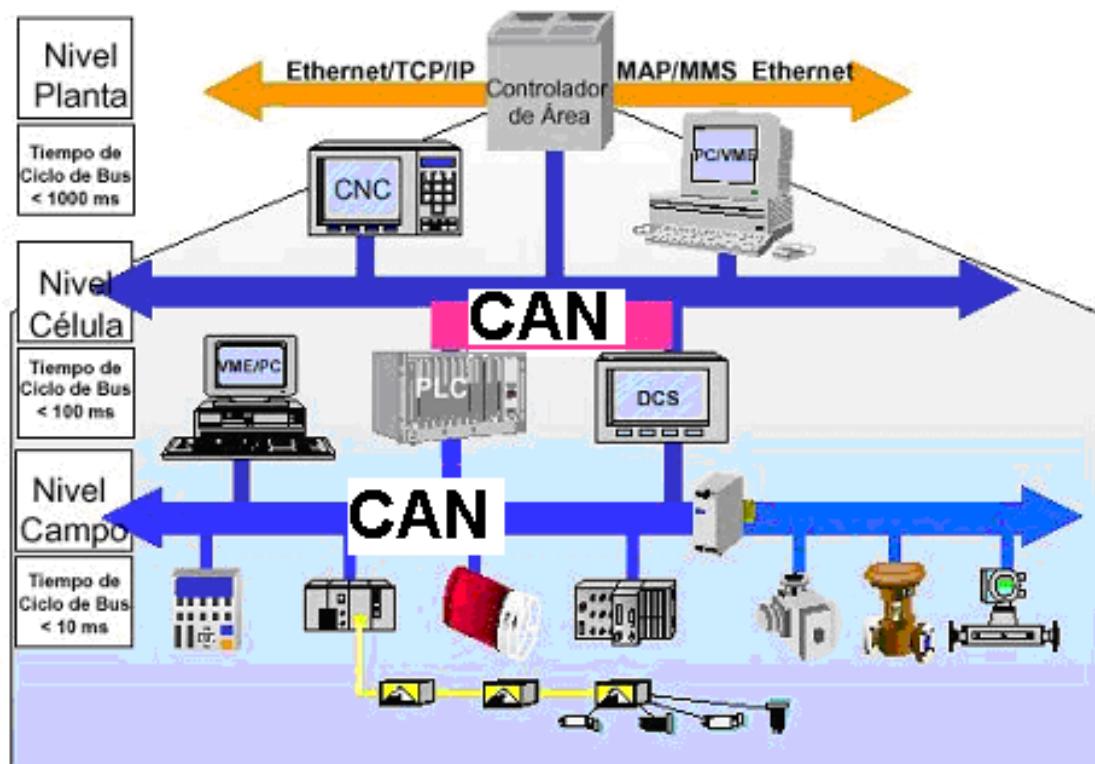
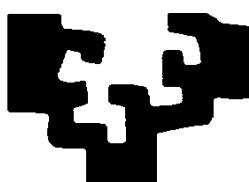


Apuntes de BUSES de CAMPO: CAN MOTSE



eman ta zabal zazu



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

Dpto. de Arquitectura y Tecnología de Computadores
Konputagailuen Arkitektura eta Teknologia Saila

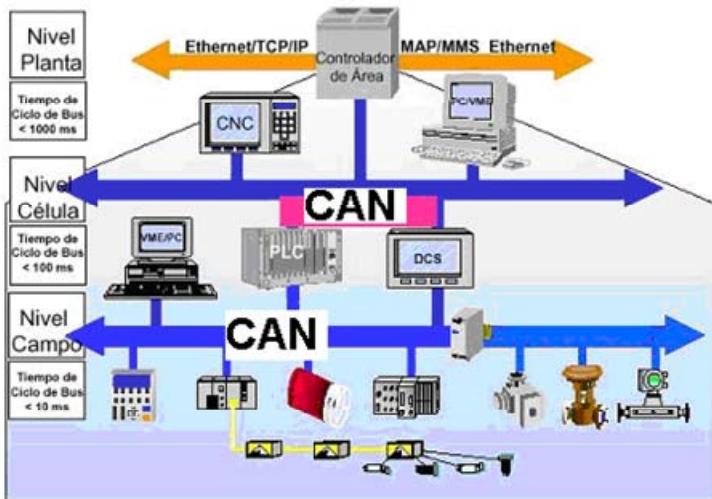
INDICE

BUSES DE CAMPO: BUS CAN.....	1
1 SISTEMAS DE TIEMPO-REAL	1
1.1 BUSES DE CAMPO	3
1.2 CARACTERISTICAS ESPECIALES DE LOS BUSES DE CAMPO	7
1.3 COMUNICACIÓN DE DATOS.....	9
1.3.1 LA CAPA FÍSICA	9
1.3.2 LA CAPA DE ENLACE.....	10
1.3.3 LA CAPA DE APLICACION	10
1.3.4 LA CAPA DE ADMINISTRACION Y CONTROL	11
1.3.5 COMUNICACIÓN CLIENTE-SERVIDOR Y PRODUCTOR-CONSUMIDOR	12
1.3.6 PROTOCOLOS ORIENTADOS AL NODO O AL MENSAJE	13
1.3.7 CONTROL DE ACCESO AL MEDIO	14
1.3.8 DETECCIÓN Y CONTROL DE ERRORES	15
1.3.9 TOPOLOGÍAS DE RED	17
1.4 BUS DE CAMPO CAN	18
1.5 EL BUS CAN Y LA ISO-11898.....	28
1.5.1 CAPA FISICA	29
1.5.2 CAPA DE ENLACE	30
1.5.3 ARBITRAJE	33
1.5.4 FORMATO DE LAS TRAMAS.....	38
Tramas de Datos.....	38
Campo de Arbitraje (<i>Arbitration Field</i>)	39
Campo de Control (<i>Control Field</i>)	40
Campo de Datos (<i>Data Field</i>)	40
Campo de Código de Redundancia Cíclica (<i>CRC Cyclic Redundancy Check</i>)	41
Campo de Reconocimiento (<i>Ack: Acknowledgement Field</i>)	42
Fin de Trama	43
1.5.5 TRAMA REMOTA	44
1.5.6 TRAMA DE ERROR	45
1.6 ERRORES	49
1.6.1 ERROR DE BIT.....	49
1.6.2 ERROR DE BIT DE RELLENO (STUFF)	50
1.6.3 ERROR CRC	51
1.6.4 ERROR DE FORMA.....	51
1.6.5 ERROR ACK	52
1.6.6 ESTADOS DE ERROR DEL NODO (CONFINAMIENTO DE FALLOS).....	53
Error-Activo	54
Error-Pasivo	55
Bus-OFF	56
Trama de Sobrecarga.....	57
Espacio entre Tramas (<i>IFS Intermisión</i>)	58
Validación de Tramas	59
Relleno de Bits (<i>Bit-stuffing</i>).	60
1.7 IMPLEMENTACION DE LA CAPA FÍSICA	61
1.7.1 REPRESENTACION FISICA DE LAS SEÑALES, REPRESENTACION DEL BIT.....	62
1.7.2 TIEMPO DE BIT Y SINCRONIZACION A NIVEL DE BIT	64
1.7.3 MEDIO DE TRANSMISIÓN	69
1.7.4 TOPOLOGÍA	71
1.7.5 ACCESO AL MEDIO	73
1.8 CARACTERÍSTICAS DE TIEMPO-REAL	74
1.9 EJEMPLO DE PROTOCOLO DE COMUNICACIÓN ROBOT HERO.....	79
CUATRO TIPOS DE SENsoRES Y DOS TIPOS DE MOTORES.....	81
1.10 RESUMEN DE FORMATOS	83

2 BUS CANOPEN	88
2.1 INTRODUCCIÓN:.....	88
2.1.1 CAL (CAN Application Layer)	89
2.2 CANOPEN.....	94
2.3 OD: DICCIONARIO DE OBJETOS (<i>CANOPEN OBJET DICTIONARY</i>)	95
2.4 SDOS: SERVICE DATA OBJECTS	104
2.5 PDOS: <i>PROCESS DATA OBJECTS (COMUNICACION IMPLÍCITA)</i>.....	109
2.6 MENSAJES PREDEFINIDOS Y OBJETOS ESPECIALES	114
2.7 GESTIÓN DE RED	121
2.8 PROCESO BOOT-UP	126

BUSES DE CAMPO: Bus CAN

Sistemas de Tiempo-Real, Buses de Campo, BUS CAN



Luis Gardeazabal, Dto. de Arquitectura y Tecnología de Computadores (v 2.0)

1

SISTEMAS DE TIEMPO-REAL

Tienen un tiempo de respuesta acotado y predeterminado

- No es suficiente que las acciones del sistema sean correctas.
- Deben producirse dentro de un intervalo de tiempo determinado.
- El sistema está conectado a un proceso externo del que recibe estímulos y a los que debe responder en un tiempo limitado para evitar que evolucione a un estado indeseable.

Bus CAN © L.G. (UPV/EHU)

2

1 SISTEMAS DE TIEMPO-REAL

Aunque el término tiempo-real se utiliza con excesiva frecuencia para referirnos a un gran número de aplicaciones que tienen una respuesta rápida ante los estímulos recibidos, la mayoría de ellos no lo son.

Una definición más exacta considera que los **sistemas de tiempo-real son aquellos que tienen un tiempo de respuesta acotado y predeterminado**¹. No es suficiente que las acciones del sistema sean correctas, sino que, además, deben producirse dentro de un intervalo de tiempo determinado.

Esto es debido a que el sistema está conectado a un proceso externo del que recibe estímulos y a los que debe responder en un tiempo limitado para evitar que evolucione a un estado indeseable.

¹ I. J. López, y J. Aranda, *Sistemas en Tiempo-Real. Introducción*. UNED, Departamento de Informática y Automática.

SISTEMAS DE TIEMPO-REAL (II)

Aunque una característica de estos sistemas debe ser la rapidez y la eficiencia, estas no definen a un sistema de tiempo-real si no existe **un análisis de los tiempos de respuesta del sistema**.

Los siguientes ejemplos de sistemas no son de tiempo-real, aunque se llaman así con demasiada frecuencia: banca *on-line*, servicio de tráfico, servicio meteorológico, etc. En cambio, si son sistemas de tiempo-real: control de robots, control de sillas de ruedas, control de procesos, control de semáforos, sistemas de navegación, etc.

Dentro de los sistemas de tiempo-real podemos encontrar sistemas de control, de procesamiento de señales y de telecomunicaciones.

Si los algoritmos del sistema los lleva a cabo un ordenador con una interfaz de E/S con el proceso, estaremos ante un sistema de tiempo-real informático. Si por el contrario el sistema de tiempo-real forma parte de un sistema más grande, viéndose como un subsistema, y se desarrolla con un hardware específico, considerándose un componente del sistema global, estaríamos ante un sistema empotrado de tiempo-real.

Para que un sistema se considere de tiempo-real todos los subsistemas que lo forman deben reunir las características necesarias para esta condición. Es decir, desde los sistemas de comunicación, pasando por todas las capas software que componen el sistema, hasta el proceso de aplicación propio del sistema deben ser de tiempo-real.

BUSES DE CAMPO (I)

Término genérico que describe un conjunto de redes de comunicación para uso industrial y orientadas al control de sistemas, cuyo objetivo es sustituir las conexiones punto a punto entre los elementos de campo y el equipo de control, y entre ellos mismos.

Son redes digitales, bidireccionales, multipunto sobre un bus serie que conectan dispositivos de campo como PLCs, transductores, actuadores y sensores.

1.1 BUSES DE CAMPO

Los avances tecnológicos en el entorno industrial proporcionan un aumento considerable de la productividad, de la calidad del producto, una disminución del tiempo de respuesta de salida al mercado, así como del coste por unidad producida. Todo ello conlleva un aumento en el volumen de negocio de la empresa. Hacia los años 70, se comenzaron a introducir computadores en el control de procesos en las tareas de vigilancia. En un primer momento el computador se encargaba de detectar anomalías y generar la correspondiente alarma. Posteriormente, se comenzó a controlar estos procesos de manera activa debido a su amplia capacidad de cálculo, sustituyendo así al cuadro de mando. La evolución posterior de los procesadores y la aparición de los microcontroladores y los controladores lógicos programables (PLCs) permitieron el desarrollo del control distribuido que liberaba al computador central del procesamiento digital de la información de los elementos participantes en el proceso de producción.

BUSES DE CAMPO (II)

Integran en un mismo sistema de comunicación todo lo relacionado con la producción de un producto, que abarca:

- Desde → el nivel administrativo de la empresa.
- Hasta → el control de cada uno de los actuadores y sensores que intervienen en el proceso de fabricación, almacenamiento o distribución.
- Pasando por la coordinación de la planta de producción y el control de la célula correspondiente.

Se necesita un sistema de control jerarquerizado por niveles.

De forma que se consigue una red de comunicación adaptada a las necesidades específicas de cada fase del proceso de producción.

El desarrollo del control distribuido en la industria va en paralelo al de las comunicaciones. Hoy en día se impone el uso de dispositivos inteligentes para el control y/o la supervisión remota de procesos de fabricación, almacenamiento o distribución. Los sistemas de comunicación empleados en estos procesos están condicionados en su diseño por la especificidad de los mismos a diferencia de las redes de datos empleadas en ofimática. Cada proceso demanda un sistema a medida para poder satisfacer sus necesidades de la mejor forma posible. Debido a la imposibilidad de crear tantos sistemas de comunicación distintos como procesos existentes (por motivos económicos, principalmente) es necesario definir estándares que solventen esta situación. Este es el motivo por el que aparecieron los buses de campo que conocemos en la actualidad.

Desde el punto de vista empresarial, la necesidad de comunicación no sólo se restringe a la producción, si no que otros departamentos de la empresa también pueden participar y tomar decisiones sobre la misma, permitiendo realizar un control global del sistema. De esta forma las decisiones que se toman en el nivel administrativo de la empresa pueden incidir sobre el desarrollo final del producto.

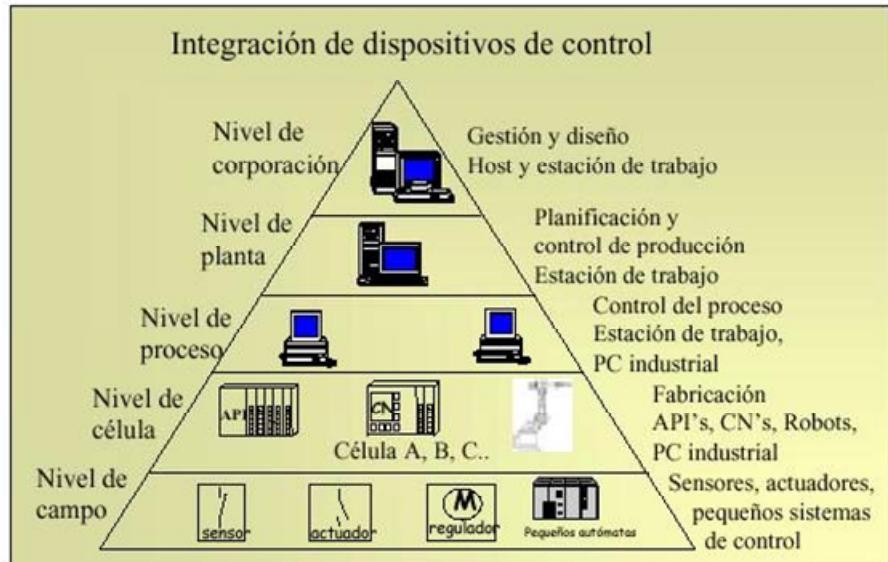


Fig.1: Arquitectura CIM (Computer Integrated Manufacturing)

Bus CAN © L.G. (UPV/EHU)

6

Para conseguir integrar en un mismo sistema de comunicación todo lo relacionado con la producción de un producto, que abarca desde el nivel administrativo de la empresa, hasta el control de cada uno de los actuadores y sensores que intervienen en el proceso de fabricación, almacenamiento o distribución, pasando por la coordinación de la planta de producción y el control de la célula² correspondiente, es necesario un sistema de control jerarquizado por niveles como el de la figura. De esta forma se consigue una red de comunicación adaptada a las necesidades específicas de cada fase del proceso de producción.

Las necesidades de cada fase son diferentes y muy dispares. Por lo tanto, se necesita que cada nivel³ esté dotado de las mismas características que la fase:

- En la fase de producción se necesita tiempo-real, inmunidad al ruido e interferencias, simplicidad de las instalaciones, etc.
- En la fase de diseño se ve la necesidad de acceder a gran cantidad de información, mientras que el tiempo de respuesta no es crítico
- En la fase de gestión, los gestores de la empresa necesitan comunicarse con los clientes, etc.

Cada uno de los niveles en los que se compone este sistema, además de llevar a cabo las labores específicas del nivel, realiza un tratamiento y filtrado de la información que pasa al nivel inmediatamente superior e inferior.

La base de la pirámide la forma el nivel de campo o nivel de control directo. Es el encargado de controlar las variables del sistema (en procesos de producción flexibles) y actuar en función de los datos recogidos según los algoritmos de control y consignas del nivel superior.

El siguiente nivel en la pirámide es el de supervisión y control de célula, que elabora la información procedente del nivel inferior e informa al operario⁴ de la célula.

El nivel de proceso se encarga de gestionar las células necesarias en la realización de uno de los procesos de la producción.

El cuarto nivel (nivel de planta) lleva a cabo labores de coordinación de la planta de producción. Controla y organiza toda el área, tratando de optimizar balances de materia y energía (flujos entre el almacén, la planta, la distribución e incluso proveedores).

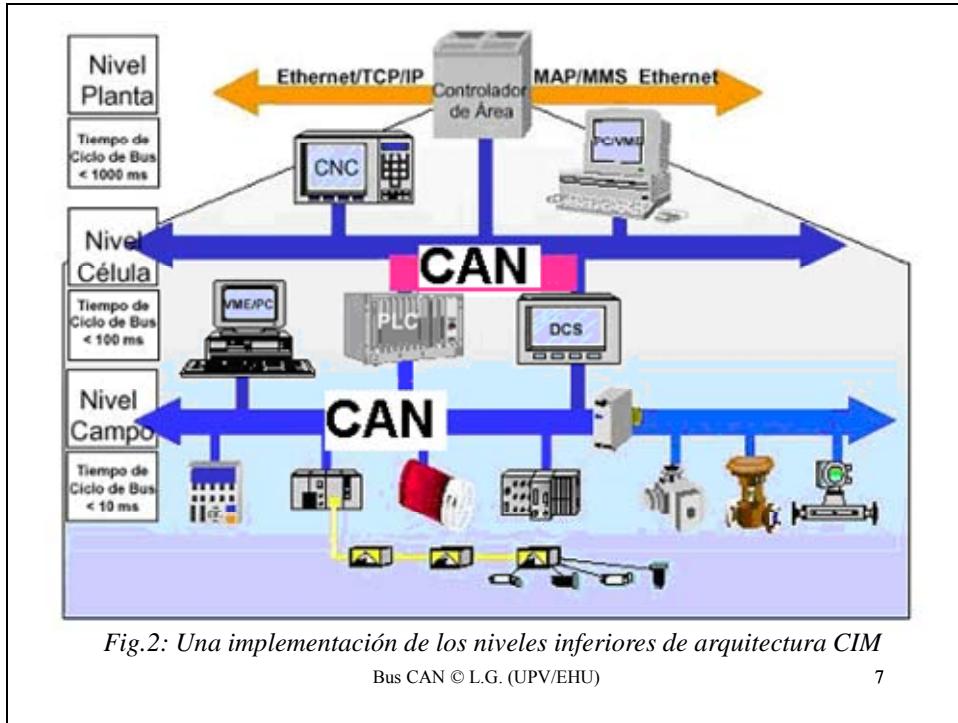
En el nivel de corporación podemos distinguir dos subniveles: el de oficina técnica y el de gestión de empresa. El primero de ellos se encarga de planificar la producción total de la empresa y diseñar mediante herramientas CAD⁵ nuevos productos generando automáticamente programas para los elementos de producción. El segundo, establece la política de producción del conjunto de la empresa en función de los recursos y costes del mercado. También se incluyen tareas de contabilidad y gestión empresarial.

² Unidad mínima de producción

³ No necesariamente una fase debe coincidir con uno de los niveles

⁴ Empleado encargado del funcionamiento de una célula concreta

⁵ Computer Aided Design



La implementación de todos estos niveles da lugar a la aparición del término CIM: *Computer Integrated Manufacturing*.

Uno de los principales problemas para el logro de esta integración lo representa la intercomunicación de los elementos del nivel de campo. Es en este nivel, donde se sitúan los buses de campo que vamos a referenciar posteriormente.

El flujo de información vertical (entre distintos niveles) se realiza a través de elementos de interconexión de redes (puentes, pasarelas, etc.) o bien de manera directa utilizando el mismo protocolo.

El flujo descendente corresponde a peticiones del nivel superior al inferior, mientras que el ascendente son informes del resultado de la realización de estas peticiones.

El flujo de información horizontal se realiza entre entidades del mismo nivel a través de sus buses y corresponde a información relativa a la tarea que realizan en conjunto.

CARACTERÍSTICAS ESPECIALES de los BUSES de CAMPO:

- Transmiten pequeñas cantidades de información.
- Cubren las necesidades de tiempo-real de las aplicaciones.
- Tienen gran compatibilidad electromagnética.
- Presentan un número reducido de estaciones.
- Son de fácil configuración.
- Presentan protocolos simples y limitados.
- Tienen bajos costes de conexión e instalación.
- Son consistentes con el modelo OSI (*Open Systems Interconnections*) de ISO.

1.2 CARACTERISTICAS ESPECIALES DE LOS BUSES DE CAMPO

Un bus de campo es el término genérico que describe un conjunto de redes de comunicación para uso industrial y orientado al control de sistemas. Su objetivo es sustituir las conexiones punto a punto entre los elementos de campo y el equipo de control, y entre ellos mismos. Son redes digitales, bidireccionales, multipunto sobre un bus serie que conectan dispositivos de campo como PLCs, transductores⁶, actuadores y sensores. Estos dispositivos suelen ser inteligentes de manera que pueden llevar a cabo funciones propias de control, mantenimiento y diagnóstico informando de cualquier anomalía producida en sí mismos o en el sistema en general.

VENTAJAS que PRESENTAN:

- Reducción del coste frente a otro tipo de sistemas.
- Ahorro en el coste de la instalación.
- Ahorro en el coste del mantenimiento.
- Ahorros derivados de la mejora del funcionamiento del sistema.
- Aumento de flexibilidad en el diseño del sistema.
- Simplificación en la obtención de la información de la planta a través de los sensores.

⁶ Convierten una magnitud física en otra.

CAPAS DE COMUNICACIÓN

Un bus de campo utiliza 3 capas del modelo OSI:

Física, Enlace y Aplicación

Muchos intentos de estandarizar el concepto de bus de campo:

- La norma IEC 1158/ISA SP50.02 (USA) y el *WorldFIP* como protocolo más acorde a su especificación.
Válido para aplicaciones simples y económicas.
- La norma EN50170/ISA SP50 (Alemania) y *Profibus*, utilizado en automatización.

La norma ISO 11898/ISO 115192 (Alemania) y el protocolo CAN.

Desarrollado inicialmente por Bosch para reducir cableado en los automóviles.

Tanto la norma, como CAN, sólo cubren la especificación de la capa física y la de enlace, dejando la de aplicación abierta a diferentes soluciones:

CANOpen, DeviceNet, CANKingdom, SDS, etc.

Bus CAN © L.G. (UPV/EHU)

10

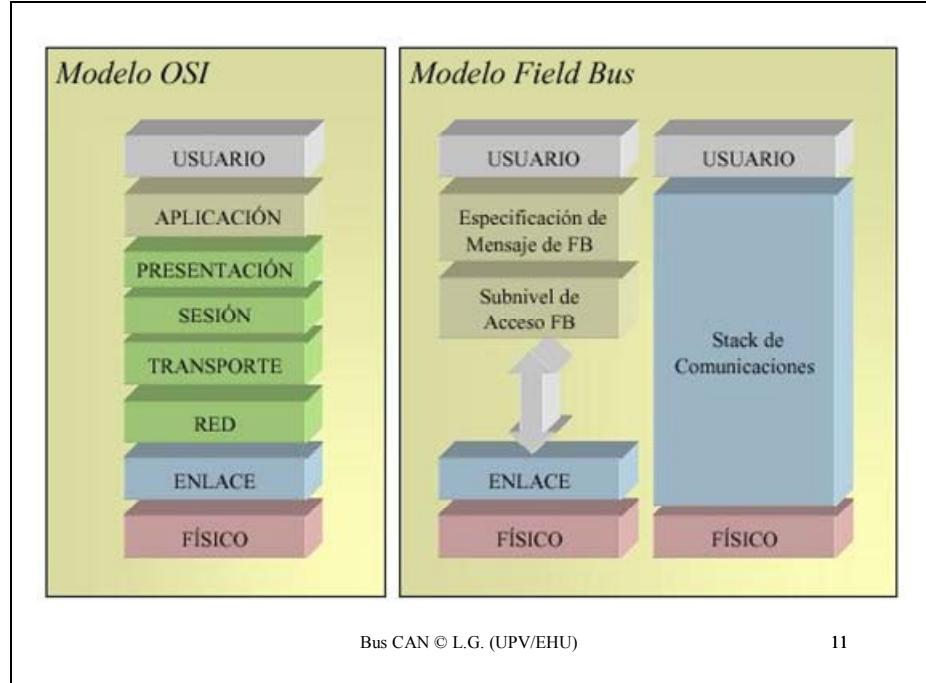
Los buses de campo sólo incluyen tres capas del modelo OSI⁷ de ISO⁸ la física, la de enlace y la de aplicación, un conjunto de servicios de administración⁹. El usuario debe conocer la capa física, para conectar los dispositivos, y los servicios de administración de la red. En la práctica, ha habido muchos intentos de estandarizar el concepto de bus de campo así como muchas soluciones industriales más o menos acordes con cada uno de esos estándares¹⁰.

⁷ OSI: Open System Interconnection.

⁸ ISO: Organización Internacional de Normalización.

⁹ Servicios que ofrece la capa de aplicación para configurar la red.

¹⁰ H. Kaschel y E. Pinto, *Análisis del estado del arte de los buses de campo aplicados al control de procesos industriales*. Universidad de Santiago de Chile, Departamento de Ingeniería Eléctrica.



1.3 COMUNICACIÓN DE DATOS

A continuación se detallan algunos conceptos relativos a la comunicación de datos en un bus de campo¹¹ para comprender que características tiene el bus CAN y que ventajas obtiene de ellas.

Como se ha comentado, un bus de campo utiliza una parte de las capas del modelo de comunicación OSI. En concreto utiliza las capas física, enlace y aplicación.

A continuación se describen las capas utilizadas y se explica la necesidad de una nueva capa de administración y control.

La CAPA FÍSICA define:

- Los parámetros eléctricos, mecánicos y funcionales de la conexión física de los dispositivos que forman la red.
- La conversión de los datos de la capa de enlace a flujo de bits o señales que circularán por ella.
- El proceso inverso, en el que se transforma la señal recibida en datos compatibles con la capa de enlace.
- Realiza funciones de sincronización y *bit-timing*, y la definición de las conexiones al medio.

1.3.1 LA CAPA FÍSICA

¹¹ Konrad Etschberger. *Controller Area Network. Basics, Protocols, Chips and Applications*, Capítulo 1: *Data communication in the Field Domain*, IXXAT Press 2001.

La CAPA de ENLACE

Construye las tramas con los datos, información de control y código de detección de errores.

- La información de control transmitida sirve al receptor para saber si una trama le interesa o no.
- El código de detección de errores, identifica si una trama está corrupta o no.

Integra la función MAC (*Medium Access Control*) que gestiona el acceso al medio según el orden de llegada y las prioridades asociadas a cada trama para que no se pierda información en el proceso.

1.3.2 LA CAPA DE ENLACE

La CAPA de APLICACIÓN

Da soporte a las aplicaciones de usuario que se comunican a través de la red.

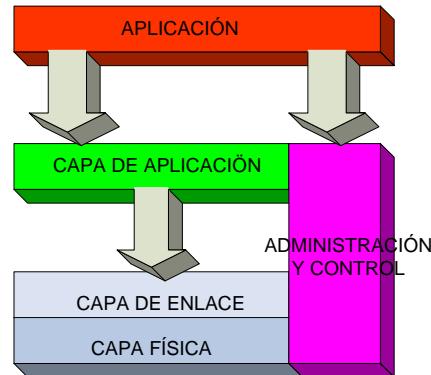
1.3.3 LA CAPA DE APLICACION

#La CAPA DE ADMINISTRACIÓN Y CONTROL

Configura los parámetros de cada una de las capas inferiores, y obtención de estado e información de errores de la capa.

Estas funciones se realizan normalmente al comienzo de la aplicación y llevan a cabo tareas como configurar el modo de operación, ajustar el *bit-rate*, etc.

Relación entre capas de comunicación del modelo OSI, administración y control, y el proceso de la aplicación



Bus CAN © L.G. (UPV/EHU)

15

1.3.4 LA CAPA DE ADMINISTRACION Y CONTROL

A diferencia de las funciones de comunicación llevadas a cabo por las capas del modelo OSI, las aplicaciones de control se ven en la necesidad de contar con una capa adicional de administración y control (ver Figura) para configurar los parámetros de cada una de las capas inferiores, así como para la obtención de meta-information relativa a las mismas (estado e información de errores de la capa). Esto es debido a la heterogeneidad de los entornos en los que se pueden emplear este tipo de aplicaciones, por lo que necesitan adecuar los parámetros de cada capa al entorno correspondiente. Estas funciones se realizan normalmente al comienzo de la aplicación y llevan a cabo tareas como configurar el modo de operación, ajustar el *bit-rate*, etc.

COMUNICACIÓN CLIENTE-SERVIDOR y PRODUCTOR-CONSUMIDOR

- Cliente-Servidor describe una relación *uno-a-uno*.
- Productor-Consumidor describe una relación *uno-a-muchos*.

Este último, es el caso del bus CAN.

Bus CAN © L.G. (UPV/EHU)

16

1.3.5 COMUNICACIÓN CLIENTE-SERVIDOR Y PRODUCTOR-CONSUMIDOR

La comunicación Cliente-Servidor describe una relación uno-a-uno, y denota las peticiones de un cliente y las respuestas de un servidor, si bien puede haber más de un cliente con el mismo servidor. Por el contrario el paradigma Productor-Consumidor describe una relación uno-a-muchos, es decir, muestra la comunicación de un productor con todos sus consumidores. Es en este último caso donde se encuentra el bus CAN, si bien es cierto que, el modelo Cliente-Servidor puede ser adoptado por los protocolos de la capa de aplicación.

PROTOCOLOS ORIENTADOS al NODO o al MENSAJE

Si el mensaje lleva información que identifica al nodo origen y al de destino, estaremos ante un protocolo orientado al nodo.

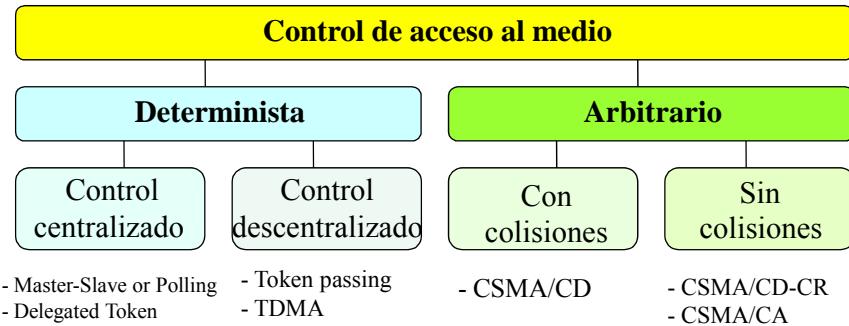
Si el mensaje sólo lleva información asociada a sí mismo (cada nodo acepta el mensaje si es del tipo que le interesa), estaremos ante un protocolo orientado al mensaje.

1.3.6 PROTOCOLOS ORIENTADOS AL NODO O AL MENSAJE

Los mensajes que circulan por un bus pueden estar orientados al nodo u orientados al mensaje, según el protocolo que se emplee. Si el mensaje lleva información que identifica al nodo origen y al de destino, estaremos ante un protocolo orientado al nodo. Por el contrario, si el mensaje sólo lleva información asociada a sí mismo (cada nodo aceptará el mensaje si es del tipo que le interesa), estaremos ante un protocolo orientado al mensaje.

CONTROL DE ACCESO AL MEDIO

Gestiona que emisor tiene el control del bus para transmitir en un momento dado y evita que señales de diferentes emisores se interfieran entre sí.



