UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

RED CONTROLADORA DE ÁREA (CAN): PROTOCOLO DE COMUNICACIÓN SERIAL DISTRIBUIDA.

M O N O G R A F Í A

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN SISTEMAS COMPUTACIONALES

P R E S E N T A

ANGEL ESCAMILLA ARROYO

ASESOR: DR. VIRGILIO LOPEZ MORALES[1]

CO-ASESOR: DR. LUIS ENRIQUE RAMOS VELASCO[1]

PACHUCA DE SOTO, HIDALGO. NOVIEMBRE DE 2005

[1] Investigador de la UAEH-ICBI-CITIS

A G R A D E C I M I E N T O S

A la Universidad Autónoma del Estado de Hidalgo por haberme dado la oportunidad de formarme profesionalmente, por la oportunidad de conocer a tantos profesores y amigos, personas que fueron sin duda muy importantes para cumplir con este logro.

A mis asesores de tesis, Dr. Virgilio López Morales y Dr. Luis Enrique Ramos Velasco. Por todo el apoyo, la tolerancia y su tiempo para poder realizar este trabajo, por las asesorías, las pláticas y la convivencia, son sin duda antes que catedráticos e investigadores de esta Universidad, dos excelentes personas, muchas gracias.

A mis Padres que tienen mucho que ver en toda mi formación, por estar ahí cuando más los necesite, por tener siempre una palabra de apoyo, por darme tantos momentos buenos y el apoyo para cursar una carrera, los amo. A mi novia Neftalí por estarme empujando siempre, por el apoyo que siempre fuiste en toda la carrera, por ser quien eres y lo que eres para mi gracias, te quiero.

A mis hermanos Victor y Vanessa por que ellos son una razón más para seguir adelante, por su comprensión y todo su apoyo gracias, los quiero mucho.

Índice general

Índice de figuras

Índice de Tablas

RESUMEN

En esta monografía se abordará los principios básicos sobre los que descansa el protocolo de comunicación serial distribuida CAN (Controller Area Network). Se empezará por su concepción, descrita en el Capítulo 1, el cual describe el camino de diseño que siguieron muchos investigadores para llegar a su elaboración. También se abordarán los diferentes problemas a resolver al diseñar e implementar una red como son: las nociones de acceso a las redes, incluyendo evidentemente los problemas de conflictos y de arbitraje, los tiempos de latencia, las nociones de elasticidad de una red, la seguridad de la información transportada; en otros términos, la estrategia de gestión de los errores (la detección, señalización y corrección de errores), las cuestiones de la topología de la red, longitudes, débitos y de soporte físico, así como las principales herramientas para resolver dichos problemas. También, se mencionará cual es y como trabaja su mecanismo de arbitraje, los tiempos de latencia, y la contención a nivel de bits y finalmente las consecuencias y capacidades que tiene una red con el uso del bus CAN, así como las nociones de elasticidad de un sistema, y las implicaciones que tiene la elasticidad de un sistema sobre la elección de un tipo de direccionamiento.

Temas muy importantes a tratar en una red con el bus CAN, el tratamiento, detección y gestión de los errores. De esta forma se llegará al punto de visualizar desde la concepción hasta su implementación, y el lugar que ocupa este bus especial en el mundo automotriz, aeronáutica, marítimo, etc.

En el Capítulo 2 se tratará principalmente los principios adoptados para la comunicación interna, describiendo de una forma más simple y comprensiva que en el documento original, que proviene de la sociedad Bosch, quien creó este potente protocolo. Se compone de 5 partes para las cuales se ha hecho un esfuerzo de presentación a fín de que la lectura y su asimilación sean menos áridas: presentación general, conformidad ISO/OSI, vocabulario, definiciones, circulación de las tramas sobre el bus (cuando todo esta bien), los casos de errores, sus detecciones, sus tratamientos, el resto del protocolo (sobre la calidad del bus, etc.), y para terminar, el CAN 2.0B.

En el tercer Capítulo se hablará ampliamente sobre la capa física sobre la cual serán transmitidas las tramas CAN. Se abordarán, los temas siguientes: la capa física descrita en el documento de referencia y el de ISO, las propiedades del bit, los tipos y estructuras de redes, los problemas de propagación de señal, de distancia y de caudal de información.

Finalmente en el Capítulo 4, se ilustra por medio de un ejemplo y circuitos integrados asociados, una posible aplicación de este protocolo, utilizando dispositivos disponibles comercialmente. En esta monografía se hizo el esfuerzo de compilar varios documentos y presentarlos de una forma clara, conservando las partes principales de las referencias integradas en esta monografía.

Contexto de la Monografía

El propósito principal de la presente monografía es la de entender los principios básicos de este protocolo de comunicación serial distribuida CAN. Se analizan en detalle el protocolo y sus diferentes capas así como los diversos problemas que se tienen que resolver al diseñar e implementar una red, la cual tiene cualquier medio físico de transmisión. Aunque gran parte de los conceptos que se manejaran en esta monografía son ya actualmente tomados en cuenta por los diferentes dispositivos e integrados que manejan este protocolo, es necesario conocer las características principales para la resolución en los problemas de conflictos y de arbitraje, así como de los tiempos de latencia y de gestión de errores.

Con el fin de hacer autocontenida esta monografía, se incluye un ejemplo de aplicación de los controladores y manejadores de este bus CAN interconectados con un microcontrolador y un sensor de presión.

OBJETIVOS.

A través de la presentación, análisis e ilustración por medio de un ejemplo de aplicación con el bus CAN, los objetivos de la presente monografía son los siguientes:

1) Analizar las nociones de acceso a las redes, con los problemas de conflicto y de arbitraje, los tiempos de latencia y seguridad.

2) Analizar el problema del tratamiento y gestión de errores.

3) Abordar algunas cuestiones principales sobre la topología de la red, longitudes y flujos de información transportada.

4) Presentar de forma detallada la capa física sobre la cual serán transmitidas las tramas CAN. E ilustrar con breve ejemplo el tipo de conexión asociada.

JUSTIFICACIÓN

La creciente necesidad de intercomunicar diferentes módulos de control, administración y seguridad, ha hecho posible el desarrollo de diferentes protocolos de comunicación. También debido a las grandes distancias que separan a estos diferentes módulos o unidades, hace que se prefiera una comunicación serial. Es así como el estudio de un protocolo de comunicación serial distribuida, como es el bus CAN, no es solamente interesante sino además nos permite conocer las bases de diseño e implementación de uno de los protocolos más utilizados actualmente en la industria.

Capítulo 1

Introducción al CAN

Introducción

La tecnología presente en nuestro mundo coexiste con la humanidad gracias al control que la humanidad tiene sobre la tecnología. Por lo tanto, este control determina la utilidad propia de la tecnología, para que en caso de que una tecnología fuera no contratable, se haría a un lado dado que no tendría ninguna utilidad. Desde el fin de la guerra mundial se ha invertido mucho tiempo y dinero en desarrollar elementos destinados al control, sin embargo al correr del tiempo, la tecnología se vuelve más compleja y por tal motivo las unidades de control se dedican solo a partes específicas del sistema en su totalidad.

Esta segmentación acarrea otra necesidad, la comunicación entre las unidades de control. Para esta tarea, se han diseñado varios protocolos de comunicación como: BATIBUS, BITBUS, EIB, FIP bus, J1850, LON, PROFIBUS, VAN, CAN. Sin embargo, en la actualidad un protocolo interesante es el Bus-CAN, principalmente porque este protocolo permite el intercambio de información entre varios participantes en tiempo real y de manera inteligente.

Esto lo realiza de manera inteligente gracias a su forma de arbitrar, es decir, todos los nodos de la red son considerados como maestros y cada uno puede enviar mensajes en cualquier momento. Cada mensaje tiene una sección de identificador y este determina la prioridad del mensaje. Los identificadores con valor más pequeño tienen la más alta prioridad. Aunque varios mensajes sean enviados al mismo tiempo, este sistema de prioridad provocará que un mensaje permanezca en el bus y no que sean destruidos.

En esta monografía se tratará el protocolo y el bus CAN por sus siglas en inglés de Controller Area Network (Red de controlador de área). Actualmente no es fácil ponerse de acuerdo al intentar definir los buses, sobre todo cuando cada uno tiene sus deseos, sus exigencias técnicas particulares y además de que existen mercados suficientemente

importantes para justificar y optimizar cada concepto, es decir reducir el costo. Tal como lo describe en las normas ISO, el bus CAN es un protocolo de comunicación serie que soporta de forma eficiente la distribución de comandos en tiempo real con un alto nivel de seguridad.

Su dominio de predilección cubre generalmente las aplicaciones de la red a una alta tasa de transmisión, alta fiabilidad de transmisión y con un concepto de cableado multiplexado de bajo costo.

1.1. Concepto de CAN.

El bus CAN fue concebido por la sociedad R. Bosch Gmbh y de estos documentos de diseño se consagrará un Capítulo de introducción para ver las grandes líneas de desarrollo de este protocolo. En este Capítulo se intentará describir el camino de diseño que siguieron muchos investigadores para llegar a la elaboración final de este proyecto.

Este Capítulo, nos permitirá comprender mejor los logros y desafíos que permitieron elaborar este protocolo tan particular y que tuviera tanto éxito, tanto en el dominio automotriz, la industria y profesional. Cada vez que se aborda el diseño de un bus los problemas a resolver son numerosos y las diferentes características de los campos de aplicación tomados como objetivos modifican el orden de los parámetros a ser tomados en cuenta e implican el desarrollo de nuevas ideas y conceptos para resolver dichos problemas.

A continuación se enumerará rápidamente los problemas inherentes a las aplicaciones de los buses de comunicación y las redes:

a) Las nociones de acceso a las redes, incluyendo evidentemente los problemas de conflictos y de arbitraje, los tiempos de latencia.,

b) Las nociones de elasticidad de una red,

c) La seguridad de la información transportadas o en otros términos, la estrategia de gestión de los errores (la detección, señalización y corrección de errores),

d) Las cuestiones que tienen que ver con las topologías, longitudes y débitos,

e) Las cuestiones de soporte físico (Cf. [Decotignie and Pleinevaux, 1993]).

1.1.1. Nociones de acceso al bus y de arbitraje.

Actualmente, los sistemas de control en tiempo real *distribuidos* basados sobre un sistema operativo situado en el interior de un solo procesador e interconectados por una red de comunicación con procesadores distribuidos, tienen una extensión importante en los sistemas paralelos. Más allá de asegurar un simple intercambio de datos, los tratamientos deben ser sincronizados, es decir que sus ejecuciones deben seguir ciertas secuencias lógicas interrelacionadas. En estos sistemas, los mensajes que tienen que

ver con la sincronización son generalmente cortos. Pueden ser creados sin importar cual sea el tratamiento que tuvieron en el sistema y deben ser simultáneamente recibidos de manera que puedan conservar la consistencia de un tratamiento paralelo (Cf. [Decotignie and Pleinevaux, 1993]).

Todas las estaciones generan independientemente mensajes en instantes aleatorios que tienen que ver con sus tareas respectivas. Las solicitudes de transmisión entran en competencia por ganar el acceso al bus, lo que nos lleva a tener tiempos de latencia variables más que constantes. (Cf. [Decotignie and Pleinevaux, 1993])

1.2. Diferentes principios de arbitraje.

(CSMA/CD,CSMA/CA).
CSMA/CD.
Por razones históricas, el proceso de arbitraje del tipo Carrier Sensor Multiple Access / Collision Detection (CSMA/CD) fue en principio planeado para resolver tales conflictos de acceso al medio, es decir al bus (Cf. [Decotignie and Pleinevaux, 1993]).

Con este sistema CSMA/CD, un mensaje de contención es detectado cuando muchas estaciones intentan acceder simultáneamente al bus cuando éste está en reposo. La transmisión es entonces detenida y todas las estaciones se retiran de la red. Después de un cierto lapso de tiempo, diferente para cada estación, cada una intenta de nuevo acceder a la red (Cf. [Decotignie and Pleinevaux, 1993]).

Es bien conocido que estas anulaciones de transferencia de datos durante la contención decrecen por el principio mismo en función de la capacidad de transporte de la red. Puede suceder igualmente que en los puntos pico del tráfico, la red este totalmente bloqueada; y esto no puede ser aceptable más que cuando se prevé utilizar la red para satisfacer las aplicaciones que se dicen son de tiempo real.
(Cf. [Decotignie and Pleinevaux, 1993]).

CSMA/CA.
En vista de los problemas indicados anteriormente, un principio entre otros fue examinado con atención. Se trata del Carrier Sensor Multiple Access / Collision Avoidance (CSMA/CA).

Este dispositivo CSMA/CA ya no funciona más con la ayuda de una contención realizada al nivel de intentar, si no más bien al nivel del bit mismo (bitwise contention -gestión del conflicto- mientras esté presente la duración del bit) [1] .

[1] Es este principio CSMA/CA que fue integrado al protocolo CAN y que se describirá su funcionamiento en detalle a lo largo de esta monografía

Para mayor detalle consulte [Decotignie and Pleinevaux, 1993].

En este caso, es posible evitar los conflictos de acceso al bus asignando un nivel de prioridad a cada uno de los mensajes transportados.
Para mayor detalle consulte [Decotignie and Pleinevaux, 1993].

En caso de contención, el mensaje de prioridad más elevado ganará siempre el bus. Naturalmente que el tiempo de latencia dependerá entonces fuertemente de los niveles de prioridad elegidos y atribuidos para cada uno de los mensajes

El problema de los tiempos de latencia.

Cuando uno hace el estudio completo de un sistema de comunicación, a fin de tomar en cuenta todos los parámetros, el tiempo de latencia se define generalmente como la duración que existe entre el instante que indica el principio de la solicitud de una transmisión y el inicio real de la acción generada por ésta (Cf. [Dominique, 1996]).

Por el momento, en una forma más simple, definiremos el tiempo de latencia (t_lat) de un mensaje como la duración que existe entre el instante que indica el inicio de la solicitud de una transmisión y el inicio real de ésta (Cf. [Dominique, 1996]).

Esta noción es difusa y muy cercana a las estadísticas, principalmente en los sistemas de tiempo real. La razón de esto es simple: solamente algunos mensajes específicos tienen realmente el tiempo de latencia garantizado y esto solamente en los picos de tráfico. Es por tanto necesario considerar dos tipos de mensajes:

- R, el número de mensajes de los cuales la latencia debe ser garantizada,
- S, los otros,

Y claro M = R + S, la totalidad de entre ellos (Cf. [Dominique, 1996]).

La curva de la Figura 1.1 da la distribución de probabilidad del tiempo de latencia en función de éste, en el caso en donde la demanda de transmisión no es solicitada más que una sola vez.

El valor particular t_cycle es el tiempo que representa un ciclo medio de actividad de la red constituido de M mensajes de largo temporal t y que contiene N bits. Las curvas dependen de la prioridad de los mensajes. La distribución de la probabilidad de prioridad 1 llega a ser 0 tan pronto como la transferencia del mensaje más largo ha sido llevado a cabo (Cf. [Dominique, 1996]).

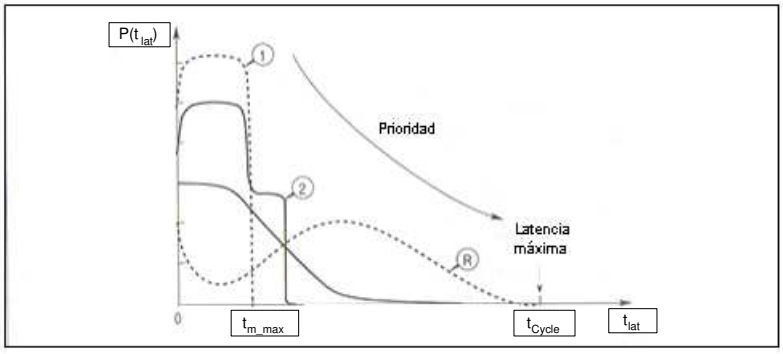

Figura 1.1: Curva de distribución

La contención a nivel de bits.

En el principio del CSMA/CA, utilizado en el bus CAN (derechos reservados desde 1975 y del mismo tipo que del bus I2C), introduce algunas restricciones en la representación de la señal al nivel de la capa física y en la geometría máxima de una red funcionando de esta manera (Cf. [Etschberger, 2000]).

Así, durante la fase de arbitraje, la señal sobre el bus debe ser (de forma en que se tenga sobre la red los bits de más alta prioridad dominando sobre aquellos de más baja prioridad) ya sea:

- Dominante: Por ejemplo, presencia de energía, presencia de corriente, luz, variación electromagnética, etc.
- Recesivo: Por ejemplo, ausencia de energía.

Por definición, cuando dos bits dominantes, y recesivos, están siendo simultáneamente transmitidos sobre el bus, el estado resultante sobre el bus debe ser el estado dominante (Cf. [Etschberger, 2000]).

Primeras consecuencias sobre la capacidad y longitud de una red.

Sabiendo que la velocidad de propagación de las ondas electromagnéticas v_prop es del orden de 200000 Km. /s en las líneas eléctricas y las fibras ópticas (ó bien que las ondas hacen alrededor de 5 ns para recorrer 1 m ó bien recorren 200 m / μs) (Cf. [Etschberger, 2000]), se mencionará rápidamente algunas consecuencias de lo que se acaba de presentar en el párrafo precedente y a los cuales se consagrará a lo largo de este Capítulo.

En principio, en un sistema funcionando por contención de bits, un bit puede viajar de un extremo de la red al otro antes de ser detectado a su llegada. Ahora bien, puede suceder que algunos micros instantes antes de su llegada la otra estación, no habiéndolo todavía detectado, es decir no habiéndolo *visto* llegar a sus terminales, decide iniciar una emisión en la otra dirección. Esto resultaría en un choque frontal (Cf. [Etschberger, 2000]).

Si designamos a t_bus como el tiempo que toma la señal en recorrer la distancia máxima de la red, la suma global de los tiempos de ida y regreso debidos a la propagación de la señal sobre el bus es de:

$$2t\_bus = (2l)(v\_prop) \tag{1.1}$$

Ejemplo: Con l = 40 m, se obtiene t_bus = 200 ns

Donde: l= longitud del bus, v_prop= Velocidad de propagación.

Así que, para que la estación que ha emitido el bit inicial esté apta a administrar los conflictos, los tiempos que debe durar el bit t_bit (bit time) deben ser más grande que el t_bus. Además, para que esté completo es necesario, incluso obligatorio, tener en cuenta los tiempos necesarios, el periodo de muestreo y tratamiento del bit en la estación a la que llega (Cf. [Etschberger, 2000]).

Para evaluar el bit de tiempo mínimo: (t_bit_min) de la red prevista, es necesario tener en cuenta (ver la Figura 1.2):

- Los retardos de salida de propagación: t_out,
- Los retardos de la entrada de la propagación: t_in,
- Los retardos debido a la sincronización: t_sync,
- La desviación de las fases debido a las tolerancias del reloj: t_clock.

Lo que da un total de un t_bit_min de:

$$t\_bit = 2t\_bus + 2t\_in + t\_sync + t\_clock \tag{1.2}$$

Ejemplo: Con una velocidad de 100 kb/ s, o sea un bit time de 10 s, la longitud de la red puede llegar a alcanzar una longitud de 900m.

Se detallará todas estas nociones en el Capítulo 2.

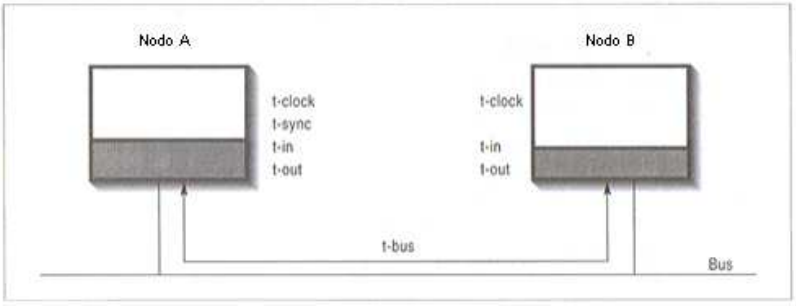

Figura 1.2: Obtención de t_bit_min

1.2.1. Nociones de elasticidad de un sistema.

La configuración de la arquitectura y la topología de un sistema distribuido es generalmente diferente de una aplicación a la otra. Además, para una misma aplicación, puede evolucionar o ser modificada en el tiempo en función de las necesidades que se deben satisfacer (Cf. [Kiencke, 1994]).

Para comprender mejor esto, se cita a título de ejemplo, el caso de las instalaciones de cadenas de producción industriales que, teniendo siempre el mismo aspecto global, requieren de ser configuradas de un tiempo al otro para adaptarse a los nuevos tipos de producción o de derivados (Cf. [Kiencke, 1994]).

Para los sistemas o las redes, se define generalmente por el término elasticidad a la aptitud de la red a tolerar un cambio de configuración con un mínimo de reprogramación posible para la transferencia de datos a ser efectuada (Cf. [Kiencke, 1994]).

Examinemos con un poco de mayor detalle los problemas ligados a la elasticidad de una red. La información que es recibida y tratada por uno o más usuarios en un sistema distribuido, debe ser creada y transmitida a una estación. No hay otra alternativa lógica fuera de esto (Cf. [Kiencke, 1994]).

Dos casos se pueden presentar:

1.- Se desea agregar nueva información. Toda nueva información requiere entonces una nueva transferencia de mensajes y, en consecuencia, una reprogramación de la comunicación. En este caso, la estación que transmitía con anterioridad esta información específica debería ser reprogramada en función de la nueva mientras que las otras estaciones deberían quedarse sin cambio alguno (Cf. [Kiencke, 1994]).

2.- Una situación diferente se produce cuando una nueva información específica que ya existía debe de ser únicamente transmitida a partir de otra estación, o bien recibida por las estaciones adicionales. En este caso la estación receptora adicional debería de ser reprogramada para recibir la información (Cf. [Kiencke, 1994]).

1.2.2. Implicación de la elasticidad de un sistema sobre la elección del principio de direccionamiento.

En el direccionamiento clásico, constituido generalmente por una dirección fuente y una dirección destino, no se ofrece una buena elasticidad estructural a un sistema. De hecho, no importa que mensaje deba ser redireccionado, éste necesita modificaciones, incluso si esto es lógicamente no obligatorio como lo hemos indicado anteriormente (Cf. [Kiencke, 1994]).

En lo que respecta al concepto CAN se decidió con el fin de asegurar una buena elasticidad del sistema, de utilizar otro principio de direccionamiento, ya no tanto basado sobre las direcciones fuente y destino, si no más bien basado sobre el contenido del mensaje, esto implica dos cosas:

1.- En principio, que un mensaje sea transmitido a todas las (otras) estaciones de la red. El término consagrado para tal principio es el de **difusión** (broadcast diffusion)

2.- Enseguida, que el tratamiento de selección del mensaje transmitido sea entonces efectuado por un filtrado que se llama de aceptación, abordo de cada estación.

Para poder realizar lo anterior, el mensaje es etiquetado (posee un **label**) por un identificador -ID (i)-, que será entonces comparado en la lista de mensajes recibidos (o que se desea recibir) en cada estación (Cf. [Kiencke, 1994]).

Esta lista contiene los punteros de las direcciones AP (i) hacia el sello de comunicación de tal forma que el contenido del mensaje pueda ser guardado (ver Figura 1.3).

De este hecho, todos los mensajes son recibidos sobre toda la extensión de la red y la consistencia de los datos es entonces garantizada en los sistemas de control distribuidos (Cf. [Kiencke, 1994]).

1.2.3. Tratamiento y gestión de errores.

Nociones positivas y negativas.

La elasticidad de los sistemas y la identificación basada sobre el contenido de los mensajes complican el tratamiento de errores cuando estos se producen.

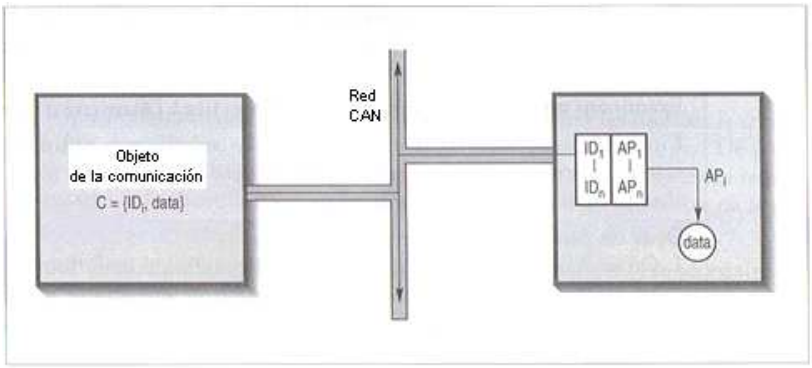

Figura 1.3: Lista de punteros

Un método clásico de no detección de errores reside, por ejemplo, en un reenvío llamado positivo de la estación receptora hacia la estación emisora cuando sucede la recepción correcta de un mensaje. La información clave contenida en el interior de tal positivo es generalmente la dirección de la estación receptora. Es el caso, por ejemplo, del bus I2C (Cf. [Kiencke et al., 1986]).

En el concepto CAN, esa noción de dirección local y el identificador desaparecieron completamente, etiquetando el mensaje que es transmitido y recibido por el conjunto de participantes de toda la red (Cf. [Kiencke et al., 1986]).

Esto impone la ejecución de una tarea local dedicada al tratamiento de los errores en cada una de las estaciones presentes sobre la red en ausencia de tales direcciones en los mensajes locales (Cf. [Kiencke et al., 1986]).

Para satisfacer a esto, el concepto de protocolo CAN utiliza la combinación de los tipos positivos y negativos. El tipo positivo ACK +, puede ser expresado bajo la forma siguiente: ACK + = ACK + (i), no importando que sea (i) (Cf. [Kiencke et al., 1986]).

Proviene de todas las estaciones (i) que hallan recibido correctamente el mensaje y expresen este reconocimiento positivo durante un mismo intervalo de tiempo bien definido (ACK, time slot).

El principio que se viene describiendo también puede expresarse de dos formas fundamentales:

- Desde un punto optimista del problema, según el cual podemos decir que el reconocimiento positivo indica que al menos una estación ha recibido correctamente el mensaje transmitido.

- Y desde un punto de vista menos optimista: el reconocimiento negativo de un mensaje debe tomar una forma tal que indicará que existe al menos un error en el sistema global (Cf. [Kiencke et al., 1986]).

En ese último caso, el mensaje que señala la presencia del error debe ser enviado sobre la red justo después de la detección de esta. Esta técnica permitirá enseguida garantizar una resincronización inmediata del sistema en el interior de una sola trama del mensaje. Este último punto es crucial para la seguridad de las aplicaciones consideradas (Cf. [Kiencke et al., 1986]).

Gestión de errores.

La combinación, la redundancia, el análisis de los reconocimientos positivos y negativos que provienen de los dispositivos dedicados al tratamiento de los errores de las diferentes estaciones son explotadas a fin de tener fuertes *presunciones* sobre el hecho de que un error proviene ya sea de la estación emisora o bien de alguna de las estaciones receptoras (Cf. [Kiencke et al., 1986]).

Por ejemplo:

1) la presencia de al menos un reconocimiento positivo que proviene de un receptor combinado a un mensaje de error significará que al menos el mensaje ha sido correctamente transmitido,

2) al contrario, la ausencia de un reconocimiento positivo conjuntamente a un mensaje de error indicará que todas las estaciones receptoras han detectado un error, y que hay una fuerte *presunción* de que el error esté localizado al nivel de la estación emisora (Cf. [Kiencke et al., 1986]).

Los mensajes de errores.

Los mensajes de errores definidos por el concepto CAN son de dos tipos: Primarios y secundarios (ver la Figura 1.4).

El balance de los errores primarios.

Cuando en una estación (o muchas simultáneamente) se detecta un error, esto llevará inmediatamente a transmitir un mensaje de error (Cf. [Kiencke et al., 1986]).

Es en ese momento, que el mensaje de error consiste en una secuencia de bits dominantes y el mensaje que está siendo en ese momento enviado es abortado.

El balance de los errores secundarios.

Una vez que se ha producido el aborto del mensaje, todas las otras estaciones que detectan una violación de las reglas del formato, transmiten un mensaje de error de su propia iniciativa, la cual se mezcla con el primero, lo que tiene por efecto de extender la duración (Cf. [Kiencke et al., 1986]).

Los balances de errores primarios son más probables para una estación donde el error se ha producido, comparativamente a un balance de error secundario que es generado en reacción a lo citado de forma precedente.

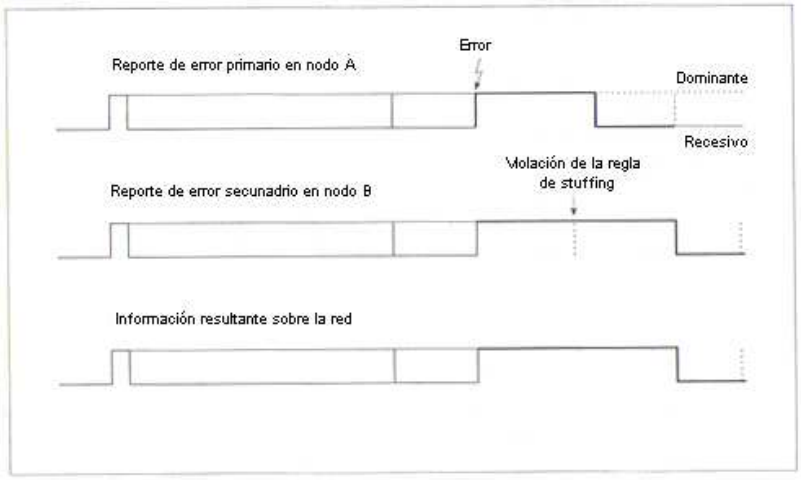

Figura 1.4: Tipos de errores

Noción de estrategia de gestión de los errores.

Aborda principalmente lo que es administrar correctamente los errores, y dentro de este concepto el fin es construir una estrategia que tenga que ver contra el tratamiento de los errores. La calidad del tratamiento dependerá fuertemente del aspecto voluntarista o no de la estrategia definida para un campo de aplicación de estados (Cf. [Kiencke et al., 1986]).

