el nodo B tras un tiempo δTm, ($0 <= \delta <= 1$) la ejecución de las instrucciones requeridas para recoger esa variable y aplicar el algoritmo de control supondrán un tiempo γTm adicional. ($0 <= \gamma <= 1$). El valor $\delta+\gamma$ determina el retardo total a la actuación y ha de tenerse en cuenta en el análisis y diseño del algoritmo de control ([40]).

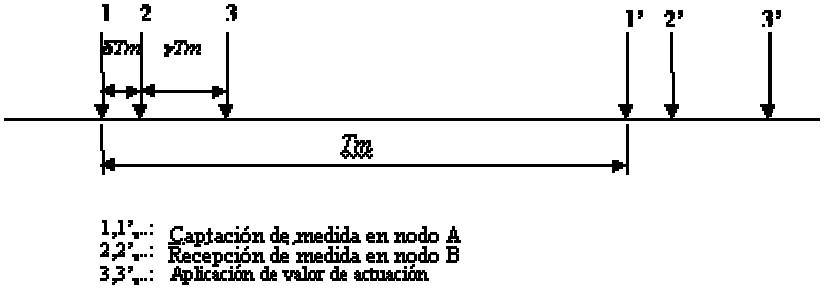


Figura 35. Tiempos en bucle de regulación digital

En CAN una solución al problema de control presentado pasa por el establecimiento de un muestreo periódico de la variable a regular, presión en este caso, en el nodo A. Esa muestra se transmite y es recibida por el nodo B que, tras recoger la muestra y aplicar el algoritmo de regulación, actúa sobre la velocidad de la bomba.

Para que el regulador se comporte de acuerdo a las hipótesis de cálculo, se han de mantener tanto el periodo de muestreo ***Tm,*** cómo el retardo a la actuación, ($\delta+\gamma$)***.Tm***, dentro de márgenes estrechos respecto a los valores previstos.

El periodo de muestreo puede basarse en un temporizador del microcontrolador del nodo A, el retardo δ***.Tm*** es el estudiado en el apartado anterior, se puede acotar para el caso peor, el segundo componente del retardo γ***.Tm*** puede determinarse basándose en características del nodo B (retardos en acceso a controlador, tiempo de cálculo del algoritmo de regulación). δ***Tm*** puede variar entre el tiempo de transmisión del mensaje que contiene la variable muestreada y la suma de ese tiempo al tiempo de transmisión del mensaje más largo de los de prioridad inferior. Para mantener el retardo a la actuación constante lo correcto es que en el nodo B se aplique la actuación basándose en una medida de tiempo desde el momento de la captación. Captación que se ha realizado en otro nodo. Ambos nodos han de sincronizarse de forma que para el nodo B sea conocido con la mayor precisión posible el instante ***t1*** en que se realiza el muestreo, de cara a poder calcular el instante ***t3*** de aplicación de la actuación: ***t3=t1+*** ($\delta+\gamma$)***.Tm***

Un método de sincronización podría basarse en un mensaje de sincronismo enviado desde el propio nodo B y que marque el instante de muestreo al A, finalmente el nodo A enviaría la muestra en otro mensaje. El mensaje de sincronismo debería tener alta prioridad, mientras que de envío de muestra no requiere máxima prioridad. El problema de este método es la incertidumbre en el retardo desde el envío de mensaje de sincronismo desde B y recepción en A.

Sí A transmite la medida de forma periódica basándose en un temporizador propio y acompaña dicha medida de una indicación del instante en que se ha tomado la muestra ("time stamp"), en la hipótesis de que el nodo B tenga un temporizador sincronizado con el de A, B podría calcular el instante ***t3***. Naturalmente, y a pesar de usar osciladores de precisión, existirá una deriva entre los temporizadores en A y B.

Este problema conduce a la necesidad de mantener relojes sincronizados en los nodos A y B. La sincronización de relojes en nodos de un bus CAN es un requisito normalmente asociado a sistemas de tiempo real distribuido, una solución se presenta en [41]. El método que se presenta a continuación se basa en dicha referencia pero con detalles de implementación particulares que ofrecen una alta precisión en la sincronización de relojes.