Bus CAN © L.G. (UPV/EHU)

18

1.3.7 CONTROL DE ACCESO AL MEDIO

El control de acceso al medio se utiliza para gestionar cual es el elemento (nodo) transmisor que tiene el control del bus para transmitir en un momento dado y evitar así que señales de diferentes transmisores se interfieran entre sí.

El método de control de acceso elegido en un bus de campo es determinante en su funcionamiento, así como en la condición de ser o no de tiempo-real. Estos métodos se pueden clasificar según la figura en cuatro grupos. Si existe una unidad central que gestiona el acceso tendremos, un control determinista centralizado. Si por el contrario, son los dispositivos conectados al bus quienes por medio de turnos gestionan el acceso, estaremos ante un control determinista descentralizado.

En los métodos arbitrarios cualquier dispositivo tomará el bus siempre y cuando éste esté libre, de forma tal que si dos dispositivos lo intentan a la vez, sólo el de mayor prioridad tomará el bus. Los métodos llamados ‘arbitrarios con colisión’, detectan si se produce alguna colisión y deshacen la situación volviendo al estado anterior. Los métodos denominados sin colisión, por el contrario, ‘impiden’ que éstas se produzcan.

Existen varias implementaciones para cada uno de estos métodos:

Control determinista centralizado

- Master-Slave or Polling
- Delegated Token

Control determinista descentralizado

- Token passing
- TDMA: Time Division Multiple Access

Control arbitrario con colisiones

- CSMA/CD: Carrier Sense Multiple Access/Collision Detect

Control arbitrario sin colisiones

- CSMA/CD-CR: Carrier Sense Multiple Access and Collision Detection with Collision Resolution
- CSMA/CA: Carrier Sense Multiple Access with Collision Avoidance

DETECCIÓN Y CONTROL DE ERRORES (I)

- Varios tipos de error dependiendo de la capa en la que se detecten.
- Si evitamos que se produzca un error, nos aseguramos de que no pase a la capa superior donde su corrección tendría peores consecuencias.
- Los sistemas de comunicación utilizan distintos mecanismos en cada capa.
- Uno de los mejores métodos en la capa física es transmitir en forma diferencial. Detecta errores a nivel de bit.

1.3.8 DETECCIÓN Y CONTROL DE ERRORES

Los entornos industriales se caracterizan por su alto grado de hostilidad para la transmisión de señales digitales, debido, principalmente, a las interferencias causadas como consecuencia de la existencia de gran cantidad de máquinas y dispositivos eléctricos. Estas interferencias (inductivas y/o capacitivas) pueden producir errores en la transmisión de las señales eléctricas entre dispositivos. Para solventar este problema los sistemas de comunicación cuentan con mecanismos para la detección y corrección de errores.

Dependiendo de la capa en la que se detecten, se diferencian varios tipos de errores en el modelo de comunicación OSI. Si corregimos o evitamos que se produzca un error, nos aseguramos de que no pase a la capa superior donde su corrección tendría peores consecuencias. Para ello, los sistemas de comunicación utilizan distintos mecanismos en cada capa con el fin de minimizar los problemas causados por los mismos. Uno de los mejores métodos de prevención de errores en la capa física consiste en transmitir la señal en forma diferencial. Este método, utiliza dos líneas de transmisión para un determinado valor lógico de la señal, transmitiendo en el mismo instante dos tensiones complementarias. De esta forma, se pueden detectar los errores que se producen a nivel de *bit*, donde una de las líneas ha sido afectada por alguna interferencia, o incluso si ambas líneas sufren la misma desviación.

DETECCIÓN Y CONTROL DE ERRORES (II)

- Es necesario añadir otros mecanismos en la capa de enlace.
- El más extendido introduce un campo adicional en la trama de los datos con información relativa al contenido de ella.
- Para que la capa de aplicación no se entere de la existencia de errores es necesario corregirlos en la capa de enlace.

Resulta imposible prevenir la totalidad de errores por mucho que se utilice el mejor medio físico de comunicación. Por lo tanto, es necesario añadir otros mecanismos en la capa de enlace que nos asegure que la trama recibida está libre de errores. El método más extendido es introducir en los datos de la trama un campo adicional con información relativa al contenido de ella, de esta forma se puede analizar si el contenido de la trama contiene lo que se espera. Con este método sólo se consigue detectar el error.

Para que la capa de aplicación no se entere de la existencia de errores en la transmisión (entre sus cometidos está detectar errores a nivel de aplicación) es necesario corregirlos en la capa de enlace. Para ello, cuando se detecta un error se solicita el reenvío de la trama al transmisor hasta que esté libre de errores. Estas retransmisiones deben ser las menos posibles para reducir la pérdida de rendimiento y no poner en peligro la función de la aplicación.

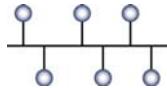
TOPOLOGÍA DE RED:

- Describe la estructura de la conexión física entre los dispositivos
- Determina el costo de la instalación, los límites de la aplicación así como los parámetros de la red.

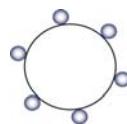
Estrella



Bus



Anillo



Árbol



Malla



Bus CAN © L.G. (UPV/EHU)

21

1.3.9 TOPOLOGÍAS DE RED

La topología de una red describe la estructura de la conexión física entre los dispositivos y determina el costo de la instalación, los límites de la aplicación así como los parámetros de la red. Las topologías más extendidas son las siguientes: estrella, bus, anillo y árbol.

BUS CAN: Un BUS de CAMPO

Bus CAN (*Controller Area Network*, Red de Área de Controlador)

- Es un bus de campo serie empleado en sistemas de control distribuido en tiempo-real.
- Desarrollado por el fabricante alemán de componentes para automóviles, *Robert Bosch GmbH* en 1984, para aplicaciones en la industria automotriz.
- Con el tiempo, adoptado para aplicaciones industriales y de control.
- Como indica su nombre, constituye una red entre controladores o microcontroladores.
- Es una red con topología de bus: sólo uno de los **Nodos (Dispositivos de Campo)** puede transmitir en un momento dado.

1.4 BUS DE CAMPO CAN

El bus CAN es un bus de campo serie muy empleado en sistemas de control distribuidos en tiempo-real. Originalmente fue desarrollado para aplicaciones en la industria automotriz, pero con el tiempo y debido a sus características, ha sido adoptado por aplicaciones industriales y de control.

Fue especificado por la compañía alemana BOSCH y estandarizado por ISO. Como su nombre indica constituye una red entre controladores o microcontroladores.

La continua aparición de dispositivos y/o sistemas electrónicos en el automóvil se debe, por un lado, al deseo de los usuarios a mayor seguridad y confort, y por otro, a los requerimientos de los gobiernos en relación a la emisión de gases, al consumo de combustible, etc. La necesidad de interconexión entre estos dispositivos debido a la complejidad de las funciones que debían desarrollar, resultaba poco eficiente la conexión punto a punto (mucho cableado con un aumento considerable del costo del sistema) como se venía realizando hasta entonces. Para solventar este problema, el fabricante alemán de componentes para automóviles, Robert Bosch GmbH, desarrolló el bus CAN en 1984.

Historia del CAN

- **1980:** La industria del automóvil requiere de un bus barato, de tiempo real y altamente robusto para comunicar diferentes componentes electrónicos. Can define únicamente las capas 1 y 2 del modelo ISO.
- **1983:** Junto con diferentes Universidades Alemanas, BOSCH desarrolla el protocolo CAN = “*Controller Area Network*“.
- **1985:** INTEL comercializa los primeros chips de CAN.
- **1986:** Se presentan en Detroit (EEUU) los primeros prototipos.
- **1987:** Fabricantes de C.I. como Intel, Motorola,... ofrecen una gama completa de microcontroladores que integran CAN.
- **1989:** Primeras aplicaciones industriales.
- **1987-1991:** Aparecen organizaciones que utilizan CAN en el automóvil (SAE, OSEK), y en aplicaciones industriales (CIA = “*CAN in Automation*”, CAN en automatización) www.can-cia.de/

Bus CAN © L.G. (UPV/EHU)

24

Historia del CAN (II)

- **1992:** MERCEDES usa un bus CAN a 500 kbps para enlazar 5 submódulos electrónicos en un Clase S.
- **1993:** CAN de Alta Velocidad (1 Mbps/Identificadores de 11 bits) estándar ISO 11898 = CAN 2.0 A.
Publicación por la CiA de las especificaciones CAL=Can Application Layer describiendo los mecanismos de transmisión sin definir cuando y como utilizarlos.
- **1995:** identificadores de 29 bits (CAN2.0 B) - Publicación por la CiA del perfil de comunicaciones DS-301 = CANopen
- **1996:** CAN se usa en la mayoría de coches de gama alta europeos.
- **1997:** CIA integra a más de 300 compañías.
- **2001:** CIA publica el perfil DS-304 el cual se puede usar para integrar componentes de nivel de seguridad 4 en un bus CANopen estándar (CANsafe).

Bus CAN © L.G. (UPV/EHU)

25

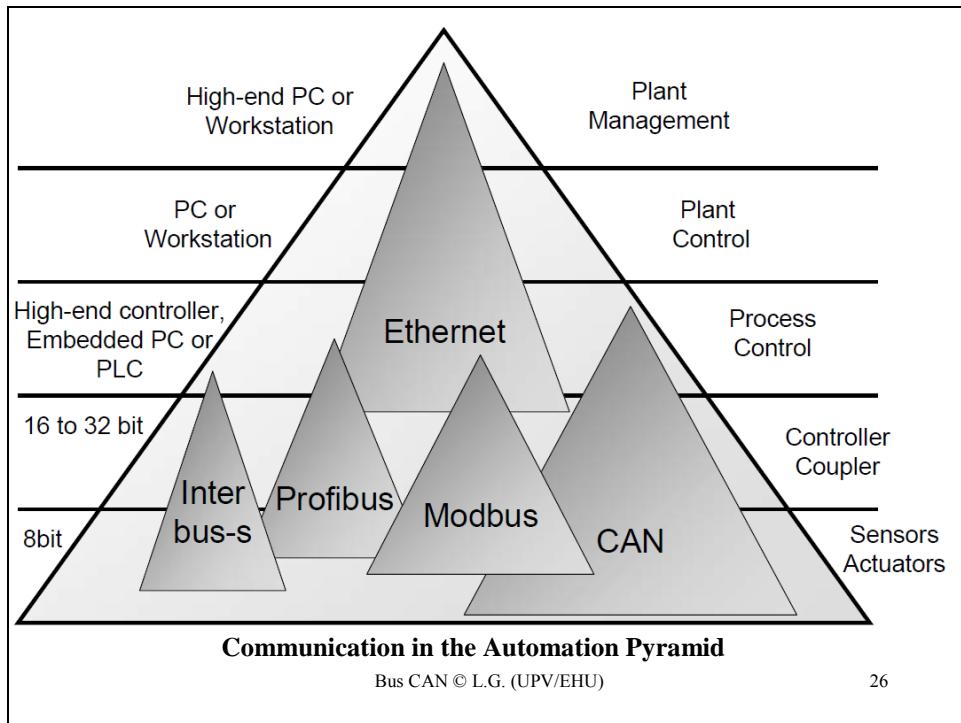
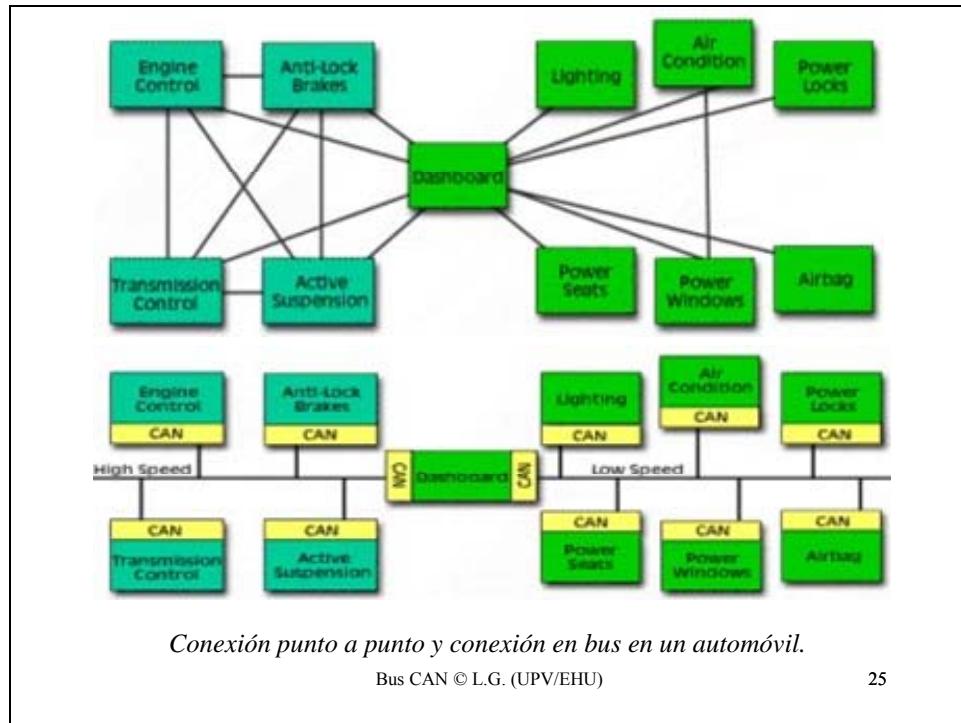
- El tiempo que cada **nodo** está transmitiendo ininterrumpidamente es muy pequeño en relación a otras redes orientadas a intercambios masivos de información, como Ethernet.
- Si un nodo necesita transmitir un mensaje crítico, se asegura que no va a tener que esperar más que un tiempo prefijado y además pequeño, que dependerá de la prioridad del mensaje y de la velocidad del bus.
- Emplea un sistema de prioridades, para que, ante una colisión de dos mensajes en el bus, prevalezca el de mayor prioridad, consiguiendo que los mensajes críticos del sistema lleguen a su destino en un tiempo determinado.

El bus CAN es una red con topología de bus. Debido a esto, sólo uno de los nodos¹² puede transmitir en un momento dado. Pero, a diferencia de lo que parece, esto no es una desventaja en el rendimiento del bus, si se tiene en cuenta las siguientes circunstancias:

El tiempo en que cada nodo está transmitiendo ininterrumpidamente (cuando un nodo consigue acceso al bus no lo libera hasta que no finaliza la transmisión del mensaje) es muy pequeño en relación a otras redes orientadas a intercambios masivos de información, como Ethernet. Por lo tanto, si un nodo necesita transmitir un mensaje crítico, se asegura que no va a tener que esperar más que un tiempo prefijado y además pequeño, que dependerá de la prioridad del mensaje y de la velocidad del bus. Si este tiempo se ajusta a las necesidades del sistema, se asegura un tiempo de respuesta dentro de lo permitido por la aplicación.

El protocolo CAN emplea un sistema de prioridades, para que ante una colisión de dos mensajes en el bus, prevalezca el de mayor prioridad, consiguiendo que los mensajes críticos del sistema lleguen a su destino en un tiempo determinado.

¹² Dispositivos de campo.



EL ÉXITO DEL BUS CAN SE DEBE:

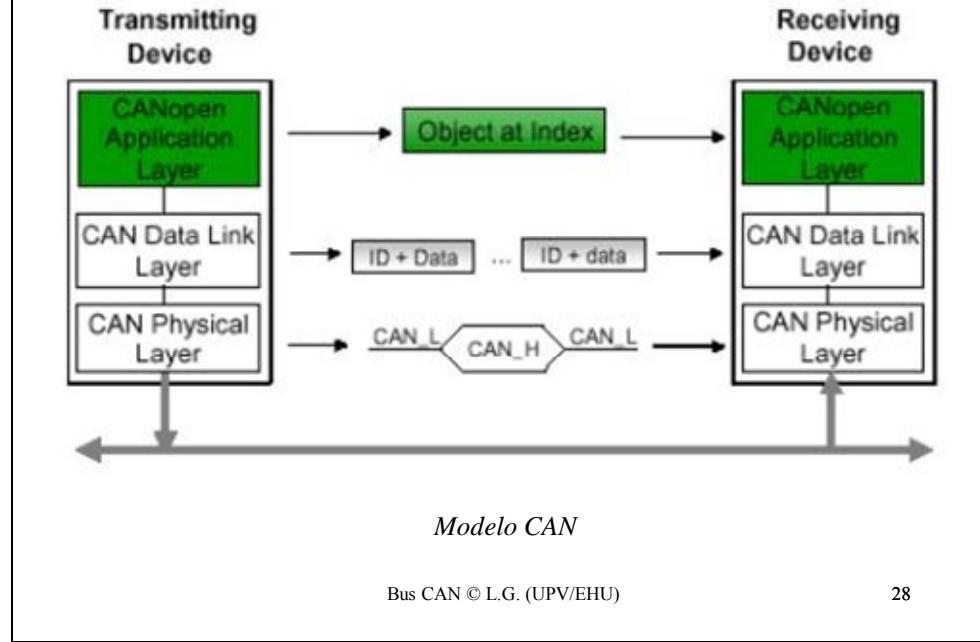
- A su elevada seguridad, gracias al sistema de arbitraje de prioridades.
- A su alto grado de inmunidad al ruido e interferencias.
- El protocolo CAN asegura que el mensaje con mayor prioridad se puede transmitir en un intervalo de tiempo muy corto (conocido).
- Otros sistemas se basan en protocolos por turnos, o en la eliminación de todos los mensajes tras detectar la colisión empleando mucho más tiempo.

El éxito del bus CAN se debe, por un lado, a su elevada seguridad, conseguida con el sistema de arbitraje de prioridades, y por el otro, gracias a su alto grado de inmunidad al ruido e interferencias (muy frecuentes en los entornos donde se emplea).

El protocolo CAN asegura que el mensaje con mayor prioridad se puede transmitir en un intervalo de tiempo muy corto (dentro de lo especificado por la aplicación y para poder mantener los requisitos de tiempo-real), mientras que otros sistemas de comunicación, se basan en protocolos por turnos, o en la eliminación de todos los mensajes tras detectar la colisión empleando mucho más tiempo.

Para reglamentar las especificaciones del bus CAN, se creó en el año 1992 el club de usuarios CiA (*CAN in Automation*), cuyo objetivo principal fue mantener al bus CAN como un sistema abierto, facilitando la comunicación entre sistemas abiertos no propietarios.

El bus CAN está dividido en las siguientes capas y subcapas:



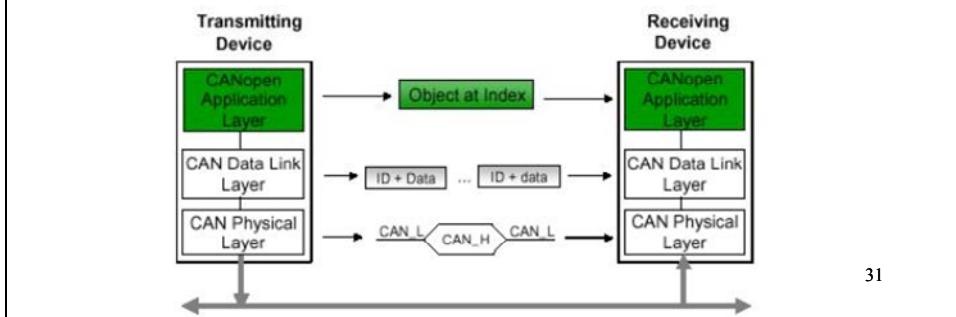
El bus CAN está dividido en las siguientes capas y subcapas:

- la subcapa de aplicación
- la subcapa de enlace
- la capa física

Las dos primeras componen el nivel de enlace del modelo OSI de ISO y tienen como función dar servicio a la capa de aplicación, de forma que la comunicación CAN sea transparente para ésta. La capa física especifica la forma de acceso al bus. En CAN, como en cualquier otro tipo de bus de campo, no existen los niveles de presentación, de sesión, de transporte, ni de red. En nivel de presentación no se requiere por no ser necesaria una conversión de los datos ya que tienen el mismo formato. El nivel de sesión no se usa en aplicaciones de tiempo-real. El nivel de transporte no hace falta, ya que los mensajes son de la misma longitud y no se dividen en paquetes más pequeños que luego hay que recomponer, y además está asegurada su corrección de datos por el nivel de enlace. El nivel de red no se requiere dado que todos los nodos reciben todos los mensajes y filtran aquellos que son de su interés.

- La **Capa FÍSICA** define las señales usadas en la transmisión, y el medio físico usado (la especificación CAN no la define, se puede optimizar según la aplicación).
- La **subCapa de ENLACE** es el núcleo del protocolo CAN: es responsable del *bit-timing* y la sincronización, el arbitraje, la aceptación de los mensajes, y de la detección y señalización de errores.
- La **subCapa OBJETO** se encarga del filtrado y manejo de los mensajes.

Bosh sólo especificó el nivel de enlace y parte del físico.



La capa física define que señales se usan para la transmisión, y el medio físico que se usa para ello. Esta capa no queda definida en la especificación CAN, por lo que puede ser optimizada según la aplicación donde se ejecute.

La subcapa de enlace es el núcleo del protocolo CAN, y es la responsable de:

- el *bit-timing* y la sincronización
- el arbitraje
- la aceptación de los mensajes
- la detección y señalización de errores

La subcapa objeto se encarga del filtrado y manejo de los mensajes.

Bosh sólo especificó el nivel de enlace y parte del físico, dejando el resto sin determinar. Posteriormente, en 1993 se crearon las normas ISO 11898¹³, para aplicaciones de alta velocidad (hasta 1 Mbps) e ISO 11519-2¹⁴, para aplicaciones de baja velocidad, que definen completamente el nivel de enlace y el físico. Es decir, si una aplicación hace uso del bus CAN puede utilizar como nivel físico el que más le convenga, pero si quiere ser compatible con las normas ISO anteriores debe ceñirse a lo especificado en ellas. Hay que observar que la capa de aplicación no está definida ni por CAN ni por ninguna de las normas ISO asociadas a él, por lo que queda abierta para cualquier protocolo CAL (*CAN Application Layer*).

¹³ "Road Vehicles - Interchange of Digital Information - Controller Area Network (CAN) for high-speed communication".

¹⁴ "Road Vehicles - Interchange of Digital Information - Controller Area Network (CAN) for low-speed communication".

CONCEPTOS BÁSICOS SOBRE EL BUS CAN (I)

- Se pueden agregar nodos a una red CAN sin necesidad de cambiar la configuración del resto de los nodos, ni de la capa de aplicación.

Esto es posible gracias a que los nodos no necesitan hacer uso de información relacionada con la configuración del sistema.

A continuación se describen algunos conceptos básicos sobre el bus CAN.

Se pueden agregar nodos a una red CAN sin necesidad de cambiar la configuración del resto de los nodos, ni de la capa de aplicación. Esto es posible gracias a que los nodos no necesitan hacer uso de información relacionada con la configuración de sistema.

CONCEPTOS BÁSICOS SOBRE EL BUS CAN (II)

- En la cabecera de los mensajes no aparece la dirección de un nodo destino, sino un identificador del contenido del mensaje.
- Cada nodo aceptará el mensaje sólo si le interesa su contenido.
- Un mismo mensaje puede ser recibido por uno o varios nodos a la vez (*multicast*) o por todos (*broadcast*).

CONCEPTOS BÁSICOS SOBRE EL BUS CAN (III)

- La consistencia del sistema está garantizada, ya que un mensaje si se recibe, lo habrán recibido todos los nodos interesados, y si no, no lo habrá recibido ninguno.

CONCEPTOS BÁSICOS SOBRE EL BUS CAN (IV)

- La prioridad de los mensajes viene determinada por su identificador, a menor valor de identificador, mayor prioridad.
- Cualquier nodo puede transmitir en cualquier momento si el bus está libre.
- Si colisionan dos o más, solo transmite el de mayor prioridad.

CONCEPTOS BÁSICOS SOBRE EL BUS CAN (V)

- Para conseguir seguridad en el sistema, cada nodo implementa mecanismos de detección y señalización de errores, así como de auto-chequeo.

Bus CAN © L.G. (UPV/EHU)

34

CONCEPTOS BÁSICOS SOBRE EL BUS CAN (VI)

- Se puede reducir el consumo de un sistema CAN pasando a modo *sleep* a aquellos nodos con actividad nula, y despertarlos (*wakeup*) cuando sea requerida su participación.

Bus CAN © L.G. (UPV/EHU)

35

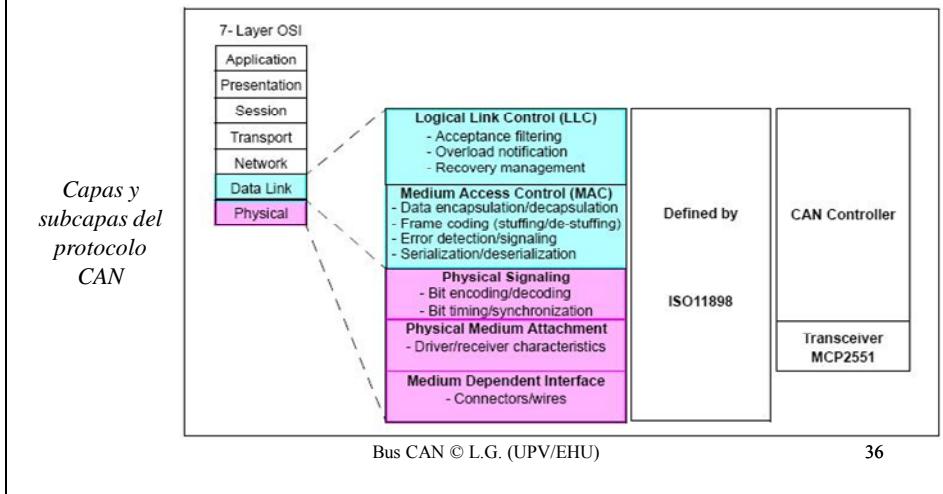
El protocolo CAN está optimizado para sistemas que necesitan transmitir y recibir cantidades de datos relativamente pequeñas (en comparación con Ethernet o USB, que están diseñados para mover bloques de datos mucho más grandes) y de forma fiable a todos los demás nodos de la red. CSMA / CD permite que cada nodo tenga una oportunidad igual para acceder al bus, así como un buen manejo de las colisiones.

Dado que el protocolo está basado en mensajes, y no está basado en la dirección, **todos los nodos en el bus reciben cada mensaje y reconocen cada uno de ellos, independientemente de la necesidad del dato o no**. Esto permite al bus operar de nodo a nodo o en formatos de mensajería *multicast* sin tener que enviar distintos tipos de mensajes.

La transmisión de mensajes rápida, robusta, con confinamiento de fallos es también una gran ventaja de CAN, porque los nodos defectuosos dejan el bus automáticamente sin permitir que un nodo tire la red. Esto garantiza que siempre está el ancho de banda disponible para la transmisión de los mensajes críticos.

EL BUS CAN Y LA ISO-11898

- En la especificación original de CAN no se definió la capa física,
- Esto permitió diferentes opciones de elección del medio y niveles eléctricos de transmisión.
- Las características eléctricas de las señales del bus se establecieron más tarde, en 1993, por el estándar ISO 11898.



1.5 EL BUS CAN Y LA ISO-11898

Muchos protocolos de red se describen utilizando las siete capas del modelo de interconexión de sistemas abiertos (OSI).

En la especificación original de CAN no se definió la capa física, lo que permitió diferentes opciones para la elección del medio y niveles eléctricos de transmisión. Las características de las señales eléctricas del bus se establecieron más tarde, en 1993, por el estándar ISO 11898. El resto de características de la capa física (y de todas las capas superiores) no están definidas por la especificación de CAN. Estas capas pueden ser definidas por el diseñador del sistema.

(ISO11898 es la norma internacional para comunicaciones CAN de alta velocidad en vehículos de carretera. ISO-11898-2 especifica las subcapas PMA¹⁵ y MDI¹⁶ de la Capa Física.)

¹⁵ PMA: Physical Medium Attachment

¹⁶ MDI: Medium Dependent Interface

La Capa FÍSICA del CAN es la responsable de la transferencia de bits entre los distintos nodos que componen la red.

Define aspectos tales como:

- Niveles de señal
- Codificación
- Sincronización
- Tiempos en que los bits se transfieren al bus

1.5.1 CAPA FISICA

Dentro de la Capa Física se distinguen tres subcapas:

- **Señalización Física** (Physical Signaling): define la codificación y decodificación de los valores a transmitir, el tiempo de bit y la sincronización a nivel de bit.
- **Conexión al Medio Físico** (Physical Medium Attachment): define las características técnicas de los transductores que se conectan al bus.
- **Interfaz Dependiente del Medio** (Medium Dependent Interface): define el tipo de medio físico que implementa el bus, la forma física y las características de los conectores.

La **Capa de ENLACE** se encarga:

- Del envío y recepción de tramas entre nodos garantizando que las tramas recibidas no tengan errores.
- Del control de flujo generando las tramas de sobrecarga correspondientes.

1.5.2 CAPA DE ENLACE

La capa de enlace define la especificación de CAN. Dentro de la capa de enlace se distinguen dos subcapas:

Control de Enlace Lógico (LLC: *Logical Link Control*): controla el flujo, filtra los mensajes que recibe el nodo y notificación de sobrecarga y funciones de recuperación para pasarlo al controlador del nodo sólo los que necesita

Control de Acceso al Medio (MAC: *Medium Access Control*): representa el kernel del protocolo CAN, codifica y decodifica la trama según la regla de *bit-stuffing*, así como de la detección y corrección de errores. Presenta los mensajes recibidos de la subcapa LLC y acepta mensajes para ser enviados a la subcapa LLC. Esta capa es responsable del formato de las tramas, la forma en la que los nodos acceden al bus, el ACK, la detección y corrección de errores.

Es decir, se encarga del envío y recepción de tramas entre nodos garantizando que las tramas recibidas estén libres de errores. También se encarga del control de flujo generando las tramas de sobrecarga correspondientes.

Dentro de la capa de Enlace se distinguen dos subcapas:

• **Control de Enlace Lógico**: controla el flujo y filtra los mensajes que recibe para pasárselos al controlador del nodo sólo los que necesita.

• **Control de Acceso a Medio**: codifica y decodifica la trama según la regla de *bit-stuffing*, así como de la detección y corrección de errores.

Bus CAN © L.G. (UPV/EHU)

40

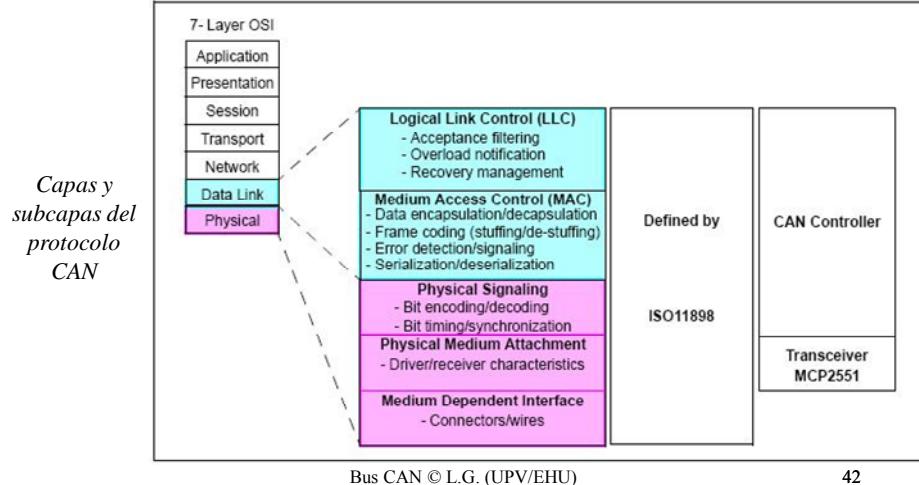
Capa de ENLACE define:

- La forma que los nodos acceden al bus.
- El formato de las tramas.