Para el protocolo CAN, fue definida (se revisará más a detalle en el Capítulo 2) una estrategia que se llama de errores (o falta) de confinamiento, en donde las grandes líneas de diseño son representadas bajo las formas gráficas representadas en las figuras 1.5 y 1.6.

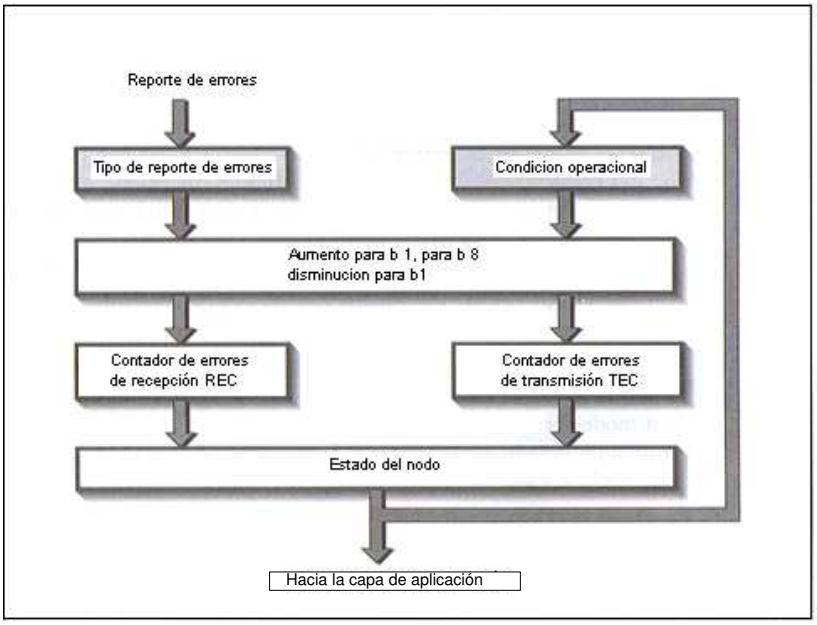

Figura 1.5: Estrategia de faltas de confinamiento

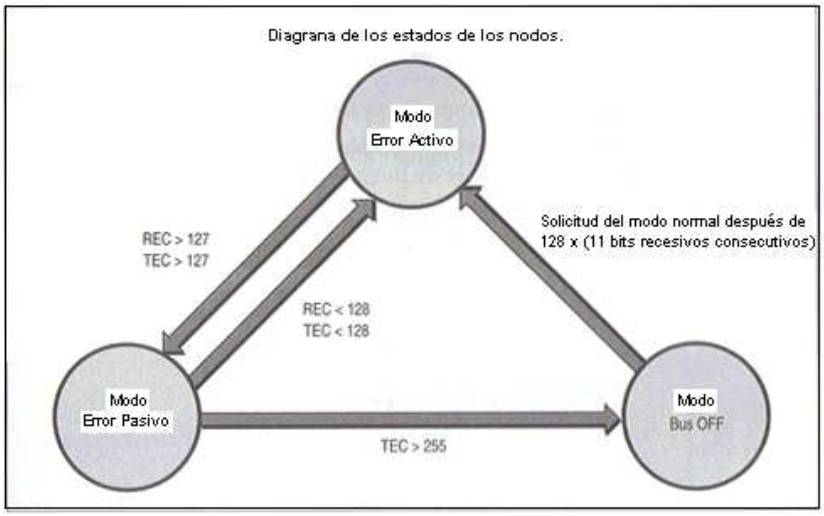

Figura 1.6: Diagrama de estados

A continuación se explicarán las grandes líneas de esta estrategia.

En principio cada estación debe poseer dos contadores de errores separados, el primero para saber que es lo que pasa durante la emisión de un mensaje, y el otro para efectuar una tarea similar cuando se produce la recepción (Cf. [Kiencke et al., 1986]).

Según el tipo de balance de error y las condiciones operacionales en las cuales se encuentra la estación en cada instante considerado, los contadores aumentan con las ponderaciones diferentes según ciertas condiciones, ó bien decrecen. Estos contadores tienen por misión el de tener en cuenta las informaciones que provienen directa o indirectamente de todas las otras estaciones (Cf. [Kiencke et al., 1986]).

Con la ayuda de esta estrategia los contadores efectúan una operación de media que da una aproximación de los valores estadísticos de la calidad de la red en un instante dado (Cf. [Kiencke et al., 1986]).

A fin de delimitar esta estrategia, también se definió que si demasiados errores eran atribuidos por estos medios estadísticos a una estación dada, su estado pasaría de un modo que se llama de error activo a un estado que se llama de error pasivo, que le impediría comunicar pero que sin embargo le permitiría todavía gestionar los errores que pudieran producirse sobre la red (Cf. [Kiencke et al., 1986]).

En presencia de errores de transmisión demasiado numerosas, la red podría bloquearse, dejando entonces todo el trafico imposible!.

A fin de evitar esto, es necesario definir que más allá de un cierto umbral (fijado a 255 para el CAN), dicha estación pasa a un nuevo estado que se dice Bus off durante la cual esta estación parece estar desconectada físicamente del bus a fin de no bloquearlo. En este caso, su puerto de entrada se queda en alta impedancia y permite así observar (función de monitor) la señal que circula en la red (Cf. [Kiencke et al., 1986]).

Aquí están entonces los principales puntos de estudio de este nuevo concepto de protocolo así como las ideas fuertes y principios puestos en marcha para resolver los problemas que generalmente se presentan en las aplicaciones.

Después de esta presentación del conjunto de este protocolo, estamos ahora listos para estudiar su funcionamiento concreto y su bus asociado.

1.3. Del concepto a la realidad.

Los grandes ejes de desarrollo del concepto CAN, han sido puestos basándose sobre el estado que guardaba en ese momento el mercado y la técnica, pero también tratando de extrapolar las tendencias del futuro, tanto a nivel de los sistemas así como al nivel de las técnicas y tecnologías que deben ser puestas en marcha para llegar a ser una realidad (Cf. [Reuss, 1988]).

Debido a la naturaleza de estos ejes, la pregunta que uno se debería de formular sería la siguiente: ¿Por qué el CAN y por que no otro protocolo?

Para esto, se darán algunas explicaciones complementarias que ayudarán a comprender y se empezará por abordar rápidamente lo que es el terreno del bus.

1.3.1. El mercado del bus.

Desde hace algunos años, muchas sociedades desarrollaron sus propias soluciones para resolver los problemas sensiblemente similares, ligados a la interconexión y comunicación entre sistemas. La casi totalidad de estas soluciones, que se llaman propietarias del hecho de diferentes intereses divergentes, condujo a que en el mercado se hiciera un mosaico de diferentes soluciones (Cf. [Reuss, 1988]).

Entre ellas, muchos buses del tipo serie existen desde hace largo tiempo. Independientemente de los mercados sobrentendidos (industrial y automotriz), los nombres más conocidos por orden alfabético son: BATIBUS, BITBUS, EIB, FIP bus, J1850, LON, PROFIBUS, VAN, etc. (Cf. [Reuss, 1988])

1.3.2. Introducción al CAN.

La amplitud del mercado automotriz (millones de piezas por año) inclinó a los fabricantes de componentes a concebir circuitos integrados que administráran el protocolo CAN. Los costos de esta forma bajaron de una forma importante, lo que no es necesariamente el caso para los otro buses, ya que ellos frecuentemente se tienen que inclinar a aplicaciones en cantidades menores sin beneficiarse de los componentes dedicados ni de precios más competitivos (Cf. [Reuss, 1988]).

De este hecho, los industriales consideraron la llegada del bus CAN con otro punto de vista, (el punto de vista muy particular de la relación desempeño y costo) y concluyeron repentinamente que este tipo de bus satisfacía plenamente sus necesidades.

1.3.3. La proposición CAN: una solución completa.

La fuerza del concepto CAN, de su promoción y su éxito viene de la voluntad de las personas implicadas en el proyecto de poner a disposición de los usuarios, en tiempo y forma, todo de lo que ellos tienen necesidad y que tienen generalmente problemas para encontrar sobre el mismo tema (Cf. [Reuss, 1988]).

De hecho, en el espacio de 2 a 3 años (periodo de tiempo bien adaptado al mercado) todos los ingredientes que permitían desarrollar inteligentemente productos CAN aparecieron y estuvieron disponibles; es decir:

- El protocolo preciso, completo escrito de forma clara,
- las normas ISO para las aplicaciones automotrices,
- las familias concurrenciales de los componentes electrónicos,
- la toma de conciencia del mercado industrial,
- la literatura técnica (artículos, libros, etc.),
- las conferencias y congresos de sensibilización, de formación,
- los grupos industriales (CiA),
- las recomendaciones complementarias para la industria que tenia que ver, por ejemplo, las tomas (CiA),
- la capa aplicativa (CAN in Automation, Honeywell, Allen Bradley.),
- las herramientas de demostración, evaluación, de desarrollo de los componentes y de la red (Cf. [Reuss, 1988]).

1.3.4. Bus CAN - bus I2C, un buen complemento.

Se acaba de evocar el bus CAN y su protocolo y puede ser cierto que entre nosotros hallamos creído que este bus iba, podía, o debía reemplazar en un cierto tiempo al bus I2C tan apreciado actualmente por muchos. La respuesta es clara: no! y este punto requiere de algunas explicaciones.

Es muy cierto, que las estructuras al nivel de los protocolos respectivos, son totalmente diferentes pero estos dos buses no compiten entre ellos. Son totalmente complementarios.

El bus I2C (Inter Integrated Circuits concebido por Philips) es un bus asimétrico con respecto a la masa y funcionalidad de forma síncrona (como un reloj SCL) sobre una distancia en principio limitada a unos metros, incluso puede llegar a una decena de metros, con una velocidad de 100 kb/s (400 kb/s en modo rápido) (Cf. [Reuss, 1988]).

El bus CAN es un bus asíncrono, con un reloj/datos multiplexados temporalmente, de una velocidad que llega a ser de hasta 1 Mb/s con una capa física flexible debido a que no esta definido por el protocolo. Se puede entonces en principio, emplearlo fácilmente con parejas diferenciales, en fibras ópticas, lo que le confiere una nueva dimensión en metros y en estructura de la red (Cf. [Reuss, 1988]).

En otros términos, el bus I2C esta concebido para quedarse en casa, sobre la tarjeta principal de su estación, mientras que el bus CAN puede llegar a darse una vuelta fuera (Controller Area Network) y permitir a su estación local comunicarse con sus otros compañeros (Cf. [Reuss, 1988]).

Así también el protocolo I2C no está protegido (estructuralmente) para nada en su calidad de datos transportados y que al contrario el bus CAN posee estructuralmente dispositivos complementarios de detección bien elaborados, señalización, gestión de errores, etc., directamente implementados sobre una capa física de silicio (Cf. [Reuss, 1988]).

Esto significa que esta realmente equipado contra las agresiones de todo tipo de parásitos eléctricos, estáticos, etc., que generalmente uno encuentra fuera de un ambiente conocido.

Lo anterior es para explicar de manera muy simple las diferencias estructurales que distinguen y complementan a estos 2 protocolos (ver la Figura 1.7) (Cf. [Reuss, 1988]).

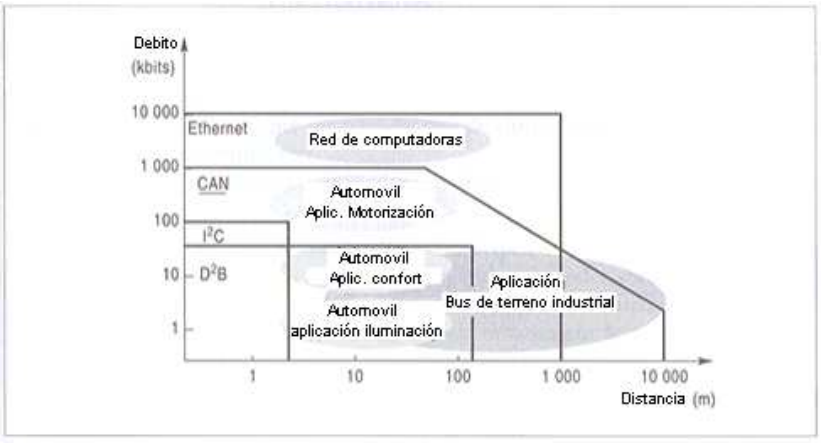

Figura 1.7: Diferencias entre CAN e I2C

Todos aquellos que han practicado con el protocolo ISO dirán que para llegar a realizar la normalización de un protocolo hacen falta años y también un gran número acelerado de procesos.

En fin, a la mitad del año 1987, la realidad tomó la forma de los primeros chips de silicio funcionales, después, en 1991, el primer coche (alemán), de alta gama, equipado de 5 unidades centrales electrónicas (ECU por sus siglas en inglés) y de un bus CAN que funcionaba a 500 Kb/s salió de las cadenas de producción (Cf. [Reuss, 1988]).

Durante el mismo periodo (1987-1991), las promociones urbanas (para las aplicaciones automotrices) y orbi (para las aplicaciones industriales) fueron creadas y aseguradas activamente por el SAE y el OSEK para la automotriz y por el CAN en Automation (CiA) reagrupamiento de industriales para las aplicaciones en todos los demás dominios (Cf. [Reuss, 1988]).

Es entonces debido a la locomotora que el mundo automotriz conoció este concepto, sin embargo su vocación típicamente industrial les ha limitado a este concepto. También, sin ignorar sus primeras aplicaciones, vamos a esforzarnos por retirar esa idea inicialmente recibida de que el bus CAN es un bus únicamente concebido para el área automotriz, para por el contrario describirlo como un bus de un buen desempeño para las redes locales industriales rápidas (Cf. [Reuss, 1988]).

Como último punto, señalaremos que cuando un sistema es concebido e implementado, es normal que un sistema similar o que la competencia salga también a la luz del día. Esto claramente se produce con la llegada de numerosos buses del mismo tipo sea en EU, Japón, Francia (bus VAN soportada para un GIE compuesto principalmente de PSA Peugeot Citroën y Renault). Faltarán otros autores que le hablarán cuando los componentes específicos estarán disponibles para el público (Cf. [Reuss, 1988]).

Esperamos que se haya sido lo suficientemente claro hasta este momento en algunos enfoques conjugados de técnicas y de marketing que permitieron hacer de este concepto una exitosa historia industrial. Veamos ahora con las consideraciones de orden técnico y con explicaciones del funcionamiento del protocolo del bus CAN.

Capítulo 2

EL CAN, SU PROTOCOLO Y SUS CARACTERÍSTICAS.

Introducción.

Dada la potencia intrínseca del protocolo CAN, se ha voluntariamente particionado, organizado y subdividido su presentación en varios capítulos.

El presente Capítulo, trata principalmente de los principios adoptados para la comunicación sobre el bus. Se compone de 5 partes para las cuales se ha hecho un esfuerzo de presentación a fin de que la lectura y su asimilación sean menos áridas que en el documento oficial y que muchas de las veces parece ser un rompecabezas: presentación general, conformidad ISO/OSI, vocabulario, definiciones, circulación de las tramas sobre el bus cuando todo esta bien, los casos de errores, sus detecciones, sus tratamientos, el resto del protocolo (sobre la calidad del bus, etc.), y para terminar, el CAN 2.0B.

Todas estas partes tienen origen en el documento de referencia de la sociedad Bosch.

Descubriremos entonces todo sobre: la estructura del bit, la sincronización, el medio y la conexión al medio y los problemas de propagación.

2.1. Las grandes definiciones.

2.1.1. Protocolo del CAN.

En esta subsección se describirá de forma simple y comprensiva el documento original que proviene de la sociedad Bosch.

En este documento, a fin de asegurar la transparencia de una realización y de una más grande flexibilidad cuando se lleve a cabo la implementación, el protocolo CAN

asegura una interconexión serie, asíncrona, multiplexada, reloj/datos temporalmente. Se presenta y se subdivide profesionalmente siguiendo sensiblemente las sesiones normalizadas de las diferentes capas del ISO/OSI (Cf. [Robert BOSCH, 1991]).

La especificación versión 2.0 comprende hasta nuestros días 2 partes que se hacen llamar A y B.

La parte A describe la trama CAN estándar la más común. Esta versión totalmente idéntica a la versión precedente que se hizo llamar 1.2. Esta trama soporta únicamente 11 bits de identificadores. Se va a lo largo de todo este Capítulo a presentar esta trama de forma detallada (Cf. [Robert BOSCH, 1991]).

La parte B se apega a describir la trama CAN según su formato extendido. Estimando que esto no era suficiente para unas aplicaciones, el valor del identificador se elevó de 11 a 29 bits (Cf. [Robert BOSCH, 1991]).

Al final de este Capítulo, se indicarán las principales diferencias entre las tramas del tipo 2.0A y 2.0B.

Observe que las versiones CAN 2.0A y 2.0B han sido concebidas de forma que se puedan asegurar una compatibilidad ascendente sin importar cuales sean las versiones anteriores del protocolo (Cf. [Robert BOSCH, 1991]).

Como se verá a continuación la estructura del protocolo del bus CAN posee implícitamente las propiedades siguientes:

- jerarquización de los mensajes,
- garantía de los tiempos de latencia,
- flexibilidad de configuración,
- recepción de múltiples fuentes con sincronización temporal,
- funcionamiento multimaestro,
- detecciones y señalización de errores,
- retransmisión automática de los mensajes alterados desde que el bus esta nuevamente en reposo,
- distinción de errores: de orden temporal o de no funcionalidad permanente en el nivel de un nodo,
- desconexión automática de los nodos defectuosos (Cf. [Robert BOSCH, 1991]).

2.2. Protocolo CAN y capas ISO.

No. de Capa	Modelo ISO/OSI	Protocolo CAN
7	Aplicación	Especificación para el usuario
6	Presentación	Vacio
5	Sesión	Vacio
4	Transporte	Vacio
3	Red	Vacio
2	Comunicación de datos	Prot. CAN (Permite acceso al medio)
1	Física	Prot. CAN (Permite acceso al medio)

Tabla 2.1: CAN vs. ISO

La Tabla 2.1 nos recuerda brevemente el desglose de la norma ISO/OSI (ISO: International Standards Organization / OSI: Open Systems Interconnection) e indica claramente las diferencias de cobertura que existe entre el contenido real requerido de la norma ISO/OSI y el contenido del documento de referencia del protocolo CAN. Como se puede marcar, en lugar de reemplazar la integridad de las 7 capas, este último no cubre la totalidad de la capa 2 Data Link Layer (capa de comunicación de datos) y una parte importante de la capa 1 Physical Layer (Capa Física) (Cf. [Organization, 1984]).

Esta Tabla requiere cuando menos alguna información complementaria que se dará a continuación.

2.2.1. Contenido de las diferentes capas ISO/OSI del CAN.

Se examinará rápidamente el contenido de las capas ISO/OSI del CAN, de forma seguida, se explicará el funcionamiento de tal sistema. Se utilizan las capas que se llaman objeto y transferencia **object, transfer** (no explícitamente indicadas en la Tabla) que tienen por misión tratar todos los servicios y funciones de la capa de interconexión de datos (Data Link Layer capa 2), definidos por el modelo ISO/OSI (Cf. [Organization, 1984]).

Capa objeto.

La capa objeto tiene por misión principal la de asegurar el filtrado de los mensajes y asegurar el tratamiento de los mensajes y de los estados.

Para esto, la capa debe:

- encontrar que mensaje debe de ser transmitido,

- decidir cuales de los mensajes recibidos vía la capa de transferencia están siendo utilizados,
- producir una interfaz en la capa aplicativa en relación con el hardware relativo al sistema (Cf. [Organization, 1984]).

Capa transfer.

La misión principal de la capa transfer es de ocuparse de la transferencia del protocolo, es decir:

- administrar y controlar la puesta en forma de la trama,
- la velocidad de transmisión y la conformidad temporal de la transferencia,
- realizar el arbitraje de los conflictos del bus,
- verificar la ausencia o la presencia de errores,
- señalar los diferentes tipos de errores (si hubiera) así como las faltas de confinamiento,
- validar los mensajes,
- proceder a despachar los mensajes (Cf. [Organization, 1984]).

Entre las numerosas tareas que tiene por misión efectuar esta capa, también esta la de decidir si el bus está libre a fin de iniciar una nueva transmisión o bien de decidir si la transmisión de un mensaje incidente esta produciéndose en ese momento (Cf. [Organization, 1984]).

Para comprender mejor esto, es necesario considerar la descomposición de las capas en subcapas tal y como se presenta en la Figura 2.1.

En términos oficiales, la capa de enlazado de datos, capa 2 (Data Link Layer) esta subdividida en dos subcapas, (Logical Link Control) y MAC (Médium Access Control). La capa física 1 (Physical Layer) comprende tres subcapas, PLS (Physical Signalling), PMA (Physical Médium Access) y MDI (Médium Depend Interface).

La subcapa MAC (Médium Access Control) representa el corazón del protocolo CAN. Ella tiene por función presentar los mensajes recibidos que provienen de la subcapa LLC (Logical Link Control) y aceptar los mensajes que deben ser transmitidos hacia la capa LLC (Cf. [Organization, 1984]).

La subcapa MAC es responsable de:

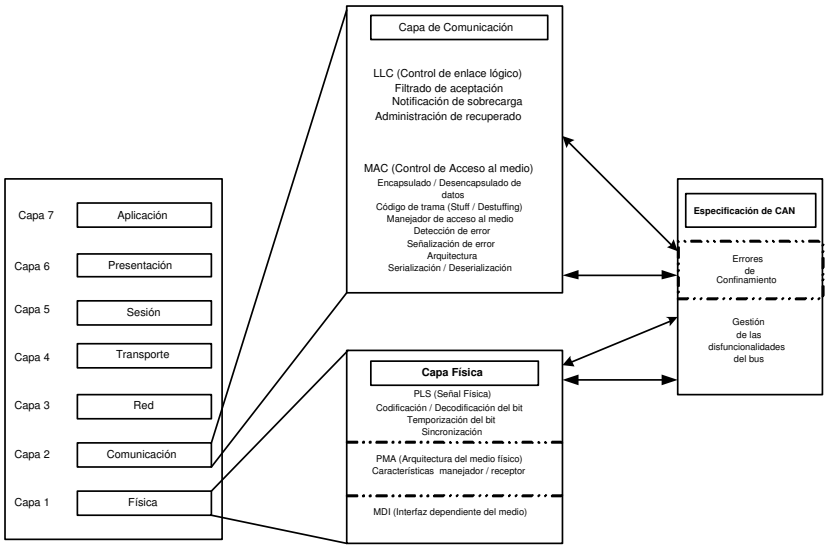

Figura 2.1: Descomposición de capas y subcapas

- la puesta en trama del mensaje,
- el arbitraje,
- la razón de satisfacción,
- la detección de los errores,
- y la señalización de los errores (Cf. [Organization, 1984]).

Nota importante.

La subcapa MAC esta supervisada por una entidad de administración (un administrador) bautizado como error de confinamiento que puede ser descrito como un mecanismo de autoverificación apto para ser la distinción entre los problemas de duración corta y las fallas permanentes.

La subcapa LLC (Logical Link Control), se encarga de:

- el filtrado de los mensajes,
- la notificación de las sobrecargas (overload),
- el procedimiento del recubrimiento de los errores.

De forma general, una vez definida, esta capa de transferencia no debe de ser modificada por un protocolo determinado.

Capa física.

La capa física define como es transmitida la señal y tiene en consecuencia como papel principal el de asegurar la transferencia física de los bits entre los diferentes nodos de acuerdo con todas las propiedades (eléctricas, electrónicas, ópticas, etc.) del sistema. Es evidente que en el interior de una misma y única red, la capa física debe de ser la misma para cada nodo[1] .

En principio esta capa:

- administra la representación del bit, (codificación, temporización, etc.),
- administra la sincronización del bit,
- define los niveles eléctricos, ópticos, de las señales,
- define el soporte de transmisión (Cf. [Organization, 1984]).

La capa de aplicación.

Un último punto también importante (que tiene que ver con el documento de referencia del protocolo CAN) es la capa de aplicación, es decir la capa 7 que esta totalmente vacía (Cf. [Organization, 1984]). La tabla de la Figura 2.2 resume claramente las funcionalidades de cada una de las capas que se acaban de analizar.

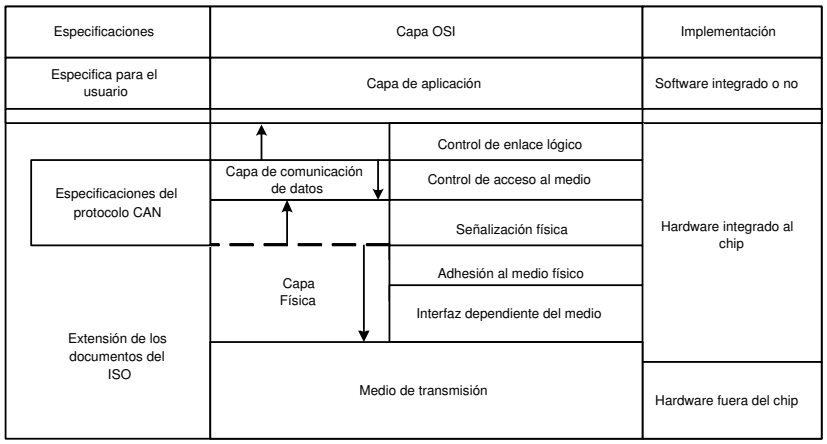

Figura 2.2: Funciones de las capas

[1] El documento de referencia de Bosch sobre el protocolo CAN no describe más que la representación detallada del bit (parte PLS). En efecto, en el interior de las especificaciones CAN, las características del driver/receiver (controlador/receptor) de la capa física no están definidas de tal suerte que el medio de transporte así como los niveles de las señales pueden ser optimizadas para una aplicación considerada.

2.3. Propiedades del CAN, vocabulario específico resumido.

En la subsección siguiente se ha conservado las siglas en inglés de ciertos términos (indicando la traducción en español) debido a que en el lenguaje técnico cotidiano de esta profesión se hace muy seguido referencia a estos vocablos.

Como en la mayor parte de los protocolos, es necesario utilizar un vocabulario de acuerdo a la situación, se presenta entonces los términos *ad hoc* y sus definiciones que tienen su origen (y adaptados a la comprensión y publicación de esta monografía) de las normas ISO que tienen que ver con el protocolo CAN (ISO 11519 y 11898) (Cf. [Peterson and Davie, 1996]).

2.3.1. Nodo (estación emisor).

ISO: subconjunto enlazado a una red de comunicación y capaz de comunicar sobre la red según un protocolo de comunicación.En el caso del protocolo CAN un nodo (estación) que genera un mensaje es llamado emisor de ese mensaje. Una estación se dice emisora hasta que el bus este en reposo (*idle*) o que la unidad halla perdido el arbitraje (Cf. [Peterson and Davie, 1996]).

2.3.2. Nodo (estación receptor).

Contrariamente al caso precedente un nodo (estación) es llamado receptor de mensaje si no es emisor de un mensaje y el bus no está libre (Cf. [Peterson and Davie, 1996]).

2.3.3. Valores del bus.

El bus puede tener uno de los valores lógicos complementarios definidos, en 0 y en 1 como es costumbre, pero bajo las formas dichas de dominante y recesivo. En el caso de una transmisión simultánea de los bits dominantes y recesivos el valor resultante del bus será dominante (Cf. [Peterson and Davie, 1996]).

De hecho, el protocolo CAN no define (ni impone) en nada el soporte físico (medio) sobre el cual el bus puede ser implementado (hilo de cobre, infrarrojo, interconexión trenzada, fibra óptica, corriente portadora.) Es por tanto difícil definir los niveles alto y bajo, 1 y 0 y, de este hecho, la escritura o la definición por ejemplo de un nivel transmitido parecería ser demasiado ambiguo salvo si hablamos de los bits dominantes o recesivos (Cf. [Peterson and Davie, 1996]).

La Figura 2.3 da algunos ejemplos concretos de estos valores e indica como por ejemplo, en los casos electrónicos convencionales se podría decir que el estado dominante

para el nivel transmitido sería el estado lógico 0 y el recesivo el estado lógico 1 (Cf. [Peterson and Davie, 1996]).

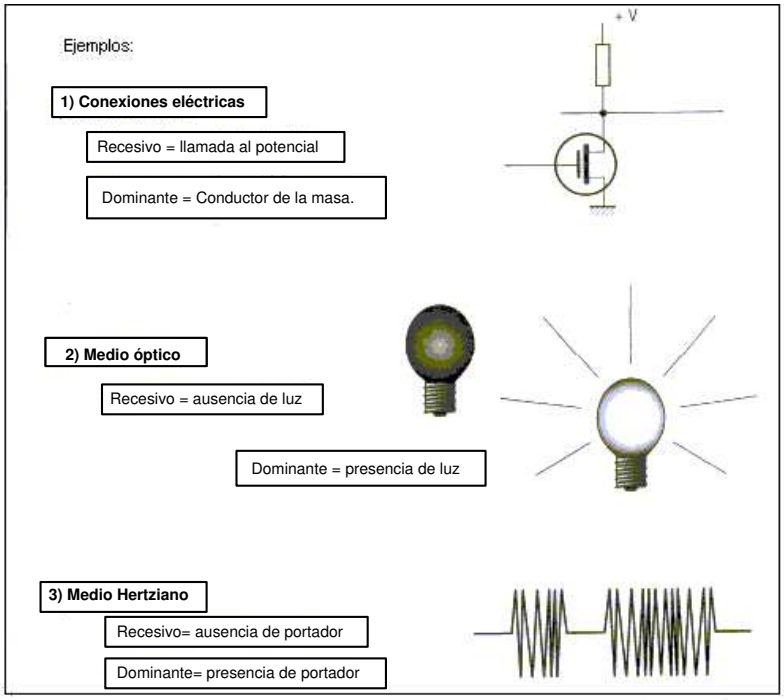

Figura 2.3: Ejemplos

2.3.4. Transferencia del bus (bite rate).

ISO: número de bits por unidad de tiempo durante la transmisión, independiente de la representación de los bits.