El mensaje que A utiliza para enviar la medida a B será un mensaje en el que además de la medida se incluye un valor de cuenta de temporizador que indica el instante ***tB*** en que B recibió el mensaje previo, ese instante coincidirá con el instante en que lo recibió A, ***tA***, en la hipótesis de retardo muy reducido de línea entre A y B (qué en un bus CAN es realista, el método de arbitraje de CAN se basa en este principio). Entonces B puede comparar el valor recibido desde A con su valor y aplicar una corrección a su contador proporcional a la diferencia (se trata de ejecutar un algoritmo de enganche de fase (PLL) digital que permita un enganche rápido, suave y sin oscilaciones). El problema de implementación práctica es la captura de los tiempos de recepción en ambos nodos, idealmente debería seguirse un método que conduzca a que el retardo entre recepción y captura sea igual para ambos nodos.

Si la captura se basa en la simple interrupción de recepción el retardo puede variar ya que, aún para controladores y CPUs iguales, en uno de los nodos la interrupción puede verse retardada por una de mayor prioridad o por un intervalo de inhibición de interrupción por acceso a recurso compartido.

En muchos microcontroladores se dispone de temporizadores con operación de captura, ante una señal externa (cambio de nivel en una patilla de entrada), el valor de cuenta de un temporizador se almacena en uno de los registros de captura (Figura 36).

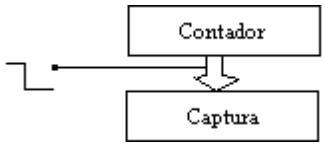


Figura 36 . Captura

El método se basa en:

a) Disponibilidad de temporizadores con captura en los microcontroladores de los nodos

b) Capacidad del controlador CAN para provocar cambio de nivel en una patilla de salida ante recepción de un mensaje concreto. Por ejemplo, el MCP2510 puede configurarse para que una de las patillas /RXnBF actúe de este modo.

Se ha de configurar los controladores, máscaras y filtro, para que ambos atiendan la recepción del mensaje de nodo A que incluye la muestra de la variable medida y el instante tA, que

ahora será el valor de captura ante recepción del mensaje con la muestra previa, el nodo B habrá memorizado el valor de esa captura previa y puede aplicar la corrección en base a la diferencia. Obsérvese que la atención a la interrupción deja de ser crítica, y tampoco lo es el retardo a transmisión, excepto en el sentido de que estén acotados superiormente para el caso peor de cara a fijar el retardo a actuación $(\delta+\gamma).Tm$. En la interrupción de recepción se guarda el valor de captura en cada uno de los nodos, en A para que pueda ser enviado en el próximo mensaje, en B para tener un valor de comparación. Una mejora adicional al método es la gestión de envío en A basada en disparo hardware por rebase de valor de cuenta en un temporizador (en el MCP2510 por actuación sobre alguna de las patillas /TXnRTS. Combinando ambas técnicas se logra alta precisión en los parámetros de tiempo fundamentales del bucle de control: el periodo de muestreo ***Tm,*** cómo el retardo a la actuación, $(\delta+\gamma).Tm$.

En controladores avanzados, que incluyan “time stamps” en cada mensaje, este método fácilmente implementable.

Por último,indicar que para la aplicación del ejemplo, el bucle de regulación distribuido puede convivir con un flujo de información entre nodos relacionado con operaciones de configuración, gestión de red etc. sin que las modificaciones, configuración de nuevos mensajes, etc. deban alterar las características del bucle de control mientras éste se base en mensajes de mayor prioridad.