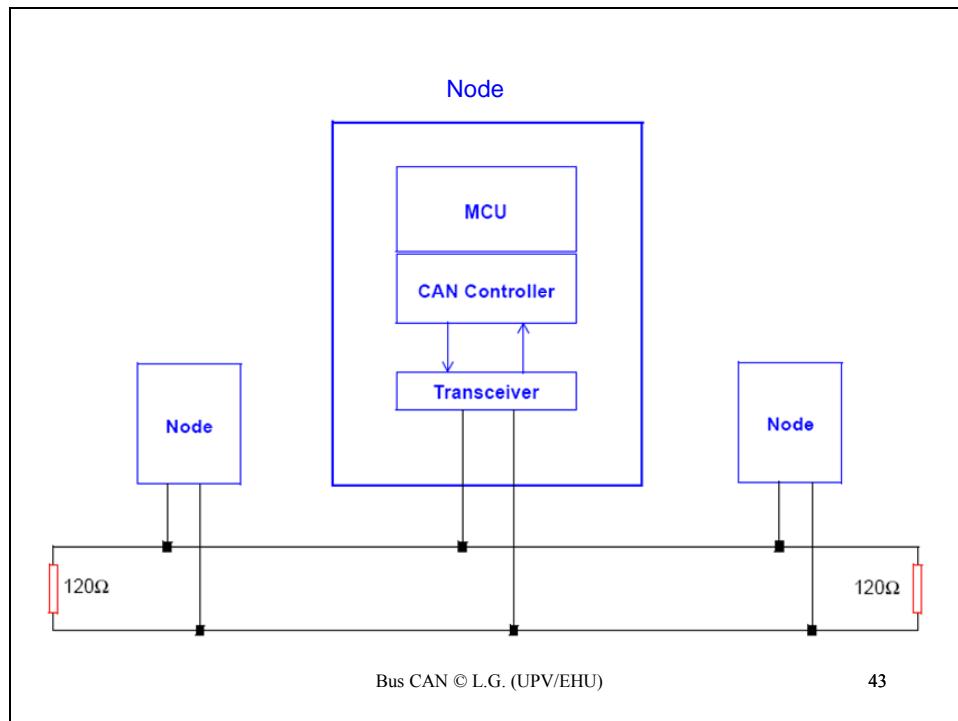
Bus CAN © L.G. (UPV/EHU)

41

- La capa de Enlace y la 1^a subcapa de la Capa Física (*Señalización Física*) están implementadas en el controlador CAN.
- La subcapa Conexión al Medio Físico está implementada en el transductor CAN.



Como se puede ver en la figura, la capa de enlace y la primera subcapa de la capa física (Señalización Física) están implementadas en el controlador CAN, mientras que la subcapa Conexión al Medio Físico está implementada en el transductor CAN. El diseñador del sistema puede elegir cualquier *driver/receiver* y el medio de transporte mientras se cumplan los requisitos de la capa física.



ARBITRAJE (I)

- Dos tramas pueden colisionar al intentar, sus correspondientes nodos, acceder al bus al mismo tiempo.
- El arbitraje de una red CAN está distribuido por todos los nodos.
- Cada nodo se encarga de comprobar si el bus está libre y si la trama que está enviando no colisiona con ninguna otra.

Bus CAN © L.G. (UPV/EHU)

45

1.5.3 ARBITRAJE

Dado que cualquier nodo puede transmitir de manera asíncrona y no existe un sistema de arbitraje por turnos, dos tramas pueden colisionar al intentar, sus correspondientes nodos, acceder al bus al mismo tiempo. El arbitraje de una red CAN está distribuido por todos los nodos. Cada nodo se encarga de comprobar si el bus está libre y de si la trama que está enviando no colisiona con ninguna otra. En el caso de que colisione con alguna, dependerá de la prioridad de cada trama para mantenerse en el bus.

ARBITRAJE (II)

- Cuando un nodo necesita enviar una trama, su controlador CAN comprueba si el bus está libre o no.
- Para asumir que el bus está libre, se deben detectar 11 bits recesivos consecutivos
- Si un nodo A coloca en el bus un **bit recesivo** y otro nodo B coloca uno **dominante**, es éste último el que prevalece en el bus.
- Cuando un nodo coloca 1 bit en el bus y su valor no corresponde con el que hay en el bus, en ese momento sabe que se ha producido una colisión y que ha perdido el arbitraje y se retira del bus.

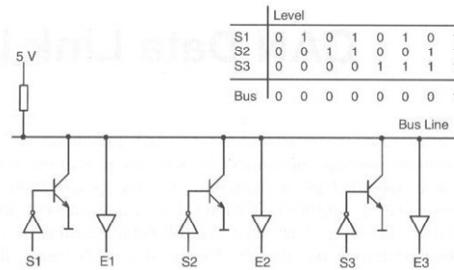
Bus CAN © L.G. (UPV/EHU)

46

Cuando el proceso de aplicación que se está ejecutando en el nodo necesita enviar una trama, su controlador CAN asociado, comprueba si el bus está libre o no. Para asumir que el bus está libre, se deben detectar 11 *bits* recesivos consecutivos, de los cuales 7 coinciden con los últimos *bits* de trama y 3 con los del espacio entre tramas. Si un nodo está esperando a enviar una trama lo hará después de estos últimos 3 *bits*.

ARBITRAJE (III)

- Si un nodo transmite sólo bits dominantes ganará a todos los demás nodos que en algún momento transmitan algún bit recesivo.
- Tomando un bit **recesivo** como un valor lógico 1 ó **5v** y un bit **dominante** como un valor lógico 0 ó **0v** (ISO 11898), el bus funciona como una *AND-Cableada*.
- Sólo si todos los bits que se envían al bus son recesivos, se verá este valor en el bus. En caso contrario el valor del bus será el de un bit dominante.
- El estado del bus cuando está libre es el de un bit recesivo.



47

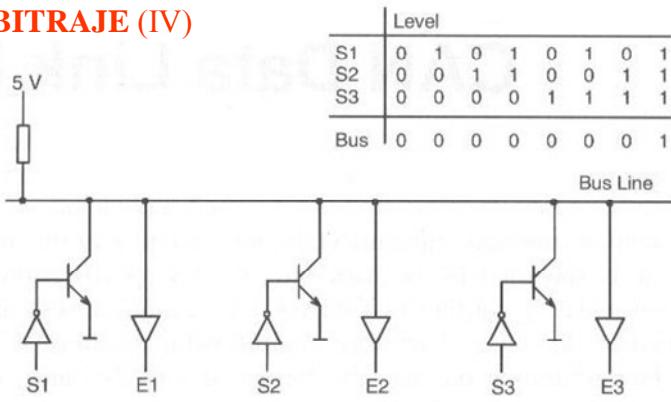
El sistema de prioridades garantiza que la trama de mayor prioridad puede seguir en el bus a pesar de haber colisionado con cualquier otra trama. Para conseguir el comportamiento deseado del sistema habrá que conceder mayor prioridad a las tramas más críticas y menor al resto.

Para garantizar el correcto arbitraje, sin la necesidad de comunicación entre los nodos, es necesario tener en cuenta una característica propia del bus. Si un nodo A coloca en el bus un *bit* recesivo y otro nodo B coloca uno dominante, es éste último, el que prevalece en el bus, de tal forma que cuando un nodo coloca un *bit* en el bus y su valor no corresponde con el que se encuentra en el bus, en ese momento sabe que se ha producido una colisión y que ha perdido el arbitraje y debe retirarse del bus. Se deduce fácilmente que si un nodo transmite sólo *bits* dominantes ganará a todos los demás nodos que en algún momento transmitan algún *bit* recesivo.

Tomando un *bit* recesivo como un valor lógico 1 ó 5v y un *bit* dominante como un valor lógico 0 ó 0v (ISO 11898), el bus funciona como una *AND-Cableada*. Sólo si todos los *bits* que se envían al bus son recesivos, se verá este valor en el bus. En caso contrario el valor del bus será el de un *bit* dominante.

El estado del bus cuando está libre es el de un *bit* recesivo.

ARBITRAJE (IV)



- A partir de este momento, ya sólo queda un nodo en el bus (CAN no permite dos identificadores de trama iguales)
- A este grupo de bits se le llama segmento de arbitraje.

Bus CAN © L.G. (UPV/EHU)

48

Cuando un nodo quiere comenzar a transmitir, transmite un *bit* dominante (*Start Of Frame*, SOF) para indicar dicha situación. Es en ese momento, cuando otro nodo podría estar haciendo lo mismo. Ambos han transmitido un *bit* dominante y ven en el bus un *bit* dominante por lo que ambos continúan adelante. Lo siguiente que deben enviar es el identificador de trama, empezando por el *bit* de mayor peso. En cuanto uno de los nodos envíe un *bit* recesivo y el otro uno dominante, el primero habrá perdido el arbitraje y será el otro quien continúe en el bus. Teniendo en cuenta los valores lógicos de cada uno de los *bits* se entiende que la trama con mayor prioridad será aquella con el menor número de identificador.

A partir de este momento, ya sólo queda un nodo en el bus (CAN no permite dos identificadores de trama iguales). A este grupo de *bits* se le llama Segmento de Arbitraje o Identificador.

Ejemplo del proceso de arbitraje en un bus CAN

Tenemos tres nodos 1, 2,y 3 con identificadores:

Nodo 1 → 65Fh = 110 0101 1111

Nodo 2 → 67Fh = 110 0111 1111

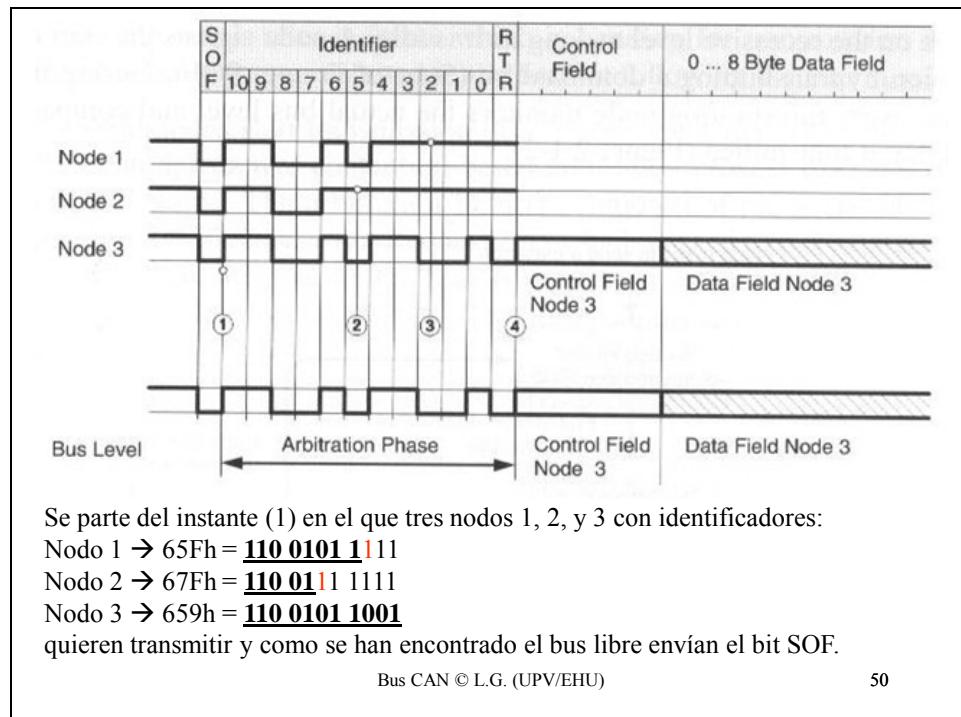
Nodo 3 → 659h = 110 0101 1001

- Quieren transmitir y
- Como se han encontrado el bus libre envían el bit SOF

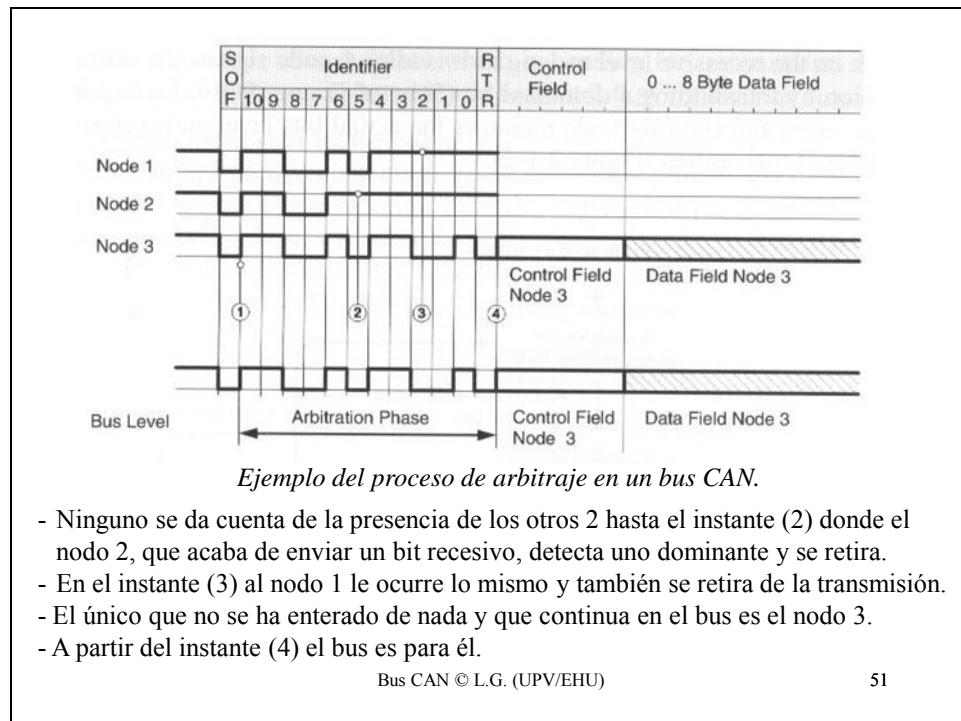
Bus CAN © L.G. (UPV/EHU)

49

A continuación se comenta el proceso de arbitraje del ejemplo que se muestra en la figura.



Se parte del instante (1) en el que tres nodos, (1, 2, y 3 con identificadores 65F, 67F, 659 respectivamente) quieren transmitir y como se han encontrado el bus libre envían simultáneamente el *bit* SOF.



Ninguno de los tres nodos se da cuenta de la presencia de los otros dos hasta el instante (2) donde el nodo 2 que acaba de enviar un *bit* recesivo detecta un *bit* dominante y se retira.

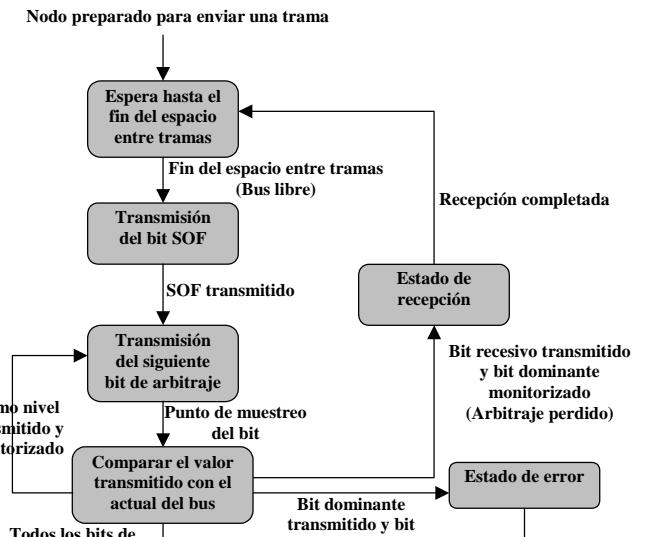
En el instante (3) al nodo 1 le ocurre lo mismo y también se retira de la transmisión. El único que no se ha enterado de nada y que continua en el bus es el nodo 3.

A partir del instante (4) todo el bus es para él.

En resumen, cuando un nodo quiere transmitir, debe esperar a que el bus esté libre. En cuanto el bus está libre, transmite cada uno de los *bits* de arbitraje mientras no detecte ninguna colisión. Si la detecta, considera que ha perdido y se retira de la transmisión, pasando al estado de recepción de la trama que le ha ganado el arbitraje. Si consigue ganar el arbitraje continua con la transmisión del resto de la trama.

RESUMEN

Diagrama de estados del arbitraje



Bus CAN © L.G. (UPV/EHU)

52

Formato de Tramas del bus CAN

El protocolo CAN distingue 4 tipos de tramas:

- **Datos**
- **Remota**
- **Error**
- **Sobrecarga**

Según el tamaño del identificador (11 ó 29 bits) se dividen en tramas **estándar** o **extendidas**.

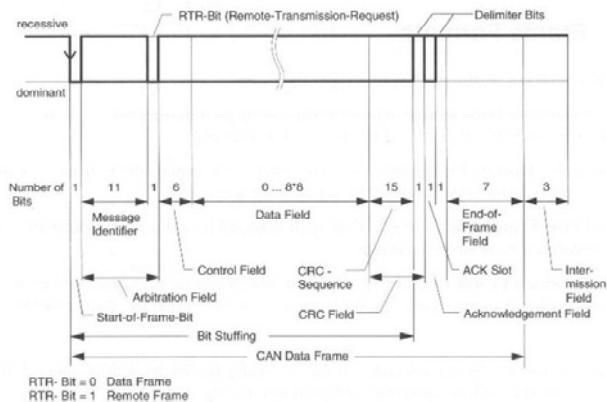
1.5.4 FORMATO DE LAS TRAMAS

El protocolo CAN distingue cuatro tipos de tramas diferentes: de datos, remotas, de error y de sobrecarga. Así mismo, según el tamaño del identificador, de 11 ó 29 bits, se dividen en tramas: estándar o básicas (11 bits) y extendidas (29 bits)

TRAMA de DATOS está dividida en 7 campos:

- Bit de Comienzo de Trama (*Start Of Frame*, SOF) (1 bit)
- Arbitraje (*Arbitration Field*) (11+1 # 29+1 bits)
- Control (*Control Field*) (6 bits)
- Datos (*Data Field*) -es opcional- (de 0 a 8 bytes)
- Código CRC (*CRC Field*) (15+1 bits)
- Bit de Reconocimiento (*Acknowledgement Field*) (1+1 bits)
- Fin de Trama (*End Of Frame Field*, EOF). (7+3 bits)

Formato de Trama de
Datos o Remota
Estándar



Tramas de Datos

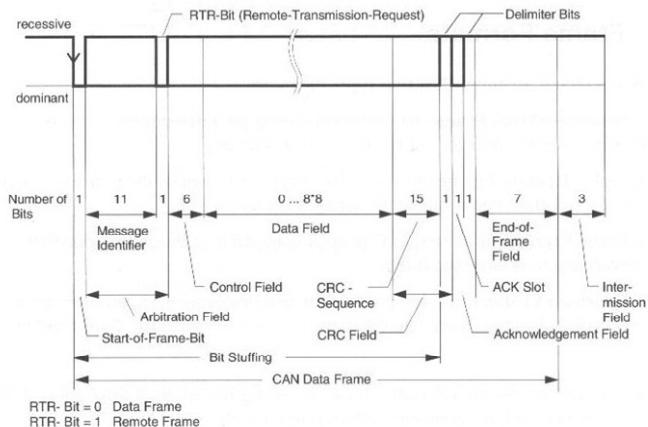
Cada trama de datos está dividida en los siguientes campos:

- *bit* de Comienzo de Trama (*Start Of Frame*, SOF)
- Arbitraje (*Arbitration Field*)
- Control (*Control Field*)
- Datos (*Data Field*) este campo es opcional.
- Código CRC (*CRC Field*)
- Reconocimiento (*Acknowledgement Field*)
- Fin de Trama (*End Of Frame Field*, EOF)

Bit de COMIENZO DE TRAMA SOF *Start Of Frame*

- Determina comienzo de trama de datos o remota mediante un bit dominante.
- Un nodo puede comenzar a transmitir sólo cuando encuentra el bus libre tras 11 bits recesivos consecutivos.
- Todos los nodos se sincronizan en el flanco de bajada de este bit.

Formato de trama de Datos o Remota Estándar



CAMPO DE ARBITRAJE (*Arbitration Field*)

- Lo forman el Identificador de Trama y el bit RTR (*Remote Transmission Requests*).
- El identificador está formado por 11 ó 29 bits, según la versión utilizada.
- Los identificadores más bajos tendrán mayor prioridad sobre los demás.
- El bit RTR indica si la trama es de datos o remota.
- CAN prioriza una trama de datos frente a una remota por lo que el valor lógico 0 (bit dominante) está asociado a una trama de datos y el valor 1 a la remota.

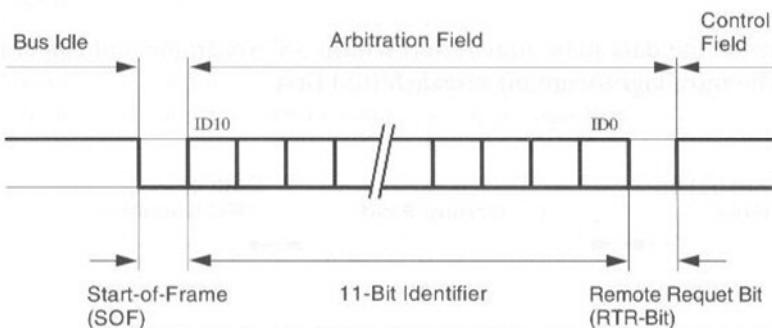


Fig.13: Formato del Campo de Arbitraje.

57

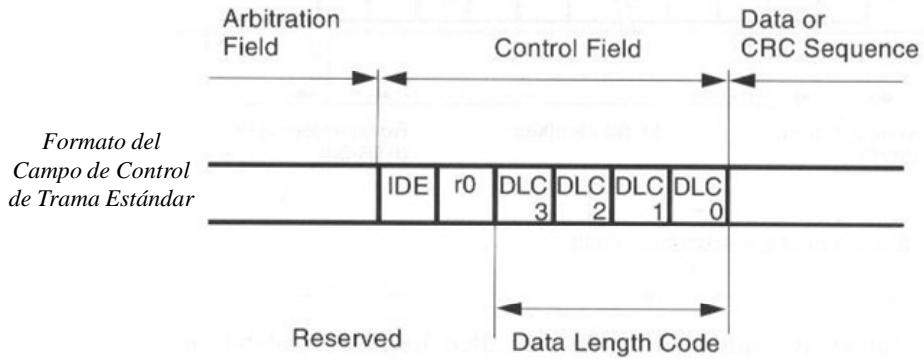
Campo de Arbitraje (*Arbitration Field*)

El campo de Arbitraje lo forman el Identificador de la Trama (*Message Identifier*) y el *bit RTR* (*Remote Transmission Requests*). El identificador está formado por 11 o 29 bits, según la versión que se esté utilizando. De esta forma podemos distinguir 2^{11} (2^{29}) tramas diferentes en la versión básica. **Los identificadores más bajos tendrán mayor prioridad** sobre los demás.

El *bit RTR* indica si la trama es de datos o remota. CAN prioriza una trama de datos frente a una remota por lo que el valor lógico 0 (bit dominante) está asociado a una trama de datos y el valor 1 (bit recesivo) a la remota.

CAMPO DE CONTROL formado por 6 bits:

- 1^{er} bit (IDE) distingue entre tramas estándar (11 bits) o extendidas (29 bits)
- Una trama estándar, transmite el bit IDE con nivel dominante:
→ ante una colisión de 2 tramas con mismo identificador, una estándar y otra extendida, gana el arbitraje la trama estándar.
- 2º (r0) está reservado para futuras extensiones del protocolo
- 4 bits: DLC3, DLC2, DLC1 y DLC0, indican nº datos (bytes) de la trama.



Bus CAN © L.G. (UPV/EHU)

58

Campo de Control (Control Field)

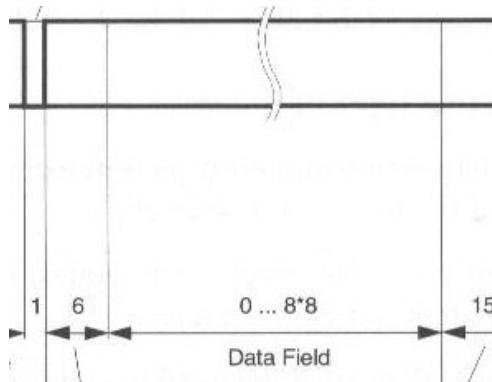
El campo de Control está formado por 6 bits:

- el primer bit (IDE) distingue entre tramas básicas o extendidas
- el segundo (r0) está reservado para futuras extensiones del protocolo
- los 4 bits restantes (DLC3, DLC2, DLC1 y DLC0) indican la longitud en bytes del campo de datos de la trama.

En una trama básica el bit IDE se transmite con un nivel dominante para que ante una colisión de dos tramas con el mismo identificador base, una básica y otra extendida, gane el arbitraje la trama básica.

CAMPO de DATOS:

- Contiene los datos que se transmiten en la trama de datos.
- Su longitud varía entre 0 y 8 bytes.



Bus CAN © L.G. (UPV/EHU)

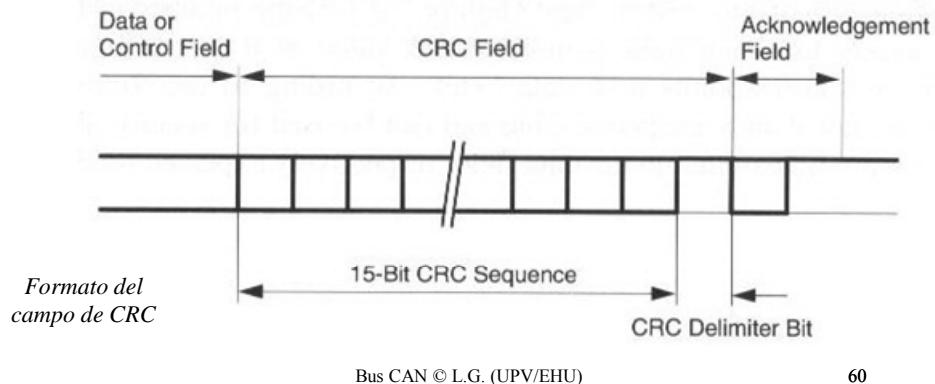
59

Campo de Datos (Data Field)

El campo de Datos contiene los datos que se transmiten en una trama de datos y su longitud varía entre 0 y 8 bytes.

CAMPO de CRC:

- Formado por un código de detección de errores de 15 bits y de 1 bit delimitador transmitido de forma recesiva.
- El nodo receptor de la trama puede detectar, mediante este código, si ha habido algún error durante la recepción de dicha trama.
- Para ello aplica el mismo algoritmo que el nodo emisor a los campos de arbitraje, control y datos, para compararlo con el código recibido.



Campo de Verificación de Redundancia Cíclica (*CRC Cyclic Redundancy Check*)

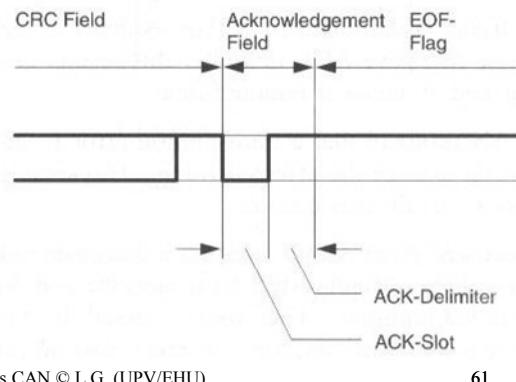
El campo de CRC está formado por un código de detección de errores de 15 bits y un bit delimitador transmitido de forma recesiva. El nodo receptor de la trama puede detectar, mediante este código, si ha habido algún error durante la recepción de dicha trama. Para ello aplica el mismo algoritmo que el nodo emisor a los campos de arbitraje, control y datos, para compararlo con el código recibido.

Bit de RECONOCIMIENTO (ACK):

Formado por 1 bit y por su delimitador recesivo.

- El transmisor envía 2 bits recesivos y **espera que algún nodo ponga a dominante el bit de reconocimiento.**
- Si esto ocurre, da por buena la transmisión de la trama, si no, considera que no se ha recibido y procede a su retransmisión.

Formato del Campo de Reconocimiento



Bus CAN © L.G. (UPV/EHU)

61

Campo de Reconocimiento (Ack: *Acknowledgement Field*)

Está formado por un *bit (ACK-Slot)* y por su delimitador recesivo (*ACK-Delimiter*). El transmisor envía dos *bits* recesivos y espera a que algún nodo ponga a dominante el *bit* de reconocimiento (*ACK-Slot*). Si esto ocurre, da por buena la transmisión de la trama, si no, procede a su retransmisión.

El nodo receptor tras recibir la trama y comprobar con el código de CRC es correcto, pone el bus en un nivel dominante en el lugar del *bit* de reconocimiento. Si varios nodos receptores sobrescriben el *bit* de reconocimiento, el nodo transmisor no es capaz de detectar el número de nodos que han recibido correctamente la trama. Por lo tanto, este mecanismo no es suficiente para garantizar el correcto funcionamiento del sistema y debe ser cada nodo quien, cuando detecte algún error, señale dicho error con una trama de error adecuada.

Por ejemplo: un nodo puede recibir un reconocimiento positivo de la trama enviada y a continuación recibir una trama de error indicando que debe retransmitir dicha trama. Esto es debido a que por lo menos uno de los receptores ha recibido correctamente la trama y por lo menos uno también no la ha recibido bien, por lo cual ha generado esa trama de error.

Después del *bit* de reconocimiento se encuentra el delimitador con un nivel recesivo.

FIN DE TRAMA

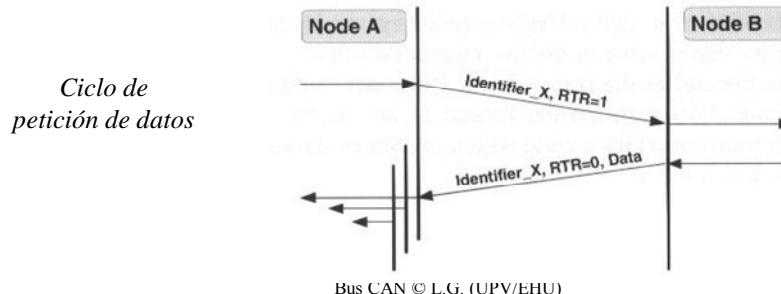
- Cada trama de datos o remota está delimitada por 7 bits recessivos consecutivos.
- Para evitar que un nodo no sincronizado con la trama actual pueda interpretar el bit de reconocimiento como un SOF e intentar sincronizarse con él, se fuerza a que el nodo detecte un error de *bit-stuffing* transmitiendo para ello por lo menos 6 bits consecutivos con el mismo nivel.
- Para permitir señalizar un posible error en el último bit de la trama se necesita un bit adicional y por este motivo el Fin de Trama está formado por 7 bits.

Fin de Trama

Cada trama, de datos o remota, está delimitada por 7 *bits* recessivos consecutivos. Para evitar que un nodo no sincronizado con la trama actual pueda interpretar el *bit* de reconocimiento como un SOF e intentar sincronizarse con él, se fuerza a que el nodo detecte un error de *bit-stuffing* transmitiendo para ello por lo menos 6 *bits* consecutivos con el mismo nivel. Para permitir señalizar un posible error en el último *bit* de la trama se necesita un *bit* adicional y por este motivo el Fin de Trama está formado por 7 *bits*.

TRAMA REMOTA

- Un nodo puede pedir una trama concreta transmitiendo por el bus una trama remota con el mismo identificador de la que desea.
- La trama puede ser recibida por cualquier nodo que la quiera, pero sólo uno se debe responsabilizar de iniciar la transmisión.
- Una trama remota está formada por los mismos campos que una trama de datos pero con la salvedad de que el bit RTR ahora tiene un nivel recesivo y que el campo de datos está vacío.
- Debe marcar el número de datos deseados.



63

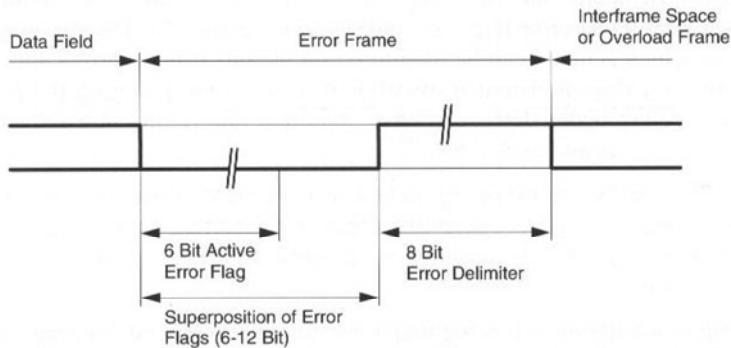
1.5.5 TRAMA REMOTA

Un nodo puede pedir una trama concreta transmitiendo, por el bus, una trama remota con el mismo identificador de la trama que desea. La trama puede ser recibida por cualquier nodo que la quiera, pero sólo uno se debe responsabilizar de iniciar la transmisión.

Una trama remota está formada por los mismos campos que una trama de datos pero con la salvedad de que el *bit* RTR ahora tiene un nivel recesivo y que el campo de datos está vacío. Eso sí, la longitud que debe marcar es la de los datos deseados.

TRAMA DE ERROR

- El nodo que detecta un error lo señala mediante una trama de error que incumple la regla de *bit-stuffing*, obligando al nodo emisor a retransmitir la trama.
- Una trama de error acaba en una secuencia de 8 bits recessivos.