El caudal (llamado a menudo velocidad por abuso del lenguaje) del CAN puede ser diferente de un sistema a otro sin embargo, para un sistema dado, el caudal debe ser uniforme y constante; atención entonces a las compatibilidades de una red a otra por que es frecuente que muchas redes coexistan en los mismos sistemas y que uno se vea obligado a crear pasarelas de una red a la otra (Cf. [Peterson and Davie, 1996]).

2.3.5. Mensajes/trama/formato.

La información transportada por el bus es enviada en un formato definido y de una longitud máxima limitada (entonces también en el tiempo por un flujo dado).

Cuando el bus está libre, cualquiera de los participantes puede iniciar una nueva transmisión.

Este punto es muy específico en el concepto CAN. De hecho en una red, un nodo no debe de preocuparse por las informaciones que tienen que ver con las configuración del sistema en el cual está sumergido (Cf. [Peterson and Davie, 1996]).

Esto tiene muchas consecuencias importantes.

Flexibilidad de los sistemas.

Los nodos deben de ser agregados (retirados) de la red sin que sea necesario que se preocupe el diseñador de la red por el hardware y/o el software dentro de la misma, lo que importa es cual nodo y las capas aplicativas (Cf. [Peterson and Davie, 1996]).

Direccionamiento del mensaje.

Para asegurar el direccionamiento de un mensaje, el contenido de un mensaje es ubicado por un identificador *identifier* (Cf. [Peterson and Davie, 1996]).

Identificador

Este identificador no indica de ninguna manera el destino del mensaje, sino describe (en el sentido de dar una idea general o precisa) el contenido, del sentido de los datos. Esto implica que cada nodo debe ser capaz de decidir si el mensaje transmitido sobre el bus le interesa o no.

Esta función es (será o deberá ser) realizada por un dispositivo electrónico de filtrado de mensajes -*messages filtering*-. En otros términos un mensaje es emitido hacia toda la red en caso de que alguno (presente) esté interesado (Cf. [Peterson and Davie, 1996]).

Notas.

No existe ningún documento (original), ISO/OSI que indique o recomiende una estructura particular que tiene que ver con los tipos y arquitectura de mensajería a utilizar.Es bueno saber que el protocolo no da estrictamente ninguna indicación sobre los valores (posibles o recomendados) a ser atribuidos a los bits que constituyen el identificador.

Para asegurar una cierta flexibilidad de concepción de las redes sobre las cuales los

numerosos participantes pueden ser llevados a conectarse, ciertos grupos corporativos (por ejemplo el sector textil) se han organizado a fin de establecer las tablas de identificadores para evitar la anarquía potencial que representa (por definición) la gran y extendida gama de valores de estos identificadores posibles.

Ejemplo de elecciones de identificadores.

A título de documentación (y para dar algunas ideas más precisas) indicaremos a continuación en las siguientes líneas algunos ejemplos de organización de identificadores que pueden ser utilizados (Cf. [Peterson and Davie, 1996]).

Frecuentemente, en una red CAN, un nodo está físicamente situado en un lugar preciso (ejemplo, sobre el edificio D, tercer piso en la esquina noreste).Supongamos que en este nodo se tenga el permiso, entre otros, de indicar a la red la temperatura exterior de este lugar.

Si se tiene un identificador para este parámetro, se hablaría de la temperatura exterior noreste en el tercer piso en el edificio D y tendríamos identificado formalmente este parámetro. El identificador asociado al mensaje que indica el valor del parámetro (ejemplo 12.2 grados) podría presentarse entonces en cualquiera de las siguientes dos formas posibles:

- 1er caso: TEMP-EXT-NE3D

- 2o caso: o bien a la inversa D3NE-TEMP-EXT (Cf. [Peterson and Davie, 1996]).

De hecho, en el primer caso, se presenta una de las formas generales de componer un identificador. Se trata de la descripción clásica sujeto/fuente, sin embargo bien hubiéramos podido elegir el segundo caso.

La elección del orden de la información, de la arquitectura y de la mensajería se efectuará cuando se tome conocimiento del hecho del procedimiento del arbitraje que esta en función del valor del peso de los bits afectados en el identificador (Cf. [Peterson and Davie, 1996]).

Para regresar al primer caso se le puede descomponer como se ilustra en la Tabla 2.2.

Hay otros métodos que son utilizados para formar los identificadores, estos dependen de los usuarios y de las aplicaciones.

Identificador en CAN 2.0A sobre 11 bits			
Partida ID_alto Sujetos		Partida ID_base Fuentes	
sobre 6 bits (64 valores) (64 parametros)		sobre 5 bits (32 valores) (32 nodos)	
Sujeto principal	**Subsujeto**	**Fuente principal**	**Subfuente**
Temperatura (sobre 4 bits)	Exterior (sobre 2 bits)	D noroeste (sobre 3 bits)	3er. piso (sobre 2 bits)

Tabla 2.2: Elección de identificadores

Pluritransmisión (Multicast)

ISO: El multicast es un modo de direccionamiento en la cual se sabe con la ayuda de una simple trama, direccionar simultáneamente a un grupo de nodos. El modo difusión (broadcast) -direccionamiento de todos los nodos - es un caso particular del modo multicast.

La consecuencia inmediata del concepto de filtrado de mensajes reside en que todos los nodos reciben simultáneamente el mensaje transmitido y que todos son aptos a reaccionar inmediatamente a este si así lo desean.

Tenemos entonces aquí un dispositivo el cual en principio lo hace apto a reaccionar en tiempo real (Cf. [Peterson and Davie, 1996]).

Consistencia de los datos.

El hecho de haber estructurado el protocolo como se enuncio anteriormente, -un mensaje recibido puede ser tratado por uno o bien sea por todos o bien por ningún nodo- la consistencia de los datos de una red CAN esta asegurada por los principios de multicast y de tratamiento de errores (Cf. [Peterson and Davie, 1996]).

2.3.6. Trama de los datos (data frame), trama de solicitud (remote frame) y dato solicitado (remote data).

La *trama de datos* es la trama que transporta los datos. El termino *remote* es frecuentemente utilizado en el protocolo CAN. Esto es debido al hecho de que un nodo, cuando envía una *trama de solicitud* señala a los otros nodos presentes sobre la red que desea recibir los datos en la forma de una trama de datos. La trama de solicitud y la trama de datos asociada son ubicadas por el *identificador* (Cf. [Charzinski, 1994]).

2.3.7. Arbitraje.

ISO: procedimiento que consiste en atribuir el soporte de comunicación (bus de señalización) a uno de los nodos intentando tomar su control. Cuando el bus está libre (si todas las otras unidades puedan iniciar al mismo tiempo) dos o muchas unidades inician simultáneamente, esto crea un conflicto de bus el cual es resuelto por un arbitraje bit a bit (no destructivo) a lo largo del contenido del *identificador*. Este mecanismo de arbitraje garantiza que no habrá pérdida de tiempo, ni pérdida de información (Cf. [Charzinski, 1994]).

Evidentemente para el caso en el que se produce la inicialización simultánea de dos mensajes conteniendo el mismo *identificador*, en este caso, una *trama de datos* prevalece sobre una trama de solicitud. Cuando sucede la fase de arbitraje, cada emisor compara el nivel del bit transmitido con el nivel de bit que normalmente debería de transmitir el mismo (con la ayuda de un dispositivo interno de supervisión del bus). Si estos niveles son idénticos, el nodo continúa emitiendo.

Cuando un nivel recesivo es enviado y un nivel dominante se observa sobre el bus, la unidad considerada pierde el arbitraje, esta debe silenciarse y no enviar ningún bit (Cf. [Charzinski, 1994]).

2.3.8. Prioridades de acceso al bus.

Dado el principio de arbitraje bit a bit, el identificador (vía su contenido) define un mensaje de prioridad básica durante el acceso al bus (Cf. [Charzinski, 1994]).

2.3.9. Funcionamiento en multimaestro.

Cuando el bus está libre, cualquier unidad puede iniciar una transmisión. La unidad de la cual el mensaje poseerá la más alta prioridad (vía el contenido binario de su identificador) ganará el acceso al bus y transmitirá su mensaje (Cf. [Charzinski, 1994]).

2.3.10. Seguridad de transmisión.

A fin de asegurar una gran calidad y seguridad de transmisión, numerosos dispositivos de señalización, de detección de los errores, y de auto prueba han sido implementados con el propósito de aumentar la fiabilidad de la información transportada con la ayuda de CAN (Cf. [Charzinski, 1994]).

Detección de error.

Con respecto a las detecciones de los errores las medidas siguientes han sido tomadas:

- El monitoreo del bus: el emisor verifica que el nivel eléctrico que el desearía imponer sobre el bus está realmente presente sobre este,
- la presencia de un CRC (cyclic redundancy code o también cyclic redundancy check),
- un procedimiento de message frame check,
- una técnica de bit stuffing (que veremos más adelante) (Cf. [Charzinski, 1994]).

Desempeño del sistema de detección de error.

Teniendo en cuenta que son detectados:

- todos los errores globales,
- todos los errores locales al nivel de los emisores,
- hasta 5 errores aleatorios repartidos en un mensaje.

La probabilidad total residual de los mensajes unidos de errores es inferior a $4.7\text{x}10^{-11}$ (Cf. [Charzinski, 1994]).

2.3.11. Señalización de los errores y tiempos de recubrimiento de errores.

Todos los mensajes confundidos como errores son señalados en el nivel de cada nodo por una bandera (flag). Tales mensajes entonces son considerados sin fundamento y rechazados. Los mensajes erróneos deben ser retransmitidos automáticamente. El tipo de recubrimiento (entre el momento en que se detecta el error y el momento donde se reinicia el nuevo mensaje) es de un máximo de 29 bits para el CAN 2.0A, (31 bits para el CAN 2.0B) si no hay otros errores detectados (Cf. [Charzinski, 1994]).

2.3.12. Errores de confinamiento.

Estos son los errores de contorno, frontera y borde. Un nodo CAN debe de ser capaz de hacer las distinciones entre las perturbaciones de duración corta y los malos funcionamientos permanentes. Los nodos considerados como defectuosos pasan entonces en modo switch off desconectándose (eléctricamente de la red) (Cf. [Charzinski, 1994]).

2.3.13. Puntos de conexión.

La conexión de comunicación en serie CAN es un bus sobre el cual un número importante de unidades pueden ser interconectadas. Sobre este principio, su número es ilimitado. No hay una relación directa entre el número de identificadores posibles y el número de nodos conectados: no hay que confundir posibilidad de los campos de identificadores, direcciones y número de participantes (Cf. [Charzinski, 1994]).

En la práctica el número total de unidades será determinado por el tiempo de retardo (debido a los fenómenos de propagación) y/o los valores de las cargas eléctricas que estos valores presentan sobre el bus (Cf. [Charzinski, 1994]).

2.3.14. Canal de conexión simple.

El bus consiste en un simple canal bidireccional que transporta los bits. A partir de los datos transportados, es posible recuperar la información de resincronización, ya que como se vio en la sección 2.3.3 el protocolo no define ni impone en nada el soporte físico sobre el cual el bus debe ser implementado (Cf. [Charzinski, 1994]).

2.3.15. Grado de satisfacción.

Todos los receptores verifican la consistencia del mensaje recibido y aprueban un mensaje consistente o bien agitan una bandera de error en caso de mensaje inconsistente.

2.3.16. Puesta en hibernación y despertador.

A fin de reducir el consumo del sistema, los elementos CAN pueden ser puestos en hibernación -*sleep mode*- y desconectar aparentemente sus controladores del bus. El *sleep mode* es terminado por un despertador automático que se inicializa desde la puesta en actividad del bus o bien por razones propias a la estación misma (Cf. [Charzinski, 1994]).

Al despertar la actividad interna de la estación es retomada, aunque en la subcapa MAC (Médium Access Control) se espera a que el oscilador local de la estación se estabilice y se sincronice sobre la actividad del bus (verificando la presencia de 11 bits recesivos consecutivos).

Una vez que esto termina los drivers del bus son autorizados a conectarse nuevamente sobre este (Cf. [Charzinski, 1994]).

Hasta aquí se ha terminado la presentación simplificada de los términos principales y característicos intrínsecos de este bus que como se ha visto, es de un alto desempeño. De este hecho, está generalmente más adaptado a situaciones profesionales de buen nivel.

2.4. Los grandes principios.

2.4.1. Un poco de electrónica.

Como ya se ha señalado, la capa física descrita en el documento de referencia no indica de ninguna manera los niveles físicos (eléctricos, ópticos) que pueden estar presentes sobre el bus. Hablar en absoluto es algunas veces difícil de comprender, a fin de ser un poco más concretos, digamos que frecuentemente en el caso de las aplicaciones con interconexión de tipo alambrado, el medio de comunicación se presenta bajo la forma de un par diferencial (paralelo o trenzado), entonces con diferentes niveles eléctricos bien precisos que definen los niveles lógicos 1 y 0.

Cuando el bus esta en reposo no hay actividad y la pantalla del osciloscopio está normalmente vacía como en el caso de todo bus asíncrono.

En principio, se comenzará por describir en detalle lo que se puede observar cuando se esta monitoreando las tramas del bus CAN estándar del tipo 2.0A.

2.4.2. Trama estándar CAN 2.0A.

La trasferencia de los mensajes es controlada con la ayuda de 4 tipos de tramas específicas y de un intervalo de tiempo que las separa (Cf. [Robert BOSCH, 1991]).

Se trata de:

- *Trama de datos*. Las tramas que transportan los datos de los emisores hacia los receptores.
- *Trama de solicitud*. Estos tipos de tramas son emitidos por una unidad presente sobre el bus para solicitar la transmisión de una trama de datos de la cual el identificador tendrá el mismo valor de aquella de la trama de solicitud.
- *Error frame* -trama de errores. Estos tipos de tramas de errores son transmitidos por cualquiera de las unidades presentes sobre el bus desde que se detecta un error sobre el mismo.
- *Overload frame* -trama de sobrecarga. Este último tipo de trama que se llama de sobrecarga es utilizado para solicitar un lapso de tiempo suplementario entre las tramas de datos o trama de solicitud precedentes y sucesivos.
- *Interframe* -intertrama.

  Las tramas de datos y las tramas de solicitud están separados (temporalmente) de las tramas precedentes por un *interframe space* (Cf. [Unruh et al., 1989]).

Todas estas tramas transportan las informaciones emitidas sobre el bus al nivel más bajo de la capa física con la ayuda de bits.

Con el fin de asegurar una mejor comprensión de ciertas fases de las diferentes tramas y de los tratamientos de errores, se examinará a continuación algunos detalles que contienen los bits y la forma de codificarlos en el caso del protocolo CAN.

2.4.3. Bit CAN.

Cuando vemos los problemas que tienen que ver con la capa física del bus CAN revisaremos las particularidades del bit CAN sin embargo desde ahora, a fin de no tener una teoría demasiado complicada vamos a definir esa noción.

Para definir un bit, hay que definir su duración y la forma en la cual está codificada en su contenido (su valor) durante el curso de esta duración. Existen múltiples formas de codificar un bit; citemos, por ejemplo, la codificación NRZ, bifase, trifase, duobinaria, etc. (ver la Figura 2.4) (Cf. [Unruh et al., 1989]).

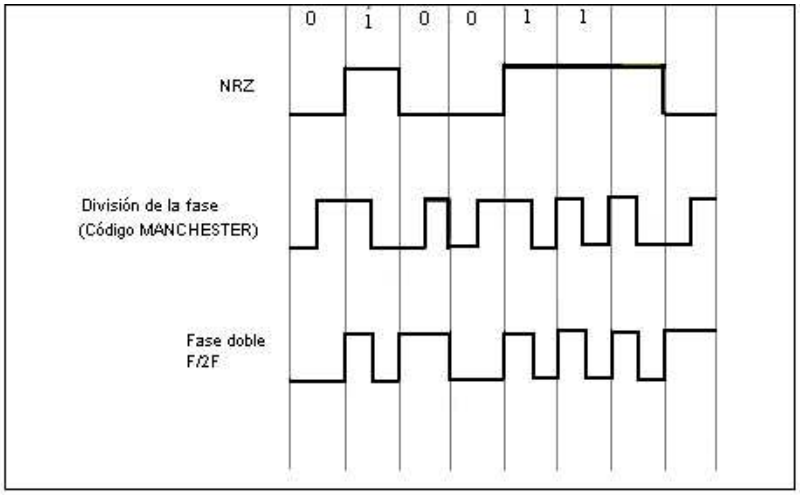

Figura 2.4: Ejemplos de códigos.

Codificación NRZ.

Con respecto al número de bits de las tramas del bus CAN, el método de codificación Non Return to Zero -NRZ- fue tomado en cuenta. Esto quiere decir que durante la duración total del bit generado, su nivel es constante, si el es dominante o recesivo (Cf. [Unruh et al., 1989]).

Todos los diseños de las tramas que presentaremos enseguida en esta monografía serán entonces representadas de esta manera. Si intentáramos observar en un osciloscopio las tramas no aparecerían de ninguna manera como se describen en este trabajo debido a que, una vez que han sido arreglados en capas (como una cebolla) en el registro de transferencia de un controlador, pasan por un convertidor particular antes de ser emitidos físicamente sobre el soporte de comunicación (Cf. [Unruh et al., 1989]).

Un proceso suplementario particular (entre muchos otros) ha sido instaurado para hacer más seguro el mensaje cuando se transporta sobre el bus. Se trata del método llamado de rellenado de bits (bit stuffing) (Cf. [Unruh et al., 1989]).

Método de Bit Stuffing

El método de bit stuffing es simple. Dado que el bit está codificado en NRZ, es posible que un mensaje en particular contenga muchos bits del mismo valor (nivel) y que pueda hacer creer a una o a muchas estaciones que hay posiblemente una anomalía sobre la red. También, ha sido voluntariamente introducido (en el nivel de transmisión) después de 5 bits de valor idéntico (sean dominantes o recesivos) un bit suplementario de valor opuesto para romper el ritmo y señalar así que todo va bien, contrariamente a lo que se pudiera pensar. Es lo que se llaman bits de *reemplazo* o también de *rellenado de bits*, y en inglés stuff. (Ver el ejemplo de las Figuras 2.5 y 2.6)

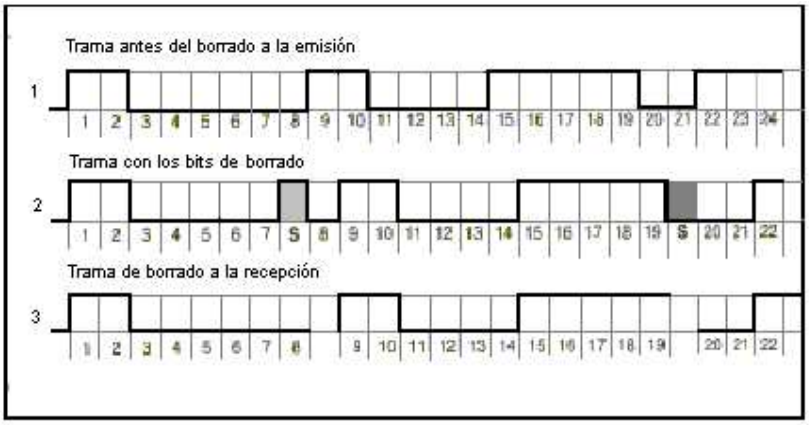

Figura 2.5: Llenado de bits

En principio, esto alarga el tiempo de transmisión de un mensaje sin embargo, participa activamente al hacer más seguro su contenido cuando se le transporta.

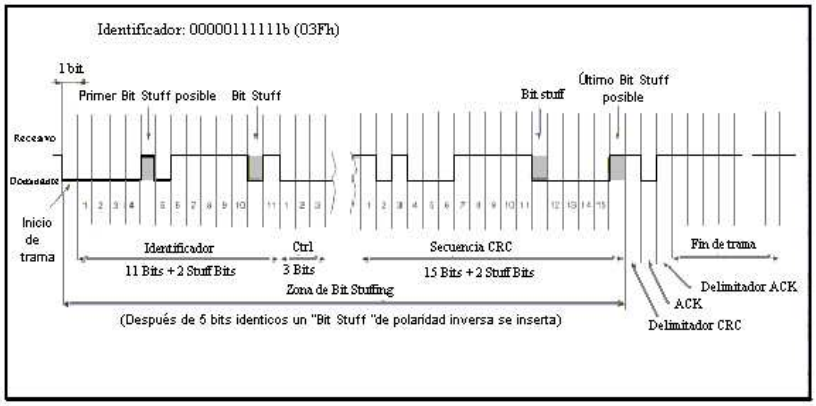

Figura 2.6: Llenado de bits

Por otro lado, se introduce en el espectro de la señal inicial NRZ (que comprende la componente continua) una componente alternativa mínima que la señal no poseía anteriormente (Cf. [Unruh et al., 1989]).

Evidentemente un receptor CAN debe de estar perfectamente al corriente de esta técnica de rellenado y procederá entonces a la función inversa de *des-stuffing* en la recepción retirando estos bits de reemplazo habiendo servido solamente para el transporte (Cf. [Unruh et al., 1989]).

La constitución hardware de los circuitos integrados dejan completamente transparente al usuario esta técnica de sobre-codificación de bits pero señala los errores si por casualidad se produjeran (Cf. [Unruh et al., 1989]).

Precisemos un último punto importante que tiene que ver con las aplicaciones que se llaman de tiempo real. En efecto, al hecho de integrar cuando se hace el intercambio, de los bits de stuffing permite crear un número mayor de transiciones en la comunicación y por tanto facilitar la sincronización de los bits a pesar del tipo de codificación de bit NRZ.

Notas:
Como se detallará más adelante solo los fragmentos de las tramas que tienen que ver con los:

- principio de la trama (Start of frame)
- Campo de arbitraje (arbitration field)
- Campo de control (Control field)

- Campo de datos (Data field)
- Secuencia de CRC (CRC sequence)

éstas estan codificadas utilizando la técnica de bit stuffing.

Cuando el emisor detecta 5 bits consecutivos de valores idénticos en el cúmulo de bits que deben de ser transmitidos, el inserta automáticamente un bit complementario en el cúmulo de bits que están siendo transmitidos (Cf. [Unruh et al., 1989]).

Los bits que quedan de los otros segmentos de las tramas de datos o trama de solicitud como son los:

- CRC delimiter
- Ack field
- End of frame,

que poseen estructuras fijas y no son rellenadas (Cf. [Unruh et al., 1989]).

Para concluir con esta sección con las *tramas de error* y *tramas de sobrecarga*, estas estructuras son fijas y por lo tanto no son codificadas por el método de *bit stuffing* (Cf. [Unruh et al., 1989]).

Veamos ahora en detalle el contenido de las diferentes tramas que pueden circular sobre el bus.

2.4.4. Trama de datos (data frame).

Empecemos por examinar la constitución de la trama de datos del tipo estándar CAN 2.0A, la más utilizada (ver ejemplos 2.7 y 2.8)

Esta trama se descompone en 7 partes principales que se llaman campos:

Y después, una octava zona que se llama interframe (Inter-trama) que hace parte integrante de la trama. Veamos más en detalle el contenido de cada uno de estos campos para una mejor comprensión del funcionamiento del bus (Cf. [Unruh et al., 1989]).

Inicio de la trama

El inicio de la trama de datos (SOF) está constituida de un solo bit dominante que señala a todas las estaciones el inicio de un intercambio, este intercambio no puede iniciar solamente si el bus estaba en reposo.Todas las estaciones deben de sincronizarse sobre el flanco antes de la transición del bit de inicio (Cf. [Unruh et al., 1989]).

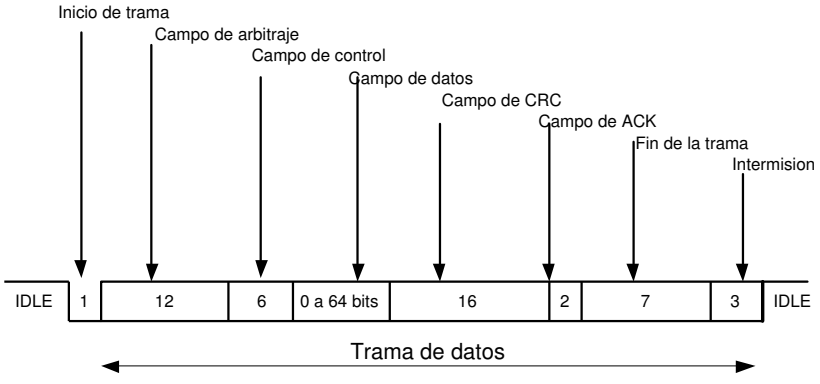

Figura 2.7: Trama de datos

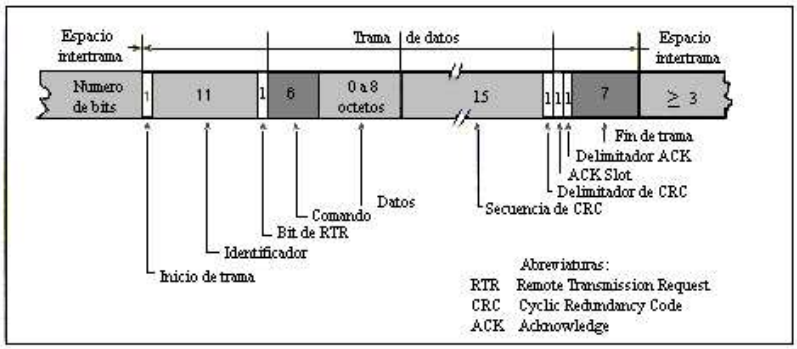

Figura 2.8: Trama de datos (continuación)

Inicio de trama	Start of frame (SOF)
Campo de arbitraje	Arbitration field
Campo de control	Control field
Campo de datos	Data field
Campo de CRC	CRC field
Campo de control	Control field
Campo de reconocimiento	ACKnowledgment field
Fin de trama	End of frame (EOF)

Tabla 2.3: Segmentos de la trama de datos

Campo de arbitraje

ISO: conjunto de bits de la trama del mensaje atribuido a cada mensaje para controlar el arbitraje.

El campo durante el cual se efectúa el arbitraje está constituido de los bits del identificador así como del bit siguiente que se le llama RTR (Remote Transmisión Request) (Cf. [Unruh et al., 1989]) (ver las Figuras 2.9, 2.10 y 2.11.)

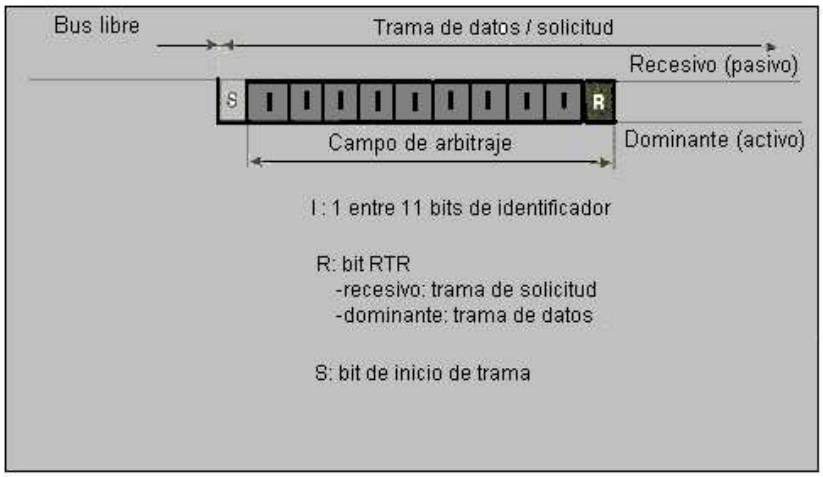

Figura 2.9: Campo de arbitraje

Identificador.

La longitud del identificador es de 11 bits y los bits son transmitidos en el orden, de ID_10 a ID_0 (el menos significativo es ID_0). Por otro lado los 7 bits los más significativos (de ID_10 a ID_4) no deben ser recesivos (Cf. [Unruh et al., 1989]).

El bit RTR.
Cuando sucede una *trama de datos*, el bit de remote transmission request (RTR) debe ser dominante (Cf. [Unruh et al., 1989]).

Campo de control.

Se constituye de 6 bits (ver la Figura 2.12) (Cf. [Unruh et al., 1989]).
Bits de reserva.