12. Bibliografía

1. "Comunicaciones y redes de computadoras", 5ª Ed, William Stallings, Ed Prentice-Hall. C15
2. "Redes de proceso distribuido" J. García Tomás, Santiago Tomás, Mario Pattini. Ed. RA-MA. 1997
3. "Las comunicaciones en la empresa, normas, redes y servicios". P.Mariño. Ed.RA-MA. 1995.
4. ISO International Standard 7498-1984. "Information processing systems-Open Systems Interconnection-Basic Reference Model.
5. LonWorks Engineering Bulletin, "Enhanced Media Access Control With LonTalk Protocol"
6. Holger Zeltwanger, "An Inside Look at the Fundamentals of CAN", Control Engineering, Enero 1995
7. Motorola, "LonWorks Technology Device Data"
8. "CAN Product Guide 1995", CiA organisation
9. LonWorks Engineering Bulletin, "LonTalk Protocol"
10. Wolfhard Lawrenz, " Worlwide Status of CAN, Present and Future"
11. IEC 870-1-1, 1-2, 1-3, 2-1, 3, 4, 5-1, 5-2, 5-3, 5-4, 5-5, 5-101, 6-2, 6-501, 6-502, 6-601 standards on Telecontrol Equipment and Systems
12. Fiber Optics Components databook. Hewlet Packard
13. "Embedded Communication Protocol options". Bhargav P. Upender. Proceedings of the Fifth Annual Embedded Systems Conference. 1993.
14. "Communication Protocols for Embedded Systems", Upender and P. Koopman, Embedded Systems Programming, November 94
15. "Profibus Technical Description" Profibus Trade Organization. Sep 1999.
16. "Interbus, the sensor-actuator bus". Interbus Club. 1997
17. "Device-Net Specification" Vol I y II. ODVA.
18. "Foundation Fieldbus, Technical Overview". Rev 2.0. Fieldbus Foundation. 1996.
19. "Smart Distributed System". Application Layer Protocol V 2.0. Honeywell 1996.
20. "Smart Distributed System". Physical Layer Specification. Honeywell 1994.
21. "Proposed Network Hierarchy for Open Control". George Thomas. Contemporary Controls.

22. "Surviving the fieldbus wars". D. Mash. EDN Europe. Abril 1999.

23. "The great fieldbus debate, is over! ". J.Pinto. Industrial Control Inteligence. Nov 1999

24. "CAN Specification" V 2.0. Robert Bosh Gmbh. 1991.

25. "CAN System Engineering" Wolfhard Lawrenz. Springer, 1997.

26. "The configuration of the CAN Bit Timing". F.Hartwich, A. Bassemir. Robert Bosh Gmbh.

27. "CANOpen Implementation". M.Farsi, M.Barbosa. Research Studies Press LTD. 2000.

28. "CANOpen. CAL-based Communication Profile". CIA Draft Standard 301

29. "CAL, CAN Application Layer". CIA Draft Standard 201..207.

30. "CANOpen. Device Profile for I/O Modules". CIA Draft Standard Proposal 401 V 1.4.

31. "CANOpen high-level protocol for CAN-bus". H.Boterenbrood. NIKHEF, Amsterdam.

32. "Tradeoffs Between Stand-Alone and Integrated CAN Peripherals". Craig Szydlowski, Intel Corporation. 1ª Conferencia CAN, 1994. CiA.

33. "MCP2510 Stand-Alone CAN Controller with SPI Interface". DS21291. 1999 Microchip Technology Inc.

34. "MCP2510 Silicon Revision A Errata Sheet". DS850059B. 1999 Microchip Technology Inc.

35. "Principles of Concurrent and Distributed Programming". M. Ben-Ari. 1990.

36. "Meeting Deadlines in Hard Real-Time Systems". L.P.Briand, D.M.Roy. IEEE Computer Society. 1999

37. "Calculating CAN response times". K. Tindell, A. Burns , A. Wellings. University of York. England. 1995.

38. "Real time descentralized control with CAN". Zuberi y Shi. 1998. University of Michigan.

39. "A perspective to the design of distributed real-time control applications based on CAN". M. Tömgren.

40. "Sistemas Discretos de Control", R. Aracil, A. Jimenez, ETSII Madrid, 1985.

41. "Implementing a distributed Real-Time clock using the CAN bus". M. Gergeleit. 1ª Conferencia Internacional CAN. 1994.

42. "CAN, Networking for Real Time Systems". Simeon Aladjem. Universidad de Tel-Aviv.

13. Algunos recursos en Internet

Ref	URL	Descripción
	www.can-cia.de	Organización de fabricantes y usuarios de CAN en redes industriales (“CAN in Automation”)
	www.profibus.com	Organización de fabricantes y usuarios de PROFIBUS.
	www.interbusclub.com	Organización de fabricantes y usuarios de INTERBUS
	www.odva.org	Open DeviceNet Vendor Association. Organización de soporte y normalización de DeviceNet
	www.fieldbus.org	Fieldbus Organization. Organización de normalización y soporte de Fieldbus
	www.ccsi.com/hart	Portal web de promoción y soporte de HART
	www.kvaser.se	Empresa promotora de la capa de aplicación CAN Kingdom
	www.bosh.de/k8/can	Página de Bosh dedicada a CAN