Formato de una trama de error activa.

Bus CAN © L.G. (UPV/EHU)

65

1.5.6 TRAMA DE ERROR

Si durante la transmisión o recepción de una trama se produce un error, el nodo que detecta el error lo señala mediante una trama de error que incumple la regla de *bit-stuffing*, obligando, de esta forma, al nodo emisor a la retransmisión de la trama. También puede ocurrir que tanto las propias tramas de error como con las de sobrecarga tengan que volver a retransmitirse.

Una trama de error está compuesta por dos campos: ‘el flag’ y ‘el limitador’. Cada nodo que detecta el error envía al bus 6 *bits* dominantes incumpliendo la regla del *bit-stuffing*. Pero, cada nodo puede detectar el error en diferente momento y enviar los *bits* sin sincronizar con los demás nodos. Es decir, cada una de las secuencias de 6 *bits* se puede solapar con las demás en algún punto de esos 6 *bits*. En el sexto *bit* de la secuencia enviada por el primer nodo salta el error de *bit-stuffing* y todos los nodos del sistema se percatan del error. Por este motivo, el flag está formado por 6 a 12 *bits* dominantes, dependiendo de si todos los nodos que detectan el error lo hacen en el mismo instante o no.

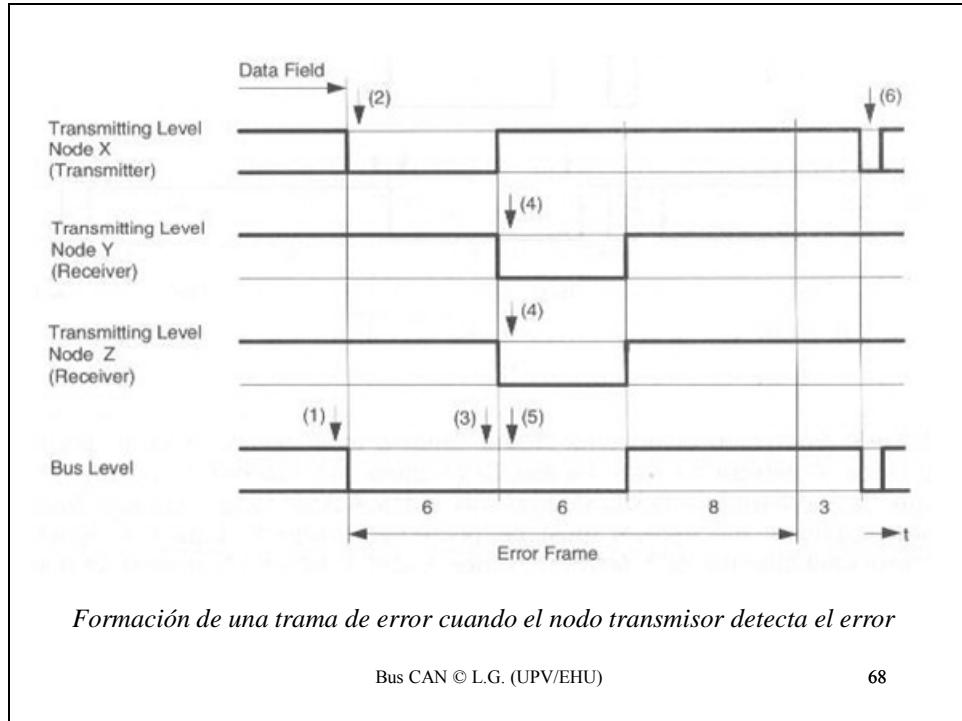
- Cada nodo puede detectar el error en diferente momento y enviar los bits de error sin sincronizar con los demás nodos.
- Es decir, cada una de las secuencias de 6 bits se puede solapar con las demás en algún punto de esos 6 bits.
- En el 6º bit de la secuencia enviada por el primer nodo salta el error de *bit-stuffing* y todos los nodos del sistema se percatan del error.
- Por este motivo, el flag está formado por 6 a 12 bits dominantes, dependiendo de si todos los nodos que detectan el error lo hacen en el mismo instante o no.

- Una trama de error inhabilita a la trama que se está transmitiendo de tal forma que tiene que ser retransmitida.
- No todos los nodos pueden generar este tipo de tramas.
- Si uno de los nodos del sistema no funciona correctamente y detecta errores donde no los hay puede corromper el correcto funcionamiento del sistema.
- La decisión de si un nodo puede o no generar tramas de error viene dada de forma automática en cada nodo.

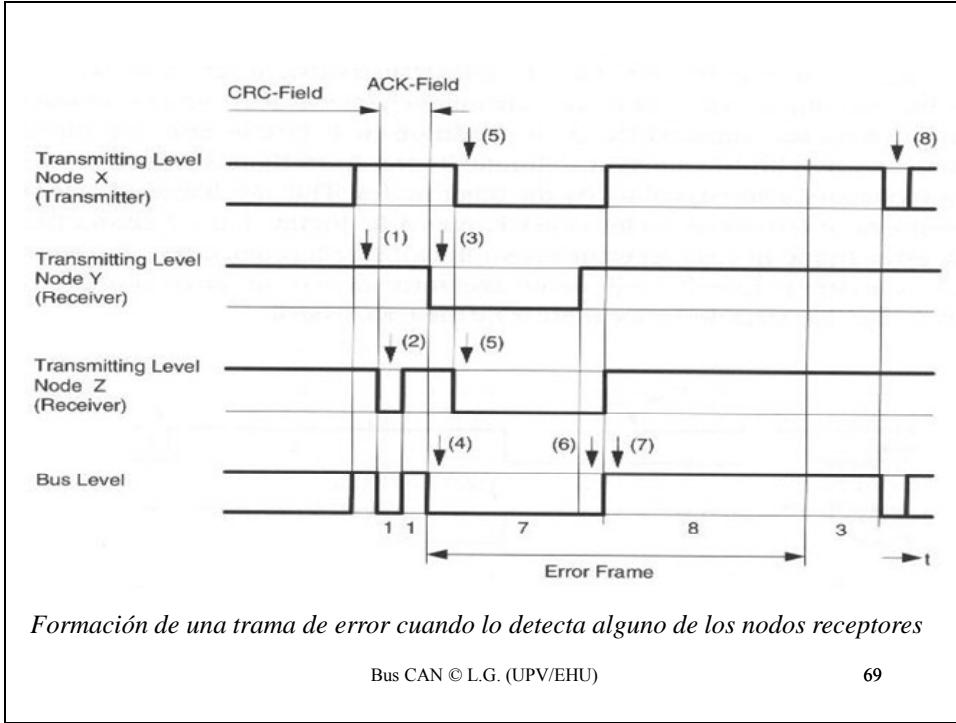
Una trama de error inhabilita a la trama que se está transmitiendo de tal forma que tiene que ser retransmitida. Este es el principal motivo, como se verá más adelante, de que no todos los nodos puedan generar este tipo de tramas. Si uno de los nodos del sistema no funciona correctamente y detecta errores donde no los hay puede corromper el correcto funcionamiento del sistema. La decisión de si un nodo puede o no generar tramas de error viene dada de forma automática en cada nodo. Cada nodo se encarga de pasar de un estado en el que puede enviar tramas de error a otro en el que no, según el número de errores que detecta.

No obstante, estos nodos pueden seguir generando tramas de error (tramas de error pasivas), pero en esta ocasión, formadas por 6 bits recesivos consecutivos, de forma que inhabiliten a las tramas que generen los demás nodos.

Una trama de error acaba en una secuencia de 8 *bits* recesivos. Cada uno de los nodos, después de generar el flag de error, transmite *bits* recesivos mientras monitoriza el bus hasta que detecta un nivel recesivo en el bus. En ese momento transmite los 7 últimos *bits* recesivos de la trama de error. Sólo el primer nodo en detectar el error monitoriza, en el bus, un nivel dominante cuando intenta transmitir el limitador de la trama. Este nivel dominante es el generado por el resto de los nodos cuando detectan el error. De esta forma un nodo sabe si ha sido el primero en detectar el error o no.



En la figura se muestra un ejemplo cuando el nodo transmisor es quien detecta el error. Durante la transmisión del campo de datos de la trama (1) el nodo transmisor detecta un error de *bit*, es decir, el valor del bus no corresponde con el que acaba de transmitir. En ese momento deja de transmitir los datos y la trama en sí para generar la secuencia de *bits* dominantes (2) que señalan el error. Los nodos receptores no se dan cuenta de esta situación hasta que el sexto *bit* dominante generado por el nodo transmisor viola la regla de *bit-stuffing* (3). En ese momento los nodos receptores no saben si el error de *bit-stuffing* se produjo de forma accidental o si por el contrario corresponde a la señalización de otro error ocurrido. Pero ellos, como nodos CAN, proceden a generar los correspondientes 6 *bits* dominantes (4). En este caso el flag de error de esta trama de error está compuesta por 12 *bit* dominantes, 6 del nodo transmisor y otros 6 de los nodos receptores. En el mismo instante en el que los nodos detectan el error de *bit-stuffing* y comienzan a transmitir el flag de error, el nodo transmisor ya ha acabado de enviar el suyo y procede a enviar el limitador de la trama de error (5). Simultáneamente, monitoriza el nivel del bus y detecta un nivel dominante creado por los flags de los demás nodos. De esta forma sabe dos cosas: que ha sido el primero en generar el error y que los demás nodos involucrados en la comunicación se han enterado. Despues de los 8 *bits* recesivos enviados, ahora sí, por todos los nodos conjuntamente, y despues de los 3 del espacio entre tramas, el nodo transmisor vuelve a comenzar la transmisión de la trama (6) y competirá de nuevo por el bus con las demás tramas que estén esperando a transmitirse.



Ahora bien, no sólo el nodo transmisor puede detectar errores, sino que también lo puede hacer cualquiera de los nodos receptores, pero en esta ocasión, como es obvio, no detectan los errores de *bit*.

En el siguiente ejemplo, se puede ver cómo se forma una trama de error cuando uno de los nodos receptores es el que detecta algún tipo de error en la comunicación.

En este caso el nodo ‘Y’ detecta un error de CRC (1), mientras que el nodo ‘Z’ activa el *bit* de reconocimiento (2) porque él sí ha recibido correctamente la trama. El nodo ‘Y’ respeta el campo de reconocimiento antes de señalizar el error (3). En el instante (4) los nodos ‘X’ y ‘Z’ detectan un error, ya que esperaban un nivel recesivo correspondiente al campo de fin de trama y en cambio detectan un nivel dominante introducido por el nodo ‘Y’ para señalizar el error detectado. En el siguiente *bit* (5) estos nodos, X y Z, señalan este nuevo error introducido a posta por el nodo ‘Y’. Cuando el nodo ‘Y’ acaba de generar el flag de error y comienza con el limitador sabe que ha sido el primero en detectar el error. Después de los 7 *bits* del limitador y de los 3 del espacio entre tramas comienza de nuevo el proceso de arbitraje con todas las tramas que quieran ocupar el bus.

En el ejemplo anterior, el tiempo que transcurre entre que se detecta el error y comienza la retransmisión de la trama es de 23 *bits* (6+6+8+3), mientras que en el ejemplo actual, el tiempo es de 18 *bits*. Este valor se conoce como Tiempo de Recuperación del Error. Si comparamos este tiempo con otros sistemas de comunicación o incluso con otros buses de campo, vemos que es sustancialmente menor. Con este mecanismo también se mejora el rendimiento del sistema en situaciones críticas. Una trama errónea no supone un cuello de botella para el resto del sistema, ya que debe entrar en competición por el bus con otras tramas, permitiendo en este caso que tramas con mayor prioridad que ella se le adelanten para garantizar el correcto funcionamiento del sistema.

ERRORES

- 5 condiciones de error definidas en CAN
(no son mutuamente excluyentes)
- 3 estados de error del nodo

1.6 ERRORES

Hay **cinco condiciones de error** definidas en el protocolo CAN (que no son mutuamente excluyentes) y **tres estados de error** en los que un nodo puede encontrarse, dependiendo del tipo y número de condiciones de error detectados.

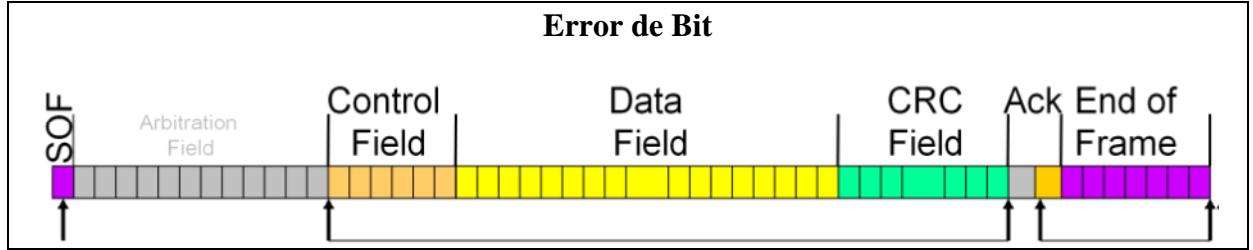
Errores Detectables

Error de BIT

- Se produce cuando un transmisor envía un bit dominante y detecta un bit recesivo.
- Si envía un bit recesivo y detecta un bit dominante cuando monitoriza el nivel del bus.

1.6.1 ERROR DE BIT

Un error de *bit* se produce cuando un transmisor envía un *bit* dominante y detecta un *bit* recesivo, o si se envía un *bit* recesivo y detecta un *bit* dominante cuando monitoriza el nivel del bus y lo compara con el valor que acaba de enviar. Una excepción es el caso en el que el transmisor envía un *bit* recesivo y detecta un *bit* dominante durante el campo de Arbitraje o de ACK.



Error de BIT de RELLENO (Bit Stuff)

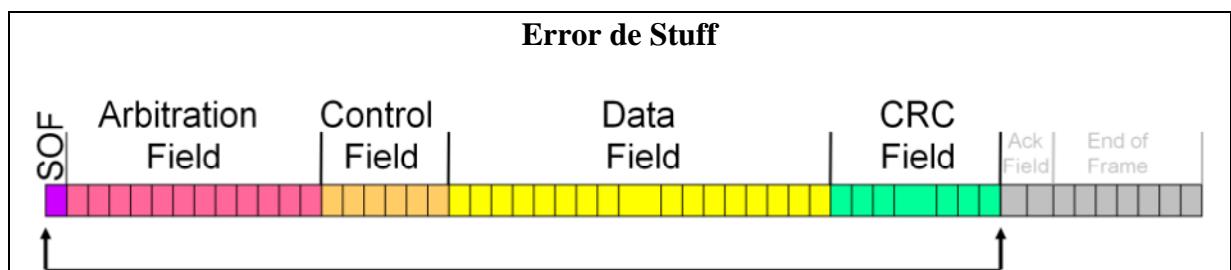
- Más de 5 bits consecutivos con igual polaridad, CAN añade un bit de polaridad opuesta en el flujo de datos.
- Los nodos receptores lo utilizan para la sincronización.
- Si, entre Inicio de Trama y el CRC Delimitador, se detectan 6 bits consecutivos con la misma polaridad, se ha violado la regla de relleno de bits.

Bus CAN © L.G. (UPV/EHU)

73

1.6.2 ERROR DE BIT DE RELLENO (STUFF)

El protocolo CAN utiliza un método de transmisión de no retorno a cero (NRZ: *Nonreturn to Zero*). Esto significa que el nivel de *bit* se coloca en el bus para todo el tiempo del *bit*. CAN es también asincrónico, y se utiliza un *bit* de relleno para permitir a los nodos receptores sincronizarse, recuperando información del reloj desde el flujo de datos. Si hay más de cinco *bits* consecutivos con la misma polaridad, CAN añade, automáticamente, un *bit* de polaridad opuesta en el flujo de datos. Los nodos receptores lo utilizan para la sincronización, pero estos *bits* de relleno se ignoran para los propósitos de los datos. Si, entre el Inicio de Trama y el CRC Delimitador, se detectan seis *bits* consecutivos con la misma polaridad, se ha violado la regla de relleno de *bits*.



Error CRC

- Se calcula un valor de 15 bits de Control de Redundancia Cíclica (CRC) por el nodo transmisor, y se envía en el campo CRC.
- Los nodos reciben el mensaje, calculan el CRC y comprueban que el valor coincide con el recibido.
- Si el valor no coincide, se genera un Error de CRC y una Trama de error.
- Si al menos un nodo no recibe el mensaje correctamente, éste se reenviará después del correspondiente tiempo entre tramas (intermisión).

Bus CAN © L.G. (UPV/EHU)

75

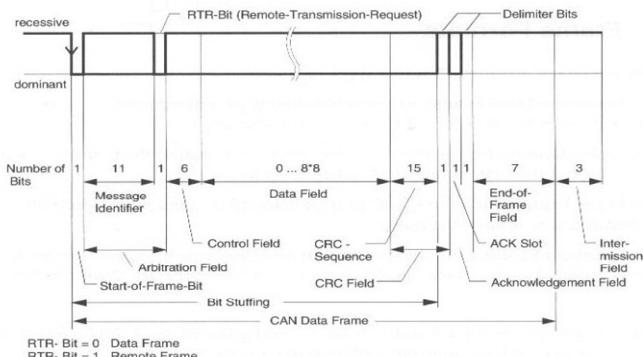
1.6.3 ERROR CRC

Se calcula un valor de 15 bits de control de redundancia cíclica (CRC: *Cyclic Redundancy Check*) por el nodo transmisor, y se envía en el campo CRC. Todos los nodos de la red reciben este mensaje, calculan el CRC y comprueban que el valor coincide con el recibido. Si el valor no coincide, se genera un Error de CRC y una Trama de error. Si al menos un nodo no recibe el mensaje correctamente, éste se reenviará después del correspondiente tiempo entre tramas (intermisión).

Error de FORMA

- Si cualquier nodo detecta un bit dominante en:
 - Fin de Trama (7bits)
 - Espacio entre tramas (3bits)
 - Delimitador de ACK (1bit)
 - Delimitador de CRC (1bit)

CAN lo define como violación de forma y genera un Error de Forma

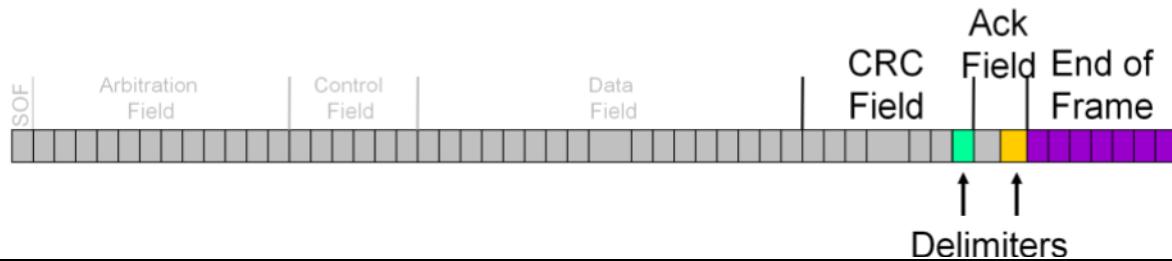


76

1.6.4 ERROR DE FORMA

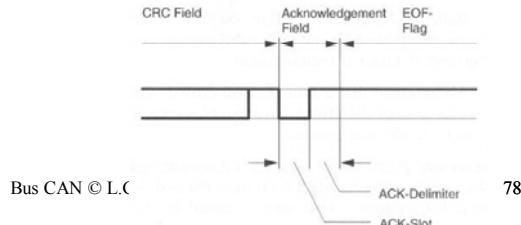
Si cualquier nodo detecta un bit dominante en uno de los siguientes cuatro segmentos del mensaje: Fin de Trama, Espacio entre tramas, Delimitador de ACK, Delimitador de CRC, el protocolo CAN lo define como una violación de forma y se genera un error de forma. Hay que tener en cuenta que para un receptor, un bit dominante durante el último bit del Fin de Trama no se trata como error de Forma.

Error de Forma



Error ACK

- En el campo ACK el emisor comprueba si el bit de ACK (que ha enviado como un bit recesivo) contiene un bit dominante.
- Este bit dominante indica que al menos un nodo ha recibido correctamente el mensaje.
- Si este bit se mantiene recesivo, indica que ningún nodo ha recibido el mensaje correctamente.
- Se genera una trama de error y el mensaje original se repetirá después del adecuado tiempo entre tramas (*intermisión*).



1.6.5 ERROR ACK

En el campo ACK de un mensaje, el nodo transmisor comprueba si el *bit* de ACK (que ha enviado como un *bit* recesivo) contiene un *bit* dominante. Este *bit* dominante indica que al menos un nodo ha recibido correctamente el mensaje. Si este *bit* se mantiene recesivo, indica que ningún nodo ha recibido el mensaje correctamente. En este caso, se genera una trama de error y el mensaje original se repetirá después del adecuado tiempo entre tramas (intermisión).

ESTADOS de ERROR del NODO (Confinamiento de fallos)

- Los errores detectados son públicos a los otros nodos a través de Tramas de Error
- Un nodo está en uno de 3 estados de error:
 - **Error-Activo** (\approx sin errores)
 - **Error-Pasivo** (\approx con pocos errores)
 - **Bus-Off** (\approx con muchos errores)

1.6.6 ESTADOS DE ERROR DEL NODO (CONFINAMIENTO DE FALLOS)

Los errores detectados se hacen públicos a todos los otros nodos a través de Tramas de Error o flags de error. La transmisión de un mensaje erróneo se aborta y la trama se repite en cuanto el mensaje gana el arbitraje en la red. Además, cada nodo se encuentra en uno de los tres estados de error siguientes: Error-Activo, Error-Pasivo, Bus-Off.

Un nodo que detecta una condición de error, lo indica enviando un *flag* de error. Para un nodo en modo "Error Activo" es un *flag* de error activo, para un nodo en "Error Pasivo" es un *flag* de error pasivo. Cada vez que cualquiera de los nodos detecta un *bit* de error, un error *stuff*, un error de forma o un error de reconocimiento, el nodo correspondiente inicia en el siguiente *bit* la transmisión de un indicador de error.

Cada vez que se detecta un error de CRC, la transmisión del flag de error se inicia en el *bit* siguiente al Delimitador ACK, a menos que ya se halla iniciado otro indicador de error.

Error-Activo

- Es el modo de funcionamiento normal, permitiendo que el nodo transmita y reciba sin restricciones.
- Toma parte activa en el bus de comunicación, incluyendo el envío de un flag de error activo, que consta de 6 bits dominantes.
- El flag de error viola activamente la regla de relleno de bits y provoca que los demás nodos envíen en respuesta un flag de error, llamado flag Eco.
- Un flag de error Activo, y el posterior error de flag Eco puede causar hasta 12 bits consecutivos dominantes en el bus: 6 del flag de error activo, y de cero a seis más del flag de error de Eco, dependiendo de cuando detecte cada nodo el error en el bus.
- Un nodo está en Error-Activo cuando:
 - el Contador de Errores de Transmisión < 128
 - y el Contador de Errores de Recepción < 128

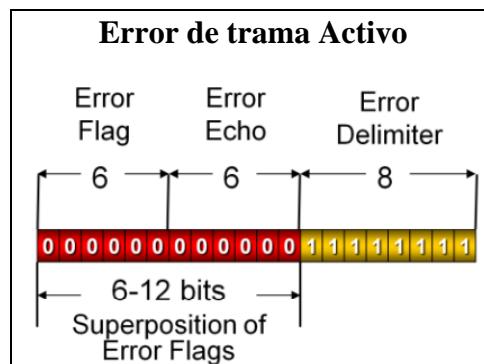
Bus CAN © L.G. (UPV/EHU)

80

Error-Activo

Un nodo está en Error-Activo cuando el Contador de Errores de Transmisión (TEC: *Transmit Error Counter*) y el Contador de Errores de Recepción (REC: *Receive Error Counter*) están, ambos, por debajo de 128. Error-Activo es el modo de funcionamiento normal, permitiendo que el nodo transmita y reciba sin restricciones.

Un nodo en Error-Activo puede tomar parte activa en el bus de comunicación, incluyendo el envío de un flag de error activo, que consta de seis *bits* dominantes. El flag de error viola activamente la regla de relleno de bits y provoca que los demás nodos envíen, en respuesta, un flag de error, llamado flag Eco. Un flag de error Activo, y el posterior error de flag Eco puede causar hasta doce bits consecutivos dominantes en el bus, seis del flag de error activo, y de cero a seis más del flag de error de Eco, dependiendo de cuando detecta cada nodo un error en el bus.

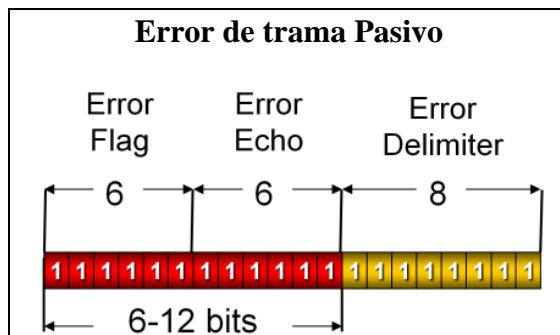


Error-Pasivo

- Un nodo pasa a Error-Pasivo cuando:
 - el Contador de Errores de Transmisión es > 127
 - ó el Contador de Errores de Recepción es > 127
- Los nodos en Error-Pasivo transmiten flags de Error-Pasivo que se componen de 6 bits recesivos
- Si el nodo en Error-Pasivo es el único transmisor que actualmente está en el bus entonces, el flag de error-pasivo violará la regla de relleno de bits y los nodos receptores responderán con un error de sus propios flags
- Cuando un nodo transmite un flag de error-pasivo y detecta un bit dominante, debe ver el bus inactivo durante 8 bits después de un *intermission* antes de reconocer el bus como disponible. Después de este tiempo, intentará retransmitir.

Error-Pasivo

Un nodo pasa a Error-Pasivo cuando el valor del Contador de Errores de Transmisión ó el Contador de Errores de Recepción es superior a 127. Los nodos en Error-Pasivo no están autorizados a transmitir flags de error-activo en el bus, pero en su lugar, transmiten flags de Error-Pasivo que se componen de seis bits recesivos. Si el nodo en Error-Pasivo es el único transmisor que actualmente está en el bus entonces, el flag de error-pasivo violará la regla de relleno de bits y los nodos receptores responderán con un error de sus propios flags (ya sea activo o pasivo en función de sus propios estados de error). Si el nodo en Error-pasivo en cuestión no es el único transmisor (por ejemplo, durante el arbitraje), o es un receptor, entonces, el flag de error-pasivo no tendrá ningún efecto en el bus debido a la naturaleza recesiva del flag de error. Cuando un nodo en error-pasivo transmite un flag de error-pasivo y detecta un *bit* dominante, debe ver el bus inactivo durante ocho bits después de un *intermission* antes de reconocer el bus como disponible. Después de este tiempo, intentará retransmitir.



Bus-OFF

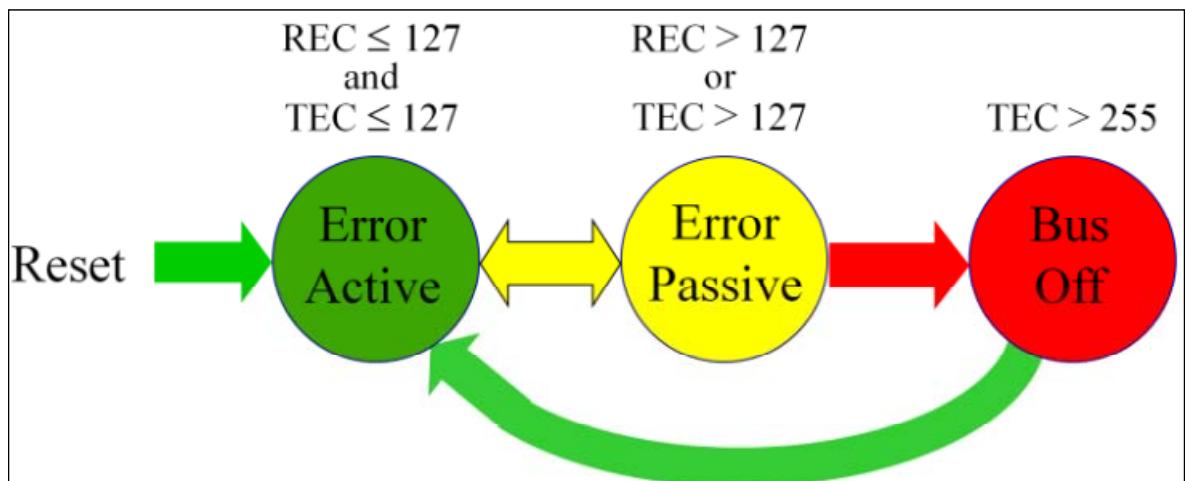
- Un nodo entra en Estado de Bus-Off cuando el Contador de Transmisión de Errores es > 255
- Recibir errores no causa el paso a Bus-Off
- En este modo, el nodo no puede enviar o recibir mensajes, reconocer mensajes, o transmitir tramas de error de cualquier tipo.
- **Así es como se consigue el confinamiento de fallos.**
- Hay una secuencia de recuperación de bus que permite a un nodo que está en bus-Off, vuelva a error-activo, y comience a transmitir de nuevo, si la condición de fallo se ha eliminado.

Bus CAN © L.G. (UPV/EHU)

84

Bus-OFF

Un nodo entra en Estado de Bus-Off cuando el Contador de Transmisión de Errores es mayor de 255 (recibir errores no causa el paso de un nodo a Bus-Off). En este modo, el nodo no puede enviar o recibir mensajes, reconocer mensajes, o transmitir tramas de error de cualquier tipo. Así es como se consigue el confinamiento de fallos. Hay una secuencia de recuperación de bus definida por el protocolo CAN que permite a un nodo que está en bus-Off, vuelva a error-activo, y comience a transmitir de nuevo, si la condición de fallo se ha eliminado.



TRAMA de SOBRECARGA

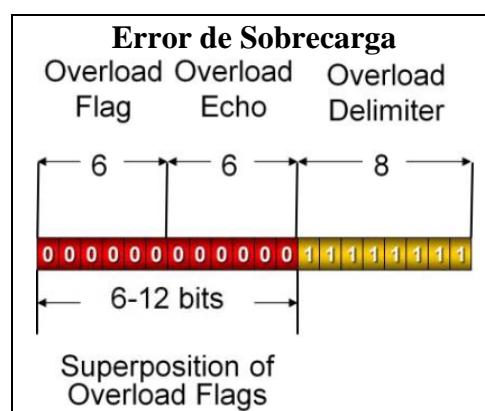
- Se utiliza para retrasar la siguiente trama de datos o remota, y cuando se detecta algún error en el espacio entre tramas.
- Se puede considerar un caso especial de trama de error que no implica la retransmisión de la trama afectada.

Bus CAN © L.G. (UPV/EHU)

86

Trama de Sobrecarga

Se utiliza para retrasar la siguiente trama de datos o remota, y cuando se detecta algún error en el espacio entre tramas. Una trama de sobrecarga se puede considerar un caso especial de trama de error que no implica la retransmisión de la trama afectada. La generación de tramas de sobrecarga está limitada al espacio entre tramas, ya que es ésta, la única característica que la diferencia de una trama de error. Por ello, debe comenzar donde el primer *bit*.



ESPACIO ENTRE TRAMAS (IFS *Intermission*)

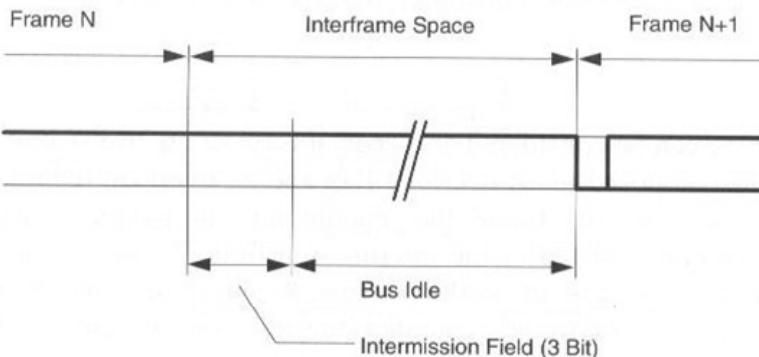
- Tramas de Datos y Remotas están separadas de las demás por el *Espacio entre Tramas*
- Está formado por 3 bits recesivos.
- Tramas de error y de sobrecarga, se transmiten sin este espacio.
- Despues del *Espacio entre Tramas* hay un número variable de bits recesivos, en el que se considera que el bus está libre.