Los 2 primeros (bits emitidos dominantes en trama 2.0A) son reservados para posteriores usos y permite asegurar las compatibilidades futuras (sobre todo aquellas de la

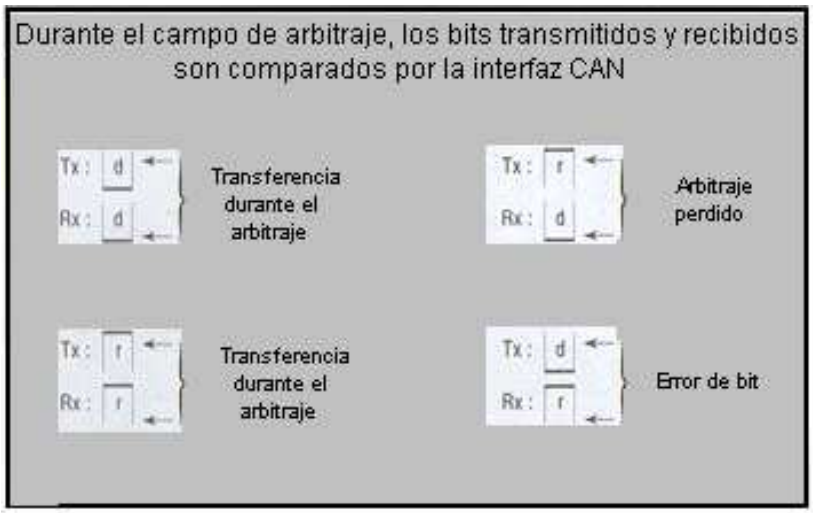

Figura 2.10: Campo de arbitraje

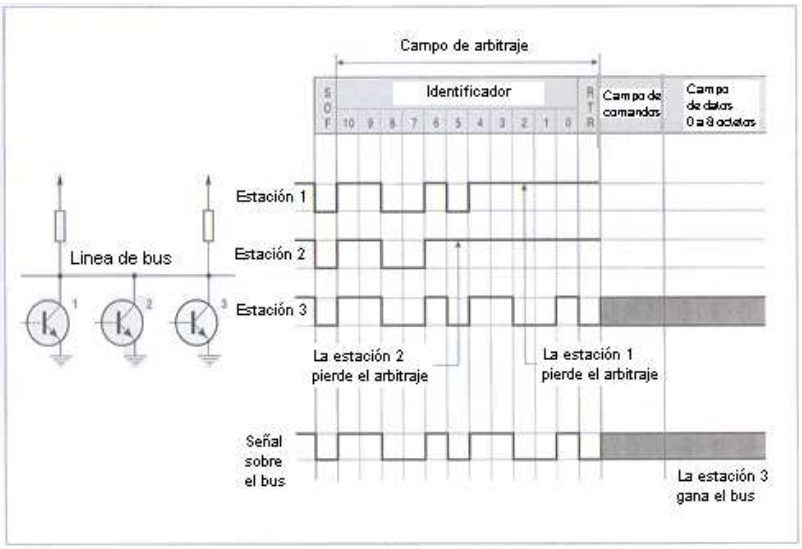

Figura 2.11: Campo de arbitraje

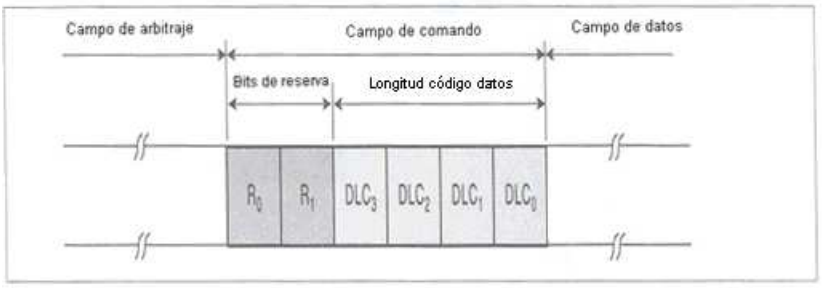

Figura 2.12: Campo de control

trama que se dice extendida CAN 2.0B que veremos al final del capítulo). Los controladores CAN deben ser aptos para tratar todas las combinaciones posibles de todos los bits del campo de control (Cf. [Unruh et al., 1989]).

Número de datos contenidos.

Los 4 últimos bits del campo de control (campo DLC- Data Length Code) indican el número de octetos que estarán contenidos en el campo de datos (Cf. [Unruh et al., 1989]).

Campo de datos.

El campo de datos es el lugar donde se encuentran los datos útiles transmitidos. Este campo puede estar compuesto de 0 bytes mínimos a 8 bytes máximos transmitidos con el MSB (Most Significant Bit) a la cabeza[2] (ver la Figura 2.13). (Cf. [Unruh et al., 1989])

Figura 2.13: Campo de datos

[2] De 0 a 8 incluido, esto nos hace 9 valores de los cuales 4 bits del DLC para definir el número de datos contenidos y no 3 bits como podría pensarse de entrada.

Campo de CRC.

CRC: Cyclic Redundancy Code (código con redundancia cíclica). Se compone de la zona CRC *sequence* seguido de un delimitador de CRC (ver la Figura 2.14) (Cf. [Unruh et al., 1989]).

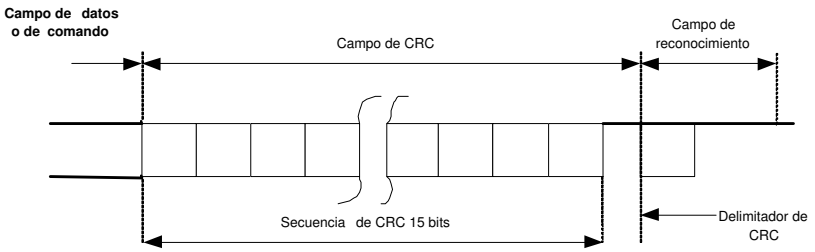

Figura 2.14: Campo de datos

Secuencia de CRC.

Para asegurar la validez del mensaje transmitido, todos los receptores deben restringirse a proceder a la verificación de una secuencia CRC generado por el emisor CAN en relación con el contenido del mensaje transmitido. Los códigos utilizados por los controladores del bus CAN son los códigos de BCH (acrónimo que tienen el origen del nombre de los autores: Bose, Chaudhuri y Hocquenghem) a los cuales se agregaron una verificación de paridad que posee los atributos siguientes:

- Longitud máxima del código: 127 bits,
- Número de dígitos de información máxima: 112 bits (un máximo de 83 son utilizados por el controlador de bus CAN),
- Longitud de secuencia de CRC: hasta 15 bits,
- Distancia de Hamming: d = 6, o sea también (d - 1) = 5, es decir que 5 errores independientes de bits son 100 % detectables en el interior del código transmitido. [3] (Cf. [Unruh et al., 1989])

La secuencia de CRC está determinada (calculada) por el emisor del mensaje con la ayuda del procedimiento siguiente:

- La flotilla de bits (no llenados o vacíos), constituidos de los bits desde el inicio de la trama hasta el fin del campo de datos (si hay), es interpretado como un polinomio

[3] La distancia de Hamming entre 2 palabras binarias de la misma longitud es el número de elementos binarios del mismo rango que difieren en las 2 palabras.

$f(x)$ con coeficientes 0 y 1 afectados cuando están presentes, efectivos o no, de cada bit. El polinomio que se obtiene de esta manera es además completado por los 0 para los 15 coeficientes menos significativos (Cf. [Unruh et al., 1989]).

- El polinomio así formado es dividido (módulo 2) por el generador polinomial siguiente, que incluye una verificación de paridad. $g(X) = X^{15}+X^{14}+X^{10}+X^8+X^7+X^4+X^3+1$. O bien transcrito de forma binaria: $g(X) = 1100010110011001$ (Cf. [Unruh et al., 1989]).
- Después de la división del polinomio $f(x)$ por el polinomio generador $g(x)$, el resto de esta división polinomial constituye la secuencia CRC de 15 bits. Es esta última la que se transmite en el interior del campo de CRC cuando sucede la secuencia de CRC (Cf. [Unruh et al., 1989]).

A manera de ejemplo haremos los cálculos en las Figuras 2.15 y 2.16.

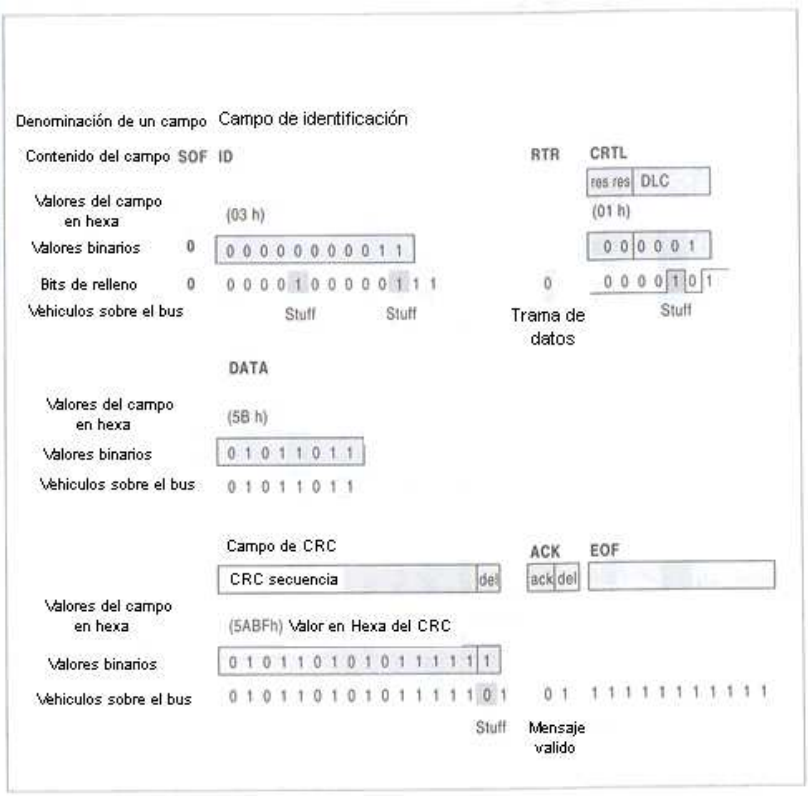

Figura 2.15: Ejemplo de la trama de datos

Complementos de información.

Más que un *checksum*, es un CRC que fue retenido para el protocolo CAN debido a los códigos cíclicos que poseen las propiedades siguientes:

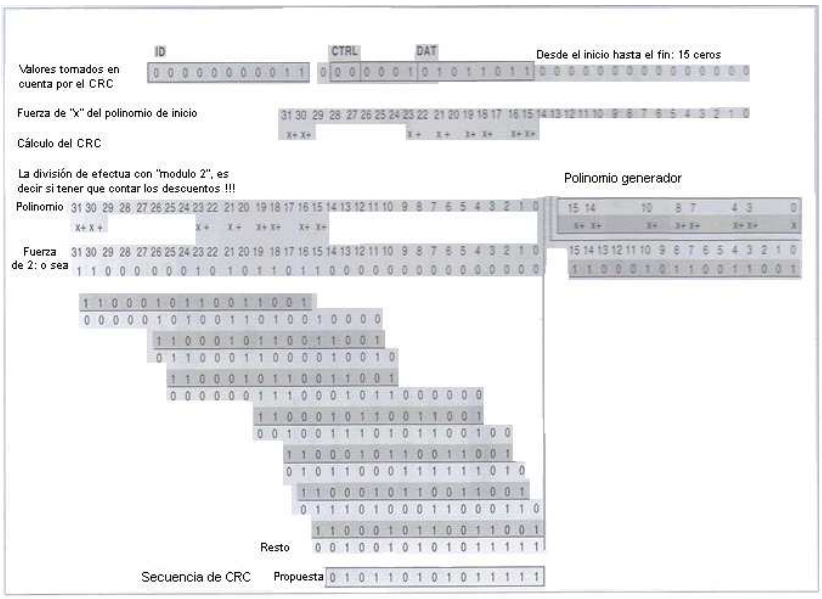

Figura 2.16: Ejemplo del cálculo del CRC del ejemplo anterior

- Simplicidad de la codificación y de la decodificación,
- Capacidad de detectar y de corregir los errores independientes o que se presentan por paquetes.

  Debemos señalar también, que entre los códigos lineales conocidos, es el de BCH que ha sido retenido debido a que, para un rendimiento dado, este código pertenece a los códigos que presentan la más grande capacidad de corrección para los errores independientes. Esta capacidad es de hecho de alto grado de satisfacción debido a que ella se sitúa muy cerca del óptimo previsto por la teoría. Además, la realización electrónica de los codificadores y decodificadores de este código es particularmente simple con la ayuda de registros desplazados (Cf. [Unruh et al., 1989]).

  La teoría y los cálculos estadísticos indican que los errores múltiples superiores a 6 bits diferentes en todo el contenido no son detectados con una probabilidad de error residual de 2-15 (aproximadamente 3 x 10^{-5}) para la parte CRC solamente (Cf. [Unruh et al., 1989]).

  El hecho de que el *start of frame* (bit dominante) esté incluido en la contraseña de código, ninguna rotación de la contraseña de código puede ser detectada en ausencia del delimitador de CRC (bit recesivo) (Cf. [Unruh et al., 1989]).

- Cuando sucede la recepción, el mensaje (una vez vaciado) que comprende: el inicio de la trama, el campo de arbitraje, el campo de control, el campo de datos, está sometido a su verificación con la ayuda de un CRC construido alrededor de un mismo generador polinomial:

$$X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1 \tag{2.1}$$

El campo termina enseguida por un *delimitador CRC* (CRC delimiter) que consiste en un simple bit recesivo [4] . (Cf. [Unruh et al., 1989])

Campo de reconocimiento.

Este campo se compone de 2 bits, el bit de ACK *slot* y ACK *delimiter* (ver la Figura 2.17).

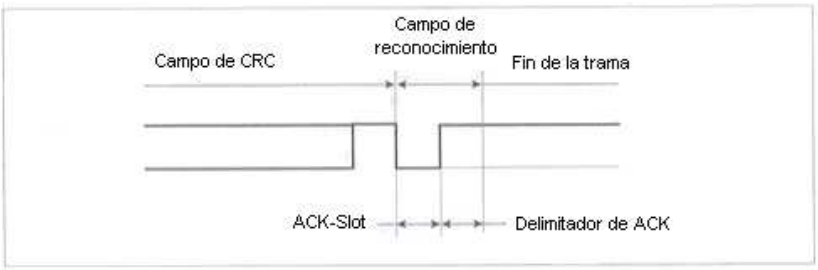

Figura 2.17: Campo de reconocimiento

Cuando sucede una transmisión, la unidad emisora emite 2 bits recesivos sobre el bus. Es decir el emisor deja el bus libre y pasa a modo de escucha en modo de receptor (Cf. [Unruh et al., 1989]).

Veamos ahora las consecuencias que son bastante interesantes de lo que venimos de enunciar.

ACK Slot.

Cada vez que un receptor recibió correctamente un mensaje -en el sentido de que no tiene un error de transmisión (incluido también el CRC)-, el mensaje está siendo considerado entonces como válido o bien informa al emisor superponiendo entonces al *time slot* del ACK *slot* un bit dominante al bit recesivo que estaba presente en ese

[4] En el protocolo CAN, se restringió la aplicación del basto concepto CRC a la sola detección de errores (y en consecuencia más tarde a sus señalizaciones) y no se extendió, como esto es normalmente el caso, a la corrección de dichos errores.

momento. Envía entonces un reconocimiento ACK.

Notas importantes.

Si la estación emisora recibe un reconocimiento, es que al menos una estación presente en la red recibió completamente el mensaje y sin error. La presencia del bit de reconocimiento no indica para nada que el receptor afectó o rechazó el contenido del mensaje (cualquiera que sea el interesado o no por el contenido del mensaje). La única conclusión que podemos sacar de esto es que en cuanto al nivel, el receptor considerado recibió físicamente en sus puertos un mensaje potencialmente utilizable y sin ningún error (Cf. [Unruh et al., 1989]).

Todos los receptores que están disponibles en la red deben permitir una señal de reconocimiento sobre la red cuando no han detectado un error cualquiera que este sea (Cf. [Unruh et al., 1989]).

En el caso contrario -el mensaje que está siendo considerado como no válido- el receptor no regresará algún reconocimiento. Además, como es bien conocido, dado que el mensaje no es válido es que contiene uno o varios errores. Para satisfacer el protocolo CAN, el receptor tendrá por obligación de señalar los errores transmitiendo una trama de error (Cf. [Unruh et al., 1989]).

Entonces tenemos que tratar con un concepto que se llama memoria compartida y en consecuencia el reconocimiento y tratamiento de los errores están definidos para proveer toda la información de una manera consistente a través de la memoria compartida. En principio, no hay ninguna razón para discriminar los diferentes receptores o los mensajes en el campo de reconocimiento. Contrariamente a otros protocolos (por ejemplo el I2C), este reconocimiento no significa para nada el hecho de que un nodo esté presente o no, o tampoco que halla un nodo interesado por el mensaje en la red (Cf. [Unruh et al., 1989]).

Veamos con un poco más de detalle estas 2 eventualidades.

1.- Presencia o ausencia de un nodo: Una consecuencia importante de lo que acabamos de describir es que si una estación está desconectada del bus (por ejemplo, en modo bus off ó también retirado del bus de forma voluntaria ó también si algunos hilos del bus están seccionados), esta estación ya no será parte integral de la memoria compartida (Cf. [Unruh et al., 1989]).

2.- Interés o no a los mensajes transmitidos (y recibidos): Es necesario entonces analizar el interés que tengan las estaciones de la red al contenido intrínseco del mensaje

transmitido (Cf. [Unruh et al., 1989]).

En estos dos últimos casos, la identificación de un nodo perdido y el interés del mensaje, será entonces el deber del diseñador de la red de tomar las precauciones adicionales al nivel de su programa aplicativo específico. Entonces será necesario, si así se desea, establecer los nodos de los procedimientos con el programa (en el nivel 7 -application layer- del ISO/OSI) para que estos reconozcan de forma programática las presencias relativas o también el interés que ellos tendrán a los mensajes emitidos y recibidos (Cf. [Unruh et al., 1989]).

Sin revelar el sujeto, es bueno saber desde ahora que las capas de programación aplicativas que permiten administrar estos problemas han sido ya desarrolladas por los grupos de usuarios que pueden tener los mismos problemas.

ACK Delimiter (Delimitador ACK).

Este segundo bit debe ser siempre recesivo, lo que tiene por consecuencia que, cuando un mensaje ha sido perfectamente recibido por todas las estaciones presentes sobre el bus, el bit de ACK slot (dominante) está rodeado por dos bits recesivos (CRC delimiter y ACK delimiter) (Cf. [Unruh et al., 1989]).

Fin de la trama de datos.

La trama de datos se termina por una bandera formada por una secuencia de 7 bits recesivos, lo que, entre paréntesis supera en 2 bits la longitud de la norma de bit stuffing.

Este campo tiene una estructura fija y la lógica de codificación (en la emisión) y la de decodificación (en la recepción) de bit stuffing son desactivadas durante la secuencia del campo de fin de la trama (Cf. [Unruh et al., 1989]).

Intertrama (Interframe).

Después viene una octava zona, que se llama de espacio interframe, que veremos con detalle un poco más adelante (Cf. [Unruh et al., 1989]). A continuación presentamos la Figura 2.18 que resume e ilustra lo esencial de las explicaciones precedentes.

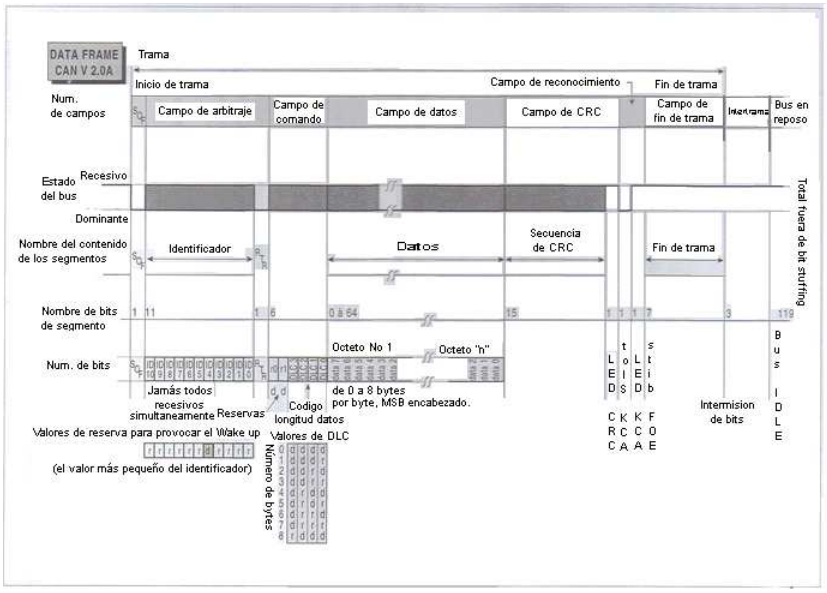

Figura 2.18: Espacio de intertrama.

2.5. Trama de solicitud (Remote frame).

Como se acaba de describir, cada uno emite sin saber si la información enviada servirá a uno de los participantes.

Puede suceder también que un nodo tenga necesidad de información de un cierto nodo de la cual no dispone en ese momento para asegurar la misión para la cual el ha sido programado. En este caso, una estación que necesita datos puede inicializar la demanda de una transmisión de los datos considerados por otro nodo enviando un trama de solicitud (Cf. [Unruh et al., 1989]). Veamos la constitución de las tramas de solicitud (ver la Figura 2.19).

Esta trama no se compone más que de 6 partes en lugar de las 7 precedentes:

- El principio de la trama,
  - El campo de arbitraje,
  - El campo de control,
  - El campo de CRC,
  - El campo de reconocimiento,
- el fin de la trama,

y después una séptima zona que se llama de espacio *interframe* (Cf. [Unruh et al., 1989]).

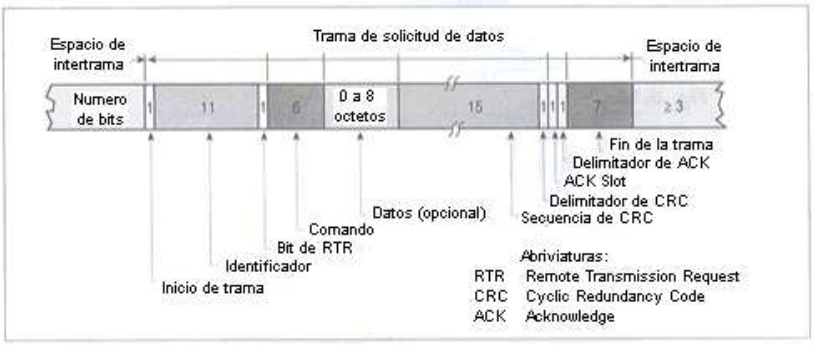

Figura 2.19: Tramas de solicitud.

Es más sencillo razonar de manera diferencial con respecto a la trama de datos que se acaba de ver en vez de repetir todo.

2.5.1. Inicio de trama.

El inicio de la trama de datos (SOF) está constituido de un solo bit dominante que señala a todas las estaciones el inicio de un intercambio. Este intercambio no puede iniciarse más que cuando el bus esté de forma precedente en reposo (Cf. [Unruh et al., 1989]).

Todas las estaciones deben de ser sincronizadas sobre el flanco antes de la transición del bit de inicio ver más adelante el mecanismo de sincronización hardware.

2.5.2. El campo de arbitraje.

El campo durante el cual se efectúa el arbitraje está constituido de los bits del identificador así como del bit que le sigue inmediatamente que se llama RTR (Remote transmission Request) (ver las Figuras 2.20 y 2.21) (Cf. [Unruh et al., 1989]).

Figura 2.20: Campo de arbitraje

Identificador.

La longitud del identificador es de 11 bits y los bits son transmitidos en el orden, de ID_10 a ID_0 (el más significativo es ID_0). Por otro lado los 7 bits los más significativos (de ID_10 a ID_4) no deben ser todos recesivos (Cf. [Unruh et al., 1989]).

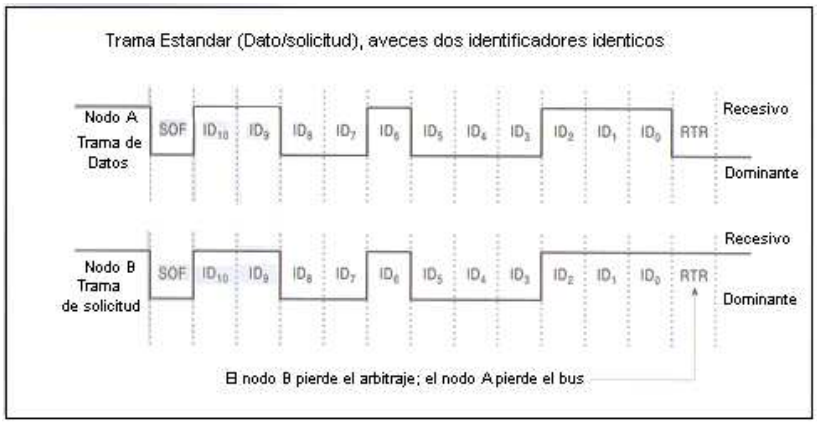

Figura 2.21: Campo de arbitraje

Bit RTR.
Contrariamente al caso precedente, el mensaje de una trama de solicitud, este bit es recesivo. Es entonces que este bit hace la diferencia entre una trama de datos y una trama de solicitud (Cf. [Unruh et al., 1989]).

Corolario.
Siendo el bit RTR por definición siempre recesivo para una trama de solicitud, este quiere decir que para un mismo identificador, una trama de datos es siempre prioritaria sobre una trama de solicitud (lo que parece razonable dado que este contiene las informaciones que la trama de solicitud se suponía va a necesitar en otro instante).

2.5.3. Campo de control.

Esta constituido de 6 bits (ver la Figura 2.22).

Bit de reserva: idem a la *trama de datos.*

Número de datos: los últimos 4 bits indican el número de bytes contenidos en el campo de datos y no en la trama de solicitud (ver un poco más adelante) no es más que la tra-

Figura 2.22: Campo de Control

ma de datos quien tendrá por misión de llevarlos enseguida [5] (Cf. [Unruh et al., 1989]).

2.5.4. Campo de datos.

No hay un campo de datos si esto no importa cual sea la longitud del mensaje que hubiera podido ser declarado en el campo de control (Cf. [Unruh et al., 1989]).

2.5.5. Campo de CRC, de reconocimiento, de fin de la trama de datos, de intertrama.

Estos campos son estructuras idénticas a las que se vieron en la trama de datos.

Para resumir todo esto examinemos la representación de un ejemplo de la trama de solicitud (ver Figura 2.23).

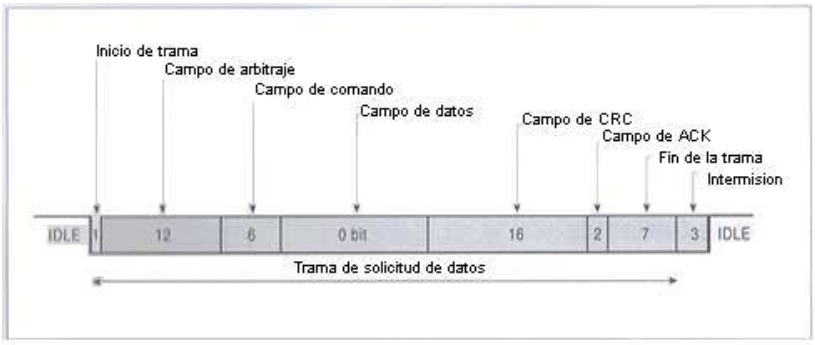

Figura 2.23: Ejemplo de la trama de solicitud

[5] Sobre este principio, hay que indicar que un número de datos no sirve para nada dado que el bit RTR es recesivo. De hecho es recomendado especificar correctamente este campo para que no se produzca un error sobre el bus si dos (microcontroladores) iniciaran una transmisión de una trama de solicitud simultáneamente.

Hasta aquí hemos supuesto que todo va bien y que los datos de las tramas de datos y/o de la trama de la solicitud de datos circulan sin problema.

Sin embargo, esto no es tan fácil, y a veces, algunos ruidos perturban las tramas, también a fin de no interrumpir el hilo de esta monografía del transporte de los datos decidimos presentar la forma en la cual se termina una trama (los interframe y los overload frame) al final de este Capítulo y de pasar directamente a la revisión de los errores (Cf. [Unruh et al., 1989]).

2.6. Los errores, detección y tratamiento.

Introducción.

Esta sección es una de las más importantes del protocolo CAN y es indispensable entender bien el mecanismo (que es bastante complicado pero de una alta eficacia) para beneficiarse completamente de todas sus ventajas. La detección y tratamiento de errores esta frecuentemente ocupada por el hecho de que los componentes que satisfacen al protocolo administran todos esos detalles internamente en los circuitos dado que ellos están conformes al protocolo, dado que el usuario final no debe de tomar ningún tipo de atención a ellos.

De hecho los modos de detección y de corrección de errores son muy potentes, y merecen estudiarse en detalle, en caso de problemas, y no tener que reinventar durante horas, en software, todo lo que ya ha sido bien realizado en hardware.

Generalidades

Dado que la complejidad del tratamiento de los errores es alta, no es tan simple como parecería presentar el contenido de esta sección sin que no tengamos que entrar en detalles un poco complicados.

Después de algunas lecturas del protocolo original y de revisar ampliamente documentos sobre este tema, y de ver el método de presentación que se debería de adoptar, se optó por tener el siguiente plan que se compone de dos grandes partes.

Enfoque general.

- Descripción de los diferentes tipos de errores que se pueden producir.
- Filosofía y estrategia global desarrolladas en el tratamiento y recubrimiento de los errores.

Detalle.

- Las técnicas de detección de los errores.
- Las técnicas (y tramas) de señalización de los errores.
- Las técnicas de recubrimientos de los errores.

2.6.1. Tipos de errores.

A fin de poder deshacerse de los ruidos, el protocolo CAN posee numerosos recursos de muy alto desempeño. En uno de los párrafos precedentes, se indicó rápidamente las principales fuentes de errores que retomaremos ahora en detalle de forma jerárquica.

Muchos tipos de errores se pueden producir:

- en el nivel de la capa física misma (bit de error y error de bit stuffing):
  - el bit mismo el cual contienen errores (ruidos, por ejemplo),
  - un error de bit stuffing por razones involuntarias (ruidos, transmisiones, olvidos, etc.) o algunas veces voluntarios, como lo veremos enseguida en la trama de error.
- Al nivel del bit y de la estructura de la trama, si todo va bien pero, por ejemplo:
  - Ella no fue reconocida provocando un *ACKnowledgement error*,
  - El valor del CRC no corresponde al que se esperaba.
- Al nivel de la capa de la trama, en el sentido en que su estructura íntima ha sido violada, (información que no esta en su lugar):
  - Error de *CRC delimiter*,
  - Error de *ACKnowledgement delimiter*,
  - Error de *end of frame*,
  - Error de *error delimiter*,
  - Error de *overload delimiter*.

En todos estos casos, la presencia de los errores será señalado por una trama de error -*error frame*- quien será generado sobre el bus para informar a quien corresponda. Antes de detallar su estructura y su mecanismo, nos hace falta un poco de filosofía y algunas definiciones que tienen que ver con lo que llamamos errores de confinamiento (Cf. [Unruh et al., 1989]).