Bus CAN © L.G. (UPV/EHU)

87

Espacio entre Tramas (IFS *Intermisión*)

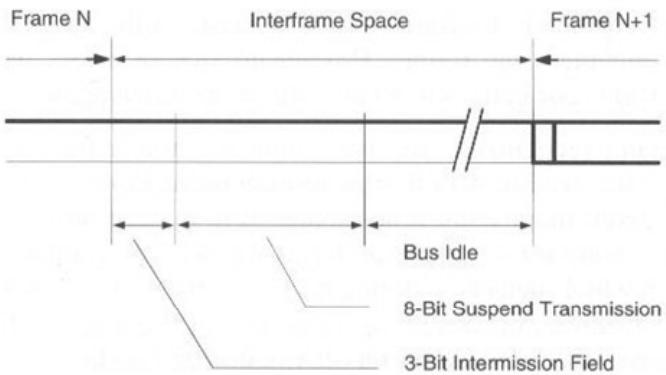
Las tramas de datos y las remotas están separadas de todas las demás por el Espacio entre Tramas, que está formado por 3 bits recesivos. Las tramas de error y de sobrecarga, como se ha visto anteriormente, se transmiten sin este espacio. Despues del Espacio entre Tramas hay un número variable de bits recesivos, en el que se considera que el bus está libre, que depende de cuando se transmite la siguiente trama.



Formato del *Espacio entre Tramas* para un nodo que esté en el estado “*error activo*”.

Bus CAN © L.G. (UPV/EHU)

88



Formato del espacio entre tramas para nodo en “error pasivo”

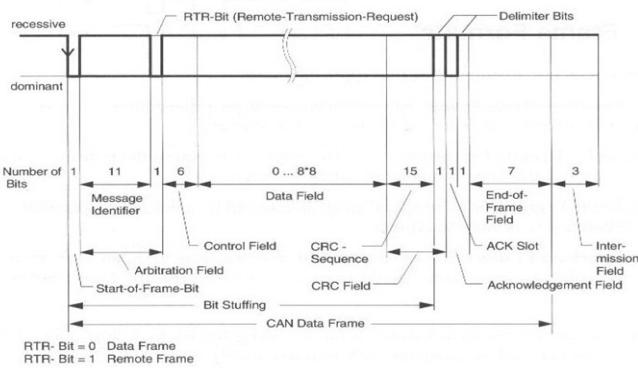
Bus CAN © L.G. (UPV/EHU)

89

En la figura 22 se muestra el formato del Espacio entre Tramas para un nodo que esté en el estado ‘Error Activo’. Es decir, aquellos nodos que pueden generar tramas de error activas por no haber superado el límite de errores permitidos. Estos nodos tienen prioridad a la hora de transmitir una trama, ya que los nodos que estén en estado ‘Error Pasivo’ después del espacio entre tramas se les impone 8 bits de suspensión de transmisión, donde como su nombre indica, no pueden comenzar a transmitir (competir por el bus) aunque tuvieran una trama en espera de ser enviada. En este intervalo se les puede adelantar cualquier otro nodo en ‘Error Activo’ .

VALIDACIÓN de TRAMAS

- El momento en el que se valida la trama es diferente para un nodo emisor y uno receptor.
- Para el emisor una trama será válida si no detecta ningún error hasta el último bit del delimitador de trama, mientras que para el receptor hasta el bit anterior.



90

Validación de Tramas

El momento en el que se valida la trama es diferente para un nodo emisor y uno receptor. Para el emisor, una trama será válida si no detecta ningún error hasta el último *bit* del delimitador de trama, mientras que para el receptor hasta el *bit* anterior.

BIT-STUFFING

- Para evitar problemas de sincronización, no se permite el envío de 6 bits consecutivos iguales.
- El protocolo agrega automáticamente un bit contrario cuando se transmite una secuencia de 5 bits del mismo valor (*bit-stuffing*).
- Los nodos que están recibiendo el mensaje quitan este bit automáticamente, de manera que al final la información se mantiene inalterable.
- Este control lo realiza la capa física del protocolo CAN.

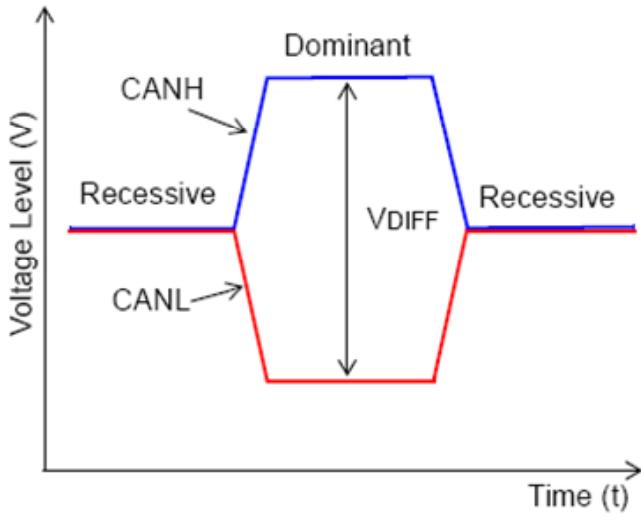
Bus CAN © L.G. (UPV/EHU)

91

Relleno de Bits (*Bit-stuffing*)

Para evitar problemas de sincronización, no se permite el envío de 6 bits iguales consecutivos. Debido a esto, se debe asegurar que la secuencia de bits de un mensaje no contenga accidentalmente dicha secuencia. Para asegurar que esto no ocurra, el protocolo agrega automáticamente un *bit* contrario cuando se transmite una secuencia de 5 bits del mismo valor (*bit-stuffing*). Los nodos que están recibiendo el mensaje quitan automáticamente este *bit*, de manera que la información al final se mantiene inalterable. Este control lo realiza la capa física del protocolo CAN.

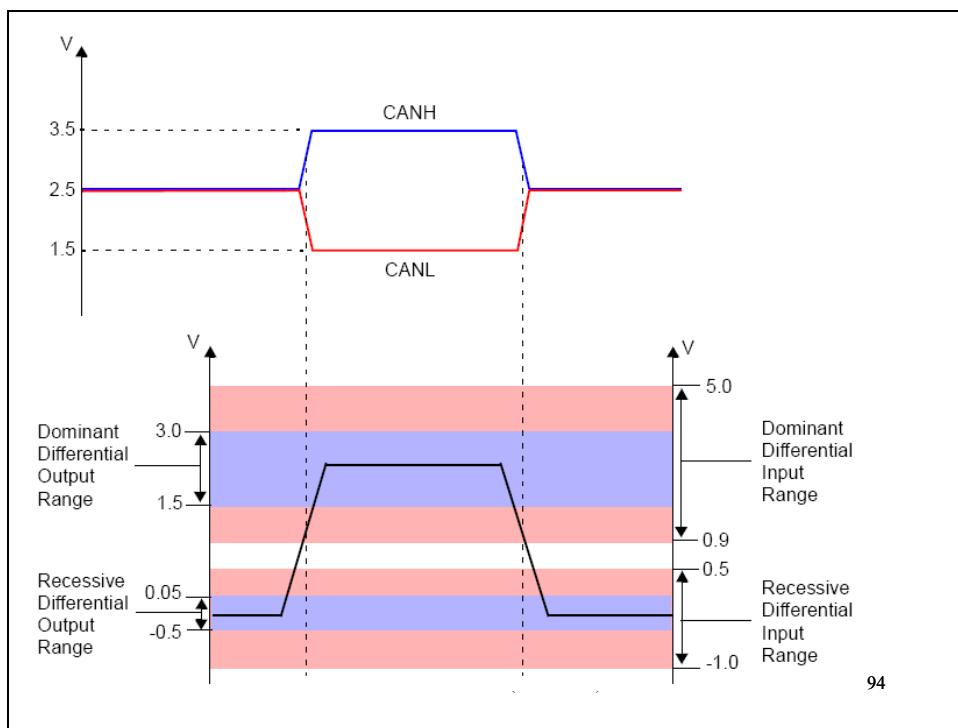
IMPLEMENTACION DE LA CAPA FÍSICA



93

1.7 IMPLEMENTACION DE LA CAPA FÍSICA

La capa física del protocolo CAN define la forma de las señales que se utilizan en la comunicación así como las características que deben reunir los dispositivos que se conectan al bus.



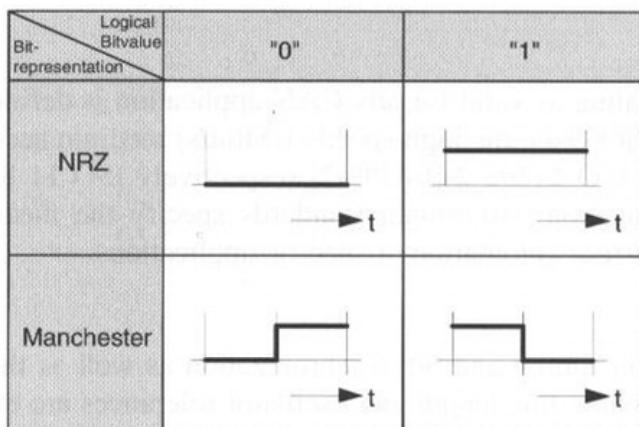
94

CAN especifica dos estados lógicos: recesivo y dominante. ISO-11898 define una diferencia de tensión para representar los estados (o bits) recesivo y dominante.

En el estado Recessivo (por ejemplo, lógica «1» en el MCP2551, TXD entrada), la diferencia de tensión entre CANH y CANL es inferior al umbral mínimo, mientras que en el estado Dominante, la diferencia de tensión entre CANH y CANL es mayor que el umbral mínimo. Un *bit* dominante sobrescribe un *bit* recesivo en el bus para lograr arbitraje no destructivo.

Representación física de las señales

Representación de un bit



Representación de un bit según NRZ y Manchester

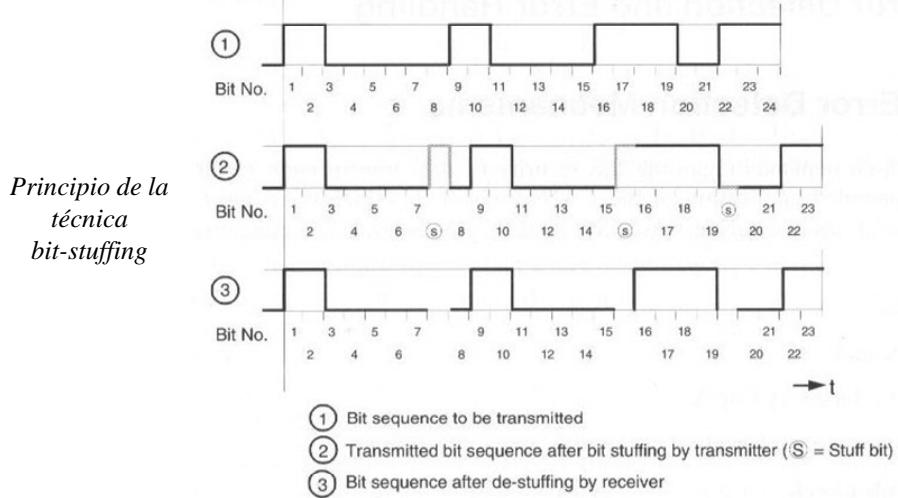
Bus CAN © L.G. (UPV/EHU)

95

1.7.1 REPRESENTACION FISICA DE LAS SEÑALES, REPRESENTACION DEL BIT

Existen varios métodos de codificación de señales digitales, con diferentes características, que les otorgan diferentes ventajas y desventajas. Una diferencia sustancial, es el número de períodos de reloj necesarios para representar un *bit*. Si tomamos como ejemplo las codificaciones Manchester y *Non-Return-to-Zero* (ver figura 25) vemos que la codificación Manchester necesita dos ciclos de reloj mientras que la NRZ solamente uno. A primera vista parece que esta última es más eficiente que la anterior ya que a la misma frecuencia de reloj es capaz de transmitir el doble número de bits. Ambos sistemas de codificación se usan sin ninguna señal externa de reloj, por lo que es responsabilidad de cada nodo que interviene en la comunicación, detectar que intervalo de tiempo de la señal corresponde a un *bit* y cual al siguiente. Una solución posible, pero no práctica, consiste en que cada nodo disponga de su propia señal de reloj configurada a la misma frecuencia que los demás, de tal forma que el tiempo de *bit* sea para todos igual, y que además comience siempre en el mismo instante de tiempo. Como es obvio y debido a las irregularidades en las señales de reloj, esta sincronización ‘a ciegas’ no es factible, no tardarían mucho en desincronizarse entre sí. Por este motivo, en la propia señal de datos debe haber algún mecanismo que permita la sincronización entre nodos. En el caso de la codificación Manchester, resulta realmente sencillo ya que cualquier secuencia de *bit* implica una transición en la señal por cada *bit* enviado. De esta forma los nodos se pueden sincronizar con cada *bit* ya sea en el flanco de bajada o de subida, según el valor lógico de cada *bit*.

- CAN aplica NRZ delimitando a 5 bits consecutivos iguales antes de introducir en el flujo de datos un 6º bit de valor contrario que obliga a una transición en la señal.



96

Por el contrario, en la codificación NRZ, sólo se produce dicha transición cuando el valor de un *bit* es diferente del siguiente. Como consecuencia durante la transmisión de una secuencia de bits con el mismo valor, los nodos no tienen forma alguna de sincronizarse explícitamente entre sí, y se deben valer de su reloj interno para la correcta deducción de los bits. Como se ha comentado, esto produce con el tiempo, una desincronización entre los nodos. Si bien es cierto que, durante un periodo corto de tiempo esas imperfecciones en los relojes internos de los nodos no impide el correcto funcionamiento del sistema. Por este motivo esta técnica es válida si se acota el tiempo (el número de bits) en que los nodos están ‘a ciegas’ a un valor que no de tiempo a los nodos a desincronizarse. A esta técnica se le conoce como *bit-stuffing*.

En el protocolo CAN se ha optado por esta última codificación delimitando a 5 bits consecutivos iguales antes de introducir en el flujo de datos un sexto *bit* de valor contrario que obliga a una transición en la señal. Este sexto *bit* introducido en el flujo de datos no interfiere en la comunicación ya que tanto el nodo transmisor como los nodos receptores aplican la misma técnica. Cuando un nodo quiere transmitir 5 o más bits consecutivos del mismo valor, introduce, cada 5 bits, un *bit* de valor contrario. Los nodos receptores, si reciben 5 bits consecutivos iguales saben que el siguiente *bit* sirve para evitar la desincronización y lo ignoran automáticamente.

TIEMPO de BIT y Sincronización a Nivel de Bit

- CAN utiliza transferencia síncrona de datos frente a asíncrona.
- Es necesario que todos los nodos implicados en la comunicación se sincronicen con cada bit, en este caso cada 5 bits.
- La resincronización es posible cuando un nodo detecta una transición en la señal.
- Hay que conocer que duración mínima debe tener 1 bit y determinar entonces la máxima velocidad de transmisión del bus.

1.7.2 TIEMPO DE BIT Y SINCRONIZACION A NIVEL DE BIT

A diferencia de otros buses de campo, el bus CAN utiliza transferencia síncrona de datos frente a asíncrona (orientado a caracteres¹⁷). Esto implica una mayor capacidad de transmisión pero complica la técnica de sincronización de *bit* para mantener una correcta comunicación. Mientras que en los sistemas asíncronos en *bit* de *start* es suficiente para sincronizarse y mantenerse sincronizado durante la transmisión de los datos, en los sistemas síncronos no lo es. Esto es debido a que la cantidad de datos que se envían después del *bit* de *start* es bastante superior a la de los sistemas asíncronos. Por este motivo es necesario que todos los nodos implicados en la comunicación¹⁸ se sincronicen con cada *bit*¹⁹. Esta resincronización es posible cuando un nodo detecta una transición en la señal.

¹⁷ Estos sistemas se caracterizan por enviar 1 o 2 *bits* de *start* y 1 o 2 *bits* de *stop* por cada 8 *bits* de datos.

¹⁸ En el caso de un bus, todos los nodos conectados a él.

¹⁹ En el caso del bus CAN por lo menos cada 5 *bits*.

TIEMPO de BIT y Sincronización a Nivel de Bit (II)

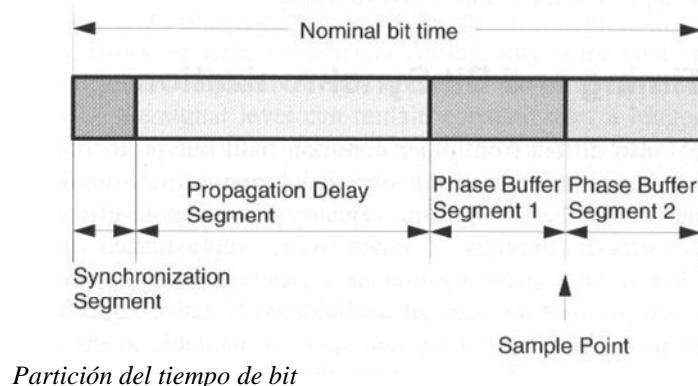
- Partiendo de que el punto más importante dentro de un bit es el de muestreo
- Es necesario reservar un intervalo de tiempo determinado antes y después de éste, que depende de las características concretas del bus CAN donde nos encontremos.
- Para evitar el desfase entre los relojes internos de cada nodo es necesario reservar a cada lado del punto de muestreo el tiempo máximo de ese desfase.

Bus CAN © L.G. (UPV/EHU)

98

Para conocer la duración mínima que debe tener un *bit* y determinar entonces la máxima velocidad de transmisión del bus, es necesario comprender algunos aspectos específicos del bus CAN. Partiendo de que el punto más importante dentro de un *bit* es el de muestreo, es necesario reservar un intervalo de tiempo determinado antes y después de éste, que depende de las características concretas del bus CAN donde nos encontremos. Para evitar el desfase entre los relojes internos de cada nodo es necesario reservar a cada lado del punto de muestreo el tiempo máximo de ese desfase.

- Hay que tener en cuenta la longitud total del bus para reservar el tiempo que tarda la señal en propagarse por el bus.
- Este tiempo debe ser el doble de lo que tardaría la señal en propagarse desde el comienzo hasta el final del bus.



Partición del tiempo de bit

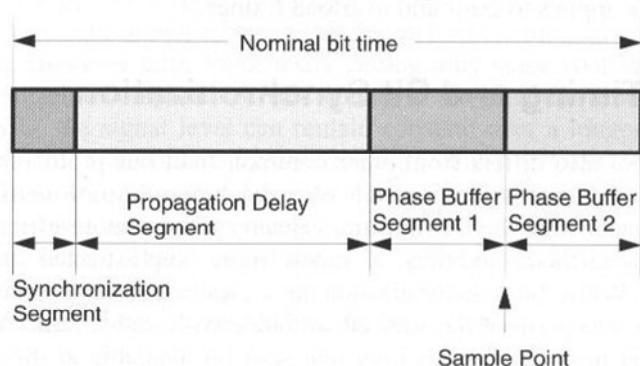
Bus CAN © L.G. (UPV/EHU)

99

Así mismo, hay que tener en cuenta la longitud total del bus para reservar el tiempo que tarda la señal en propagarse por el bus. Este tiempo debe ser el doble de lo que tardaría la señal en propagarse desde el comienzo hasta el final del bus. Es necesario que sea el doble debido al sistema de arbitraje usado en CAN así como a la técnica de reconocimiento de tramas.

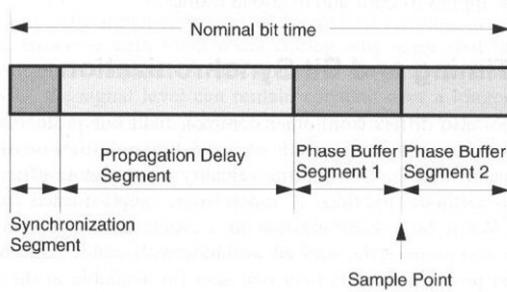
- La duración de cada segmento es múltiplo de TQ (*Time Quantum*)
- TQ es múltiplo de la frecuencia del reloj interna de cada nodo
- La duración de cada TQ debe ser la misma para cada nodo.
- Esto permite que la frecuencia de reloj de cada nodo pueda ser diferente.

La longitud de cada segmento es múltiplo de TQ (*Time Quantum*) que a su vez es un múltiplo de la frecuencia del reloj interna de cada nodo. La duración de cada TQ debe ser la misma para cada nodo. Esto permite que la frecuencia de reloj de cada nodo pueda ser diferente.



- En el segmento de sincronización es donde debe ocurrir la transición de la señal, si es que ocurre, de lo contrario es necesario alargar o disminuir el tiempo de bit para asegurar que el punto de muestreo sea el correcto.

En el segmento de sincronización es donde debe ocurrir la transición de la señal, si es que ocurre, de lo contrario es necesario alargar o disminuir el tiempo de *bit* para asegurar que el punto de muestreo sea el correcto. La variación de la longitud del tiempo de *bit* se alarga en el '*Phase Buffer Segment 1*' o se disminuye en el '*Phase Buffer Segment 2*'. Con este mecanismo cada nodo puede sincronizarse con la señal transmitida.



- La longitud de cada segmento viene determinada por las características físicas del bus.
- La longitud del bus y la tolerancia de los relojes implican unos valores mínimos determinados, normalmente se parte de la velocidad de transmisión requerida, y sobre ella se ajustan los valores de todos los segmentos.

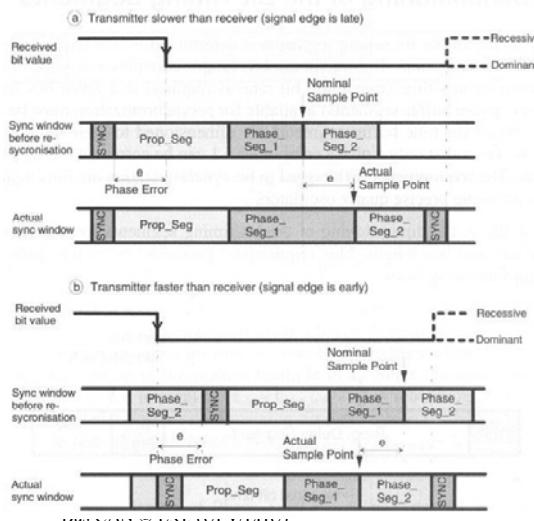
Bus CAN © L.G. (UPV/EHU)

102

La longitud de cada uno de los segmentos viene determinada por las características físicas del bus. Si bien la longitud del bus y la tolerancia de los relojes implican unos valores mínimos determinados, normalmente se parte de la velocidad de transmisión requerida, y sobre ella se ajustan los valores de todos los segmentos.

- Para conseguir variar la longitud del tiempo de bit se alarga el *Phase Buffer Segment 1* o se disminuye el *Phase Buffer Segment 2*.
- Con este mecanismo cada nodo puede sincronizarse con la señal transmitida.

Principio de resincronización

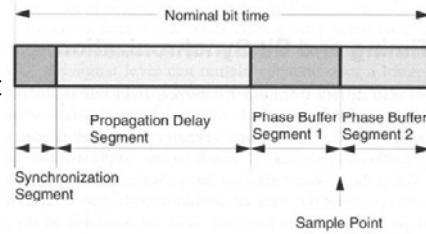


103

- TQ es una unidad de tiempo derivada del periodo del oscilador.
- Los módulos tienen un *prescaler* programable, con valores entre 1 a 32.
- TIME QUANTUM = $m * \text{MINIMUM TIME QUANTUM}$
 $m = \text{valor del prescaler}$

Longitud de los segmentos de tiempo:

- SYNC_SEG = 1 TQ.
- PROP_SEG = 1,2,...,8 TQ.
- PHASE_SEG1 = 1,2,...,8 TQ.
- PHASE_SEG2 es el máximo de PHASE_SEG1 y el tiempo de procesamiento de la información
- El tiempo de procesamiento de información es = < 2 TQ
- El nº total de TQ del tiempo de bit es programable desde 8 a 25



El TQ (*Time Quantum*) es una unidad de tiempo fija derivada del periodo del oscilador.

MEDIO de TRANSMISIÓN

- Debe tener las características necesarias para representar los niveles de señal requeridos por el sistema de arbitraje de CAN.
- El bus debe estar en nivel dominante sólo cuando uno o más nodos estén transmitiendo dicho nivel y, en nivel recesivo en todos los demás casos.
- Esta característica se consigue fácilmente tanto con medios eléctricos como ópticos.

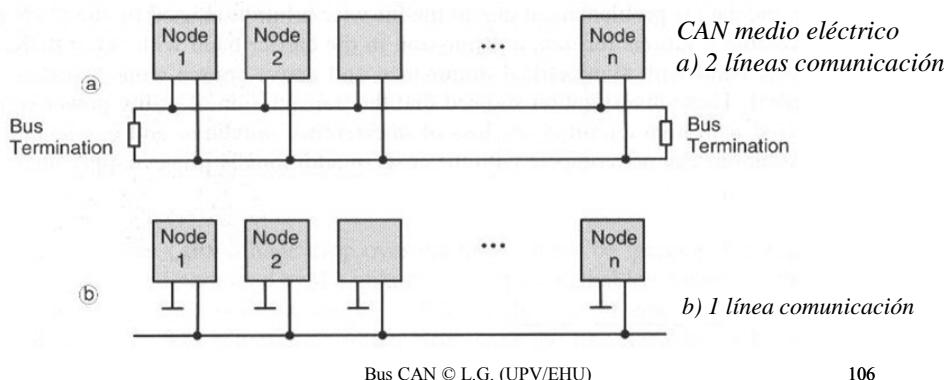
Bus CAN © L.G. (UPV/EHU)

105

1.7.3 MEDIO DE TRANSMISIÓN

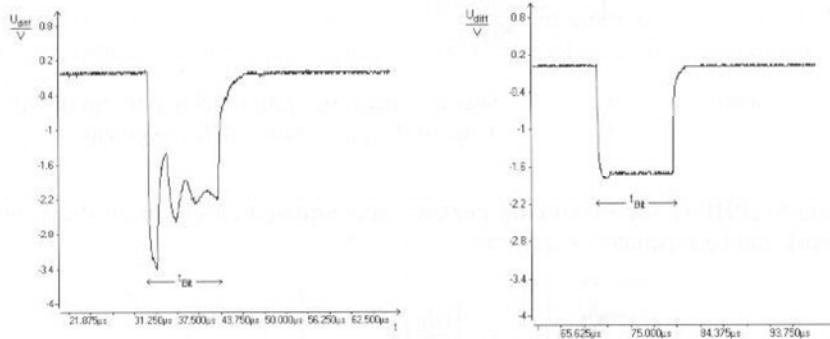
MEDIOS ELÉCTRICOS

- Las redes CAN se implementan normalmente de forma diferencial utilizando dos hilos de comunicación.
- También es posible implementar sistemas de un solo hilo, incluso sistemas que por las mismas líneas de comunicación alimentan a todos los nodos.



Medios Eléctricos

- La transmisión mediante dos líneas posibilita la transmisión diferencial y a su vez una menor tasa de error.
- El bus debe contar con un terminador de bus en cada extremo para evitar la reflexión de las señales.

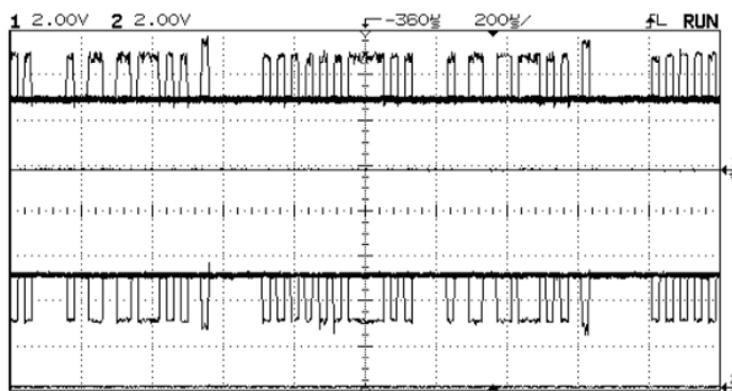


Señal recibida por un nodo en un bus sin terminadores y con ellos

Bus CAN © L.G. (UPV/EHU)

107

La transmisión mediante dos líneas posibilita la transmisión diferencial y a su vez una menor tasa de error. El bus debe contar con un terminador de bus en cada extremo para evitar la reflexión de las señales. La interferencia electromagnética inducida por las dos líneas se puede compensar utilizando par trenzado en lugar de líneas paralelas. Con un sistema apropiado de corrección de errores es posible mantener la comunicación con ruido ambiental (dentro de unos márgenes), e incluso con ambas líneas cortocircuitadas o con alguna rota.



Comunicación diferencial por un medio eléctrico

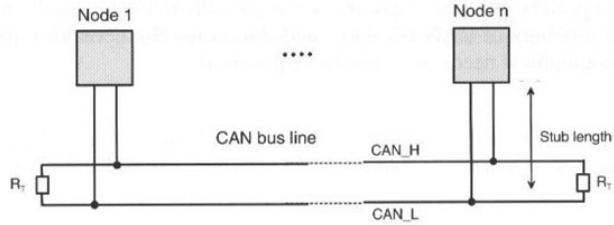
Bus CAN © L.G. (UPV/EHU)

108

La mayoría de redes CAN están implementadas con un sistema de dos líneas, ya que es un sistema sencillo que aporta muchas características de seguridad y existen una gran cantidad de circuitos que implementan la interfaz de acceso al bus.

En los sistemas con solo una línea de comunicación hay que tener especial atención con las interferencias del entorno para garantizar una comunicación rápida y segura.

TOPOLOGÍA



Topología de un bus CAN según la ISO 11898-2.

- Especifica que los terminadores de bus sean resistencias de 120 Ohm cada una
- Para una velocidad de transmisión máxima de 1 Mbit/s, la longitud total del bus no debe superar los 40 m ni el *stub length* los 30 cm.
- Es posible solventar estas limitaciones gracias a dispositivos como repetidores, puentes de red o puertas de enlace según las necesidades concretas del sistema.

Bus CAN © L.G. (UPV/EHU)

109

1.7.4 TOPOLOGÍA

La ISO 11898-2 especifica la topología que se muestra en la siguiente figura. También especifica que los terminadores de bus deben ser resistencias de 120 Ohm cada una y que para una velocidad de transmisión máxima de 1 Mbit/s, la longitud total del bus no debe superar los 40 m ni el *stub length* los 30 cm. Como es evidente, estos valores se pueden aumentar a costa de sacrificar la velocidad de transmisión.

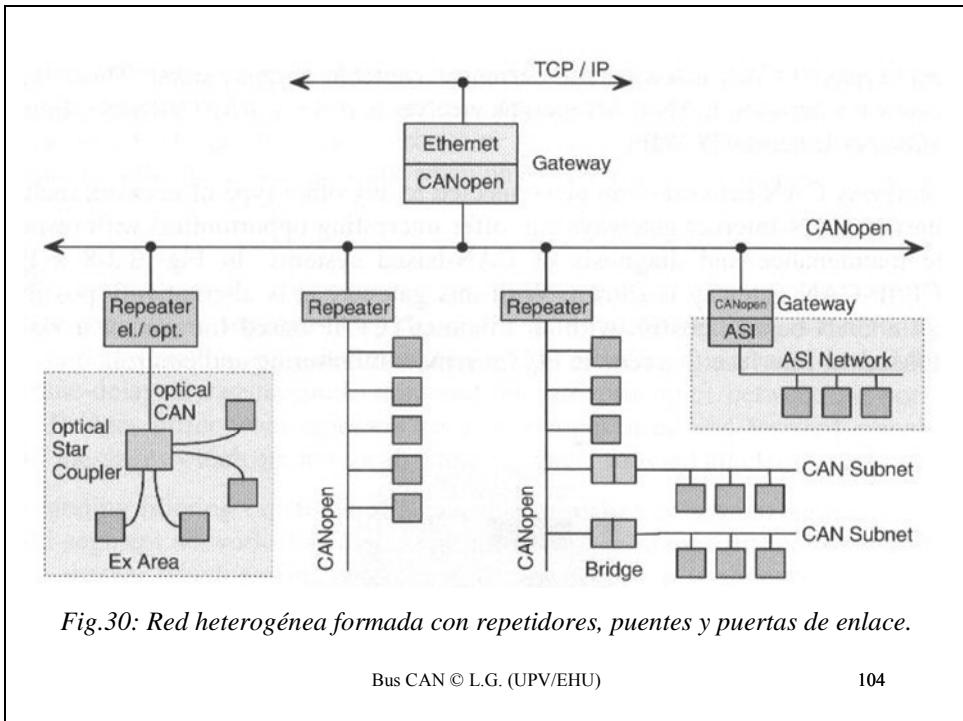


Fig.30: Red heterogénea formada con repetidores, puentes y puertas de enlace.

Bus CAN © L.G. (UPV/EHU)

104

En entornos industriales que requieren una alta tasa de transferencia y la longitud viene determinada por la localización de los nodos a conectar, si ésta supera los 40 m, es posible solventar estas limitaciones gracias a dispositivos como repetidores, puentes de red o puertas de enlace según las necesidades concretas del sistema.

Con un repetidor es posible unir dos segmentos de un bus CAN que no supere cada uno los 40 m. El repetidor transfiere las señales de un segmento al otro y viceversa. A la hora de diseñar un sistema con repetidores hay que tener en cuenta los retardos que introducen éstos y sumárselos al retraso de propagación de la señal.

A diferencia de los repetidores, los puentes aíslan dos segmentos de tal forma que el tráfico local a cada segmento se queda en él y no interfiere en el otro. Sólo los mensajes que tienen como destino algún nodo del segmento contrario cruzarán el puente hacia él.

Las puertas de enlace comunican una red CAN con cualquier otra red adaptando ambos protocolos para una comunicación bidireccional. En su interior transforman las tramas CAN en las correspondientes al protocolo implementado que pueden variar en longitud y número. En la siguiente figura se puede observar una red heterogénea donde se han utilizado tanto repetidores como puentes y una puerta de enlace.

ACCESO AL MEDIO

- Para conectar un nodo CAN al bus existen diferentes circuitos (*transceiver*) que implementan el acceso según las diferentes especificaciones.

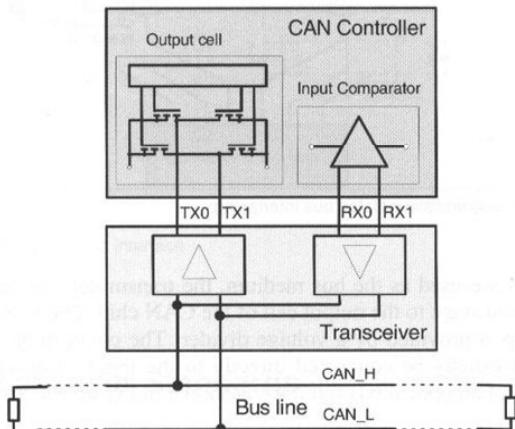


Figura 31: Nodo CAN con circuito de acceso al medio.

Bus CAN © L.G. (UPV/EHU)

105

1.7.5 ACCESO AL MEDIO

Para conectar un nodo CAN al bus existen diferentes circuitos que implementan el acceso según las diferentes especificaciones. Están caracterizados por incluir un amplificador de señal tanto en el envío como en la recepción.

CARACTERÍSTICAS DE TIEMPO-REAL

- El bus CAN está diseñado para sistemas de control, y en ningún caso para intercambio masivo de información.
- Cuando nos referimos a él como sistema de comunicación válido para aplicaciones en tiempo-real, nos referimos a aplicaciones de control y no por ejemplo, a aplicaciones de voz o video en tiempo-real.
- El sistema de arbitraje del CAN es el principal mecanismo para conseguir tiempo-real

1.8 CARACTERÍSTICAS DE TIEMPO-REAL

A continuación, veremos las características que hacen al bus CAN ser un sistema de comunicaciones sobre el que se pueden implementar sistemas de control en tiempo-real²⁰. También veremos como acotar el tiempo que transcurre desde que el nodo origen quiere enviar un mensaje hasta que llega al nodo destino.

Antes de comenzar, conviene recordar que el bus CAN está diseñado para sistemas de control, y en ningún caso para intercambio masivo de información. Por lo tanto, cuando nos referimos a él como sistema de comunicación válido para aplicaciones en tiempo-real, nos estamos refiriendo a aplicaciones de control y no por ejemplo, a aplicaciones de voz o video en tiempo-real.

²⁰ M. A. Livani y J. Kaiser, *EDF Consensus on CAN Bus Access for Dynamic Real-Time Application*. University of Ulm, Departament of Computer Structures.

CARACTERÍSTICAS DE TIEMPO-REAL (II)

- El controlador de un nodo coloca en el buffer de salida del controlador CAN el mensaje que quiere transmitir, y será éste quien se encargue de transmitir el mensaje cuando el bus esté libre y sea el mensaje de mayor prioridad en ese momento.
- Una de las características de los sistemas en tiempo-real, es poder acotar el tiempo de respuesta a un estímulo dado.
- Para ello, hay que determinar el tiempo máximo en el que el sistema debe responder a los eventos del exterior.

El sistema de arbitraje que utiliza CAN es el principal mecanismo para conseguir tiempo-real. El controlador de un nodo coloca el mensaje que quiere transmitir en el buffer de salida del controlador CAN, y es éste el encargado de transmitir el mensaje cuando el bus está libre y sea el mensaje de mayor prioridad en ese momento.

CARACTERÍSTICAS DE TIEMPO-REAL (III)

En CAN si se conoce el tiempo máximo que puede transcurrir desde que un nodo quiere enviar un mensaje hasta que llega a su destino, se puede determinar el comportamiento del sistema en cualquier situación dada.

- A este tiempo máximo le llamaremos **latencia máxima** del mensaje.
- La **latencia** de un mensaje está compuesta por
 - el tiempo que tiene que esperar en el buffer a ser transmitido
 - el tiempo que dura la transmisión en sí del mensaje.

Una de las características de los sistemas en tiempo-real, es poder acotar el tiempo de respuesta a un estímulo dado²¹. Para ello, hay que determinar el tiempo máximo en el que el sistema debe responder a los eventos del exterior. En CAN, si se conoce el tiempo máximo que puede transcurrir desde que un nodo quiere enviar un mensaje hasta que llega a su destino, se puede determinar el comportamiento del sistema en cualquier situación dada. A este tiempo máximo le llamaremos latencia máxima del mensaje.

²¹ K. Tindell, A. Burns y A. Welling, *Calculating Controller Area Network (CAN) Message Response Times*. University of York, Department of Computer Science, Cork, England.

CARACTERÍSTICAS DE TIEMPO-REAL (IV)

- El tiempo de espera depende por un lado, del tiempo que tarde en acabar la transmisión del mensaje que está ocupando el bus en ese momento (**tiempo de bloqueo**), y por otro, del número de mensajes de mayor prioridad que se quieran transmitir en ese intervalo de tiempo (**interferencia**).
- El tiempo que dura la transmisión real del mensaje en cuestión, depende de su longitud y de la velocidad de transmisión del bus.
- El tiempo de **bloqueo** será el máximo cuando el mensaje que está ocupando el bus justo acaba de comenzar a transmitirse y es el de mayor longitud posible.

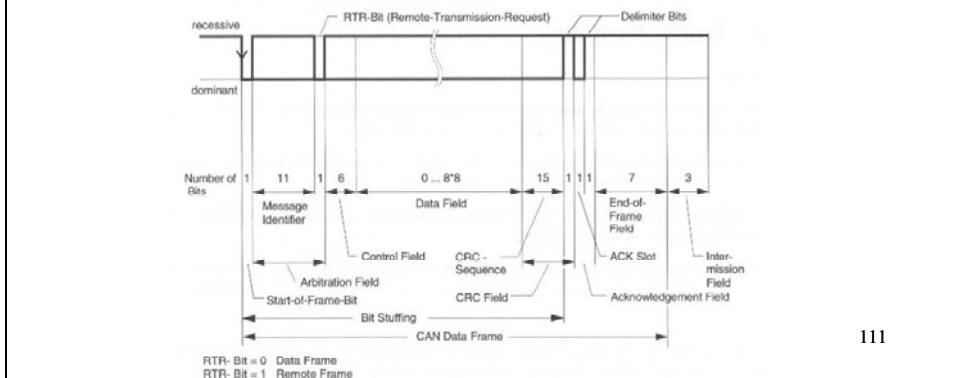
Bus CAN © L.G. (UPV/EHU)

110

La latencia de un mensaje está compuesta por el tiempo que tiene que esperar en el buffer a ser transmitido y el tiempo que dura la transmisión del mensaje en sí. El primero de ellos depende, por un lado, del tiempo que tarde en acabar la transmisión del mensaje que está ocupando el bus en ese momento (tiempo de bloqueo), y por otro, del número de mensajes de mayor prioridad que se quieran transmitir en ese intervalo de tiempo (interferencia). El segundo de ellos, el tiempo que dura la transmisión real del mensaje en cuestión, depende de su longitud y de la velocidad de transmisión del bus (de manera indirecta, tanto el tiempo de bloqueo como la interferencia también dependen de la longitud de los mensajes implicados y de la velocidad de transmisión).

CARACTERÍSTICAS DE TIEMPO-REAL (V)

- Para calcular la mayor longitud de un mensaje hay que tener en cuenta que los mensajes CAN pueden transmitir hasta 8 bytes de datos y en la versión estándar de identificadores de 11 bits (CAN 2.0A) tienen además 47 bits adicionales de control.
- Hay que añadir los bits correspondientes al *bit-stuffing*, que en el caso peor, puede aparecer uno cada 5.



111

Para calcular la latencia máxima de un mensaje (que quedará formulada en función de su prioridad y su longitud) debemos tener en cuenta el máximo de los tiempos anteriormente descritos. El tiempo de bloqueo será el máximo cuando el mensaje que está ocupando el bus justo acaba de comenzar a transmitirse y es el de mayor longitud posible. Para calcular la mayor longitud de un mensaje hay que tener en cuenta que: los mensajes CAN pueden transmitir hasta 8 bytes de datos y en la versión básica de identificadores de 11 bits (CAN 2.0A) tienen además 47 bits adicionales de control. También hay que añadir los bits correspondientes al *bit-stuffing*, que en el caso peor, puede aparecer uno cada cinco. La regla del *bit-stuffing* se aplica a los datos y a 34, de los 47 bits de control.

- La regla del *bit-stuffing* se aplica a los datos y a 34 de los 47 bits de control.

$$T_{\max} = \left(\left\lfloor \frac{34+8*8}{5} \right\rfloor + 47 + 8*8 \right) \tau_{bit} = 130 \tau_{bit}$$

Duración máxima de un mensaje

τ_{bit} = Tiempo de transmisión de un bit.

- El tiempo necesario en transmitir el mensaje que estamos analizando dependerá de su longitud, más concretamente del número de bytes de datos.

$$T_{\max}^n = \left(\left\lfloor \frac{34+8*n}{5} \right\rfloor + 47 + 8*n \right) \tau_{bit}$$

Duración de un mensaje de 'n' datos

- Para calcular la *interferencia*, para el caso peor, debemos considerar que aparece un mensaje para cada prioridad mayor.

$$I_{\max}^p = \sum_{\forall j < p} T_{\max}^j$$

- En estas condiciones, la latencia máxima será la suma de estos tres valores.

$$L_{\max}^{p,n} = T_{\max} + I_{\max}^p + T_{\max}^n \quad Fórmula\ 5$$

= Mensaje en curso + Mensajes con mayor prioridad + Propio Mensaje

Para calcular la interferencia (nº de mensajes antes del nuestro) suponemos que en el intervalo de tiempo entre el momento en que se quiere transmitir el mensaje y hasta que se consigue, no se repiten los mensajes de la misma prioridad. Por otro lado, para el caso peor, debemos considerar que aparece un mensaje para cada una de las prioridades mayores.

$$L_{\max}^{p,n} = T_{\max} + I_{\max}^p + T_{\max}^n$$

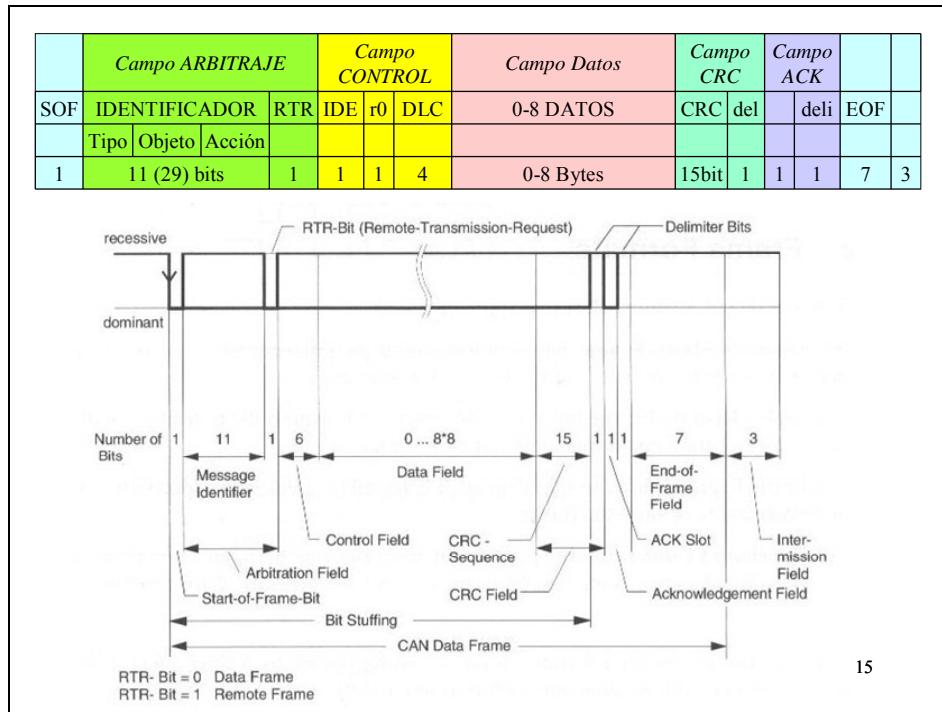
Tiempo total hasta que se termina de transmitir el mensaje de prioridad 'p' y 'n' datos

Protocolo de comunicación robot HERO

Bus CAN © L.G. (UPV/EHU)

114

1.9 EJEMPLO DE PROTOCOLO DE COMUNICACIÓN ROBOT HERO



Protocolo de comunicación robot HERO

(4 tipos de sensores y 2 motores)

Identificador Estándar CAN (11 bits)

-**Tipo de dato (3 bits) – Tipo Objeto (4 bits) – Acción (4bits) -**

Tipo (3 bits)

0x00 – 0x03	- Reservado (mensajes de prioridad alta)
0x04	- Control
0x05	- Datos
0x06 – 0x07	- Reservado

Objeto (4bits)

0x00	- Dato sensor de entrada (SENSOR_IN) - No usado
0x01	- Dato sensor de salida (SENSOR_OUT)
0x02	- Dato motor de entrada (MOTOR_IN)
0x03	- Dato motor de salida (MOTOR_OUT)
0x04– 0x0F	- Reservado

Acción (4 bits)

0x00	- Calibrar (CALIBRATE)
0x01	- Habilitar (ENABLE)
0x02	- Deshabilitar (DISABLE)
0x03 – 0x0F	- Reservado

Bus CAN © L.G. (UPV/EHU)

116

Parámetros (Datos) D1 .. D8 (de 0 a 8 bytes)

= Sensores

Tipo (8 bits)

0x00	- Contacto
0x01	- Distancia – Infrarrojo – Digital
0x02	- Distancia – Infrarrojo – Analógico
0x03	- Distancia - Ultrasonido
0x04 - 0xfe	- Reservado
0xff	- Todos

Identificador (8 bits)

0x00 - 0x0f	- Número de sensor
0x10 - 0xfe	- Reservado
0xff	- Todos

Valor (8 bits)

0x00	- Valor no disponible (sensor sin valor correcto)
0x01 - 0xFF	- Valor medido

= Motores

Tipo (8 bits)

0x00	- Paso a paso
0x01	- DC
0x02 - 0xfe	- Reservado
0xff	- Todos

Identificador (8 bits)

0x00 - 0x0f	- Número de motor
0x10 - 0xfe	- Reservado
0xff	- Todos

Valor (8 bits)

0x00 – 0xFF	- Valor medido
-------------	----------------

Bus CAN © L.G. (UPV/EHU)

117

Cuatro tipos de sensores y dos tipos de motores

	Campo ARBITRAJE			Campo CONTROL			Campo Datos		Campo CRC		Campo ACK			
SOF	IDENTIFICADOR		RTR	IDE	r0	DLC	0-8 DATOS		CRC	del		deli	EOF	
	Tipo	Objeto	Acción											
1	11 (29) bits		1	1	1	4	0-8 Bytes		15bit	1	1	1	7	3

Identificador Estándar CAN (11 bits)

= **Tipo (3 bits)** 0x00 – 0x03 - Reservado (mensajes de prioridad alta)

0x04 - Control

0x05 - Datos

0x06 – 0x07 - Reservado

= **Objeto (4bits)** 0x00 - Dato sensor de entrada (SENSOR_IN) - No usado

0x01 - Dato sensor de salida (SENSOR_OUT)

0x02 - Dato motor de entrada (MOTOR_IN)

0x03 - Dato motor de salida (MOTOR_OUT)

0x04– 0x0F - Reservado

= **Acción (4 bits)** 0x00 - Calibrar (CALIBRATE)

0x01 - Habilitar (ENABLE)

0x02 - Deshabilitar (DISABLE)

0x03 – 0x0F - Reservado

Petición de respuesta – RTR (1 bit) 0 Trama datos / 1 Remota

Longitud – DLC (Campo de control) nº de datos (6 bits)

Parámetros (Datos) D0 .. D8 (de 0 a 8 bytes)

= **Sensores Tipo (8 bits)**

0x00 - Contacto

0x01 - Distancia – Infrarrojo – Digital

0x02 - Distancia – Infrarrojo – Analógico

0x03 - Distancia - Ultrasonido

0x04 - 0xfe - Reservado

0xff - Todos

Identificador (8 bits)

0x00 - 0x0f - Número de sensor

0x10 - 0xfe - Reservado

0xff - Todos

Valor (8 bits)

0x00 - Valor no disponible (sensor sin valor correcto)

0x01 - 0xFF - Valor medido

= **Motores Tipo (8 bits)**

0x00 - Paso a paso

0x01 - DC

0x02 - 0xfe - Reservado'

0xff - Todos

Identificador (8 bits)

0x00 - 0x0f - Número de motor

0x10 - 0xfe - Reservado

0xFF - Todos

Valor (8 bits)

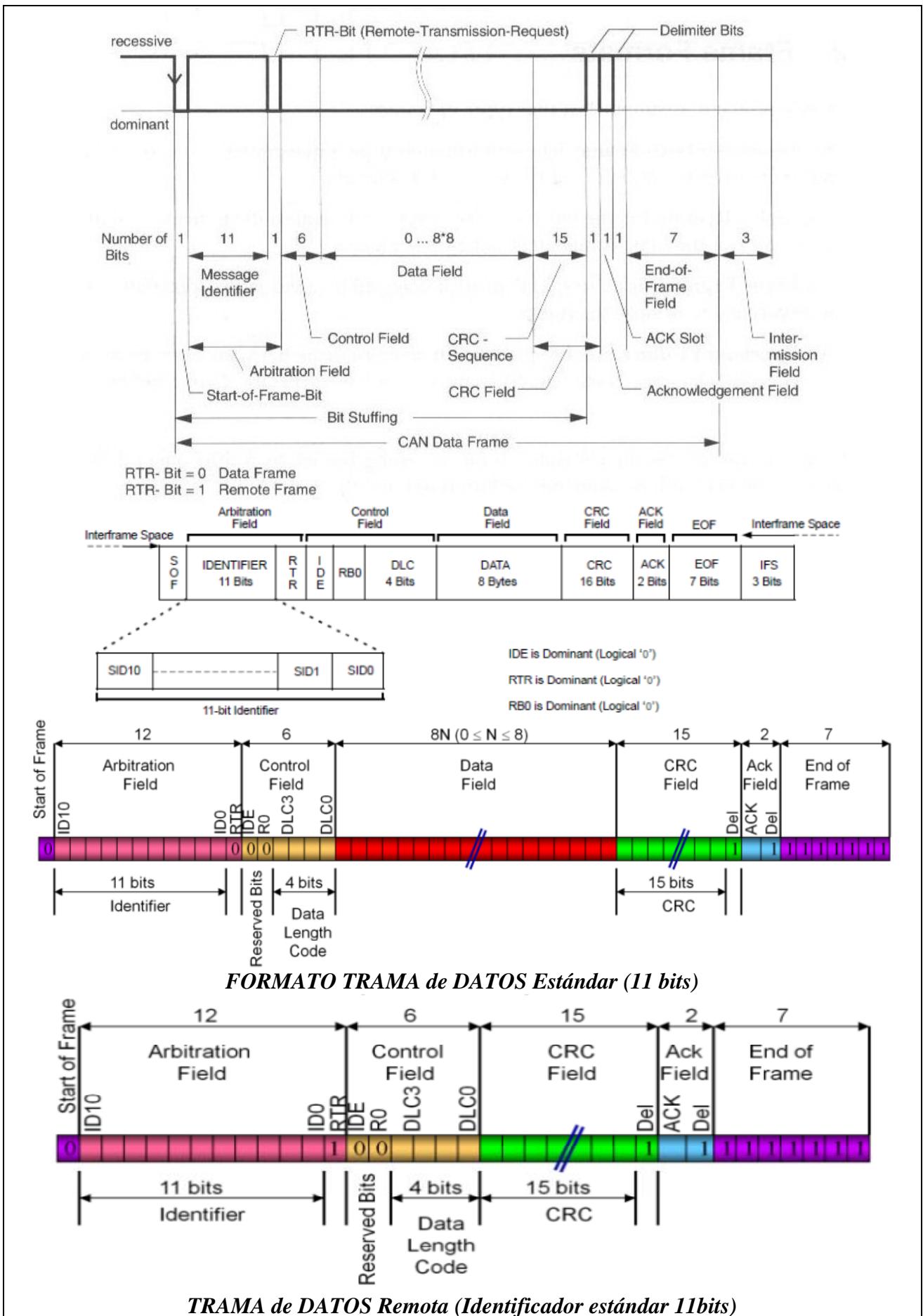
0x00 – 0xFF - Valor medido

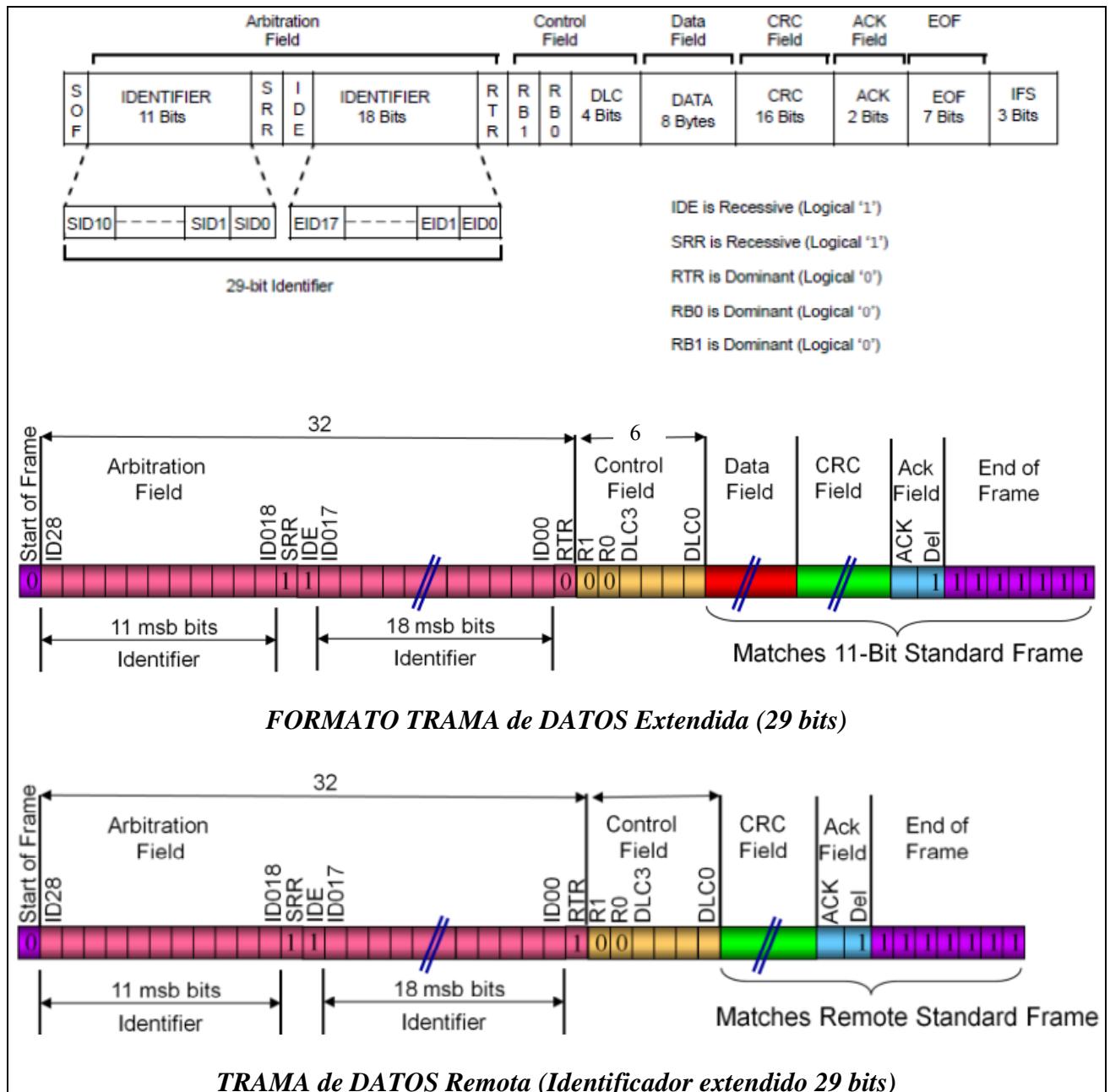
Parámetros definidos en ROBOT.h

Comando	S_ID value				Parámetros				vEmitor	Receptor
	Binary	Hex	RTR	DLC	D1	D2	D3 .. D7	D8		
CTRL_SENSOR_CALIBRATE	100 0000 0000	0x400	0	2	Tipo	Identificador	N/A	N/A	MODULO_SENSOR	MODULO_LCD
CTRL_SENSOR_ENABLE	100 0000 0001	0x401	0	2	Tipo	Identificador	N/A	N/A	MODULO_SENSOR	MODULO_LCD
CTRL_SENSOR_DISABLE	100 0000 0010	0x402	0	2	Tipo	Identificador	N/A	N/A	MODULO_SENSOR	MODULO_LCD
CTRL_MOTOR_CALIBRATE	100 0001 0000	0x410	0	2	Tipo	Identificador	N/A	N/A	MODULO_MOTOR	MODULO_LCD
CTRL_MOTOR_ENABLE	100 0001 0001	0x411	0	2	Tipo	Identificador	N/A	N/A	MODULO_MOTOR	MODULO_LCD
CTRL_MOTOR_DISABLE	100 0001 0010	0x412	0	2	Tipo	Identificador	N/A	N/A	MODULO_MOTOR	MODULO_LCD
DATA_SENSOR_OUT_CONTACT	101 0001 0000	0x510	0	0 .. 8	Contacto_S1	Contacto_S2	Contacto_S3 .. Contacto_S7	Contacto_S8	MODULO_SENSOR	MODULO_LCD
DATA_SENSOR_OUT_DIGITAL	101 0001 0001	0x511	0	0 .. 8	Distancia_S1	Distancia_S2	Distancia_S3 .. Distancia_S7	Distancia_S8	MODULO_SENSOR	MODULO_LCD
DATA_SENSOR_OUT_ANALOG	101 0001 0010	0x512	0	0 .. 8	Distancia_S1	Distancia_S2	Distancia_S3 .. Distancia_S7	Distancia_S8	MODULO_SENSOR	MODULO_LCD
DATA_SENSOR_OUT SONAR	101 0001 0011	0x513	0	0 .. 8	Distancia_S1	Distancia_S2	Distancia_S3 .. Distancia_S7	Distancia_S8	MODULO_SENSOR	MODULO_SENSOR
DATA_MOTOR_IN_STEPPER	101 0010 0000	0x520	0	0 .. 8	Posicion_M1	Position_M2	Posicion_M3 .. Posicion_M7	Posicion_M8	MODULO_MOTOR	MODULO_LCD
DATA_MOTOR_IN_DC	101 0010 0001	0x521	0	0 .. 8	Power_M1	Direccion_M1	Power_M2 .. Power_M4	Direccion_M4	MODULO_MOTOR	MODULO_LCD
DATA_MOTOR_OUT_STEPPER	101 0011 0000	0x530	0	0 .. 8	Posicion_M1	Estado_M1	Posicion_M2 .. Posicion_M4	Estado_M4	MODULO_MOTOR	MODULO_LCD
DATA_MOTOR_OUT_DC	101 0011 0001	0x530	0	0 .. 8	r.p.m._M1	r.p.m._M2	r.p.m._M3 .. r.p.m._M7	r.p.m._M8	MODULO_MOTOR	MODULO_LCD
DATA_MOTORS_IN	101 0100 0000	0x540	0	2	Velocidad	Direccion	Posicion, N/A	N/A	MODULO_LCD	MODULO_MOTOR

Tipo-objeto-cción

1.10 RESUMEN DE FORMATOS





11 bits → 2.048 mensajes

29 bits → más de 536 millones de mensajes (536.870.912)

MODULO CAN DEL PIC24H

Ver PIC24H Family Reference Manual (Section 21, DS70226)

Recepción del mensaje:

- 32 Búferes de mensajes, los 8 primeros pueden ser de transmisión o de recepción. El resto sólo de recepción.
- 16 filtros de aceptación para el filtrado de mensajes
- 3 registros máscara de filtro de aceptación para el filtrado de mensajes

Transmisión de Mensajes:

- 8 búferes de mensaje configurables para transmisión de mensajes

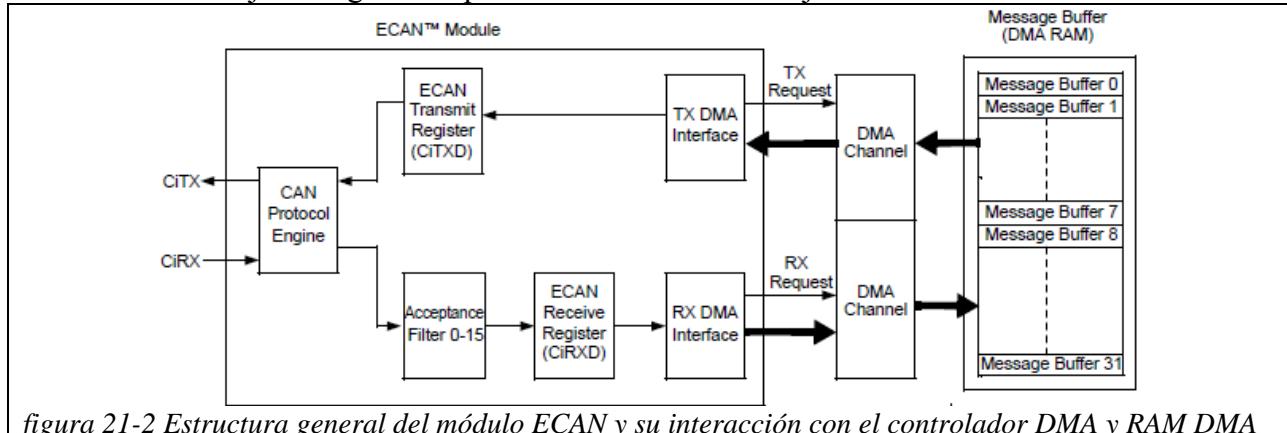


figura 21-2 Estructura general del módulo ECAN y su interacción con el controlador DMA y RAM DMA

BUFERES DE MENSAJE ECAN

Los búferes de mensaje ECAN se encuentran en DMA RAM. No son Registros de funciones especiales ECAN (FSR). La aplicación de usuario debe escribir directamente en el área de memoria DMA RAM que está configurada para Búferes de mensaje ECAN. La ubicación y el tamaño de la zona de búfer las define el usuario.

A continuación se muestra cómo se organizan las palabras del búfer de mensaje para transmisión y recepción.