2.6.2. Boletín de salud de una red.

De hecho, dejando de lado el hecho de que los errores se pueden producir, es siempre útil saber:

- De que tipo son,
- Si se producen raramente, si ellos son realmente muy molestos,
- Que los microcontroladores que administran la red deben ser informados cuando hay perturbaciones durables,
- Y cuando la actividad del bus regresa a la normalidad (Cf. [Unruh et al., 1989]).

Cuando hay perturbaciones persistentes, la parte del controlador del bus pasa en modo que llamamos bus off y la parte CPU local puede entonces tomar los valores convenidos por defecto. La presencia de perturbaciones menores (o de duración corta) sobre el bus no afecta la parte del controlador del bus.

Para realizar esto con el término confinamiento se sobrentiende un enorme mecanismo que tiene por meta el de adaptarse a determinar si un nodo:

- no esta perturbado para nada,
- esta un poco perturbado,
- esta un poco más que gravemente perturbado,
- esta tan perturbado que el debe conmutar a *bus off* (Cf. [Unruh et al., 1989]).

2.6.3. Mecanismo de tratamiento de los errores de confinamiento.

Una de las finalidades de este mecanismo consiste en permitir la detección de los errores y perturbaciones del hardware, sin embargo, también y sobre todo proceder a su localización a fin de poder intervenir con precisión.

Las reglas del tratamiento de los errores de confinamiento están descritas en el protocolo CAN de tal forma que los microcontroladores siendo los más próximos del lugar donde se produce el error, reaccione con la más alta prioridad lo más rápidamente posible (es decir llega a estar en error pasivo o *bus off*). De esta forma, los errores pueden ser más fácilmente localizados y pueden llegar a minimizarse sus influencias sobre las actividades normales del bus (Cf. [Unruh et al., 1989]).

Para entrar en el estándar todos los microcontroladores conformes al protocolo CAN deben obligatoriamente poseer dos contadores internos bien distintos:

- el *transmit error counter*,
- el *receive error counter*,

Quienes tendrán por misión el de registrar (contar) los errores que se producen cuando suceden las transmisiones y recepciones (Cf. [Unruh et al., 1989]).

Veamos un poco más en detalle este mecanismo:

- si el mensaje es transmitido o recibido correctamente, el contenido del contador respectivo decrece,
- si el mensaje viene con errores, el contenido del contador respectivo aumenta,

Además:

- los contadores de errores no practican métodos proporcionales de conteo,
- un error provoca un aumento del contador considerado, en una cantidad mayor de lo que pudiera haber decrecido el mismo contador según si este hubiera recibido o emitido un mensaje correctamente.

Y además:

- sobre un largo periodo de tiempo esto puede llevar a un aumento de los contadores de errores incluso si hay menos de mensajes incorrectos que de mensajes correctos; en este caso, el nivel de los contadores de errores refleja la frecuencia relativa de los problemas que se han producido sobre el bus,
- la relación aumento/disminución de los contadores -dicho de otra forma la ponderación del conteo de los puntos- depende de la relación mensajes correctos/mensajes incorrectos sobre el bus; por definición, este protocolo fija su valor a 8 (Cf. [Unruh et al., 1989]).

2.6.4. Descontando los puntos.

Llegamos al final de esta parte sin embargo antes de tomar decisiones, es necesario contar los puntos.

No importa cuales sean las reglas exactas para descontar los puntos (los cuales se dan en la tabla de la Figura 2.24), se puede enunciar rápidamente las consecuencias de los totales obtenidos (Cf. [Unruh et al., 1989]).

Una representación gráfica de los resultados y consecuencias es dada en las Figuras 2.25 y 2.26.

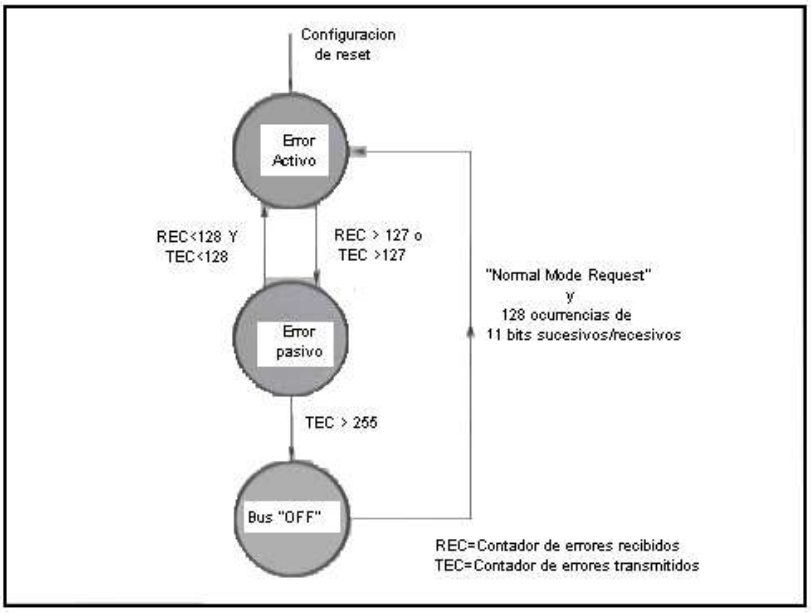

Figura 2.24: Descontando puntos

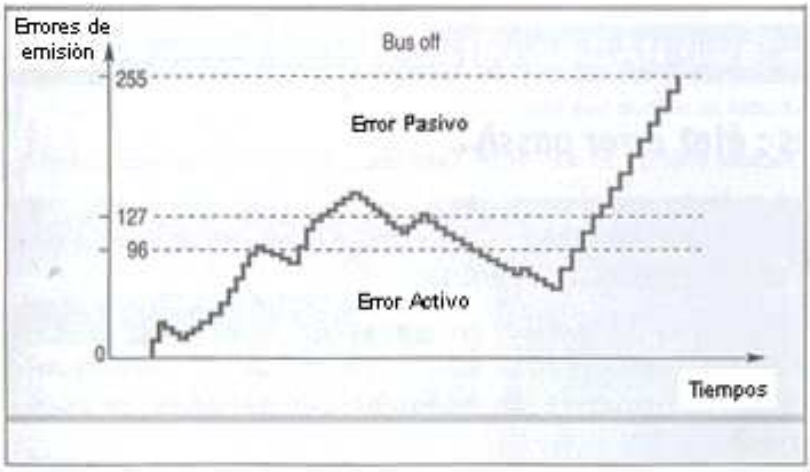

Figura 2.25: Representación de los estados

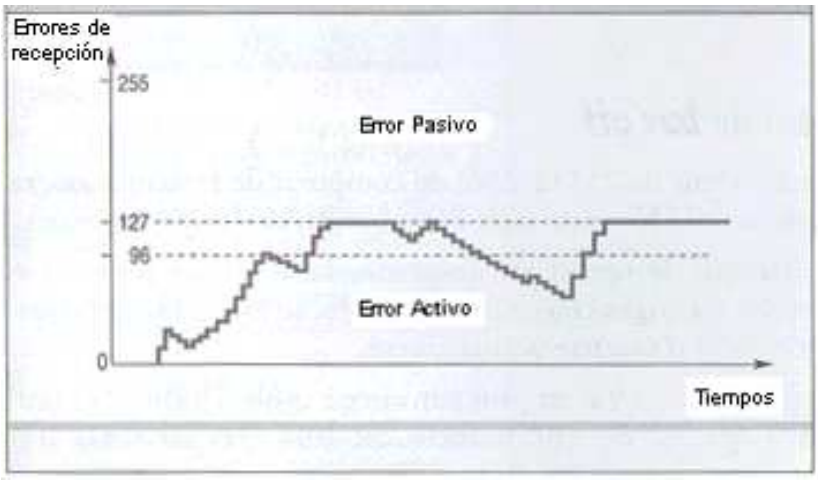

Figura 2.26: Representación de los estados

2.6.5. De 0 a 127 incluido: estado de error activo.

Cuando los valores de uno y otro de los contadores de errores están comprendidos entre 0 y el 127 incluido, se dice que la estación (el nodo) CAN funciona en estado (modo) error activo.

Esto significa que el nodo en cuestión continuará recibiendo y emitiendo normalmente pero que también en el caso en donde un error sea detectado, transmitirá un *active error flag* durante la trama de error (Cf. [Unruh et al., 1989]).

Nota: Es recomendado en el protocolo de comenzar a preocuparse en el momento en donde el contador llega a 96 puntos (warning limit), indicando una combinación significativa de condiciones de error. Generalmente, en este nivel se produce una interrupción al interior del microcontrolador CAN que se llama de error status o de error interrupt que provoca la solicitud de gestión de estas perturbaciones (Cf. [Unruh et al., 1989]).

2.6.6. De 128 al 255 incluido: estado de error pasivo.

Cuando los valores contenidos en uno u otro de los dos contadores están comprendidos entre 128 y 255 incluido, se dice el nodo CAN funciona en estado (modo) error pasivo *error passive*.

Esto significa que el nodo en cuestión continuará recibiendo y emitiendo normalmente pero que sin embargo, también que en el caso de detección de un error, continuará transmitiendo únicamente los *passive error flag* durante la trama de error [6] (Cf.

[6] Una condición de error que deja un nodo que llegue a estar en error pasivo obliga al nodo a enviar

[Unruh et al., 1989]) .

2.6.7. Más alla de 255: estado de bus off.

Más allá del valor de 255 (>= 256) del contador de transmisión, se dice que el nodo CAN entra en el estado de *bus off* (Cf. [Unruh et al., 1989]).

Eso significa que el nodo en cuestión, cansado de no poder transmitir o recibir mensajes con éxito, empieza a callarse y cesa de recibir y emitir normalmente. En este caso la unidad *bus off* ya no está autorizada a tener influencia o actividad sobre el bus. Estas etapas de control del bus (los drivers) deben de ser electrónicamente desconectados del bus.

Puede ser, sobre el principio, que en este estado de silencio indefinido si nadie se interesa en este nodo, el decida entrar nuevamente en la red teniendo en cuenta los errores precedentes, y volviendo a colocar sus contadores a 0 (Cf. [Unruh et al., 1989]).

El protocolo autoriza a un nodo bus off a regresar en error activo (habiendo puesto todos los contadores de errores a 0) después de que este no halla observado errores sobre el bus, 128 ocurrencias de 11 bits recesivos cada uno (demostrando así que muchos mensajes de los cuales los bits de *ACKnowledge delimiter + end of frame +* aquellos de la intermisión hallan pasado bien y que aparentemente el bus ha encontrado nuevamente su buen camino) (Cf. [Unruh et al., 1989]).

2.6.8. Consecuencias de los errores.

Localización de un nodo defectuoso.

En este párrafo, es importante notar la forma en la que los contadores de errores de transmisión y recepción presentes en un nodo toman los puntos localmente según sea que el nodo receptor (mensaje erróneo en sus propios conectores) o emisor (error de su parte o imposibilidad de llegar a emitir correctamente tal y como se espera, debido por ejemplo a problemas de la red) y por lo tanto de volver a cuestionar su lugar en la red (Cf. [Unruh et al., 1989]).

Las Figuras 2.27 y 2.28 dan un ejemplo preciso de lo que acabamos de enunciar, mostrando claramente como, y cuales son los estados de emisión o recepción, un *nodo de error* aumenta siempre más sus puntos que los otros nodos; de aquí que hay una gran facilidad de ubicación por sus ausencias sucesivas (en el orden de su tránsito en

un active error flag. Esto sucede en el caso cuando el nodo que estaba en error activo (error active) con un contador a 125, por ejemplo, y, sobre un error de 8 unidades de conteo de más, pasa a 133 en error pasivo.

modo error pasivo después desapariciones sucesivas por los tránsitos a modo *bus off*) sobre una red y, en consecuencia, tomar decisiones *ad hoc* (Cf. [Unruh et al., 1989]).

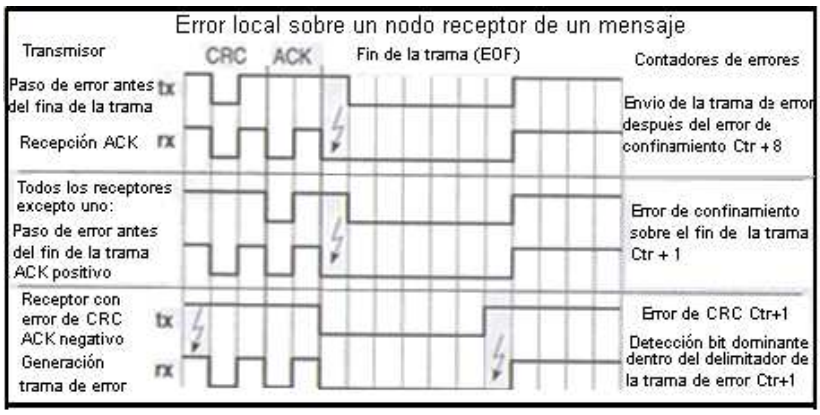

Figura 2.27: Errores de confinamiento, CRC y reconocimiento

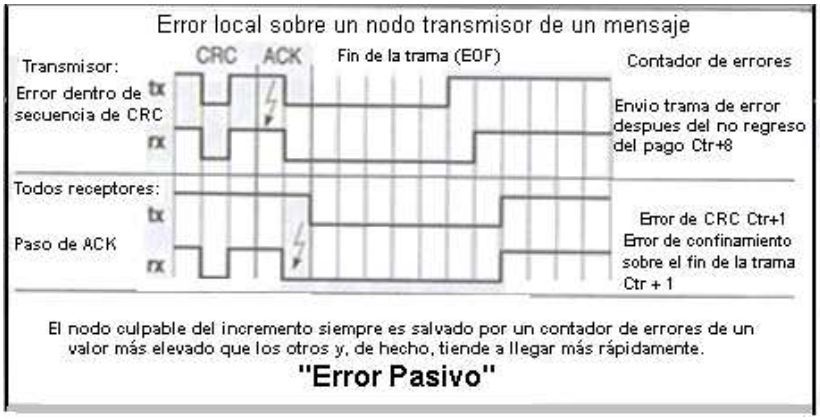

Figura 2.28: Errores de confinamiento, CRC y reconocimiento

Estadísticas que tienen que ver con la salud de la red.

Es evidente, que en todo instante, el conocimiento del estado de los contadores internos documenta e informan la salud de la estación y del estado de la red inmediata a esta estación. Además, como cada estación debe informar a la red de las catástrofes locales, existe la posibilidad para cada estación, si ella dispone del contenido preciso de esos contadores, de efectuar sus propias estadísticas de calidad de la red (Cf. [Unruh et al., 1989]).

En nuestros días, así como se presenta, el protocolo CAN no impone el poner a disposición del administrador/controlador CAN las informaciones tan finas como la que acabamos de enunciar. La única forma de tener una idea de los valores internos de los contadores consiste en observar sobre el bus los diferentes error flags (banderas de error) que están ligadas a través de ciertas relaciones a estos valores pero, teniendo cuidado ya que la decodificación de estas informaciones no es para nada simple (Cf. [Unruh et al., 1989]).

Evidentemente, tener disponibles de forma externa, sobre los bordes del componente, las informaciones de interrupción que corresponden a los saltos de los valores de 96, 128, 255 sería extraordinario (Cf. [Unruh et al., 1989]).

¿Y por que no mejor todavía? Disponer de dos registros pequeños internos en donde se pueda leer cuando uno lo desee los contenidos de los contadores de errores de transmisión y de recepción a fin de satisfacer todos los deseos de estadística sería realmente fabuloso.

Para esto, en las siguientes generaciones de los componentes CAN vendrán precisamente todas estas modificaciones.

Ahora que el problema está bien enfocado en lo que tiene que ver con los errores, sus implicaciones y su administración, veamos como se efectúan las detecciones, señalización y tratamiento de recubrimiento.

2.6.9. Detección de errores.

Después de haber establecido este tema sobre la gestión de los errores que va a ser aplicado, pasemos al estudio de los mecanismos de detección de estos.

Existen 5 tipos de errores diferentes, que no se excluyen mutuamente.

Errores de la capa física (errores de bit y de bit stuffing).

Bit con un error - bit error.

Un emisor CAN verifica en todo instante si el nivel de bit que desea emitir sobre el bus corresponde bien al que desea enviar. Si esto no corresponde, lo señala por un *bit error*, con las excepciones siguientes:

Cuando sucede el envío de un bit recesivo durante la flotilla de bits llenados del campo de arbitraje o durante el ACK slot (Cf. [Unruh et al., 1989]).

De hecho, durante estas fases de tiempo, el bit recesivo puede ser aplastado por un bit dominante sin que esto sea un error; en este caso, el microcontrolador CAN interpreta habitualmente esta situación, no como un error de bit si no: bien sea como una perdida de arbitraje en un primer caso, o bien como una respuesta posible de reconocimiento; por lo tanto, durante el *ACKnowledge slot*, solamente los microcontroladores CAN receptores (y no los emisores) son capaces de reconocer un bit de error (Cf. [Unruh et al., 1989]).

Un emisor que envía un *error flag* pasivo y que detecta un bit dominante. No lo debe interpretar como un bit de error.

Error de bit stuffing - stuff error.

En el campo del inicio de trama, de arbitraje, de control de datos y de CRC, la presencia de *bit stuffing* existe.

Hay dos maneras de observar un error de *bit stuffing*:

- Una perturbación genera más de 5 bits consecutivos de la misma polaridad (esta perturbación puede ser voluntaria para crear un error de *bit stuffing* a fin de señalar a todos los participantes una anomalía),
- Una perturbación falsifica uno o más de 5 bits precedentes del bit de *stuff*: en este caso habría que, o se debería sin embargo no hay reconocimiento de error de bit stuffing por los receptores (Cf. [Unruh et al., 1989]).

¿La razón? otras detecciones de errores tales que las verificaciones del CRC, violación del formato en lo que tiene que ver con los receptores CAN, se encargarán de señalarlo. Así estará incluso para los errores de bit de los emisores CAN. Por lo tanto, finalmente este tipo de error no pasará desapercibido.

2.6.10. Errores de reconocimiento y de CRC.

En el nivel de bit y de la estructura de la trama todo va bien, sin embargo existe un error de reconocimiento o de CRC.

Error de reconocimiento - ACKnowledgment error.
Este error es detectado por un emisor CAN cuando no observa sobre el bus un bit dominante durante el tiempo reservado al ACKnowledgment slot.
(Cf. [Unruh et al., 1989])

Error de CRC - CRC error.
El receptor calcula el valor de CRC de la misma forma en que lo ha efectuado el emisor (el resto de la división). Hay entonces un error si, después del cálculo y la

verificación, el valor del CRC no corresponde a lo que se esperaba (paridad incluida) (Cf. [Unruh et al., 1989]).

2.6.11. Otros errores.

Existe un error en el nivel de la capa de la trama en el sentido de que su estructura íntima ha sido violada (informaciones que no están en sus lugares).

Estos son los errores de estructura, ya sea:

- Errores de *CRC delimiter*,
- Errores de *ACKnowledgment delimiter*,
- Errores de *end of frame*,
- Errores de *error delimiter*,
- Errores de *overload delimiter*.

Durante la transmisión de estos campos, una condición de error será reconocida si un nivel dominante es detectado en el lugar de un nivel recesivo (Cf. [Unruh et al., 1989]).

En resumen.

El protocolo CAN utiliza los mecanismos de detección de errores siguientes:

- El *monitoring* del bus,
- El *cyclic redundancy check*,
- El *message frame check*,
- El *bit stuffing*,
- El reconocimiento,
- El *error signaling*,

Veamos cada uno de estos mecanismos de detección en el mismo orden.

Monitoreando el bus.
El emisor de bits compara la señal que el desea emitir con la señal que el observa físicamente sobre la línea física del bus.

En caso de no ser conforme (o en conformidad), el emisor emite entonces una trama de error si la señal sobre el bus es diferente a la señal emitida, con la excepción de la

fase de arbitraje (durante la transmisión del identificador del mensaje) y aquella del *ACKnowledgment slot*.

De esta forma los bits de errores que afectan todas las estaciones presentes sobre el bus no pueden conducir a errores que no sean détectables debido a que estos estarían siendo detectados por el emisor de la trama (Cf. [Unruh et al., 1989]).

Código de redundancia cíclica.

Los 15 bits del CRC son calculados a partir de todos los bits del SOF hasta el último bit de los datos. El código BCH utilizado para crear el CRC da una distancia de Hamming de 6 que tiene también en cuenta una verificación de paridad en la secuencia de bits llenados (stuffed) (Cf. [Unruh et al., 1989]).

Message frame check.

Los campos de SOF, RTR, IDE, DLC, los delimitadores y el campo de EOF deben ser consistentes con la especificación del CAN. Si uno de los campos del formato fijo de la trama recibida (excepto el último bit del EOF) no es conforme al estándar, el receptor envía una trama de error y no acepta la trama recibida (Cf. [Unruh et al., 1989]).

Bit stuffing.

La violación de la regla de bit stuffing entre el SOF y el CRC debe ser considerada como un error (Cf. [Unruh et al., 1989]).

Reconocimiento.

El emisor de la trama de datos o de la trama de solicitud trata una ausencia de reconocimiento como un error y destruye el campo de EOF emitiendo al mismo tiempo una trama de error (Cf. [Unruh et al., 1989]).

Error signaling.

Cada estación que detecta un error inicializa una trama de error de tal forma que las otras estaciones presentes sobre la red ven una violación de la regla de bit stuffing o del formato fijo de los delimitadores o de los campos de EOF. Observe que, una vez más, es en ese momento que todas las estaciones que ven un error voluntario de *bit stuffing* responderán enviando ellas también una trama de error (Cf. [Unruh et al., 1989]).

La Figura 2.29 resume los procesos de detección de errores según la situación de emisor o de receptor CAN.

Veamos ahora las fases de señalización y de información presentes sobre la red vía una trama de error que es dedicada y representativa de los eventos precedentes.

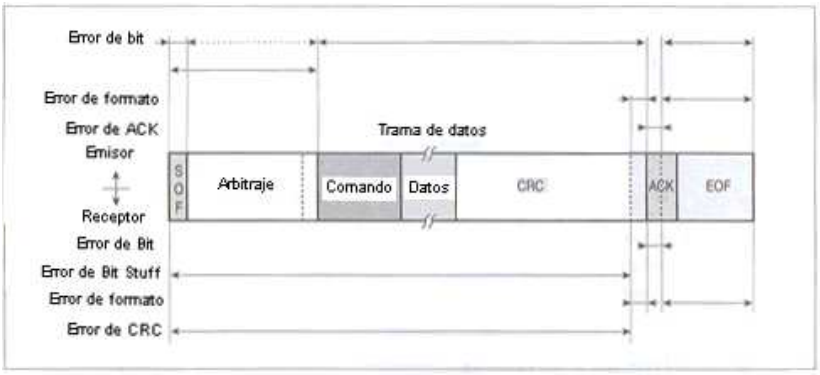

Figura 2.29: Tipos de errores

2.6.12. Señalización de los errores.

Grandes principios.

La detección de los errores que ha sido efectuada, una vez que se ha llevado a cabo esto es razonable de una forma general de rendir cuentas a la red:

- Para informaciones y acciones a los participantes de la red, pero también,
- Para indicar de forma simultánea a los otros participantes el estado local en la cual se encuentra en ese instante (teniendo en cuenta el estado de sus propios contadores de errores que ya han sido detectados) y en consecuencia para dar una cierta idea de la calidad local de la red en donde se está en ese momento generando (Cf. [Unruh et al., 1989]).

Para esto, un nodo que detecta una condición de error tiene por misión la de señalarlo transmitiendo un *error flag* diferente según los estados instantáneos de sus propios contadores de errores:

- Para un nodo en estado *error passive* (error pasivo): un *passive error flag* (bandera de error pasivo),
- Para un nodo en estado *error active* (error activo): un *active error flag* (bandera de error activo) (Cf. [Unruh et al., 1989]).

Antes de explicar cual es el contenido de las tramas de los errores, se da desde ahora el mecanismo particular que rige el instante preciso de su inicialización.

2.6.13. Inicialización de las tramas de error.

Notas importantes.

El disparo de las tramas de error tiene lugar en momentos diferentes según los orígenes de los errores.

Si estamos en presencia de:

- De un bit de error,
- De un error de *bit stuffing*,
- De un error de estructura,
- De un error de reconocimiento,
- La transmisión de un *error flag* se inicia desde el bit siguiente a partir de la estación que ha detectado el error.

Todas las veces que una estación detecta un error de CRC, la transmisión de un error flag inicia en el bit siguiente al *ACKnowledgment delimiter*, a menos que un error flag halla ya iniciado en otra parte sobre la red por alguna otra condición (Cf. [Unruh et al., 1989]).

2.6.14. Error frame (trama de errores).

Veamos la constitución de la trama de error (ver Figura 2.30).

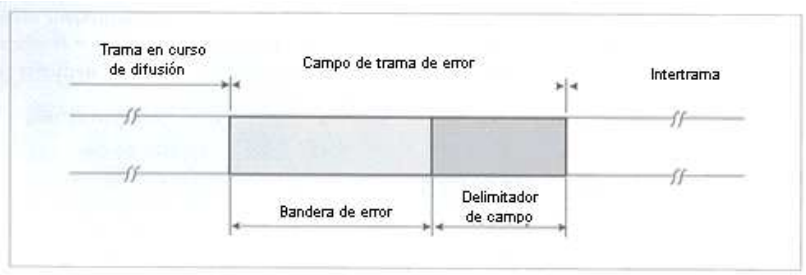

Figura 2.30: Error frame

Esta trama no tiene más que dos campos:

- Un campo de banderas (flags) de errores,
- Un delimitador de campo,

- Y después viene la zona que se hace llamar de espacio interframe.

Campo de las banderas de error.
Este primer campo está constituido por la superposición de las banderas de error las cuales han contribuido a las diferentes estaciones presentes sobre le bus.

Hay dos casos de banderas de errores[7] :

- Los *active error flag* (banderas de errores activos),
- Los *passive error flag* (banderas de errores pasivos).

Banderas de error activo.
Una estación *error active* que detecta una condición de error la señala transmitiendo un *active error flag*. Por definición, una *active error flag* está constituido de 6 bits dominantes consecutivos (ver la Figura 2.31.) (Cf. [Unruh et al., 1989]).

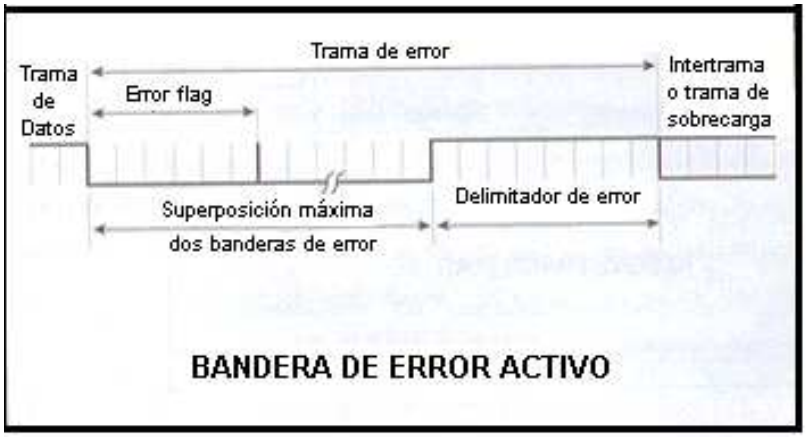

Figura 2.31: Trama de errores

Por su constitución misma, los *active error flags* transgreden la ley del *bit stuffing* aplicado desde el *start of frame* hasta el *CRC delimiter* dado que los 6 bits consecutivos son idénticos o bien, destruyen el formato fijo y definido del campo de ACK así como aquel del campo *end of frame* de los cuales los bits no están más que normalmente en estado recesivos (Cf. [Unruh et al., 1989]).

En consecuencia, todas las otras estaciones presentes sobre la red detectan también una condición de error y, de su lado, inician también la transmisión de un nuevo error

[7] Para los términos que definen con precisión a lo que se le llama estaciones de error activo y error pasivo, ver los párrafos precedentes.

flag (activo o pasivo según sus propios estados). Es decir esto llega a ser un sabotaje en cadena (Cf. [Unruh et al., 1989]).

La secuencia de bits dominantes que podemos observar entonces sobre el bus resulta de la superposición de la participación de los diferentes *error flag* provistos por todas las estaciones individuales (y que cada una de entre ellas puede examinar -monitorear- en tiempo real) (Cf. [Unruh et al., 1989]).

El documento CAN autoriza una longitud total de esta secuencia que puede variar de entre un mínimo de 6 bits (la longitud del active error flag) y un máximo limitado a 12 bits a fin de no bloquear indefinidamente el bus (Cf. [Unruh et al., 1989]).

Banderas de error pasivo.
Una estación que se dice de *error passive* que detecta una condición de error intenta señalarla a través de la transmisión de un passive error flag.

Por definición un *passive error flag* está constituido de 6 bits recesivos sucesivos a menos que esos bits no sean aplastados por los bits dominantes que provienen de otros nodos (ver la Figura 2.32) (Cf. [Unruh et al., 1989]).

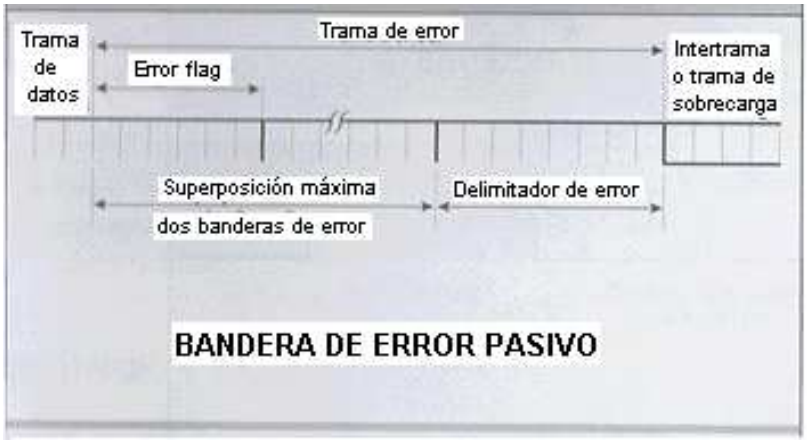

Figura 2.32: Trama de errores

Por principio de la emisión de los bits recesivos, un passive error flag no es capaz de interrumpir un mensaje en curso entre diferentes otros controladores presentes sobre el bus, pero este tipo de error flag puede ser ignorado (aplastado por los otros controladores) (Cf. [Unruh et al., 1989]).