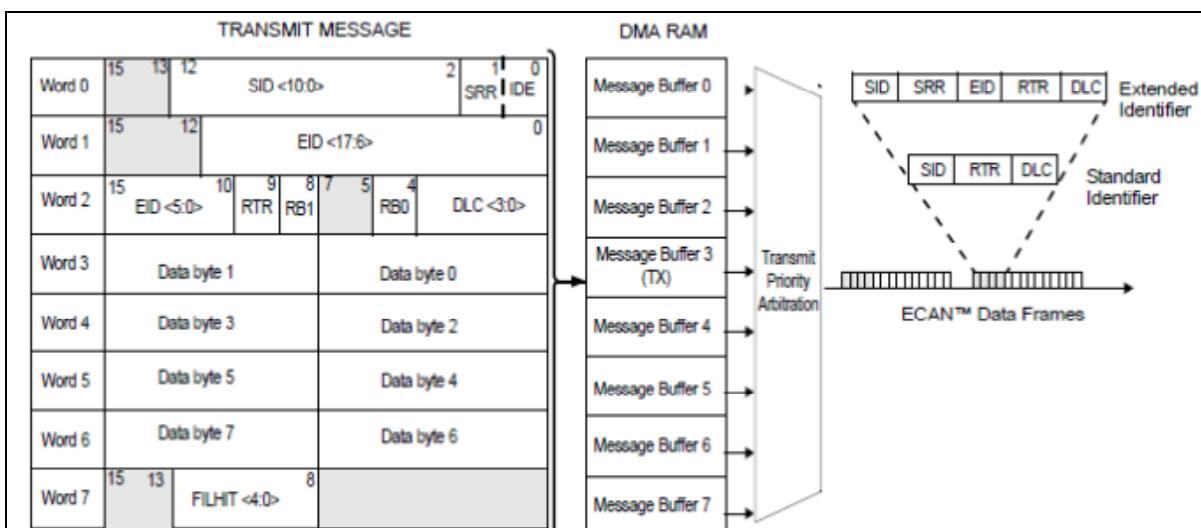


Figura 21-8: ECAN™ Transmission

SID<10:0> (Standard Identifier bits): bits de Identificador Estándar

SRR (Substitute Remote Request bit): Petición Remota Suplente para tramas extendidas

IDE (Extended Identifier): Bit de Identificador Extendido

EID<17:6>: EID<5:0> (Extended Identifier bits): Bits de ID Extendido

RTR (Remote Transmission Request): bit de petición de Transmisión Remota

RB1: Reservado, se debe poner a '0' para protocolo CAN.

RB0: Reservado, se debe poner a '0' para protocolo CAN.

DLC (Data Length Code): nº de bytes del mensaje

FILHIT<4:0>(Filter Hit Code)(CiVEC<12:8>): Indica el nº del filtro que dio lugar a escribir en ese buffer.

FILTROS DE ACEPTACIÓN

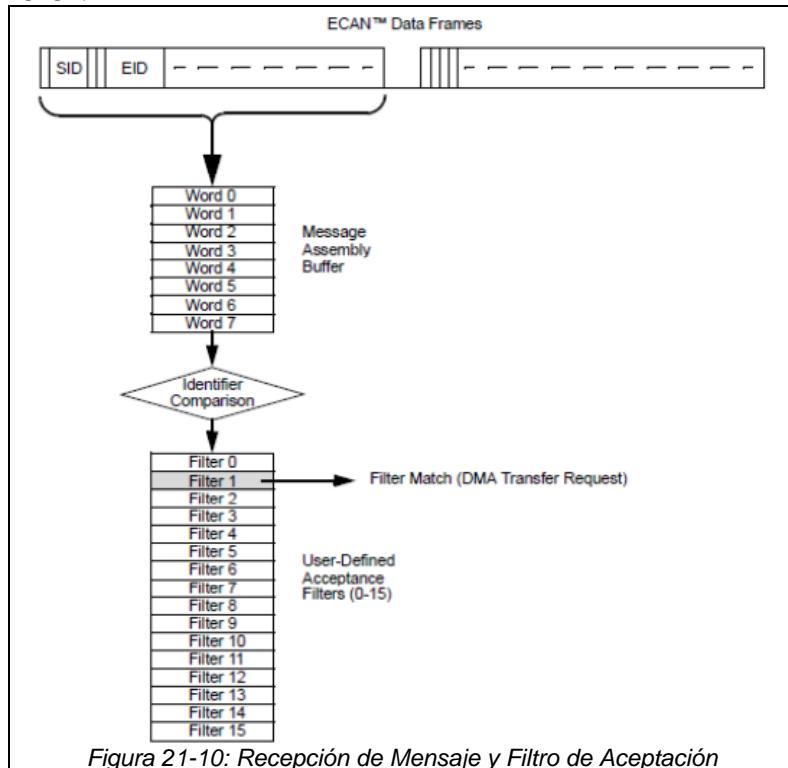


Figura 21-10: Recepción de Mensaje y Filtro de Aceptación

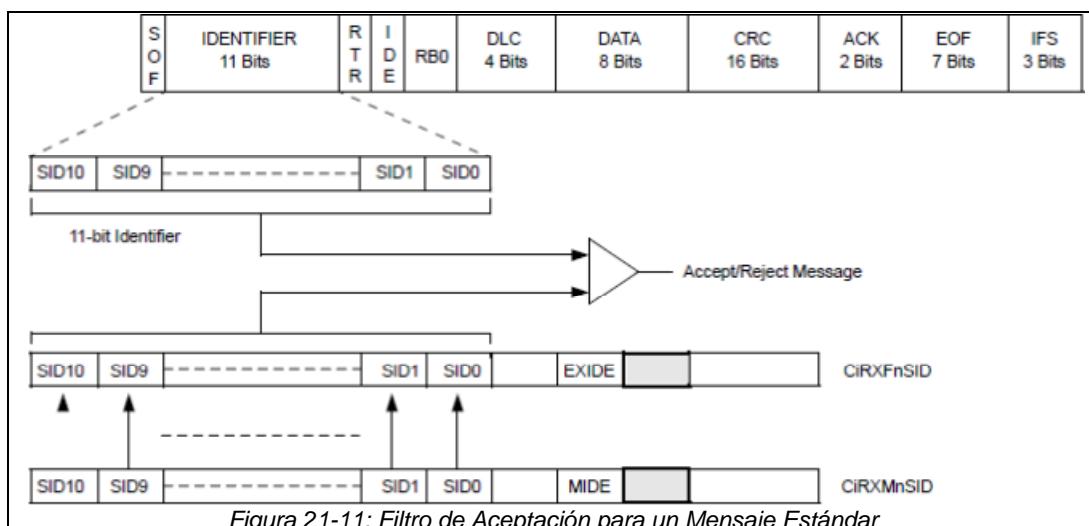


Figura 21-11: Filtro de Aceptación para un Mensaje Estándar

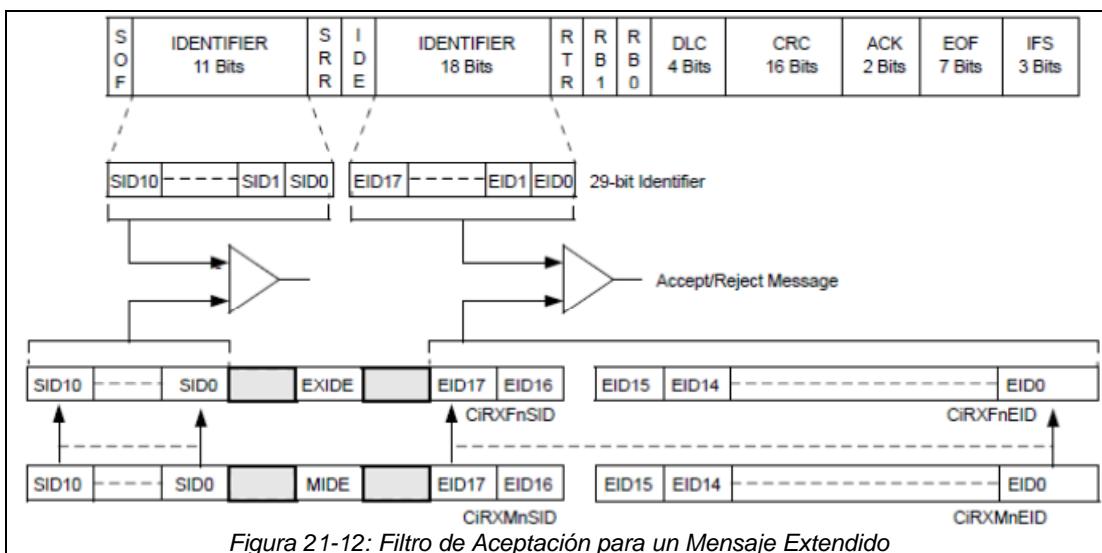


Figura 21-12: Filtro de Aceptación para un Mensaje Extendido

Los filtros de Aceptación 0-15 pueden activarse o desactivarse de forma individual utilizando el bit *Enable Filter* (FLTENn) en el registro CAN *Acceptance Filter Enable* (CiFEN1 <15:0>).

Los filtros de Aceptación 0-15 especifican identificadores que deben estar contenidos en un mensaje entrante para que sus contenidos pasen a un búfer de recepción.

Cada uno de estos filtros se compone de dos registros: uno para Identificadores Estándar y otro para los Identificadores Extendidos. Estos registros se identifican como:

- CiRXFnSID: ECAN Filtro de Aceptación ‘n’ para registro identificador estándar (n= 0-15)
- CiRXFnEID: ECAN Filtro de Aceptación ‘n’ para registro identificador extendido (n = 0-15)

MASCARAS de FILTRO de ACEPTACIÓN

La máscara de filtro de aceptación determina que bits, en los identificadores de mensaje entrante, se comparan con los bits de los filtros de aceptación.

Los filtros de aceptación seleccionan opcionalmente una de las máscaras de filtro de aceptación usando los bits de selección de mascara *Mask Source Select* (FnMSK <1:0>), en los registros CiFMSKSEL1 y CiFMSKSEL2:

- CiFMSKSEL1 <FnMSK>: Selección de la máscara para Filtros 0-7
- CiFMSKSEL2 <FnMSK>: Selección de la máscara para filtros 8-15

FnMSK<1:0> Selection y Valores

00 Selecciona mascara 0

01 Selecciona mascara 1

10 Selecciona mascara 2

11 Reservado

La máscara de bits determina qué bits debe aplicar el filtro.

Si cualquier bit de la máscara es ‘0’, ese bit es automáticamente aceptado, independientemente del bit del filtro.

(0 → NO compara Filtro con Mensaje, 1 → Compara)

Acceptance Filter/Mask Truth Table

Máscara (SIDn/EIDn)	Filtro (SIDn/EIDn)	Mensaje (SIDn/EIDn)	Acepta / Rechaza bit n
0	x	x	Aceptado
1	0	0	Aceptado
1	0	1	Rechazado
1	1	0	Rechazado
1	1	1	Aceptado

3 máscaras, 16 filtros

21.7.1.3 Selección del Tipo de Mensaje

EXIDE and MIDE Selections

EXIDE	MIDE	Selection
0	1	Filtro de aceptación para comprobar identificador estándar
1	1	Filtro de aceptación para comprobar identificador extendido.
X	0	Filtro de aceptación para comprobar identificador estándar/extendido

2 BUS CANOPEN

BUS CANopen

INTRODUCCIÓN:

Entre las capas de aplicación más extendidas de CAN encontramos:

- **CANopen** de ámbito europeo, proporcionada por CiA (Can in Automation)
- **DeviceNet**: desarrollado por Allen-Bradley actualmente abierto y gestionada por ODVA
- **SDS** de Honeywell
- **OSEK** orientada a sistemas distribuidos de tiempo real en vehículos
- **CAN Kingdom** de la empresa sueca Kvasser, orientada a automatización de maquinaria
- **LIN, Profibus, Fielbus, FipWordFip, ...**

1

2.1 INTRODUCCIÓN:

Entre las capas de aplicación más extendidas de CAN encontramos:

CANopen

Version History		
Version	Date	Comments
1.x	1998	First draft versions
2.0	1999	Change of title; change in chapter numbering; addition of CAN message syntax details
2.0a	7 April 1999	Corrected error in table 7 and 8 captions
3.0	20 March 2000	Several additions in accordance with CiA DS301 CANopen Communication Profile Version 4.0 (update from 3.0)

2

CAL (CAN Application Layer)

- Fue una de las primeras especificaciones producidas por CiA (*CAN in Automation*)
- Basada en un protocolo existente desarrollado originalmente por Philips Medical Systems.
- Ofrece un ambiente orientado a objetos para el desarrollo de aplicaciones distribuidas de cualquier tipo, basadas en CAN.

3

2.1.1 CAL (CAN Application Layer)

Es una capa de aplicación desarrollada por Philips en su división de electromedicina (*Philips Medical Systems*) y adoptada y desarrollada posteriormente por CiA.

La capa CAL está compuesta por 4 servicios:

- **CMS** (*CAN-based Message Specification*)
- **NMT** (*Network ManagementT*)
- **DBT** (*DistriBuTor*)
- **LMT** (*Layer ManagementT*)

4

CAL provee 4 servicios de la capa de aplicación: CMS, NMT, DBT y LMT.

- **CMS (CAN-based Message Specification)**
Ofrece objetos de tipo Variable, Evento y Dominio para diseñar y especificar cómo se accede a la funcionalidad de un dispositivo (un nodo) a través de su interfaz CAN.
Ej.: transmitir más de 8 bytes

5

- **CMS (CAN-based Message Specification)** define 8 niveles de prioridad en sus mensajes, cada uno con 220 COB-IDs, ocupando desde el 1 al 1760.
Los identificadores restantes (0, 1761-2031) se reservan para NMT, DBT y LMT:

COB-IDs (Communication Object Identifier)

CAL	
COB-ID	Función
0	NMT start/stop services
1 – 220	CMS objects priority 0
221 – 440	CMS objects priority 1
441 – 660	CMS objects priority 2
661 – 880	CMS objects priority 3
881 – 1100	CMS objects priority 4
1101 – 1320	CMS objects priority 5
1321 – 1540	CMS objects priority 6
1541 -1760	CMS objects priority 7
1761 – 2015	NMT Nodo Guarding
2016 - 2031	NMT, LMT,DBT services

Distribución de los COB-ID (11-bit) en CAL.

6

COB-ID = Identificativo del mensaje

1/ **CMS (CAN Based Message Specification)**

- **NMT (*Network Management*)**

Proporciona servicios para la gestión de la red: inicializar, arrancar o parar nodos, o detectar fallos.

Está implementado en el concepto de maestro-esclavo, sólo un NMT maestro.

7

2/ **NMT (*Network Management*)**: proporciona servicios para la gestión de la red: inicializar, arrancar o parar nodos, cambiar y supervisar el estado de los nodos, detectar fallos de nodo, estado de la comunicación, escritura y lectura de datos de configuración.

Este servicio está implementado en el formato maestro-esclavo, sólo un NMT maestro.

Se parte de que cualquier nodo ha de contener en origen un conjunto mínimo de objetos predefinidos. A partir de este conjunto, el nodo puede adquirir una configuración más concreta desde un nodo remoto que actúe como configurador.

Un mensaje NMT es un mensaje con identificador 0. Esto otorga al mensaje NMT la máxima prioridad posible. El mensaje NMT siempre consta de una carga útil de dos bytes en la trama CAN. El segundo byte contiene el ID del nodo al que se dirige. Es posible dirigirse a todos los dispositivos con un mensaje NMT con el ID 0 de nodo reservado.

La configuración de un nodo queda definida en dos documentos electrónicos escritos en ASCII y que pueden ser datos de entrada para sistemas de configuración y prueba de confirmación de un nodo:

- La hoja de datos electrónica EDS (*Electronic Data Sheet*), que establece el mapa de objetos,
- El archivo de descripción de dispositivo DCF (*Device Configuration Files*) que determina valores para esos objetos.

- **DBT** (*DistriBuTor*)

Ofrece una distribución dinámica de los identificadores CAN: COB-ID (*Connection Object Identifier*).

Está implementado en el concepto de maestro-esclavo, sólo un DBT maestro.

8

3/ **DBT** (*DistriBuTor*)

- **LMT** (*Layer Management*)

Permite cambiar ciertos parámetros de las capas: ej.: la dirección-NMT de un nodo, la velocidad del bus, etc.

9

4/ **LMT** (*Layer Management*)

- Este estándar asume CAN2.0A (*CAN Standard Message Frame*) con identificadores de 11 bits.
- Se puede utilizar CAN2.0B (*CAN Extended Message Frame*) con identificadores de 29 bits.
- En ese caso, se mapean los 11 bits definidos anteriormente con los 11 bits más significativos del identificador de tramas extendidas.
- De esta manera los rangos de la tabla quedan más amplios.

10

NMT, DBT y LMT son servicios de configuración y gestión del bus de campo. Se gestionan de acuerdo a un protocolo maestro-esclavo y solo se implementa la parte del maestro en nodos con capacidad de configuración y gestión.

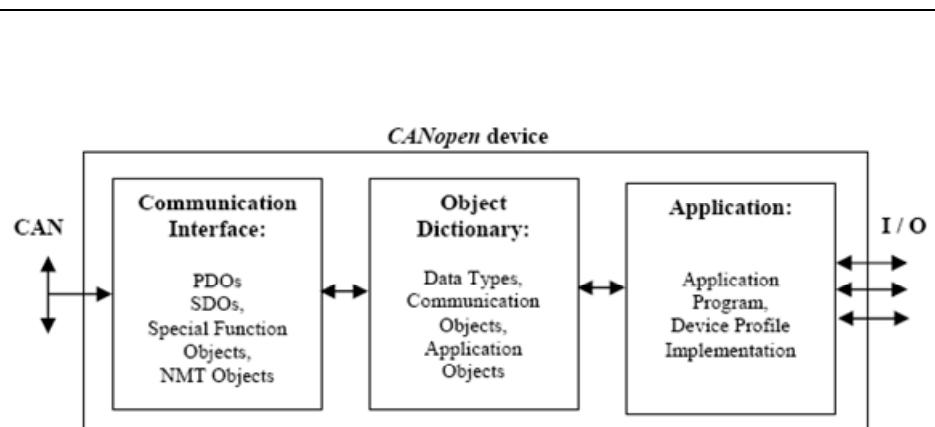
CMS es un servicio básico de cualquier nodo.

CANopen

- CAL (*CAN Application Layer*) proporciona todos los servicios de gestión de red y mensajes del protocolo pero no define los contenidos de los objetos CMS ni los tipos de objetos (define cómo, pero no qué).
- Aquí es donde CANopen entra en juego.
- CANopen está construido por encima de CAL, utilizando un subconjunto de sus servicios y protocolos de comunicación.
- Proporciona una implementación de un sistema de control distribuido utilizando los servicios y protocolos de CAL.
- El concepto central de CANopen es el **DICCIONARIO DE OBJETOS (OD: Object Dictionary)**.
Es un concepto utilizado por otros buses de campo.
No forma parte de CAL.

11

2.2 CANopen



12

Un equipo CANopen se puede dividir en tres partes:

- **Comunication interface:** provee los servicios de envío y recepción de objetos de comunicación por el bus.
- **Object dictionary:** define todos los tipos de datos, objetos de comunicación y objetos de aplicación utilizados por el equipo. Esta información se encuentra en el archivo EDS.
- **Application:** Contiene la funcionalidad de control del equipo y la interfaz con el hardware del equipo.

OD: DICCIONARIO de OBJETOS

(CanOpen Objet Dictionary)

Es el elemento más importante al definir la funcionalidad del nodo.

Es 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.

13

2.3 OD: DICCIONARIO DE OBJETOS (CANopen OBJET DICTIONARY)

DICCIONARIO DE OBJETOS de CANopen (OD)

- Es un grupo ordenado de objetos.
- Describe completamente y de forma estandarizada la funcionalidad de cada dispositivo y permite su configuración mediante mensajes (SDO) a través del propio bus.
- Cada objeto se direcciona utilizando un índice de 16 bits.
- Para permitir el acceso a elementos individuales de las estructuras de datos también existe un subíndice de 8 bits.

14

El OD de un dispositivo CANopen es el elemento más importante al definir la funcionalidad del nodo. El OD 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.

Contiene todos los parámetros del equipo que son accesibles mediante comunicaciones por medio del bus.

Cada objeto del diccionario se puede acceder mediante:

- Un índice de 16-bit
- Un subíndice de 8-bit.

Estructura General del Diccionario Objeto CANopen	
Index	
0000	<i>not used</i>
0001 - 001F	Static Data Types (standard data types, e.g. Boolean, Integer16)
0020 - 003F	Complex Data Types (predefined structures composed of standard data types, e.g. PDOCommPar, SDOParameter)
0040 - 005F	Manufacturer Specific Complex Data Types
0060 - 007F	Device Profile Specific Static Data Types
0080 - 009F	Device Profile Specific Complex Data Types
00A0 - 0FFF	<i>reserved</i>
1000 - 1FFF	Communication Profile Area e.g. Device Type, Error Register, Number of PDOs supported
2000 - 5FFF	Manufacturer Specific Profile Area
6000 - 9FFF	Standardised Device Profile Area e.g. "DSP-401 Device Profile for I/O Modules" [3]: Read State 8 Input Lines, etc.
A000 - FFFF	<i>reserved</i>

Nombre simbólico	Descripción	Código
NULL	Entrada vacía	0
DOMAIN	Segmentos de datos extenso y sin estructura (por ejemplo un programa cargable)	2
DEFTYPE	Definiciones de tipos simples	5
DEFSTRUCT	Definición de estructura de 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

Clases de objeto

En la primera parte del OD 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. El rango relevante de objetos va desde el índice 1000 al 9FFF.

CANopen está definido en documentos que describen perfiles. Un perfil define para cada objeto: su función, nombre, índice, subíndice, tipos de datos, si es obligatorio u opcional, si es de ‘sólo lectura’, ‘sólo escritura’ o ‘lectura / escritura’, etc. Los perfiles definen que objetos OD son obligatorios y cuáles no. En el caso más general, una entrada del OD es traducible a una estructura (o registro) en un lenguaje de programación.

- Hay un perfil de comunicaciones (*communication profile*) donde están descritos todos los parámetros relacionados con las comunicaciones.
- Hay varios perfiles de dispositivos (*device profiles*) donde se definen los objetos de un dispositivo en particular.

La descripción de un objeto con un índice dado incluye:

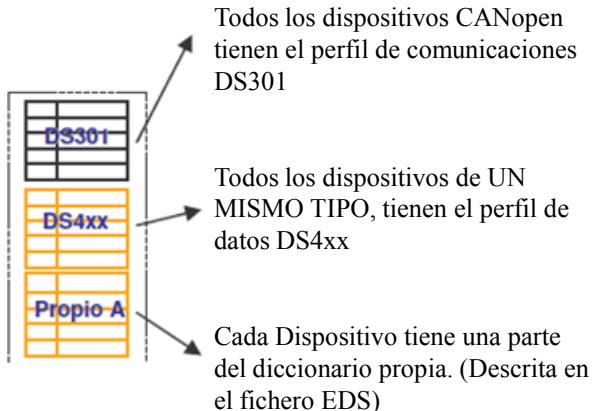
- El nombre de clase (DEFTYPE, VAR, ARRAY, etc.) que determina un código numérico de objeto que establece su funcionalidad
- El nombre del objeto con una descripción textual
- El tipo de la entrada de diccionario. Puede estar predefinida
- Los atributos de acceso (lectura, escritura, lectura-escritura: constante, para entrada de proceso, 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

PERFILES

La parte de configuración de las comunicaciones sigue la norma DS 301

La parte de datos sigue la norma DS 4xx (xx dependiendo del tipo de dispositivo)

Existe una tercera parte propia del dispositivo (Definida en el fichero EDS)



15

Hay una serie de parámetros comunes a todo equipo CANopen que vienen definidos según el perfil del equipo: DS301, DS4xx.

Todo equipo con perfil DS4xx contiene los parámetros del perfil DS301.

Perfiles estándar CANopen:

- CANopen communication profile (DSP-301): (perfil básico) define una serie de parámetros necesarios para comunicar con un equipo CANopen. Todos los equipos CANopen requieren tener implementado el DSP-301. Índices asignados dentro del OD en el rango 1000h-1FFFh
- CANopen device profiles (DSP-4xx): define la funcionalidad de un tipo de equipo en particular y cómo acceder a esa funcionalidad por medio del bus CAN. Los perfiles disponibles hasta ahora son: DSP 401 (módulos E/S), DSP 402 (Control de drives y Motion), DSP 403 (HMI), DSP 405 (Equipos programables IEC1131) y DSP 406 (Encoders). Índices asignados dentro del OD en el rango 6000h-9FFFh
- CANopen Application profiles: Define la funcionalidad y las relaciones de comunicación de todos los equipos CANopen del bus. Perfiles de aplicación disponibles: DS414 (maquinaria textil), DS417 (Ascensores) y DS422 (Sistemas municipales).

Profile number	Device class
DS-401	Generic I/O modules
DS-402	Drives and motion control
DS-403	Not allocated
DS-404	Measuring devices and closed loop controllers
DS-405	IEC 61131-3 programmable devices
DS-406	Encoders
DS-407	Public transportation - Passenger Information Systems
DS-408	Fluid Power Technology - Hydraulic drives and proportional valves
DS-409	Vehicle door control
DS-410	Declinometers
DS-412	Medical Devices
DS-413	Truck Gateways
DS-414	Weaving Machines
DS-415	Road Construction Machinery
DS-416	Building Door Control
DS-417	Lift Control Systems
DS-418	Battery Modules
DS-419	Battery Chargers
DS-420	Extruder Downstream Devices

En la capa aplicación se definen los siguientes grupos de perfiles:

Hay una serie de parámetros que son de libre definición por parte del fabricante.

La descripción del OD se realiza en forma de un archivo EDS (*Electronic Data Sheet*) en formato ASCII. Este archivo sigue una sintaxis estricta y se puede utilizar en todos los configuradores de bus CANopen (Sycon etc.).

Para que el Maestro pueda conocer el Diccionario de Objetos de cada nodo, necesita poder acceder al EDS localmente:

Indice	Objeto	Descripción
0000	No usado	
0001 - 001F	Tipos de datos estáticos <i>(Static Data Types)</i>	Contiene definiciones de tipos de dato estándar como boolean, enteros 16, string, punto flotante, etc. Se incluyen como referencia no pueden ser leídas ni escritas
0020 - 003F	Tipos de datos complejos <i>(Complex Data Types)</i>	Contiene definiciones de estructuras predefinidas, compuestas de tipos estáticos, comunes a todos los dispositivos. e.g. PDOCommPar, SDOParameter
0040 - 005F	Tipos de datos específicos del fabricante <i>(Manufacturer Specific Complex Data Types)</i>	Contiene definiciones de estructuras predefinidas, compuestas de tipos estáticos, específicas de un dispositivo en particular.
0060 - 007F	Tipos de datos estáticos específicos del perfil del dispositivo <i>(Device Profile Specific Static Data Types)</i>	Definiciones de tipos de datos básicos específicos para el perfil del dispositivo.
0080 - 009F	Tipos de datos complejos específicos del perfil del dispositivo <i>(Device Profile Specific Complex Data Types)</i>	Definiciones de estructuras específicas para el perfil del dispositivo
00A0-0FFF	reservado	
1000 - 1FFF	Rango para el perfil de comunicaciones Communication Profile Area (e.g. Device Type, Error Register, Number of PDOS supported)	Contiene parámetros de configuración del bus CAN. Estas entradas del diccionario son comunes a todos los dispositivos
2000 - 5FFF	Rango para el perfil específico del fabricante <i>Manufacturer Specific Profile Area</i>	Contiene las extensiones al perfil estándar realizadas por el fabricante.
6000 - 9FFF	<i>Standardised Device Profile Area (e.g. "DSP-401 Device Profile for I/O Modules" : Read State 8 Input Lines, etc.)</i>	Contiene todos los objetos de datos comunes a un tipo de perfil que pueden ser leídos o escritos desde la red. Algunas de las entradas son obligatorias (funcionalidad requerida) mientras que otras son opcionales (funcionalidad opcional).
A000 - FFFF	Reservado	

13

- El rango relevante de objetos va del índice 1000 al 9FFF.
- Para cada nodo existe un OD, diccionario de objetos, que contiene todos los parámetros que describen el dispositivo y su comportamiento en la red.
- CANopen está definido en documentos que describen perfiles

18

- Un perfil define para cada objeto:
 - Su función
 - Nombre
 - Índice
 - Subíndice
 - Tipos de datos
 - Si es obligatorio u opcional
 - De “sólo lectura”, “sólo escritura” o “lectura / escritura”
 - etc.

19

- Hay un perfil de comunicaciones (*communication profile*) donde están descritos todos los parámetros relacionados con las comunicaciones.
- Hay varios perfiles de dispositivos (*device profiles*) donde se definen los objetos de un dispositivo en particular.
- Los perfiles definen que objetos OD son obligatorios y cuales no.

20

COMUNICACIÓN CANopen

El modelo de comunicaciones de CANopen define

4 tipos de mensajes (objetos de comunicación):

- **Mensajes (Objetos) Administrativos**

- *Service Data Objects (SDO)*

- *Process Data Objects (PDO)*

-**Mensajes Predefinidos o Especiales**

21

COMUNICACIÓN CANopen

El modelo de comunicaciones de CANopen define 4 tipos de mensajes (objetos de comunicación)

Mensajes Administrativos

Configuran las distintas capas de la red:

- Inicialización
- Configuración
- Supervisión de la misma.

Se basa en los servicios LMT, NMT y DBT de la capa CAL

(Layer Management) (Network Management) (DistriBuTor)

22

1/ Mensajes Administrativos (*Administrative message*)

Mensajes: Service Data Objects (SDO)

- Objetos o mensajes de servicio, utilizados para leer y escribir cualquiera de las entradas del Diccionario de Objetos de un dispositivo.
- Corresponden a mensajes CAN de baja prioridad.
- Son mensajes de más de 8 bytes

23

2/ Objetos de comunicación para transferencia de datos de servicio u objetos de comunicación explícita (**SDO: Service Data Objects**). Tienen como fin la transmisión de parámetros y la petición de lectura y escritura. Ofrecen acceso al diccionario de objetos de los dispositivos remotos.

Mensajes: Process Data Objects (PDO)

- Objetos o mensajes de proceso, utilizados para el intercambio de datos de proceso, es decir, **datos de tiempo real**.
- Corresponden a mensajes CAN de alta prioridad.
- Mensajes de 1 a 8 bytes

24

3/ Objetos de comunicación para transferencia de datos de proceso u objetos de comunicación implícita que se utilizan para leer/escribir rápidamente datos de proceso, para aplicaciones de tiempo real (**PDO: Process Data Objects**)

Utiliza el modelo productor/consumidor de CAN ampliado mediante transferencias síncronas:

Mensajes Predefinidos o Especiales

- De sincronización, de Emergencia y *Time Stamp*.
- Permiten sincronizar los dispositivos (objetos SYNC) y generar notificaciones de emergencia en forma opcional.

25

4/ Mensajes Predefinidos o Especiales

Comparación entre SDOs y PDOs

SDOs (<i>Service Data Objects</i>)	PDOs (<i>Process Data Objects</i>)
<ul style="list-style-type: none">- Información de mayor volumen (se pueden realizar transferencias de más de 8 bytes)- Transferencia de baja prioridad- Se direccionan en el OD (índice + sub índice)- Transferencia asíncrona	<ul style="list-style-type: none">- Información de menor volumen (transferencia de menos de 8 bytes)- Transferencia de alta prioridad- Diversos objetos se pueden mapear en una trama CAN- Transferencia síncrona, asíncrona y cíclica (periódica)

26

SDO (*Service Data Objects*):

- Configuración de dispositivos y transferencia de muchos datos.
- Comparando con los PDOs, son mensajes de baja prioridad.
- Los SDO, para acceder al OD del dispositivo, implementan modelos de comunicación cliente-servidor o punto-a-punto.
- Para un SDO, el servidor es el dispositivo cuyo diccionario está siendo accedido, mientras que el otro dispositivo, el que inicia la actividad, es el cliente.
- Los mensajes son de 8 bytes, aunque no es necesario que todos los bytes sean datos significativos.
- Los mensajes requieren confirmación.
- Cada SDO implica el intercambio de 2 tramas CAN con diferentes identificadores.

27

2.4 SDOs: SERVICE DATA OBJECTS

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 no tiene límites de longitud. Si la carga útil no cabe en la trama CAN, se dividirá en varias tramas CAN. Es posible acceder a los datos de los nodos mediante SDOs en cualquier momento y sin haber configurado nada previamente.

Un SDO se soporta sobre dos mensajes CAN (2 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 tengan que ser completados con datos inútiles.

En un SDO se incluye un índice y un subíndice para direccionamiento del OD. Por medio de esta 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 por la norma:

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

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

En una red CANopen se pueden tener hasta 256 canales SDO (2 identificadores por canal).

Configuración de dispositivos y transferencia de cantidad de datos.

Comparando con los PDOs, los SDO son mensajes de baja prioridad. Los SDO implementan modelos de comunicación cliente-servidor o punto-a-punto, para acceder al diccionario objeto del dispositivo. Para un SDO, el dispositivo a cuyo diccionario está siendo accedido es el servidor mientras que el otro dispositivo, el que inicia la actividad, es el cliente.

Los mensajes son de 8 bytes, aunque no es necesario que todos los bytes sean datos significativos. Los mensajes requieren confirmación. Cada SDO implica el intercambio de 2 tramas CAN con identificadores diferentes.