Después de haber detectado una condición de error, un controlador en estado del

modo error passive espera durante 6 bits consecutivos de polaridad idéntica y cuando los encuentra, los interpreta como un error flag (Cf. [Unruh et al., 1989]).

La estación *error passive* espera 6 bits consecutivos de la misma polaridad que comienza al inicio del *passive error flag*.

El *passive error flag* es terminado cuando estos 6 bits idénticos han sido detectados (Cf. [Unruh et al., 1989]).

Delimitador de error.

El error delimiter está constituido de 8 bits recesivos.

Después de la transmisión de un error flag cada estación envía los bits recesivos y monitorea el bus hasta que es detectada una transición de un nivel dominante a un recesivo (Cf. [Unruh et al., 1989]).

En este instante, cada controlador CAN ha terminado de transmitir su error flag y de forma adicional emitido el primero de los bits del error delimiter. Todos los controladores CAN pueden entonces iniciar la secuencia de los 7 bits recesivos que quedan para terminar de construir los 8 bits del error delimiter (Cf. [Unruh et al., 1989]).

Después de este evento y de un campo de intermisión, todos los controladores *error active* presentes sobre la red pueden iniciar una transmisión simultáneamente.

Si un error es detectado y señalado durante la transmisión de una trama de datos o de una trama de solicitud, el mensaje en curso es puesto en una cola y una retransmisión del mensaje es inicializado (Cf. [Unruh et al., 1989]).

Si un controlador CAN se da cuenta de algunas desviaciones de un error de trama, una nueva trama de error es transmitida. Muchas tramas de errores consecutivas pueden llegar a provocar que el controlador CAN tenga un estado de *error passive* y de dejar la red no bloqueada (Cf. [Unruh et al., 1989]).

Nota. La forma correcta de terminar un *error frame*, es cuando una estación de *error passive* puede tener la necesidad del bus en modo *bus idle* durante un tiempo al menos equivalente a 3 bits (si al menos hay un error local en una estación receptora en modo *error passive*). De este hecho el bus no debería ser cargado al 100 % de forma permanente.

Después de todos estos largos discursos la Figura 2.33 nos da los ejemplos concretos de lo que se puede producir según las diferentes hipótesis.

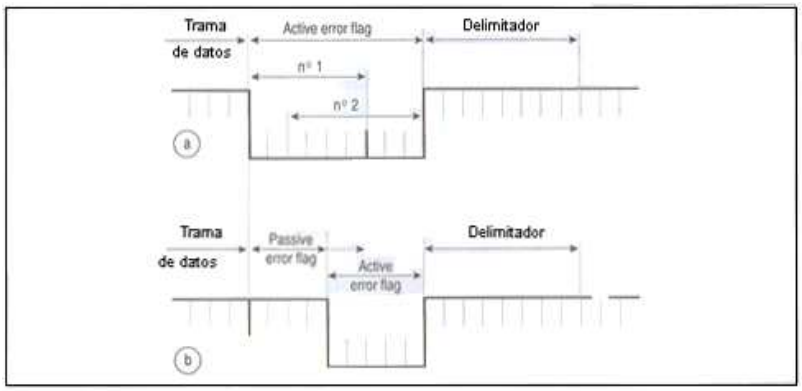

Figura 2.33: Ejemplos

Recubrimiento de errores.

El recubrimiento de errores se efectúa por la retransmisión automática de la trama perturbada hasta el momento donde ella llega a pasar correctamente y que no hay más ya el mensaje de error (Cf. [Unruh et al., 1989]).

Esto puede durar eternamente, bueno casi, si por casualidad los errores fueran de forma perseverantes, pero tarde o temprano, los contadores de errores aumentarían, el circuito que produce el error (o que se da cuenta) pasaría el primero en bus off. Después de haber acabado con este candidato, no debería haber más errores y, el mensaje entonces debería pasar de forma adecuada. En el caso donde los problemas perseveran: se realiza la misma situación con otro candidato (Cf. [Unruh et al., 1989]).

¿Que pasa si todos los candidatos (nodos) dan problemas? Pues bien todos desaparecerían uno a la vez y solamente quedaría un solo nodo a querer comunicarse sobre la red (bien que nadie contestara). Es decir nuevamente contaríamos los errores, etc. Y también el último nodo estaría en modo bus off de esta forma todos los nodos estarían hibernando. El bus se encuentra momentáneamente bloqueado debido a que ninguna estación no puede ver pasar las famosas 128 ocurrencias de 11 bits recesivos necesarios para despertarse (Cf. [Unruh et al., 1989]).

Es este el *time out* y su inseparable gestión los que deben intervenir para asegurar el despertador: es entonces cuando se da un *power on reset* del circuito integrado. Es un poco brutal pero eficaz (Cf. [Unruh et al., 1989]).

Validez de los mensajes.

El instante preciso donde el mensaje es considerado como válido es diferente según sea la forma en la que esté, emisor o receptor del mensaje (Cf. [Unruh et al., 1989]).

Para el emisor del mensaje.

Para el emisor del mensaje, el mensaje es considerado como válido si no hay un error detectado hasta el fin del *end of frame*. Si el mensaje ha sido corrompido, una retransmisión seguirá automáticamente y se efectuará de acuerdo con la regla de prioridades de los mensajes (Cf. [Unruh et al., 1989]).

La retransmisión debe reiniciarse tan pronto como el bus esté en modo idle (neutro), de forma en la que sea capaz de competir por el acceso al bus con los otros mensajes.

Para los receptores del mensaje.

Para los receptores del mensaje, el mensaje es considerado como válido si no hay un error detectado hasta antes del último bit de el end of frame (Cf. [Unruh et al., 1989]).

En todos los otros casos los mensajes son considerados como no válidos.

2.7. Desempeño de los tratamientos de errores.

No esta fuera de nuestro propósito de interesarnos por los desempeños del tratamiento de los errores en un protocolo. De hecho, esta parte no se incluye en el protocolo mismo, sin embargo ella subyace dentro del protocolo debido a que participa a definir, por ejemplo, el tipo de CRC y el número de bits que lo constituyen. Además, esta rúbrica anima las numerosas controversias entre los diferentes protocolos debido a que ellas tocan en el funcionamiento real de las redes quienes, muy a menudo, no importa lo que pueda parecer, no son estrictamente idénticos (Cf. [Unruh et al., 1989]).

Para ver en detalle este tópico, los grupos de investigadores (normalmente independientes), de las grandes escuelas o universidades desarrollan las teorías para poder calcular, justificar, destruir de una forma consiente las cualidades y virtudes de tales o cuales artificios de los protocolos (Cf. [Unruh et al., 1989]).

Es así que en las subsecciones siguientes citaremos los principales resultados conocidos y admitidos por la profesión.

2.7.1. Errores y desempeños.

Clases de errores.

Generalmente, los errores residuales están clasificados en grandes clases.

Errores que no perturban la longitud de la trama.
Ver la Figura 2.34.

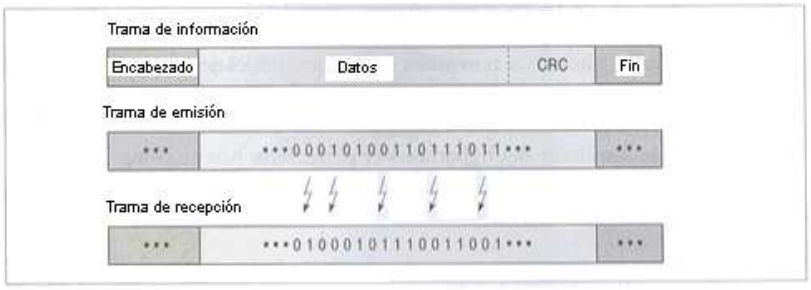

Figura 2.34: Clases de errores

En esta clase, todos los campos de la trama transmitida son interpretados como válidos por la estación receptora. Los campos afectados por los errores de bits no pueden ser aquellos que pertenecen a IDE, DLC y DATA (Cf. [Unruh et al., 1989]).

Errores que participan a la modificación de la longitud de la trama.

Ver la Figura 2.35

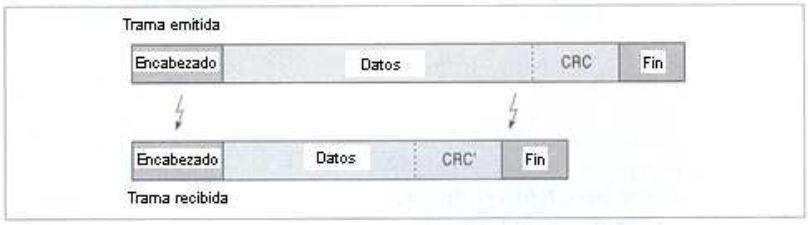

Figura 2.35: Clases de errores

Si los campos SOF, RTR, IDE o DLC son modificados por los errores de bit, el receptor espera una trama de longitud diferente a la trama originalmente emitida e interpreta la trama incidente de forma diferente que aquella que originalmente estaba prevista.

En función de los cambios producidos por los errores, el receptor puede esperar una trama de longitud más larga o más corta. Sin embargo, la probabilidad de error residual es mucho más débil para los aumentos que para las reducciones de longitud de las tramas debido a estos errores (Cf. [Unruh et al., 1989]).

2.7.2. Transformación de error.

Otros tipos de errores de bits pueden producirse. Citemos por ejemplo el hecho que los parásitos que afectan el contenido de los datos sin derogar a las reglas de *bit stuffing*, o también que un par de bits con error suprima la presencia de un *bit de stuff* (Cf. [Unruh et al., 1989]).

2.7.3. Análisis de las causas de errores.

Una vez clasificados y repertoriados todos los casos de errores probables, llega a ser entonces posible el poder pasar a un análisis cuantificable. Los cálculos que tienen que ver con este análisis llegan a sobrepasar largamente el marco de trabajo de esta monografía. Para poder entrar más en detalle, o si usted quiere consultar una comunicación mucho más especializada sobre este tópico, favor de consultar las referencias bibliográficas al final de esta monografía (Cf. [Unruh et al., 1989]).

En resumen, el análisis se compone de:

- Sumario de las hipótesis y los parámetros que se toman en cuenta,
- la modelación del canal de transporte,
- la elaboración de la formula matemática de probabilidad del error residual,
- los resultados:
  - De la probabilidad del error residual normalizado,
  - De la contribución de cada una de las clases de error,
  - De la influencia de la longitud del campo de datos,
  - De la influencia del formato de la trama y del número de estaciones presentes sobre la red (Cf. [Unruh et al., 1989]).

La Figura 2.36 resume los resultados obtenidos.

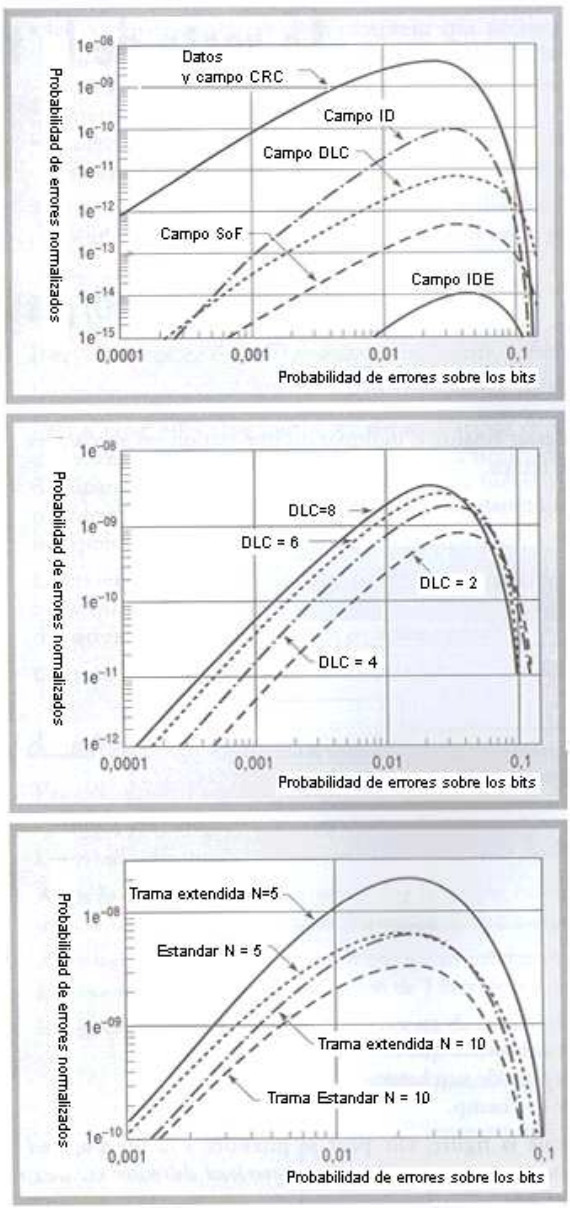

Figura 2.36: Obteción de resultados

2.8. El resto de la trama.

2.8.1. Overload frame(trama de sobrecarga).

Esta trama tiene por finalidad la de indicar que una estación está sobrecargada durante un cierto lapso de tiempo (Cf. [Unruh et al., 1989]).

Veamos ahora su constitución (ver la Figura 2.37)

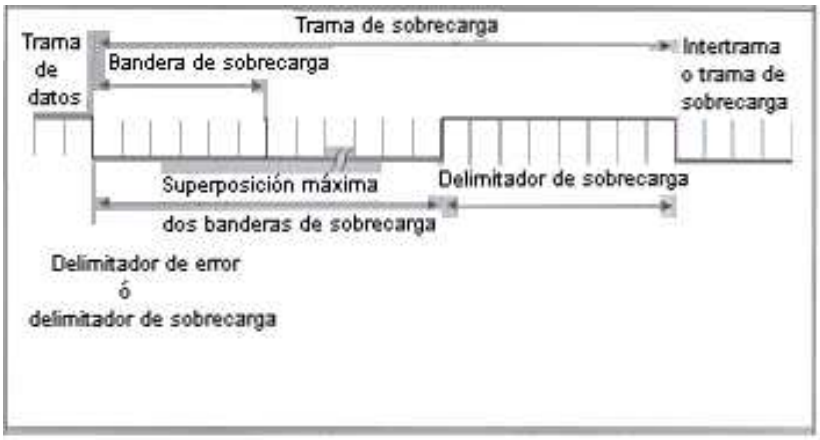

Figura 2.37: Trama de sobrecarga

Esta trama no contiene más que dos campos:

- El campo de los flags (banderas de sobrecarga),
- El delimitador de campo.

Como lo indica la figura, ella puede producirse al final de un *end of frame* o de un *error delimiter* o también de algún otro como *overload delimiter* en el lugar del principio de la intertrama.

Viene enseguida la zona que se llama de espacio intertrama o también una trama de sobrecarga. Hay dos tipos de condiciones de carga que llevan a la transmisión de una bandera de sobrecarga (Cf. [Unruh et al., 1989]):

- Las condiciones internas de un receptor que necesita un cierto tiempo (retardo) para aceptar la próxima trama de datos o trama de solicitud. En ese caso, el inicio de una trama de sobrecarga es solamente autorizado al primer bit time de la intermisión que tendrá lugar.
- La detección de un bit dominante durante la fase de intermisión. En este caso el inicio de la trama de sobrecarga tiene lugar justamente después de la detección del bit dominante (Cf. [Unruh et al., 1989]).

A fin de no bloquear el bus de forma indefinida solamente las dos tramas de sobrecarga consecutivas pueden ser generadas para retardar las tramas de datos o de solicitud siguientes (Cf. [Unruh et al., 1989]).

Overload flag(OLF).

Se constituye de 6 bits dominantes consecutivos, como la bandera de error activo (Cf. [Unruh et al., 1989]).

La estructura del OLF destruye la estructura definida del campo de intermisión.

Esto tiene por efecto que todas las demás estaciones que detectan también una condición de sobrecarga e inicializan también una transmisión de OLF, en el caso donde un bit dominante sería detectado durante el tercer bit de la intermisión localmente de un nodo, los otros nodos no interpretaran el OLF correctamente si no que interpretarán los primeros de estos 6 bits dominantes como un inicio de trama (Cf. [Unruh et al., 1989]).

El sexto bit evita aquí también la sacrosanta ley del *bit stuffing* y en consecuencia todas las demás estaciones detectan una condición de error e inicializan la transmisión de una bandera de error. Nuevamente volvemos a encontrar el sabotaje en cadena (Cf. [Unruh et al., 1989]).

Overload delimiter (Delimitador de sobrecarga).

Por definición el delimitador de sobrecarga esta constituido de 8 bits recesivos consecutivos, como el error delimiter. El delimitador de sobrecarga tiene la misma estructura que el delimitador de error (Cf. [Unruh et al., 1989]).

Después de la transmisión de una bandera de sobrecarga la estación examina el bus hasta que ella detecta una transición que señala el paso de un bit dominante a un bit recesivo (Cf. [Unruh et al., 1989]).

Es en este instante que, cada estación sobre el bus termina de enviar su bandera de sobrecarga y todas las demás estaciones inicializan una transmisión de 7 bits recesivos (Cf. [Unruh et al., 1989]).
La Figura 2.38 da un ejemplo concreto de lo que se puede producir.

Periodo de intertrama (interframe).

Las tramas de datos y las tramas de solicitud están separadas de las tramas precedentes (de cualquiera de los tipos que ellas sean: datos, solicitud, de error, de sobrecarga) por un campo de bits llamado espacio de intertrama (Cf. [Unruh et al., 1989]).

Figura 2.38: Ejemplo

De forma contraria, las tramas de sobrecarga y las tramas de error no son precedidas por un espacio de intertrama y los múltiples overloads frames no están separados por un espacio de intertrama (revisar las figuras que ya han sido presentadas y observar en detalle estas fases de funcionamiento del bus) (Cf. [Unruh et al., 1989]).

El espacio de intertrama se compone de dos o tres campos según sea el caso (ver la Figura 2.39)

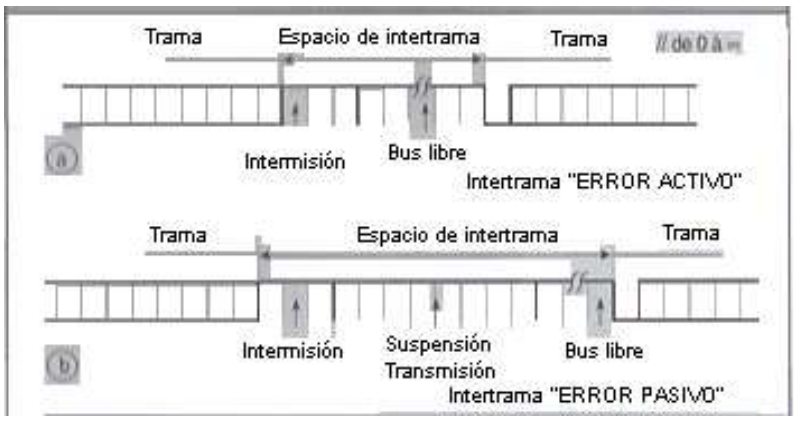

Figura 2.39: Espacio intertrama

Estos son:

- El campo de bits intermisión,
- El campo de bits de *bus idle*,

y, para las estaciones de error pasivo que han sido transmisores del precedente mensaje, de una campo de bits de suspensión de transmisión (Cf. [Unruh et al., 1989]).

Campo de intermisión.
Este campo está constituido de tres bits recesivos.

Durante la intermisión, ninguna estación está autorizada a iniciar una transmisión de la trama de datos o de la trama de solicitud. La sola acción que puede ser llevada a cabo es de poder señalar una condición de sobrecarga (Cf. [Unruh et al., 1989]).

Campo de *bus idle.*
La duración durante la cual el bus es idle (en reposo, en espera) puede ser arbitrariamente elegida. Durante ese lapso de tiempo el bus está libre y no importa cual estación que halla tenido alguna cosa que transmitir puede tener acceso al bus (Cf. [Unruh et al., 1989]).

Un mensaje que estaba en espera de transmisión durante la transmisión precedente puede entonces inicializar desde el primer bit siguiente a la intermisión. Para la detección de un bit dominante sobre el bus durante la intermisión es interpretada como el inicio de la trama de una nueva transmisión (Cf. [Unruh et al., 1989]).

Campo de transmisión suspendida (Suspend Transmission).

Después de que una estación de error pasivo halla transmitido un mensaje, ella envía 8 bits recesivos siguientes al campo de intermisión antes de inicializar la transmisión de otro mensaje o de reconocer que el bus está libre.

Si durante ese tiempo, una transmisión (debido a otra estación) inicia, la estación entonces llega a ser receptor de este mensaje (Cf. [Unruh et al., 1989]).

2.8.2. Modo de hibernación y despertador (sleep mode y wake up).

A fin de reducir el consumo de energía, un elemento que participa en la red CAN puede meterse de modo hibernación (sleep mode), es decir no tener mas actividad interna habiéndose desconectado de sus drivers del bus de la línea (Cf. [Unruh et al., 1989]).

El *sleep mode* es terminado por un despertador (wake up) provocado por alguna actividad sobre el bus o por las condiciones internas propias al sistema local. En el despertador, la actividad interna del circuito es reiniciada (no importa que la capa de transferencia espere que el oscilador del sistema se haya estabilizado) y el circuito esperará entonces a ser sincronizado el mismo sobre la actividad del bus (verificando los 11 bits recesivos) antes que los bus drivers sean posicionados de nuevo en *on the bus* (Cf. [Unruh et al., 1989]).

Esto quiere decir que es posible que se pierda uno o algunos mensajes, cuando sucede la fase de despertador de un participante, sabiendo, que estos no serán reconocidos, será necesario de reemitir los mensajes perdidos. A fin de subsanar este problema y de

despertar los nodos que han hibernado en un sistema, un mensaje especial de despertador ha sido reservado. Este mensaje contiene el identificador de la más baja prioridad es decir: rrr rrrd rrrr en la cual r es recesivo y d dominante (Cf. [Unruh et al., 1989]).

2.8.3. Inicio y despertador (start up y wake up).

Una última nota que tiene que ver con el párrafo precedente y con el wake up, no en funcionamiento normal, si no más bien a la puesta en marcha por la primera vez (start up).

Si durante la fase de inicio del sistema, solamente un nodo está activo (on line) y este nodo transmite algunos mensajes, no habrá ningún reconocimiento. De este modo, se detectará un error y se repetirán los mensajes. En ese caso, puede llegar a pasar un error passive sin embargo no aún *bus off* (Cf. [Unruh et al., 1989]).

Con esto terminamos precisamente en análisis de todos los errores posibles y los propios intercambios que pueden darse en una red CAN. Lo que hay que asegurar en todos los casos es que para los analistas de las redes existe y facilita también todos los trabajos de análisis y diseño para poder determinar el contenido de un intercambio (Cf. [Unruh et al., 1989]).

2.9. CAN 2.0B.

Introducción.

Después de haber lanzado en el mercado el primer concepto de protocolo CAN (en la versión 1.2), fue evidente que el campo de identificadores de 11 bits de la trama estándar podia presentar algunos problemas en ciertos casos de aplicaciones bien específicas y que, para dar un mayor confort de utilización a un sistema, era importante concebir una trama extendida que tenía en cuenta un campo de identificadores más grande (29 bits). Para esto, hacia falta modificar el formato de las tramas de datos y el de la solicitud de datos. Lo que fue hecho, asegurando una compatibilidad ascendente y rebautizando la primera trama estándar 1.2 en que 2.0A y la otra llamada extendida de 2.0B.

2.9.1. CAN 2.0A y 2.0B.

En los párrafos siguientes presentaremos solamente las diferencias que existen entre el protocolo CAN 2.0A y 2.0B.

CAN 2.0B e ISO.

La descripción del protocolo CAN 2.0B esta copiado del modelo de referencia ISO/OSI (Cf. [Robert BOSCH, 1991]).

2.9.2. Tramas.

Las nociones de las tramas de datos, de petición, de error, de sobrecarga, etc. son las mismas (Cf. [Robert BOSCH, 1991]).

Formato de las tramas.

Las principales diferencias entre los formatos de las tramas residen en el número de bits que constituye el campo del identificador y las significaciones de los bits de reserva (ver Figura 2.40) (Cf. [Robert BOSCH, 1991]).

Formato estándar -CAN 2.0A- identificador 11 bits. Formato extendido -CAN 2.0B- identificador 29 bits[8] .

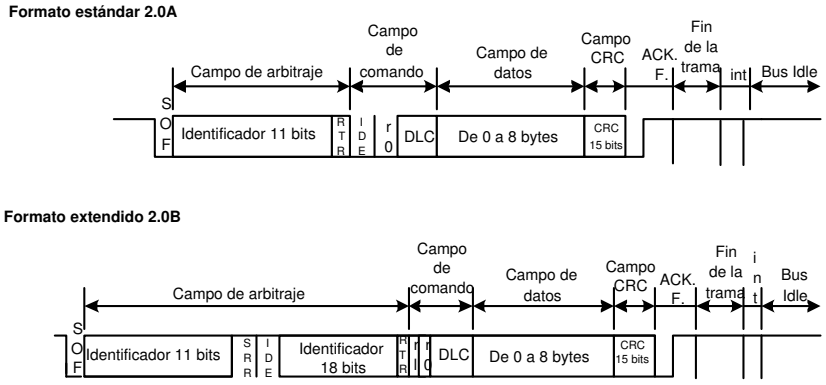

Figura 2.40: Formato de las tramas.

Identificadores y campo de arbitraje.
Los campos de arbitraje 2.0A y 2.0B son en principio diferentes (ver las Figuras 2.41 y 2.42).

Para el CAN 2.0A (Figura 2.41), este campo esta compuesto de 11 bits renombrados en la trama CAN 2.0B, ID_28 a ID_18 y del bit RTR (Bit del Remote Transmission Request).

[8] Observe que el formato extendido ha sido construido de forma tal que los dos formatos puedan coexistir simultáneamente sobre una misma red.

Esta parte alta del campo del identificador lleva el nombre de base ID y constituye la prioridad de base de la trama extendida (Cf. [Robert BOSCH, 1991]).

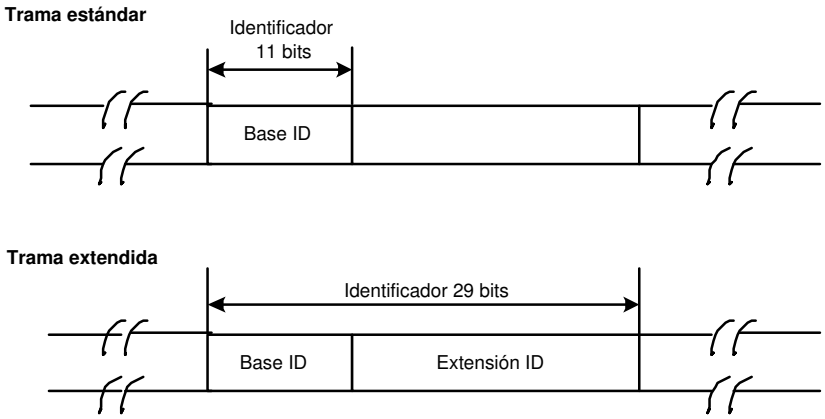

Figura 2.41: Identificadores y campo de arbitraje

Para el CAN 2.0B (Figura 2.42), este campo está compuesto de 29 bits renombrados de ID_29 a ID_00 y de 3 bits el antiguo RTR (el cual se movió de lugar) y de 2 nuevos bits SRR (Bit de Substitute Remote Request) e IDE (bit de Identifier Extensión) recuperados en los bits de reserva de la trama CAN 2.0A del campo de control los cuales estaban ahí para este caso desde hace largo tiempo (Cf. [Robert BOSCH, 1991]).

Notas: observe los lugares relativos y los nombres de los bits RTR, IDE y SRR respectivamente en la trama estándar 2.0A y la trama extendida 2.0B.

Observe también que en la nueva trama 2.0B la aparición de 2 nuevos bits de reserva (r0 y r1) todavía sin definir hasta nuestros días. Las Figuras 2.43 y 2.44 resumen sus funciones y acciones respectivas.

El campo de control.

A pesar de que todavía está constituido de 6 bits, después de las modificaciones de los nombres de los bits entre la trama 2.0A y 2.0B, el campo de control posee una nueva estructura (Cf. [Robert BOSCH, 1991]).

2.9.3. Obligaciones de un controlador CAN 2.0B.

Por definición, en la especificación oficial, un controlador que soporta el formato extendido 2.0B debe soportar sin restricción el formato estándar 2.0A.

Veamos a continuación la especificación original.

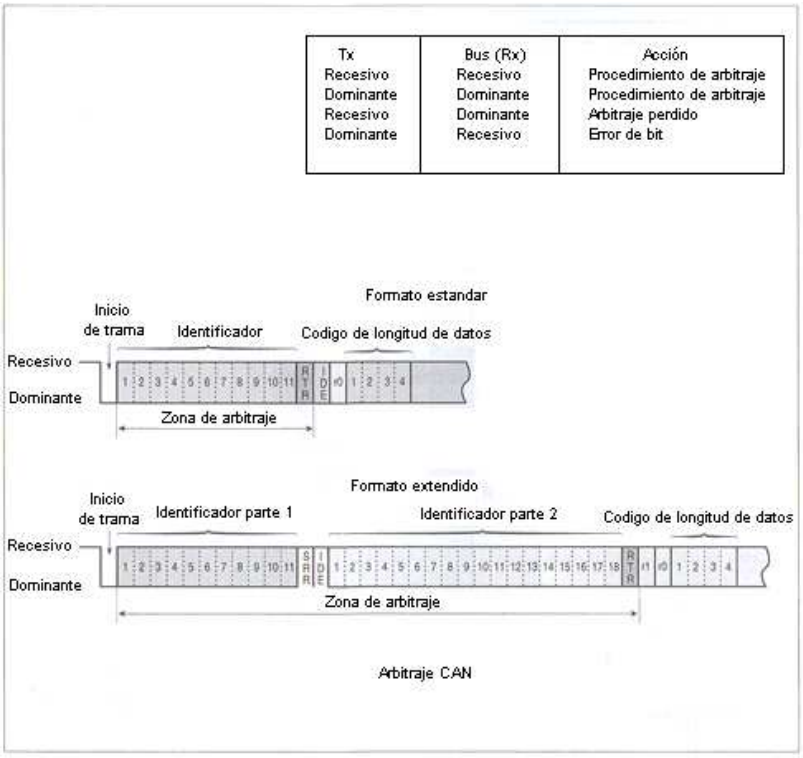

Figura 2.42: Identificadores y campo de arbitraje

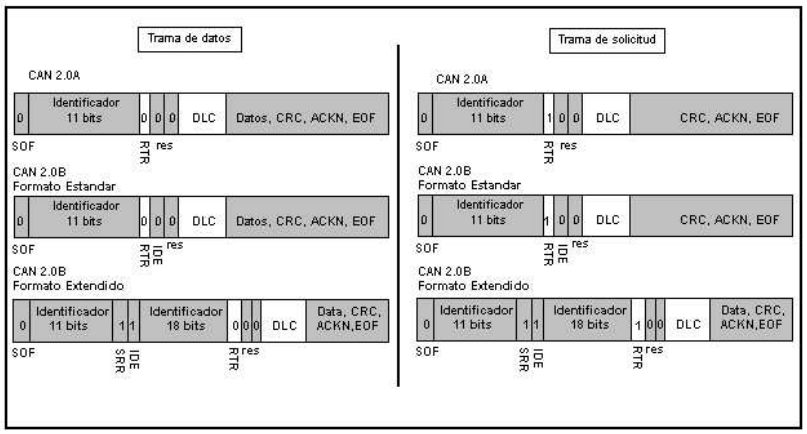

Figura 2.43: Bits de reserva

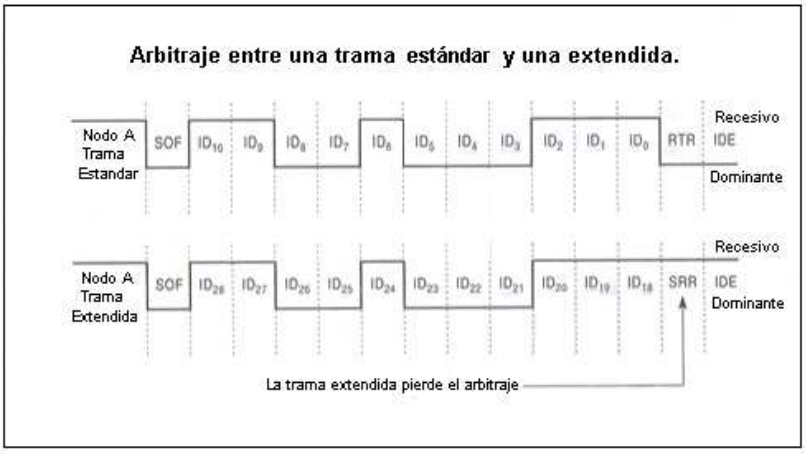

Figura 2.44: Bits de reserva

"Los controladores son considerados conformes con la especificación de la parte B del protocolo versión 2 si tienen al menos las propiedades siguientes:

- El controlador soporta el formato estándar,
- El controlador puede recibir mensajes en formato extendido. Con esto se sobrentiende que las tramas extendidas no pueden ser destruidas por el solo hecho de sus formatos. Sin embargo no es necesario que el formato extendido sea soportado por los nuevos controladores."(Cf. [Robert BOSCH, 1991])

A continuación se verán los problemas de coexistencia y compatibilidades de los circuitos CAN 2.0A y 2.0B sobre una misma red (Cf. [Robert BOSCH, 1991]).

2.9.4. Compatibilidades de CAN 2.0A y 2.0B.

Se presentará enseguida algunas notas que tienen que ver con las compatibilidades entre estos 2 formatos 2.0A y 2.0B del protocolo CAN para la realización de los componentes electrónicos.

En efecto, en razón de las aplicaciones específicas de la extensión de la red, es bastante frecuente que un usuario quiera hacer circular sobre una misma red las tramas estándar del tipo 2.0A y las tramas de formato extendido 2.0B (Cf. [Robert BOSCH, 1991]).

En este caso habrá sobre la red ciertos componentes que funcionan únicamente bajo el formato 2.0A y que, en cada envío de una trama extendida 2.0B, la estructura desconocida para estos (al nivel de los bits de servicio), generarán sistemáticamente una trama de error, dando como resultado errores catastróficos (Cf. [Robert BOSCH, 1991]).

Es necesario introducir ciertas nociones, como las que se llaman CAN 2.0A activo y 2.0B pasivo.

Se puede descomponer esto en algunas clases de componentes.

En el caso de componentes CAN 2.0A se puede encontrar por ejemplo:

- 2.0A activo solo.
- 2.0A activo y 2.0B pasivo (no hay una trama de error generada en caso de transmisión de una trama reconocida del tipo 2.0B) (Cf. [Robert BOSCH, 1991]).

Para las componentes CAN 2.0B, el protocolo impone que todas las tramas 2.0A y 2.0B sean reconocidas y tratadas automáticamente desde su recepción. Con respecto a la emisión, la elección del tipo 2.0A o 2.0B es a elección del usuario.
(Cf. [Robert BOSCH, 1991]).

Para los campos de aplicación, la Figura 2.45 resume la utilización en el mercado industrial en 1995.

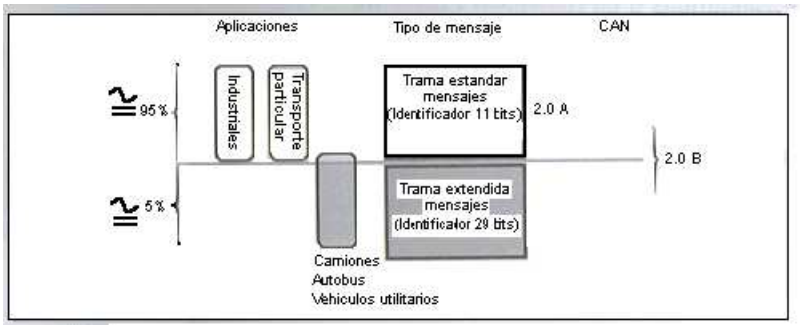

Figura 2.45: Campos de aplicación.

Capítulo 3

LA CAPA FÍSICA CAN.

Introducción

En los Capítulos precedentes, se han descrito los principios generales del protocolo CAN (principalmente la capa dos del ISO) y los componentes dedicados que permiten gestionarlo y concebir adecuadamente algunos sistemas.

No se ha hablado por el momento (al menos ha sido poco) de la capa física sobre la que tarde o temprano circularán las tramas CAN, y tampoco de sus numerosas aplicaciones.

En este Capítulo, su propósito principal es el describir concretamente las primeras aplicaciones en muchos capítulos que comprenden:

- Un poco de la teoría que tiene que ver con la capa física y los problemas que surgieron por los diferentes tipos de medios de desarrollo e interconexión (la capa 1 del ISO y la capa 0),
- los ejemplos de realización de CPU CAN con la ayuda de un microcontrolador estándar y de un gestionador de protocolo, después con la ayuda únicamente de un microcontrolador dedicado CAN.

Veamos entonces a detalle el contenido de este capítulo, el cual se ha subdividido en cuatro partes principales como son:

- la capa física CAN descrita en el documento de referencia y el de ISO,
- las propiedades del bit, es decir, las cualidades fundamentales y necesarias en el soporte físico utilizado,
- los tipos y estructuras de redes,
- los problemas de propagación de señal, de distancia y de caudal de información.

3.1. Capa física CAN.

Introducción.

En esta sección se supondrá que el lector esta bien informado del protocolo, de la gestión de las tramas y de todas las particularidades y singularidades para una rápida introducción a estos temas). A continuación se conocerá la estructura general del protocolo si bien hay todavía bastantes preguntas básicas (en el sentido de las capas del ISO) y concretas a plantearse y a resolver antes de poner definitivamente en función este bus en el hardware. Estas son algunas de las preguntas que seguramente el principiante en la estructuración de un protocolo sobre una implementación física realizaría (Cf. [Site, 1999]):

- ¿que estructura tiene realmente el bit?
- ¿en que momento se dice que es un 1 ó un 0?
- ¿cual es el mejor soporte físico para el transporte de protocolo?
- ¿los hilos ordinarios son suficientes para transportar la información?
- las fibras ópticas ¿también nos pueden servir?
- ¿cuál es el caudal óptimo de una aplicación dada?
- ¿hasta que distancia se puede esperar que esto funcione?
- ¿existe una o varias tomas normalizadas?
- ¿cómo controlar la línea?
- ¿cómo protegerla?.

Todas estas preguntas son clásicas, y tienen que ver con la capa física y con los problemas asociados de caudal, de elección de interfaz física apropiada, de toma estándar y como siempre el eterno debate flujo/distancia/medio (Cf. [Site, 1999]).

3.2. ISO y capa física CAN.

De forma frecuente, los usuarios de CAN esquematizan la capa física de forma muy simple y no tienen en cuenta más que el aspecto de la línea; una elección normal dado de que es la que se percibe en primer lugar. En este trabajo se propone no jugar el papel de un usuario estándar durante algunos instantes y de desmenuzar la línea CAN en detalle (Cf. [Site, 1999]).

Para esto, se introduce en una forma de presentación más académica, los documentos que el International Standardization Organization (ISO) editó y que permiten preparar la estandarización de las capas de comunicación de datos (Data Link Layer) y de las capas físicas (Physical Layer) para las aplicaciones precisas (vehículos automotrices) que pueden ser fácilmente aplicadas o transpuestas a otros dominios. Estos documentos describen la arquitectura CAN en términos de las capas jerarquizadas de acuerdo con las definiciones del modelo de la referencia de base para los Open System Interconnection (OSI) y propone las especificaciones que tienen que ver con las capas de base siguientes [1] (Cf. [Site, 1999]):

- Capa de enlace de datos (Data Link Layer) DLL
  - Subcapa de enlace lógico (Logical link control sub-layer) LLC
  - Subcapa de acceso al medio (Medium access control sub-layer) MAC

  Estas capas están descritas en los capítulos precedentes.

- Capa Física (Physical layer) (capa 1) PL
  - Physical signaling sub-layer PLS (códificación del bit, temporización del bit, sincronización)
  - Physical medium attachment sub-layer PMA (Características de los niveles de control -drivers- y de recepción)
  - Médium dependent interface sub-layer MDI (Conectores)

- Medio de transmisión (Transmisión médium)
  - Bus cable
  - bus terminación networks (Cf. [Site, 1999]).

3.3. Características necesarias en el soporte físico.

Aunque el tipo de soporte físico que sirve para transportar las tramas CAN no sea explícitamente definido y sea más bien dejado al gusto de cada diseñador en el protocolo CAN, a fin de respetar la estructura y arquitectura del CAN, es necesario que el medio elegido:

- Sea capaz de representar los bits que poseen los estados dominantes y recesivos sobre el soporte de transmisión,

[1] Las capas situadas encima de la capa de data link no hace parte de los documentos ISO. Ellas deben de ser establecidas y especificadas por el diseñador del proyecto.

- Esté en el estado recesivo cuando un nodo emite un bit recesivo o cuando se de la ausencia de transmisión de los nodos,

- Soporte la función del nivel de transmisión, del mismo hecho de su implementación,

- Esté en el estado dominante si alguno de los nodos emite un bit dominante que aplaste al bit recesivo (Cf. [Site, 1999]).

3.4. El bit CAN.

En esta subsección se aborda las especificaciones de la subcapa de señalización física (physical signaling sub-layer specification -PLS) del ISO.

En esta parte se proponen los problemas que tienen que ver con el bit (codificación, duración, sincronización) y aquellas de los débitos brutos y netos de la red (Cf. [Site, 1999]).

3.4.1. Codificación del bit CAN.

Como ya se ha señalado, el bit es fundamentalmente codificado según el principio de non return to zero (NRZ)[2] (ver la Figura 3.1.) (Cf. [Site, 1999])

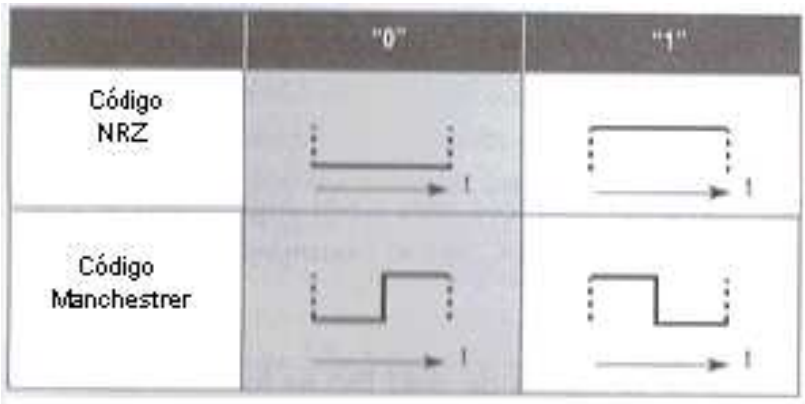

Figura 3.1: Código del bit CAN

[2] El procedimiento de bit stuffing descrito en el Capítulo no cambia en nada el tipo de codificación del bit mismo.

3.4.2. Bit time.

Es el lapso de tiempo que dura la presencia efectiva del bit sobre el bus y se denomina bit time (tb).

Las funciones que fueron anteriormente definidas son llevadas a cabo por una entidad electrónica realizada en lógica probabilística, que se dice de bit timing, y que están contenida en los circuitos integrados teniendo por misión la de administrar el protocolo CAN, bajo las siguientes líneas:

- Administración del bus (bus management) - ejecutada en el interior del espacio de tiempo que dura el bit (tb o también bit time frame) tales como las restricciones de sincronización de las estaciones (Electronic Central Units -ECU) dispuestas sobre la red,
- Compensación de los retardos de propagación,
- Definición de la posición exacta del punto de muestreo (Cf. [Site, 1999]).

3.4.3. Nominal bit time.

Cuando se habla de nominal bit time, se dice por definición duración nominal de un bit.

En principio, este valor no puede ser más que ideal y teórico. Esto es el diseñador del sistema desea dar un valor nominal al bit time de su red proyectada.

Cada estación de la red debe de ser concebida para poseer este valor de nominal bit time (tb) sin embargo, para el diseñador, teniendo en cuenta que los componentes que constituirán concretamente cada una de ellas (y sobre todo del hecho de las precisiones de sus propios osciladores, de su tolerancia, de la variación de la temperatura y el tiempo, etc.), el valor instantáneo del bit no será jamás realmente nominal en todo instante, es por tanto que habrá que tenerlo todo el tiempo en cuenta.

La duración nominal del bit esta habitualmente construida electrónicamente a partir de un número determinado de ciclos de sistema que tiene origen de lo que debería constituir el reloj interno de cada estación (Cf. [Site, 1999]).

Está es igual, por definición, a:

$$Nominalbittime = 1/nominalbitrate \tag{3.1}$$

En el documento oficial del protocolo del CAN, se indica que el nominal bit time se subdivide en muchos segmentos de tiempo que no se superponen entre ellos. Estos son, en el orden cronológico (ver la Figura 3.2):

- El segmento de sincronización,
- El segmento de tiempo de propagación,
- El segmento número uno que se llama phase buffer 1,
- El segmento número dos que se llama phase buffer 2 (Cf. [Site, 1999]).

De los cuales se darán a continuación las definiciones.

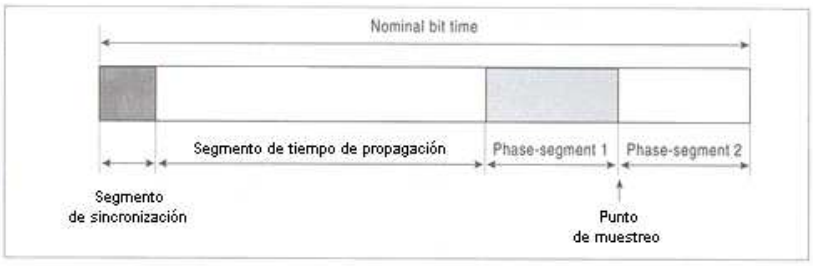

Figura 3.2: Nominal bit time

Segmento de sincronización.

El segmento de sincronización es una parte del bit time utilizado para sincronizar los diferentes nodos dispuestos sobre el bus. Por definición, el flanco antes de un bit incidente *se espera* que se produzca en el interior de este segmento de tiempo (Cf. [Site, 1999]).

Nota importante.

La palabra *se espera* se justifica por el hecho de que el bit time nominal es igual al tiempo que el bit debería durar pero que sin embargo, debido a la codificación del bit efectuado en NRZ, puede suceder que al momento en donde se espera su presencia (temporal de un flanco), esta no exista (físicamente) debido, a que la señal (bit eléctricamente presente sobre el bus), continua a tener su valor anterior y no se produce la señal física de transmisión. A pesar de todo esto, en ese instante hay que considerar que el principio de un nuevo bit esta probablemente produciéndose en ese instante.

Segmento del tiempo de propagación.

Esta parte del *nominal bit time* esta concebida para tomar en cuenta y compensar el tiempo de retardo debido a los fenómenos físicos de propagación de la señal sobre el medio utilizado y los tipos de topologías empleadas para realizar la configuración geométrica de las redes. Para fijar desde ahora estas ideas y para que el CAN funcione correctamente, se dice que este valor representa un tiempo igual al doble (ida + regreso) de la suma de los tiempos de propagación presentes debidos al medio y de los retardos aportados por los diferentes niveles de entrada y salida de los participantes dispuestos también sobre el bus (Cf. [Site, 1999]).

Segmentos de phase buffer 1 Y 2.

Esta pareja de buffers[3] de fases (phase buffer segment 1 y 2) es utilizado para compensar las variaciones, errores o modificaciones de fase y/o de posición de los flancos de las señales que pueden producirse cuando se existe un intercambio sobre el medio: jitter, variaciones de umbrales de detectores de niveles de los comparadores de entrada de los circuitos integrados, integraciones y deformaciones de las señales a lo largo de la línea que forma el bus (Cf. [Site, 1999]).

Estos dos segmentos están concebidos para ser alargados o recortados por el mecanismo de resincronización como se indicará más adelante.

Es así como terminamos el inventario de la terminología de estos segmentos; En la siguiente sección se examinará con detalle las razones y fenómenos físicos que han llevado a definirlos.

Muestreo, punto de muestreo y tratamiento del bit.

¿Para que serviría transportar las informaciones si no es para recibirlas, leerlas e interpretarlas correctamente? Para esto, es necesario proceder a un número importante de manipulaciones cuando sucede la llegada de un bit para poder asegurar su integridad (Cf. [Site, 1999]).

Punto de muestreo.

Este instante - el punto de muestreo, *sample point*- no debe producirse ni demasiado temprano ni demasiado tarde durante el lapso de tiempo que dura el bit. La elección y/o la posición temporal correcta del punto de muestreo son esenciales para asegurar el buen funcionamiento, y por lo tanto el éxito de una transmisión (Cf. [Site, 1999]).

[3] El término buffer de fase se debe de interpretar en un sentido físicamente equivalente a los enganches mecánicos situados entre los vagones de un tren cuya misión es la de asegurar las compensaciones y absorciones de los golpes debido a las diferencias de velocidades instantáneas entre los vagones.

La respuesta al porque resulta crucial está elección es simple. Basados en el principio que para poder resolver los fundamentos del protocolo (fases de arbitraje y de reconocimiento), los problemas debidos a la deformación de la señal que representa el bit y los retardos debidos a los fenómenos de propagación de la señal sobre el medio, se deberían leer e interpretar lo más tarde posible, el valor del bit time para estar lo más seguro posible de su integridad (Cf. [Site, 1999]).

Es necesario entonces, no esperar al último momento, debido al menos a las 2 razones siguientes:

- Es necesario tomar en cuenta todas las tolerancias que participan al establecimiento de la duración del bit,
- Existe de forma inherente, un tiempo necesario que se debe conceder para proceder al calculo del valor del bit cuando se produce su muestreo.

En conclusión, el instante en donde debería producirse el muestreo del bit debería situarse más bien al final que al principio del tiempo que dura el bit (Cf. [Site, 1999]).

La posición de este instante preciso (el punto de muestreo) sitúa por definición exactamente la frontera entre el segmento número 1 de phase buffer y el segmento de la fase número 2 (Cf. [Site, 1999]).

Para asegurar esto, cuando se concibe un proyecto, es importante conservar en memoria otros puntos que son importantes y que se mencionan a continuación, y que tiene que ver con la precisión del instante del muestreo del bit.

Estos puntos importantes abordan los siguientes problemas:

- Una inicialización de la trama (SOF) obliga a todos los controladores presentes sobre el bus a proceder a una sincronización del tipo hardware sobre el primer paso de un nivel recesivo a dominante,
- Es necesario tomar en cuenta también la presencia de las fases eventuales de arbitraje durante las cuales muchos controladores pueden transmitir simultáneamente,
- Para evitar una posición temporal incorrecta del punto de muestreo del bit, y por lo tanto una eventual lectura errónea de su valor, es también necesario incluir la presencia de buffers temporales de sincronización adicionales situados de una parte y de la otra de ese instante (buffers de fase 1 y 2). Las principales razones que pueden llevar a muestreos incorrectos son:
  - Sincronización incorrecta debida a las breves impulsiones (spikes),

  - Ligeras variaciones de la(s) frecuencia(s) del (los) oscilador (es) de cada uno de los microcontroladores CAN de la red que producen los errores de fase (Cf. [Site, 1999]).

- El *time segment* de fase 2 (de concepción idéntica a la fase segmento 1 de protocolo CAN) que tienen por misión la de constituir:

  - Un segmento de buffer de tiempo dispuesto después del punto de muestreo con lo cual el objetivo es el de participar también a la (re) sincronización del bit si es necesario,
  - Una reserva de tiempo llamada comúnmente *information processing time* (tiempo de tratamiento de la información); entre el lapso de tiempo situado iniciando inmediatamente después del punto de muestreo (es decir al principio del segmento de tiempo de la fase 2) el cual es necesario en el cálculo del valor del bit (Cf. [Site, 1999]).

Nota relacionada con la razón del cálculo del bit.

Con el fin de mejorar la calidad de captura y validación del bit (precedencia de parásitos, fase eventual de arbitraje), es mejor efectuar un filtraje numérico realizando un muestreo del valor del bit muchas veces a partir del instante de *sample point*. Para información, existe un circuito (82c200) que es capaz de hacer esto en programación, si se desea, y efectuar un muestreo múltiple del valor del bit un número impar de veces a fin de determinar el valor del bit con la ayuda de una lógica mayoritaria. Es lo que se le llama generalmente un enriquecimiento del protocolo (Cf. [Site, 1999]).

3.5. Nominal bit time.

Cada uno de los segmentos precedentes que han sido detallados determina la calidad, el flujo de información, la longitud, de una red CAN y existen numerosas iteraciones entre ellos, lo que puede provocar en el lector un malestar debido a que no siempre se puede tener una presentación fácil, clara y limpia.

Al final de este Capítulo, encontrará una Figura que resume todos los párrafos que se expondrán, sin embargo esta Figura no parecería muy clara si no se han leído estos párrafos.

La secuencia que seguirán las explicaciones se limitan a:

- Indicar la construcción del bit time,
- Describir el papel de los diferentes segmentos,
- Explicar sus interrelaciones,

- Y finalmente determinar los valores en un caso de aplicación.
  (Cf. [Semiconductor, 1994b]).

3.5.1. Construcción del bit time.

Todo lo que se acaba de describir, es necesario asociarlo necesariamente a los segmentos de los valores temporales y sus medidas relativas respectivas.

El *nominal bit time* de una red CAN es fija y definida para un proyecto y cada estación debe ser capaz de construir localmente su propio *nominal bit time* con la ayuda de sus propios recursos. Cada corte local del *nominal bit time* podrá por lo tanto ser estructuralmente diferente de estación a estación. Claramente es entonces preferible que el jefe de proyecto, responsable de la concepción de la red, armonice el conjunto de los cortes temporales y todos los nominal bit time individuales (Cf. [Semiconductor, 1994b]).

A fin de no llegar a una cacofonía sin nombre, el protocolo obliga a respetar ciertas reglas de construcción del bit time las cuales no se pueden omitir. Entre ellas, se indica de forma precisa, la de definir como base de la construcción del *bit time* de una estación local, al más pequeño elemento de tiempo de referencia utilizable por esta -llamada mínimo *time quantum*- sin embargo igualmente algunas derivaciones de estas reglas se presentarán el la siguiente sección.

Mínimo time quanta de una estación.
El valor mínimo *time quantum* está generalmente derivado del reloj presente en el sistema - clock - presente en la estación (tomando por ejemplo una señal que viene del cuarzo que sirve del reloj al microcontrolador que administra la funcionalidad del nodo) (Cf. [Semiconductor, 1994b]).

Pasemos a la etapa siguiente: la definición del time quantum del (nominal bit time).

Time quantum del bit time.
El time quantum es una unidad de tiempo fijo, derivada y obtenida partiendo del time quantum, a partir de la cual va a ser construida el bit CAN.

La realización concreta de esta unidad de tiempo es efectuada con la ayuda de (pre) divisores programables (por valores enteros, m) de los cuales la gama de división se debe situar entre 1 y 32. A partir de un mínimo time quantum (por ejemplo aquel del reloj, clock, de la estación) el time quantum puede tener una duración de: $Timequantum = mxmnimotimequantum$, en donde m es el valor del factor de división de el (pre) divisor (Cf. [Semiconductor, 1994b]).

Corte del bit time.

El protocolo CAN define los valores siguientes:

- El segmento de sincronización vale un time quantum (este valor es fijo),
- El segmento de tiempo de propagación vale de 1 a 8 time quanta[4] ,
- El segmento número 1 del phase buffer vale de 1 a 8 time quanta,
- La information processing time debe ser inferior o igual a 2 time quanta,
- El segmento número 2 del phase buffer es igual al máximo del valor del segmento número 1 más el information processing time (Cf. [Semiconductor, 1994b]).

Notas: A fin de adaptarse lo mejor posible a los desempeños de los componentes electrónicos con los parámetros de una red CAN, los valores que constituyen estos segmentos son generalmente programables con la ayuda de registros específicos.

En el caso concreto del circuito integrado que gestiona el protocolo el PCA 82C 200, por razones tecnológicas, su fabricante juzgo útil definir un nuevo segmento de tiempo -el time segment 1 (TSEG 1) que caracteriza el conjunto de los segmentos de compensación de los tiempos de propagación y del buffer de sincronización fase segmento 1 del protocolo CAN (Figura 3.5). Para la utilización y la aplicación general CAN, esto llega a ser prácticamente lo mismo.

Es importante que los osciladores de los diferentes ECU estén coordinados o armonizados para ofrecer al sistema un time quantum especificado lo mejor posible.

3.5.2. Corte del bit time.

Nuestros cuatro segmentos están ahora en su lugar y no nos resta más que definir el número de quanta que se dedican a los segmentos de propagación y de los phase buffer 1 y 2. Una vez que esto es conocido, es más fácil deducir la duración del nominal bit time y por lo tanto el caudal de información bruta de la red (Cf. [Semiconductor, 1994b]).

Sin embargo, en este nivel, el conjunto de los elementos que constituyen la red (medio, componentes, topología) y que intervienen, llega a ser urgente el tener que proceder a una clasificación rigurosa de los parámetros y de proceder con un cierto orden y una metodología. Enseguida, se examinan los diferentes segmentos con el fin de clasificarlos (Cf. [Semiconductor, 1994b]).

[4] El número total de time quanta contenidos en un bit time que puede ser elegido por programación no debe ser ni inferior a 8 ni superior a 25 (ver ejemplos en figuras 3.3 y 3.4). Es en este momento, en el que se puede decir que la arquitectura del nominal bit time esta perfectamente definida.