- Hay 2 mecanismos para transferir SDOs:

– Transferencia Expedita:

Transmite mensajes con longitud de datos menor o igual a 4 *bytes*

Se envía un único mensaje CAN y se recibe la confirmación del servidor

– Transferencia en Segmentos:

Transmite mensajes de más de 4 *bytes*

Se aplica fragmentación, se parten los datos en múltiples mensajes

El cliente espera confirmación del servidor por cada segmento

28

Un SDO que se transmite del cliente al servidor bajo este protocolo tiene el siguiente formato:

Client → Server / Server → Client			
<i>Byte 0</i>	<i>Byte 1-2</i>	<i>Byte 3</i>	<i>Byte 4-7</i>
<i>SDO Command Specifier</i>	<i>Objet Index</i>	<i>Objet Subindex</i>	**

** hasta 4 bytes de datos en transferencia expedita, o un contador de 4 bytes en transferencia segmentada o parámetros relacionados con la transferencia de bloques

Client → Server / Server → Client	
<i>Byte 0</i>	<i>Byte 1-7</i>
<i>SDO Command Specifier</i>	Hasta 7 bytes de datos (<i>segment transfer</i>)

Los SDOs siempre acceden a datos de aplicación a través del OD. 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.

SDO: Transferencia Explícita

Un SDO que se transmite del cliente al servidor en este protocolo tiene el siguiente formato:

- **Command Specifier** (1 byte): contiene información sobre si es subida o descarga de datos, petición o respuesta, transferencia expedita o en segmentos, tamaño de los datos y el *toggle bit*:
 - *Client Command Specifier* (3 bits): 001 para *download* y 010 para *upload*.
 - Ignorado (1 bit): se pone a 0.
 - Número de *bytes* / Ignorado (2 bits): número de *bytes* que no contienen datos. Es válido si los siguientes dos bits son 1. Si no, vale 0. En *upload* se ignora.
 - *Transfer Expedited* / Ignorado (1 bit): indica si se trata de una transferencia expedita (=1) o una transferencia en segmentos (=0). En *upload* se ignora.
 - *Block Size Indicated* / Ignorado (1 bit): si está a '1' es que el tamaño de los datos está especificado, si no, no. En *upload* se ignora.

Client → Server / Server → Client

Byte 0	Byte 1-2	Byte 3	Byte 4-7
SDO Command Specifier	Objet Index	Objet Subindex	Dato

29

Transferencia Explícita

- **Command Specifier** (1 byte): contiene información sobre si es subida o descarga de datos, petición o respuesta, transferencia expedita o en segmentos, tamaño de los datos y el *toggle bit*:
(3 bits <7:5>) *Client Command Specifier*: 001 para *download* y 010 para *upload*.
(1 bit <4>) Ignorado: se pone a 0.
(2 bits <3:2>) Número de bytes / Ignorado: número de *bytes* que no contienen datos. Es válido si los siguientes dos *bits* son 1. Si no, vale 0. En *upload* se ignora.
(1 bit <1>) *Transfer Expedited* / Ignorado: indica si se trata de una transferencia expedita (a 1) o una transferencia en segmentos (a 0). Tiene que estar siempre a 1 para este tipo de transferencia. En *upload* se ignora.
(1 bit <0>) *Block Size Indicated* / Ignorado: si está a 1 es que el tamaño de los datos está especificado, si no, no. En *upload* se ignora.
(4 bytes) Ignorado / Datos: si es un *download* (escritura) se ignora ya que es una confirmación. Si es una lectura o *upload*, contiene el valor leído del diccionario de objetos del servidor

- **Index** (2 bytes): índice de la entrada del diccionario de objetos del servidor que el cliente desea acceder mediante el SDO actual.
- **Subindex** (1 byte): subíndice de la entrada del diccionario de objetos del servidor que el cliente desea acceder.
- **Datos / Ignorado** (4 bytes): datos que se desean enviar al servidor (el valor de una entrada en el diccionario si se está escribiendo). Si es *upload* se ignora.

Client → Server / Server → Client

Byte 0	Byte 1-2	Byte 3	Byte 4-7
<i>SDO Command Specifier</i>	<i>Objet Index</i>	<i>Objet Subindex</i>	<i>Dato</i>

30

- El servidor, al recibir el mensaje anterior, contesta con un mensaje con el siguiente formato:
- **Command Specifier** (1 byte):
 - *Server Command Specifier* (3 bits): 011 para *download* y 010 para *upload*.
 - Ignorado (1 bit): se pone a 0.
 - Ignorado / Número de bytes (2 bits): si es un *download* se ignora.
 - Ignorado / Transfer Expedited (1 bit): si es un *download* se ignora.
 - Ignorado / Block Size Indicated (1 bit): si es un *download* se ignora.
- Ignorado / Datos (4 bytes): si es un *download* (escritura) se ignora ya que es una confirmación.
- Si es una lectura o *upload*, contiene el valor leído del diccionario de objetos del servidor.

31

Transferencia en Segmentos:

Command Specifier (1 byte):

- *Client Command Specifier* (3 bits): 001 *download* y 000 *upload*.
- *Toggle Bit* (1 bit): vale ‘0’ o ‘1’ alternativamente segmentos consecutivos
- Nº de bytes / Ignorado (3 bits): indica el nº de bytes que no contienen datos, o cero si no especifica el tamaño.
- *Last Segment Indication* / Ignorado (1 bit): =1 si se trata del último segmento del SDO que se está transmitiendo y 0 si hay más segmentos.

Client → Server / Server → Client

Byte 0	Byte 1-7
<i>SDO Command Specifier</i>	Hasta 7 bytes de datos (<i>segment transfer</i>)

32

Transferencia en segmentos

- **Command Specifier (1 byte):**

(3 bits) *Client Command Specifier*: 001 para *download* y 000 para *upload*.

(1 bit) *Toggle Bit*: es un bit que vale 0 o 1 de forma alternada en segmentos consecutivos.

(3 bits) Nº de bytes / Ignorado: indica el nº de bytes que no contienen datos, o cero si no especifica el tamaño.

(1 bit) *Last Segment Indication* / Ignorado: =1 si se trata del último segmento del SDO que se está transmitiendo y 0 si hay más segmentos.

PDO (Process Data Objects):

- Intercambia datos en tiempo real. (de 1 a 8 bytes)
Un productor → a varios consumidores.
- Cada PDO se describe mediante 2 objetos del diccionario:
 - PDO *Communication Parameter*: contiene el COB-ID que utiliza el PDO, el tipo de transmisión, tiempo de inhibición y temporizador.
 - PDO *Mapping Parameter*: contiene una lista de objetos del OD contenidos en el PDO, incluyendo su tamaño en bits.
- El contenido de un mensaje PDO está predefinido

(COB: Connection Objects)

33

2.5 PDOs: PROCESS DATA OBJECTS (COMUNICACION IMPLÍCITA)

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 debe haberse definido previamente sin ambigüedad a través del OD. El intercambio de PDOs sigue el modelo productor-consumidor y se utilizan tanto los modelos ‘push’ como ‘pull’.

Cada PDO se describe mediante 2 objetos del diccionario:

- **PDO Communication Parameter**: contiene el COB-ID que utiliza el PDO, el tipo de transmisión, el tiempo de inhibición y el temporizador.
- **PDO Mapping Parameter**: contiene una lista de objetos del OD contenidos en el PDO, incluyendo su tamaño en *bits*.

El contenido de un mensaje PDO está predefinido.

Los PDOs pueden ser organizados con amplia flexibilidad y en la capa de aplicación se han contemplado tipos y modos de inicialización de PDO que permiten configurar sistemas de adquisición 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

Un PDO de un dispositivo tiene un COB-ID único.

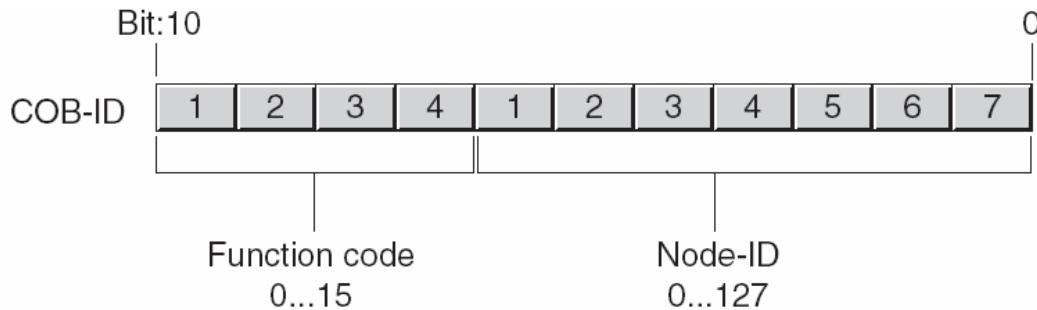
El COB-ID, en realidad es el Identificador del mensaje CAN y es una combinación entre el número de nodo CANopen y el servicio de comunicaciones que se desea (4+7 bits). Corresponde a los canales por los que los dispositivos envían y reciben información.

Tiene dos funciones principales:

- Arbitraje del bus especificando las prioridades de transmisión. El objeto de comunicación con el COB_ID más bajo es el de mayor prioridad en la red
- Identificación de los objetos de comunicación, ya que el equipo de red reconoce el tipo de servicio que se demanda (NMT, SYNC, TxPDO, RxPDO , SDO, EMCY)

Se compone de 2 partes:

- Código de función, 4 bits
- Node-ID, 7 bits



Los PDOs se configuran en el OD por medio de entradas de rango 0x1400 – 0x1BFF en los que se fijan dos entradas de diccionario por PDO: el parámetro de configuración y el de 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.

Índice	Subíndice	Descripción	Tipo de datos
1xxx	0x00	Nº Objetos Mapeados	Unsigned8
	0x01	Objeto 1	Unsigned32
	0x02	Objeto 2	Unsigned32
	Unsigned32
	N	Objeto n	Unsigned32

Formato del objeto de mapeo

31	18	15	8	7	0
Indice de 16 bits		Subíndice de 8 bits		Longitud	

Formato de campo de objeto:

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.

- CANopen define varios mecanismos de comunicación para la transmisión de PDOs:
 - **Transmisión asíncrona:**
 - **Transmisión sincrónica:**

34

PDO: TRANSMISIÓN ASÍNCRONA:

- La transmisión se produce por una solicitud remota.
- La transmisión se produce por un evento específico definido en el perfil del dispositivo.

35

- **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 la aplicación) o por un evento relacionado con un objeto de comunicación, bien sea agotamiento de una temporización asíncrona o disparo por una solicitud remota. Es decir por una solicitud remota o por un evento específico definido en el perfil del dispositivo.

PDO → TRANSMISIÓN SINCRÓNICA:

- **APERIODICA:**
 - La transmisión se dispara por petición remota de otro dispositivo
 - La transmisión se dispara por un evento del nodo especificado en el perfil del dispositivo.
- **PERIODICA:**
 - Periódicamente, cada “n” (1-240), se transmiten mensajes

36

La transmisión puede organizarse desde diferentes categorías:

- **Transmisión síncrona.** La transmisión se inicia sólo ante la 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 la 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 y transmisión síncrona aperiódica.
 - **Transmisión síncrona periódica o cíclica:** se transmiten periódicamente en cada instante de sincronización o cada ‘n’ (1-240) mensajes
 - **Transmisión síncrona aperiódica:** en la que la transmisión, ante la recepción de mensaje de sincronización, sólo se inicia si se ha producido un determinado evento en el nodo.

Distintos modos de transmisión de PDOs, definidos por el *Transmission Type* (entero de 8 bits) del *Communication Parameter*

Trans-Mission Type	Condition to trigger PDO (B=both needed, O=one or both)			PDO Transmission
	SYNC*	RTR*	Event*	
0	B	-	B	Sync, acyclic
1-240	O	-	-	Sync, cyclic
241-251	-	-	-	reserved
252	B	B	-	Sync, after RTR
253	-	O	-	Async, after RTR
254	-	O	O	Async, manufacturer specific event
255	-	O	O	Async, device profile specific event

*SYNC = objeto SYNC recibido

*RTR = recibida trama RTR

*Event = cambio de valor de un dato, temporizador...

37

- Un PDO puede tener asignado un tiempo de inhibición que define el tiempo mínimo que debe pasar entre 2 transmisiones consecutivas del mismo PDO.
Forma parte del *Communication Parameter*.
Está definido como un entero de 16 bits en unidades de 100 microsegundos.
- Un PDO puede tener asignado un tiempo, cuando la transmisión es periódica, como entero de 16 bits en unidades de 1 ms

38

Se pueden configurar tiempos de inhibición y generación de PDOs de forma que se creen ventanas virtuales de tiempo asignadas a cada PDO. De esta forma se consigue que la información se actualice en los instantes correctos sin sobrecargar el bus más de lo necesario.

Un PDO puede tener asignado un tiempo de inhibición que define el tiempo mínimo que debe pasar entre 2 transmisiones consecutivas del mismo PDO. Forma parte del *Communication Parameter*. Está definido como un entero de 16 bits en unidades de 100 microsegundos.

Un PDO puede tener asignado un tiempo, cuando la transmisión es periódica, como entero de 16 bits en unidades de 1 ms.

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

Los PDOs se configuran a través del OD. Esta configuración 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. Estos datos reflejarán uno o varios objetos de la aplicación. En el diccionario de objetos se define el parámetro de comunicación del PDO.
- 2/ Fijar los parámetros del PDO: Identificador del mensaje, tipo de transmisión, etc. Se define en el OD 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 propiedades 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.

MENSAJES PREDEFINIDOS y ESPECIALES

De SINCRONIZACION (SYNC)

- En una red CANopen, hay un dispositivo que es el productor de objetos SYNC y una serie de dispositivos consumidores.
- Cuando los consumidores reciben el mensaje del productor, abren su ventana de sincronismo en la que pueden ejecutar sus tareas sincrónicas.
- Esto permite coherencia temporal y coordinación entre dispositivos.

39

2.6 MENSAJES PREDEFINIDOS Y OBJETOS ESPECIALES

En una red CANopen existen algunos objetos singulares que se recogen en el OD, entre ellos:

Objeto de Sincronización (SYNC) (0x1005).

Elemento básico de sincronización de PDOs, se transmite según el concepto maestro-esclavo desde un nodo único. Es un mensaje de alta prioridad sin datos.

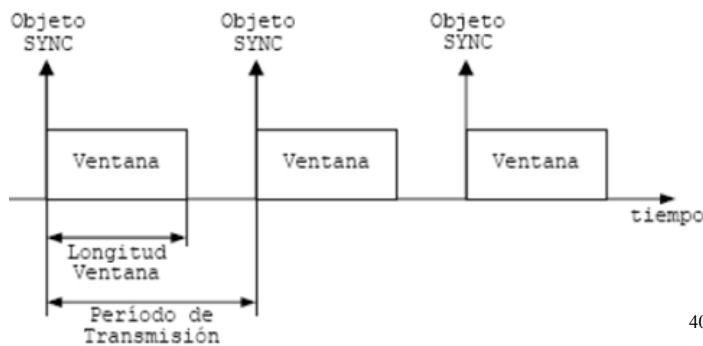
En una red CANopen, hay un dispositivo que es el productor de objetos SYNC y una serie de dispositivos consumidores.

SYNC sincroniza las acciones de los nodos. Contiene el COB-ID 128, lo que puede generar algo de imprecisión debido a la prioridad.

Cuando los consumidores reciben el mensaje del productor, abren su ventana de sincronismo en la que pueden ejecutar sus tareas sincrónicas. Esto permite coherencia temporal y coordinación entre dispositivos.

Ejemplo:

- Un conjunto de sensores leen las variables del proceso controlado en forma coordinada ...
- El comportamiento de estos mensajes está determinado por 2 parámetros:
 - **La longitud de la ventana (Synchronous Window Length)** puede ubicarse en la entrada 0x1007 del diccionario.
 - **El período de transmisión**, se encuentra en la posición 0x1006.



40

MENSAJES PREDEFINIDOS y ESPECIALES (II)

MENSAJE de Temporización Absoluta (*Time Stamp*) (0x1013)

- Proporciona una referencia de tiempo común

41

Temporización Absoluta (*Time Stamp*) (0x1013).

Es 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).

MENSAJES PREDEFINIDOS y ESPECIALES (III)

MENSAJE DE EMERGENCIA

- Se envían cuando ocurre un error interno en un dispositivo.
- Se transmiten al resto de dispositivos con la mayor prioridad.
- Pueden usarse como interrupciones o notificaciones de alertas.
- Se implementan como objetos CMS del tipo “*Stored Event*”

(CMS: CAN-based Message Specification)

42

Objeto de Emergencia EMCY.

Emitido ante un error o fallo (corriente, tensión, temperatura, comunicación, ...) interno del nodo. En la especificación se definen los códigos de error aplicables. La reacción de los consumidores de EMCY depende de la aplicación. El estándar CANopen define varios códigos de emergencia. El objeto EMCY se transmite en una sola trama de CAN con ocho bytes.

Un mensaje de emergencia tiene 8 *bytes*.

Emergency-sender ⇒ Emergency-receiver(s)

COB-ID	Byte 0-1	Byte 2	Byte 3-7
0x080 + Node_ID	Emergency Error Code	Error Register (Object 0x1001)	Manufacturer specific error field

Emergency Error Code (2 bytes): código del error que causó la generación del mensaje de emergencia.

- Se trata de fallos internos de los dispositivos por lo cual, los errores se relacionan con fallos de tensión, de corriente, del software del dispositivo, del adaptador del bus CAN, etc.

43

Emergency Error Code	Meaning
00xx	Error Reset or No Error
10xx	Generic Error
20xx	Current
21xx	current, device input side
22xx	current, inside the device
23xx	current, device output side
30xx	Voltage
31xx	mains voltage
32xx	voltage inside the device
33xx	output voltage
40xx	Temperature
41xx	ambient temperature
42xx	device temperature
50xx	Device hardware
60xx	Device software
61xx	internal software
62xx	user software
63xx	data set
70xx	Additional modules
80xx	Monitoring
81xx	Communication
8110	CAN overrun
8120	Error Passive
8130	Life Guard Error or Heartbeat Error
8140	Recovered from Bus-Off
82xx	Protocol Error
8210	PDO not processed due to length error
8220	Length exceeded
90xx	External error
F0xx	Additional functions
FFxx	Device specific

Clasificación de los tipos de errores

Error Register (1 byte):

- Entrada con índice 0x1001 del diccionario de objetos.
- Cada bit=1 del registro indica una condición de error distinta

Bit	Error type
0	generic
1	current
2	voltage
3	temperature
4	communication
5	device profile specific
6	reserved (=0)
7	manufacturer specific

Emergency-sender ⇒ Emergency-receiver(s)

COB-ID	Byte 0-1	Byte 2	Byte 3-7
0x080 + Node_ID	Emergency Error Code	Error Register (Object 0x1001)	Manufacturer specific error field

45

Registro de error (Error Register) (0x1001) (1 byte).

Manufacturer-specific Error Field (5 bytes): Puede usarse para información adicional sobre el error.

Los datos incluidos y su formato son definidos por el fabricante del dispositivo.

MENSAJES PREDEFINIDOS y ESPECIALES (IV)

Mensajes de Node/Life Guarding

- Se utilizan para saber si un nodo está operativo.
- La comunicación se basa en el concepto de maestro-esclavo.
- El NMT maestro monitoriza el estado de los nodos (*node guarding*). Opcionalmente los nodos pueden monitorizar el estado del NMT maestro (*life guarding*).
- Comienza en el NMT esclavo después de haber recibido el primer mensaje de *node guarding* del NMT maestro.
- Sirve para detectar errores en las interfaces de red de los dispositivos, pero no fallos en los dispositivos en sí (ya que estos son avisados mediante mensajes de emergencia).
- Se puede implementar de dos maneras distintas: NMT *Node Guarding* o *Heartbeat*, pudiendo utilizarse sólo una de ellas a la vez.

NMT: Network ManagementT

46

Mensajes de Node/Life Guarding

NMT Node Guarding

- El maestro pregunta, cada cierto tiempo, a los esclavos si están activos.
- Si alguno no contesta en un tiempo determinado significa que está caído y se informa de ello.
- Si los esclavos también monitorizan al maestro, informan de que el maestro está caído si no reciben mensajes de *Node Guarding* de él durante un determinado intervalo.
- El maestro interroga a los esclavos mediante una trama remota (RTR) con la siguiente estructura:



47

NMT Node Guarding

Los NMT esclavos responden con el siguiente mensaje:

COB-ID	Byte 0
0x700 + Node_ID	bit 7: <i>toggle</i> , bit 6-0: <i>state</i>

- El *toggle bit* (bit 7) es un bit que va alternando su valor en cada mensaje

48

COB-ID	Byte 0
0x700 + Node_ID	bit 7: <i>toggle</i> , bit 6-0: <i>state</i>

Los bits del 0 al 6 indican el estado del nodo.

Value	State
0	Initialising
1	Disconnected *
2	Connecting *
3	Preparing *
4	Stopped
5	Operational
127	Pre-operational

49

HEARTBEAT

- Cada nodo manda un mensaje *Heartbeat* cada cierto tiempo para informar de que está operativo.
- En este caso, el mensaje de *Boot-up* se considera que es el primer mensaje de *Heartbeat*.
- Si el NMT maestro deja de recibir estos mensajes durante un tiempo determinado significará que el nodo está caído.
- Tienen la siguiente estructura:

COB-ID	Byte 0
0x700 + Node_ID	<i>state</i>

<i>state</i>	Meaning
0	Boot-up
4	Stopped
5	Operational
127	Pre-operational

50

HEARTBEAT

GESTION de RED

- La gestión de red en CANopen (NMT) sigue una relación maestro-esclavo basada en nodos.
- Requiere la existencia de un dispositivo en la red que ejerza la función de maestro (NMT-maestro).
- Cualquier otro nodo tiene funcionalidad NMT-esclavo, definida por su identificador de nodo.
- Se consideran los siguientes grupos funcionales:
 - Servicios de inicialización de nodos
 - Servicios de supervisión
 - Servicios de configuración

51

2.7 GESTIÓN DE RED

Iniciación y Comutación de estado

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

52

Iniciación y Comutación de Estado

- El NMT-Maestro emite órdenes de cambio de estado del tipo:
 - Arranque de nodo
 - Parada de nodo
 - Paso a pre-operacional
 - Reinicialización de nodo
 - Reinicialización de comunicación.

53

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'*)

Supervision del Estado de los Nodos

- Se basa en un doble proceso:
 - El NMT-Maestro detecta dispositivos ausentes manteniendo una base de datos.
 - 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.

54

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.

A su vez, los NMT-esclavos, comprueban constantemente la existencia de un NMT-Maestro en la red. Se ha definido también una función alternativa de latido (*Hearbeat*) basada en relaciones productor consumidor. El productor de latido emite un mensaje cíclico que puede ser tenido en cuenta por los consumidores y dar lugar a la generación de un evento.

Configuración

- La configuración por defecto de CANopen se basa en la asignación de identificadores de mensajes a los 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.
- 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.

55

Configuración

Está definido un diccionario de objetos por defecto, que establece los objetos mínimos requeridos por un nodo CANopen antes de la configuración. Se incluye un objeto de emergencia, un SDO, un máximo de 4 PDOs de recepción y 4 de transmisión, un objeto de sincronización, un objeto de temporización, etc.

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 continua ampliación y evolución y pueden obtenerse de la organización CiA.

56

Los Perfiles

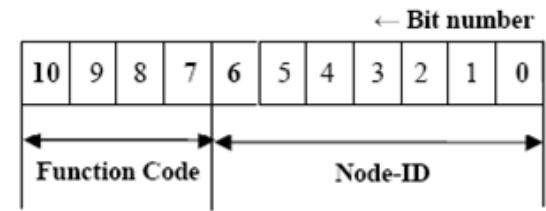
Los Perfiles CANopen Predefined Connection Set

- CANopen define la distribución de los identificadores de mensaje (*Predefined Connection Set*) de manera que hay un mensaje de emergencia por nodo, mensajes de sincronización y *time stamp*, 1 SDO (ocupando 2 identificadores), mensajes NMT y 4 PDOs de transmisión y 4 de recepción por dispositivo.

57

El identificador de 11 bits se divide en dos partes:

- 4 bits para el código de función
- 7 bits para el identificador de nodo (Node-ID)



- La distribución de los identificadores se corresponde con una estructura del tipo maestro - esclavo.
- El maestro conoce los Node-IDs de todos los esclavos conectados (máximo 127).
- Los esclavos no pueden comunicarse entre sí porque no conocen sus identificadores.

58

La distribución de los identificadores se corresponde con una estructura del tipo maestro - esclavo. El maestro conoce los Node-IDs de todos los esclavos conectados (máximo 127) y realiza la conexión punto a punto. Los esclavos no pueden comunicarse entre sí porque no conocen sus identificadores.

Broadcast objects of the CANopen Predefined Master/Slave Connection Set			
Object	Function code (ID-bits 10-7)	COB-ID	Communication parameters at OD index
NMT Module control	0000	000h	—
SYNC	0001	080h	1005h, 1006h, 1007h
TIME STAMP	0010	100h	1012h, 1013h

Peer-to-Peer objects of the CANopen Predefined Master/Slave Connection Set				
Object	Function code (ID-bits 10-7)	COB-ID *	COB-ID	Communication parameters at OD index
EMERGENCY	0001	081h -0FFh	000 1000 0001 - 000 1111 1111	1024h, 1015h
PDO 1 (transmit)	0011	181h -1FFh	001 1000 0001 - 001 1111 1111	1800h
PDO 1 (receive)	0100	201h -27Fh	010 0000 0001 - 010 0111 1111	1400h
PDO 2 (transmit)	0101	281h -2FFh	010 1000 0001 - 010 1111 1111	1801h
PDO 2 (receive)	0110	301h -37Fh	011 0000 0001 - 011 0111 1111	1401h
PDO 3 (transmit)	0111	381h -3FFh	011 1000 0001 - 011 1111 1111	1802h
PDO 3 (receive)	1000	401h -47Fh	100 0000 0001 - 100 0111 1111	1402h
PDO 4 (transmit)	1001	481h -4FFh	100 1000 0001 - 100 1111 1111	1803h
PDO 4 (receive)	1010	501h -57Fh	101 0000 0001 - 101 0111 1111	1403h
SDO (transmit/server)	1011	581h -5FFh	101 1000 0001 - 101 1111 1111	1200h
SDO (receive/client)	1100	601h -67Fh	110 0000 0001 - 110 0111 1111	1200h
NMT Error Control	1110	701h -77Fh	111 0000 0001 - 111 0111 1111	1016h, 1017h

* The COB-ID range covers the allowed Node-ID range from 1 to 127 (los 11 bits incluyen los 4 de Function Code).

CANopen Identifier Distribution

La localización de los identificadores CAN (COB-IDs) se puede realizar de tres formas

- Usando el Predefined Master/Slave Connection Set
- Modificando los PDO cuando el nodo está en estado Pre-Operational
- Con los servicios CAL DBT (DistriBuTor)

60

La localización de los identificadores CAN o COB-IDs, se puede realizar de tres formas diferentes:

- 1/ Usando el *Predefined Master/Slave Connection Set*: con una localización de los identificadores por defecto, sin necesidad de configuración; configuración del contenido de datos PDO (PDO mapping) si el dispositivo lo soporta.
- 2/ Modificando los identificadores PDO después del encendido, cuando el dispositivo está en estado Pre-Operacional
- 3/ Usando los servicios CAL DBT (*DistriBuTor*): nodos o esclavos conectados a una red CANopen son inicialmente identificados su Nodo-ID. El Nodo_ID puede configurarse por interruptores en el propio dispositivo o usando servicios de CAL LMT

Proceso BOOT-UP

- Aparte de los mensajes predefinidos, CANopen incluye una serie de mensajes para la administración y monitorización de los dispositivos en la red.
- Están implementados en la capa CAL y reciben el nombre de servicios de gestión de red (*Network ManagementT*, NMT).
- Se trabaja con un modelo de comunicaciones maestro-esclavo en el cual un dispositivo es el NMT maestro y el resto los NMT esclavos.

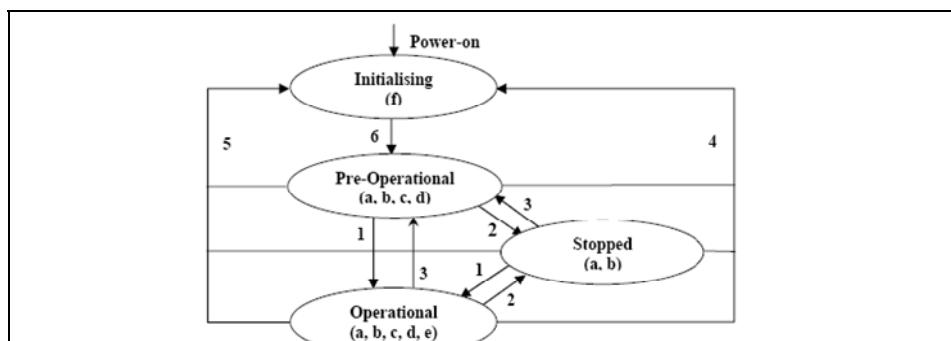
61

2.8 PROCESO BOOT-UP

En el proceso de inicialización de la red, CANopen soporta un proceso ‘extended boot-up’ que es opcional y un proceso ‘minimum boot-up’ que tiene que ser soportado por todos los nodos. BOOT-UP es una tarea adicional del objeto EMCY. Se envía con el COB-ID 700h+Node-ID y 1 byte de datos (00h).

Los servicios NMT se usan para llevar a todos o a los nodos seleccionados a varios estados de operación simultáneamente. Este mensaje CAN consiste de una cabecera con COB-ID=0 y dos bytes de datos: 1 byte con la petición de servicio (NMT *comand specifier*) y el segundo con el Node-ID o cero que direcciona a todos los nodos simultáneamente.

Los dispositivos que solo soportan *boot-up* mínimo, o dispositivos con capacidad mínima, entran en el estado *Pre-Operational* automáticamente



Las letras en paréntesis indican objetos permitidos en cada estado:

a. NMT, b. Node Guard, c. SDO, d. Emergency, e. PDO, f. Boot-up

Transiciones entre estados (mensajes NMT):

1: Start_Remote_Node (command specifier 0x01)

2: Stop_Remote_Node (command specifier 0x02)

3: Enter_Pre-Operational_State (command specifier 0x80)

4: Reset_Node (command specifier 0x81)

5: Reset_Communication (command specifier 0x82)

6: Inicialización terminada, entra en Pre-Operational al mandar mensaje de Boot-up

- Un dispositivo NMT esclavo puede encontrarse en alguno de los siguientes estados:
 - *Initialising*
 - *Pre-operational*
 - *Operational*
 - *Stopped*

63

Initialising:

- Al encender el dispositivo pasa directamente a este estado
- Después de realizar la inicialización el nodo transmite el mensaje de *Boot-up* y pasa al estado *Pre-Operational*

COB-ID	Byte 0
0x700 + Node_ID	0

64

Initialising

Pre-operational:

- En este estado el dispositivo puede ser configurado mediante SDOs.
- Puede enviar y recibir SDOs, mensajes de emergencia, de sincronización, *time stamp* y mensajes NMT.

65

Pre-operational

Operational:

- El dispositivo ya ha sido configurado y funciona normalmente.
- Todos los objetos de comunicación están activos, también puede enviar y recibir PDOs.

66

Operational

Stopped:

- Todos los objetos de comunicación dejan de estar activos
- No puede enviar ni recibir PDOs ni SDOs, sólo mensajes de NMT (Network ManagementT).

67

Stopped

NMT		000h
Sync	Emergency	080h
TimeStamp		100h
	PDO	180h
		200h
		280h
		300h
		380h
		400h
		480h
		500h
	SDO	580h
		600h
		680h
	Guarding	700h
		780h
LSS		7FFh

68

Communication object	COB-ID(s) hex	Slave nodes
NMT node control	000	Receive only
Sync	080	Receive only
Emergency	080 + NodeID	Transmit
TimeStamp	100	Receive only
PDO	180 + NodeID 200 + NodeID 280 + NodeID 300 + NodeID 380 + NodeID 400 + NodeID 480 + NodeID 500 + NodeID	1. Transmit PDO 1. Receive PDO 2. Transmit PDO 2. Receive PDO 3. Transmit PDO 3. Receive PDO 4. Transmit PDO 4. Receive PDO
SDO	580 + NodeID 600 + NodeID	Transmit Receive
NMT node monitoring (node guarding/heartbeat)	700 + NodeID	Transmit
LSS	7E4 7E5	Transmit Receive

69