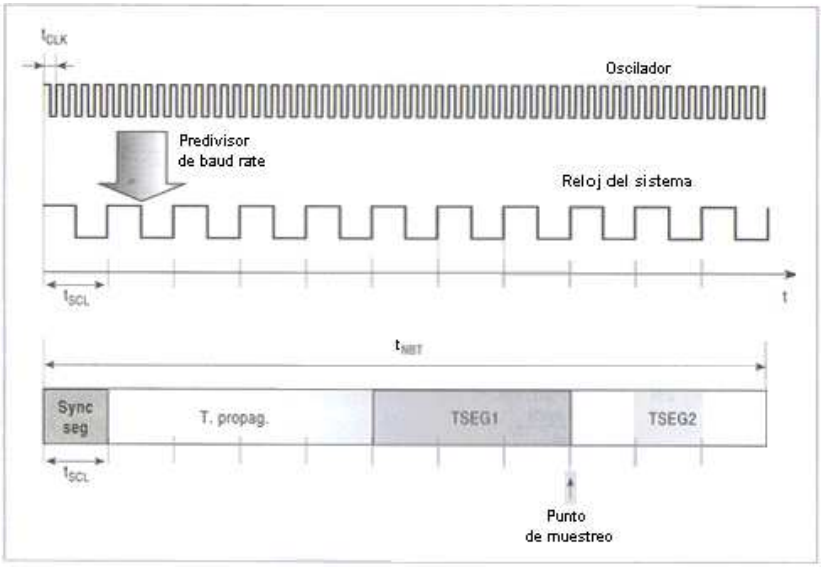

Figura 3.3: Ejemplos

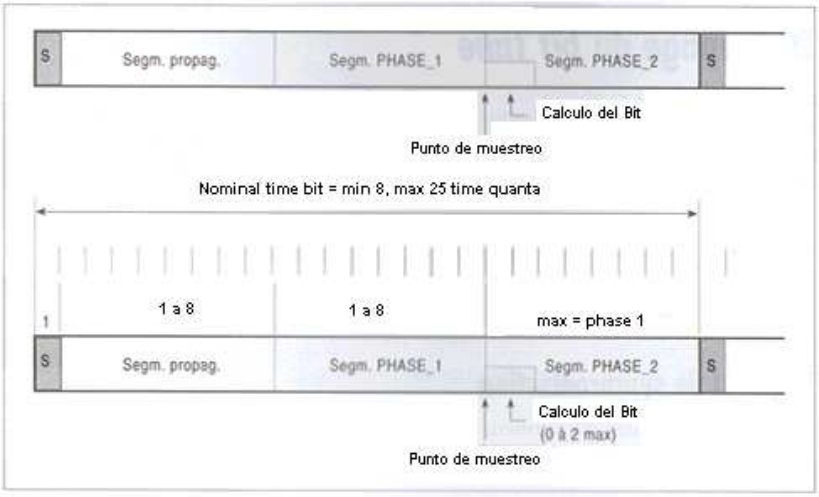

Figura 3.4: Ejemplos

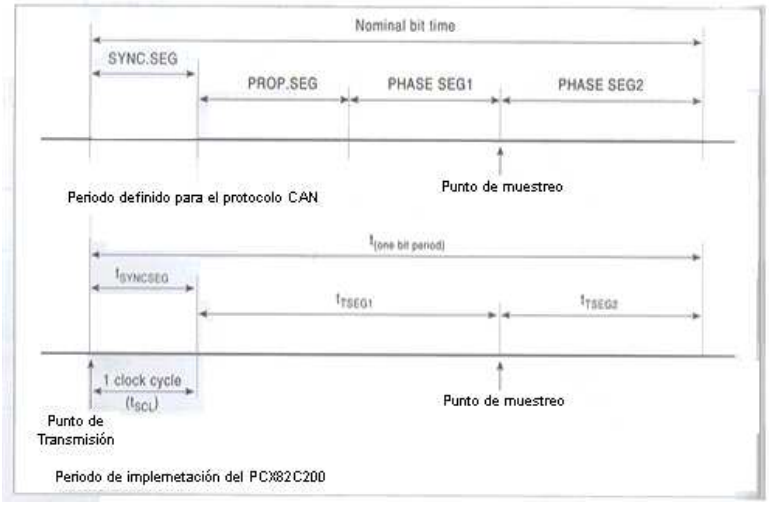

Figura 3.5: Periodo implementación PCX82C200

Segmento de sincronización.

La duración del segmento es fija y siempre vale 1 quantum.

Segmento que se llama de propagación.

Los parámetros que toman en cuenta para definir el valor del segmento (independientemente de las tolerancias de los relojes de las diferentes estaciones) son:

- El medio (y su elección),
- La velocidad de propagación de la señal sobre este medio, y la longitud máxima de la red (medio),
- La topología de la red,
- Los parámetros eléctricos de entrada /salida de los componentes, la interfaz de la línea,
- La calidad de las señales recibidas,

Todos estos parámetros tienen algunas propiedades del medio (por ejemplo, topología), que se han reagrupado sobre el mismo tema: los problemas relacionados al fenómeno de propagación (Cf. [Semiconductor, 1994b]).

Segmento de los buffer 1 y 2.

Independientemente de los medios, longitud, etc., citados antes los parámetros son:

- Los valores nominales de las frecuencias de reloj de cada nodo,
- Las tolerancias de estos relojes,
- Las derivas en temperatura y temporales de estos relojes,
- Los *peores casos*,

Todos estos parámetros tienen que ver con la calidad de éxito de la sincronización o resincronización del número de bits totales incidentes en función de las tolerancias, derivas, etc., que se encontrará en la parte ligada con los problemas que se llaman de sincronización (Cf. [Semiconductor, 1994b]).

3.6. CAN y propagación de la señal.

Hay mucha teoría que ha sido escrita sobre los diferentes aspectos teóricos en las redes. En esta monografía no es nuestro propósito escribir de nuevo todas estas partes teóricas si no más bien de sacar a luz las particularidades del protocolo CAN en sus aplicaciones en las redes locales (Cf. [Semiconductor, 1994a]).

Si se habla de redes se tiene que hablar necesariamente de estructura, topología y medio. No se pueden evitar estos grandes estándares y los comentarios que ahí se encuentran.

3.6.1. Tipo, topología y estructura de las redes.

No importa cual sea el tipo de medio utilizado, su topología es en una gran parte dictada por las posibilidades que ofrece el protocolo que sirve de transporte de información.

En el caso que nos ocupa, la topología puede ser variada (Cf. [Semiconductor, 1994a]).

Topología en bus.

El bus es la topología más fácil a ser puesta en marcha y la más expandida para este tipo de protocolo. Algunas de sus particularidades son mencionadas a continuación.

En principio, todas las estaciones deberían estar conectadas lo más cercanas al bus (ver la Figura 3.6.) Otro bus del tipo backbone ó sea columna vertebral (ver la Figura 3.7) está muy próximo de la que se ha presentado precedentemente en la Figura 3.6, sin embargo varía ligeramente, en lo que tiene que ver con sus propiedades eléctricas debido a los flujos de información elevados (Cf. [Semiconductor, 1994a]).

De hecho, hay que pensar en términos de las líneas eléctricas y por lo tanto evocar y tener en cuenta los componentes que vayan a ser conectados sobre estas líneas.

Esto quiere decir que existe una frontera, un poco difusa pero sin embargo muy real, entre los flujos de información bajos (donde estos fenómenos pueden ser considerados como inexistentes) y las relaciones ó flujos de información mucho más rápidas en donde entra concretamente en el dominio de las líneas de transmisión. Esta es la eterna relación que existe entre las longitudes de onda de las señales que son transportadas y las distancias que deben recorrer (Cf. [Semiconductor, 1994a]).

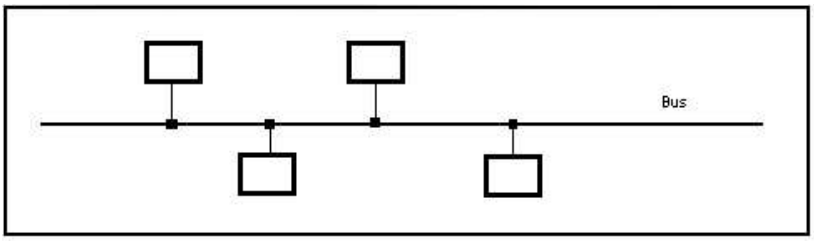

Figura 3.6: Topología bus

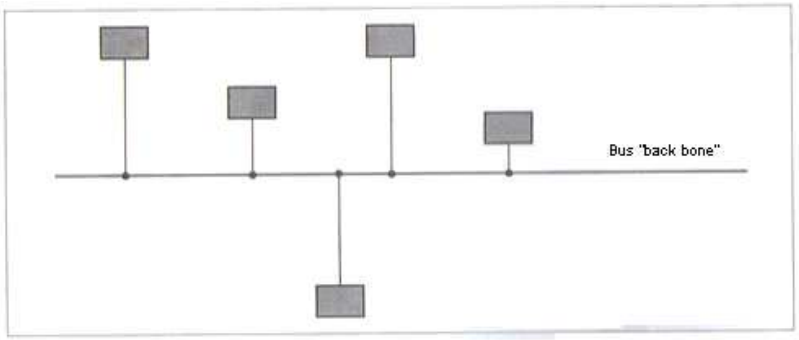

Figura 3.7: Bus *back bone*

Para información.

Debemos tener en cuenta que para un flujo de información binario NRZ (onda de señales cuadradas) de 1 Mb/s (o sea para una frecuencia equivalente de las señales cuadradas de 500 kHz) se propagan por ejemplo sobre un par diferencial a 200000 Km/s, la frecuencia fundamental (sinusoidal) de la onda debería tener una longitud de onda de:

$$\lambda = vT, \lambda = 2x108x(5x105) - 1 = 400metros \tag{3.2}$$

La onda cuadrada que es formada de las armónicas impares, las otras longitudes de onda de la señal de base serían de: 400/3, 400/5, 400/7 metros. Es decir alrededor: 133, 80, 57 metros lo cual es un poco corto (Cf. [Semiconductor, 1994a]).

En este sentido el protocolo CAN ha sido bien planeado, al instaurar las recomendaciones especificas para los modos CAN low speed (velocidad inferior a 125 Kb/s) y CAN high speed (de 125 Kb/s a 1 Mb/s) (Cf. [Semiconductor, 1994a]).

Una cuestión que podría ser planteada debido a los principios técnicos y/o económicos es aquella que se formula cuando se desea trabajar en la frontera de los 2 nodos - algo así como un médium speed- alrededor de 250 Kb/s. En este caso, sin querer penalizar los costos, el problema es tan complejo como aquel del modo high speed puesto que es absolutamente necesario pensar en la línea y considerar las impedancias características, las adaptaciones/desadaptaciones de impedancia, los rebotes, las tasas de las ondas estacionarias, los vientres y nodos de tensión, las terminales (en valores de impedancia y en posiciones físicas.) Y es también absolutamente necesario interesarse a las respuestas de las señales numéricas de la línea que ha sido así constituida (respuesta impulsional, under/overshoot, diagrama de red) y en consecuencia en la seguridad de los datos transportados (Cf. [Semiconductor, 1994a]).

Precisamente hay que observar que lo que tiene que ver con la seguridad del transporte de los datos, es decir, en los capítulos precedentes tienen que ver con los errores y sus tratamientos y con la ayuda de ellos nos dan una idea más precisa (Cf. [Semiconductor, 1994a]).

Topología en estrella.

En una red o en una parte de una red de forma seguida se presenta sobre una forma topológica de estrella (ver la Figura 3.8). Esta configuración no formula ningún problema estructural (Cf. [Semiconductor, 1994a]).

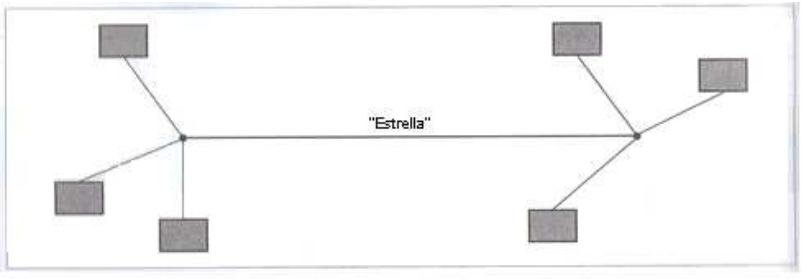

Figura 3.8: Topología en estrella

Topología en anillo.

Cuando se desea disponer de una más grande seguridad y fiabilidad del funcionamiento de los sistemas en caso de falla mecánica de la red (el caso de los cables que pueden ser seccionados, por ejemplo), frecuentemente se llega a proceder a (des) doblamientos físicos de los buses entre dos o más estaciones (ver la figura 3.9). Esto conduce tarde o temprano a una topología del tipo de anillo (esto fuera de toda noción de protocolo, por ejemplo del tipo de una ficha) (Cf. [Semiconductor, 1994a]).

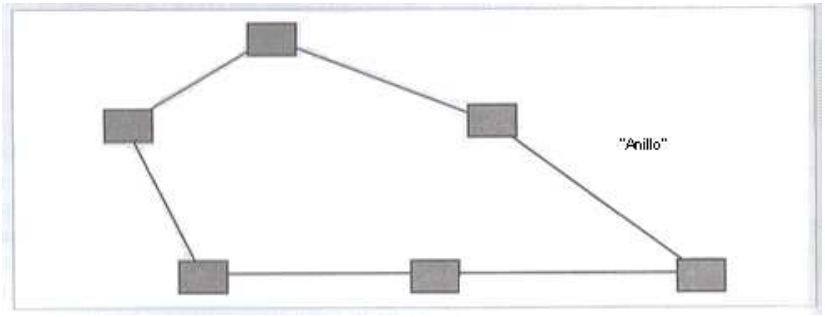

Figura 3.9: Topología en anillo

En el caso del protocolo CAN, esta topología particular viene a ser una vez más examinada con atención para la estimación y la administración de los tiempos de propagación, para resolver de forma precisa el problema de la fase de arbitraje del bit (Cf. [Semiconductor, 1994a]).

3.6.2. Tiempo de propagación.

Si hay una línea, entonces hay un fenómeno de propagación de la señal sobre la línea, por lo tanto de velocidad de propagación, y en consecuencia hay un retardo debido al tiempo pasado a la propagación de la señal sobre el medio, impedancia característica, y de adaptación (y desadaptación), y en consecuencia nodos y vientres de ondas, ondas estacionarias, rebotes al final de la línea y colisiones posibles sobre el bus entre las señales incidentes y las señales reflejadas (Cf. [Semiconductor, 1994a]).

Para interesarse a lo que tiene que ver con los tiempos de propagación, es necesario de tomar y asumir algunas hipótesis [5] .

[5] Los valores de estos parámetros son generalmente fácilmente accesibles con la ayuda de otras técnicas de los diversos elementos involucrados. Estos parámetros siendo conocidos, se puede pasar entonces a la determinación del valor del segmento de propagación que se debe asignar a un nodo.

De hecho, consideremos como conocidos o definidos:

- La elección del medio (por ejemplo, por razones de costo),
- La velocidad de propagación de la señal sobre el medio elegido,
- La longitud del medio (por razones propias a la aplicación),
- El tiempo de retardo interno de los componentes de circuito de emisión y de recepción (Cf. [Semiconductor, 1994a]).

3.6.3. Estimación física del valor del segmento de propagación.

La Figura 3.10 sintetiza el ejemplo característico de los fenómenos que se estudiarán en el caso más desfavorable (worst case) teniendo en cuenta los caminos recorridos, los retardos debidos a las propagaciones de las señales y a las tolerancias y derivas de los osciladores de cada uno de los participantes (Cf. [Semiconductor, 1994a]).

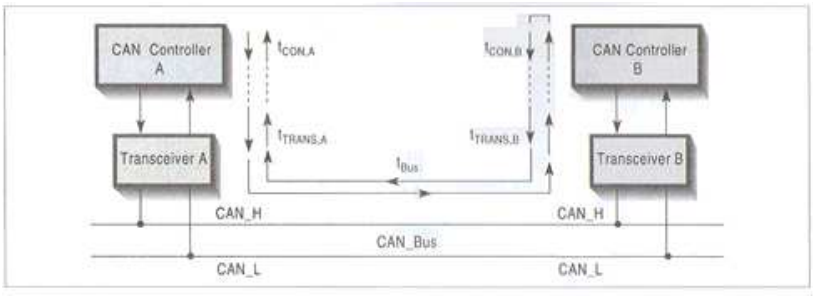

Figura 3.10: Ejemplo

En una misma red, la distancia más grande entre dos controladores CAN determina el tiempo de retardo máximo de la propagación de ida.

Aun más, para asegurar un buen funcionamiento del protocolo durante las fases de arbitraje y de reconocimiento, un lapso de tiempo más elevado debe ser previsto para compensar la totalidad de los tiempos de retardo de propagación que entran en cuenta durante un intercambio sobre la red (Cf. [Semiconductor, 1994a]).

Para comprender mejor lo anterior tome el siguiente ejemplo, donde se ven las fases de comunicación:

- Emisión de un bit Recesivo por el nodo A,
- La señal originada en el nodo A se propaga hacía el nodo B,

- Algunos micro instantes antes de llegar al destino, el nodo B no habiendo observado todavía la presencia del bit incidente que proviene de A inicia la emisión de un bit dominante,
- La señal que proviene de B se propaga hacía A,
- La colisión se produce en un lugar muy próximo a B y esto tiene por efecto de llevar a la línea al estado dominante (nivel transmitido) en ese lugar,
- La señal resultante (dominante) se propaga entonces hacía A y toma un cierto tiempo, sensiblemente igual a aquel de ida, para llegar,
- La estación A, que recibe un bit dominante durante su emisión de un bit recesivo, es entonces apto, según su estado instantáneo, a definir si esto representa para ella una perdida de arbitraje (si el bit que ella emitía hace parte del identificador), o bien un bit de error (Cf. [Semiconductor, 1994a]).

De hecho, para asegurar correctamente este tipo de mecanismo y el buen funcionamiento del protocolo, la duración del bit emitido por la estación A debe ser entonces igual o superior a la suma global de todos estos tiempos. Se puede decir, si abusamos un poco del lenguaje, que esta suma de tiempos representa el tiempo máximo de propagación necesaria que tomará la señal para efectuar la ida y vuelta de un controlador al otro. Para ser más precisos habría que tener en cuenta los tiempos necesarios a los tratamientos y cálculos a ser efectuados por cada estación dispuesta en las extremidades de la topología de la red (Cf. [Semiconductor, 1994a]).

Citemos dos ejemplos característicos que pueden producirse: dos controladores CAN sobre un bus simple durante una fase de arbitraje, y el caso en donde dos nodos están presentes sobre una red en anillo (ver la Figura 3.11).

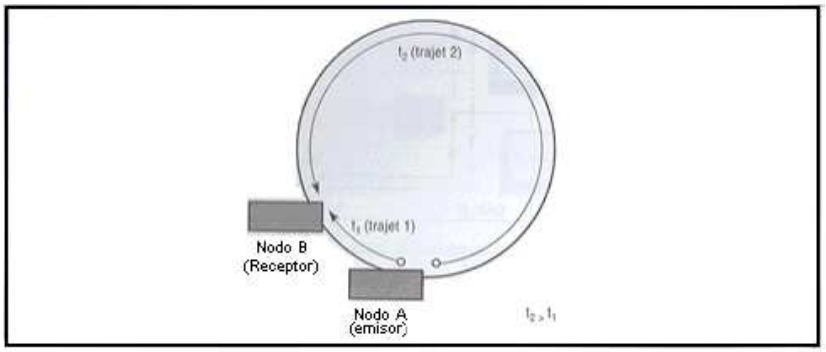

Figura 3.11: Ejemplo

3.6.4. Definición precisa de la duración del segmento de propagación.

Con la ayuda de las explicaciones anteriores, llega a ser evidente que, para un medio determinado, la duración temporal que va a afectar al segmento de propagación del bit CAN participa activamente en la evaluación de la longitud máxima de la red, y viceversa (Cf. [Semiconductor, 1994a]).

Para definir con precisión la duración mínima a ser atribuida al segmento de propagación, seg_T_prop, es necesario tomar en consideración la totalidad de las contribuciones temporales de cada uno de los elementos de la red (ver Figura 3.12).

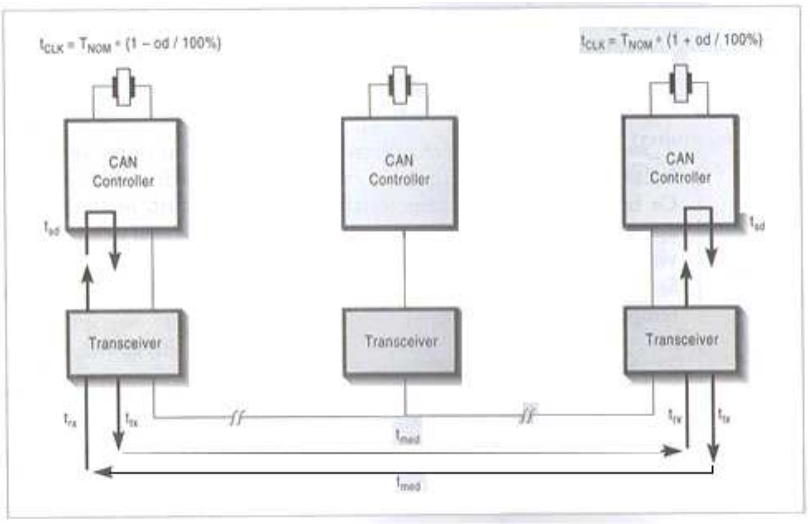

Figura 3.12: Ejemplo

Siguiendo la señal, desde su salida del emisor hasta la llegada efectiva en la entrada de la estación receptora, se encuentra sucesivamente:

- El tiempo de retardo puesto por el controlador de emisión para sacar la señal en sus terminales,
- El tiempo de retardo que hace la interfaz de emisión para generar la señal sobre el medio,
- El tiempo necesario al transporte de la señal sobre el medio,
- El tiempo que hace la interfaz de recepción para transitar la señal hacia el controlador de recepción,
- El tiempo que hace el controlador de recepción para tratar la señal incidente (Cf. [Semiconductor, 1994a]).

Frecuentemente, se dividen estos parámetros en dos clases, aquellos que tienen que ver más específicamente con las propiedades físicas del medio utilizado, T_med, y aquellos propios al conjunto de características electrónicas presentes sobre la red, T_elec.

La suma de estos dos componentes, T_res, caracteriza el tiempo total que hace un bit para ir realmente de una estación a la otra, es decir (ver Figura 3.13) (Cf. [Semiconductor, 1994a]):

$$T\_res = T\_med + T\_elec \tag{3.3}$$

Con T_elec = suma de los tiempos de retardo T_xxxx.

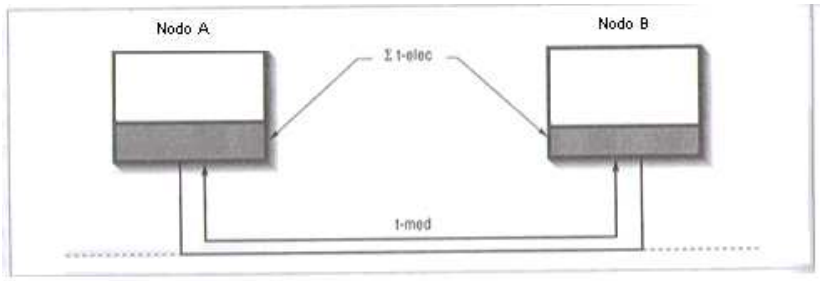

Figura 3.13: Suma de los tiempos de retardo

T_med

T_med representa el tiempo necesario para el transporte de la señal sobre el medio en el caso de la más grande distancia entre dos controladores CAN en la red.

Este tiempo depende de las características físicas del medio utilizado, de su aptitud de transportar más o menos rápidamente la señal y en consecuencia de la velocidad de propagación intrínseca (Cf. [Semiconductor, 1994a]).

Si llamamos v_prop a esta velocidad y que el medio posee una longitud L, entonces ese tiempo es:

$$T\_med = L/v\_prop \tag{3.4}$$

Si tomamos el caso clásico del medio por hilos (par diferencial, por ejemplo) la velocidad de propagación es sensiblemente igual a 0.2 m/ns (200 m/s ó, dicho de otra forma posee un tiempo de propagación 5 ns/m) (Cf. [Semiconductor, 1994a]).

T_elec

Tres elementos participan a la cuantificación de este tiempo.

T_sd: que representa la suma de los tiempos de retardo debidos al tratamiento de la señal realizados en las etapas de salida (a la salida) y de entrada (a la llegada) de los controladores CAN de las estaciones de salida y de llegada.

T_tx : que representa el tiempo de retardo que hace la interfaz de línea de misión para generar la señal sobre el bus.

T_rx: que representa el tiempo que hace la interfaz de la línea de recepción para hacer transitar la señal hacia el controlador de recepción.

Los fabricantes de los componentes de interfaz de línea (drivers de línea) indican con precisión los valores de los elementos de T_tx, T_rx en sus hojas técnicas de características cuando las señales incidentes son propias (es decir bien cuadradas) (Cf. [Semiconductor, 1994a]).

Ese hecho es excepcional, a continuación se examino un poco más de cerca lo que pasa cotidianamente en la realidad.

T_qual_sign

Muy frecuentemente, el medio utilizado esta formado de pares diferenciales (debido a sus grandes aptitudes a rechazar las señales parásitas inyectadas en modo común). En la recepción, la interfaz de entrada de un circuito integrado se compone generalmente de un amplificador operacional, montado/seguido, asociado de un comparador.

Claramente, este último posee sus propias características de umbral y de histéresis. Estos elementos, calculados en tensión, deben ser traducidos en tiempos equivalentes (ver la Figura 3.14) en función de la forma y calidad de las señales incidentes (habitualmente idealizado en toda la literatura) (Cf. [Semiconductor, 1994a]).
De hecho, es más bien estas señales como las representadas en la Figura 3.15, debido sobre todo a la influencia de las capacitancias e inductancias repartidas a lo largo de la línea (Cf. [Semiconductor, 1994a]).

Nota.
La disimetría tiempo de elevación/tiempo de descenso, voluntariamente exagerados sobre la Figura esta aquí para hacernos pensar en la presencia de dos problemas a veces completamente distintos.

Evidentemente, el comparador de entrada hará su mejor trabajo para cambiar rápidamente los niveles eléctricos, sin embargo, como se muestra en la Figura 3.15, los tiempos de retardo no son debidos a la propagación pura, ni a las características propias de los comparadores, si no a la calidad de las señales incidentes.

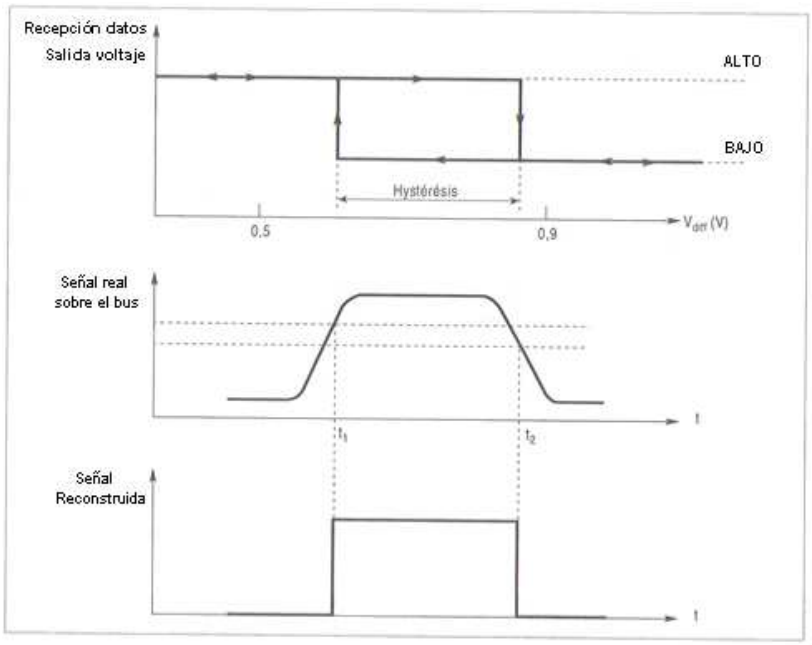

Figura 3.14: Cálculo de umbral e histéresis

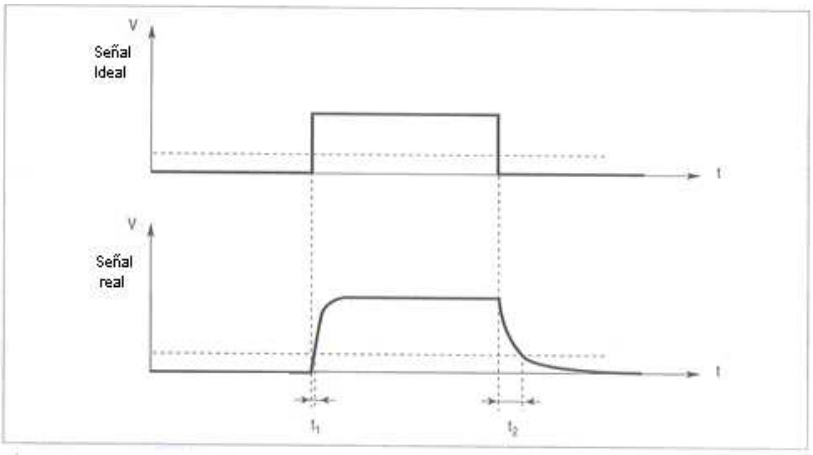

Figura 3.15: Representación de señales

Se puede entonces asimilar estos nuevos conceptos de tiempo de retardo eléctrico, T_qual_sign, en el equivalente de los tiempos de propagación o de las distancias suplementarias del medio.

Teniendo en cuenta todas estas observaciones, un orden de valor del conjunto T_elec puede ser estimado a una centésima de segundos.

Se obtiene finalmente el valor total mínimo de la duración del segmento de tiempo de propagación.

Ó sea, la duración mínima del segmento de tiempo de propagación:

$$seg\_T\_propmin \geq 2x(T\_med + T\_sd + T\_tx + T\_rx + T\_qual\_sign) \quad (3.5)$$

El factor 2 es dividido a la ida y el regreso de la señal para tener en cuenta la particularidad del protocolo cuando sucede el arbitraje y el reconocimiento (Cf. [Semiconductor, 1994a]).

Una vez que este valor determinado es bien conocido, el valor elemental del *time quantum* definido precedentemente, es bastante fácil determinar el número mínimo de *time quantum* necesarios a la elaboración del segmento de propagación (entre 1 mínimo y 8 máximo) del *nominal bit time.*

3.6.5. La relación entre medio/velocidad de transferencia/distancia recorrida.

El mismo problema se plantea frecuentemente en los corolarios siguientes:

- ¿Cual es la distancia máxima que puede recorrer el CAN para una velocidad X y un medio Y?
- ¿Cuál es la velocidad máxima utilizable para una red de longitud X sobre un medio Y?

Basta tomar en cuenta las nociones enunciadas precedentemente pero presentarlas de forma diferente.

Distancia máxima.
Para una velocidad y un medio dado, la lectura de la norma CAN muestra que la distancia entre dos nodos depende fuertemente de los parámetros siguientes:

- la velocidad de propagación propia al soporte físico utilizado para realizar la línea,

- el retardo que introduce el nivel de salida de la estación emisora,
- el retardo que introduce el nivel de entrada de la estación receptora,
- la velocidad binaria, bit rate, nominal deseada (ella depende entonces del valor máximo de la duración mínima del segmento de propagación),
- el instante preciso en donde es efectuado el muestreo de la señal y la forma en la cual se mide el valor del bit durante su presencia física,
- las frecuencias y tolerancias de los diferentes osciladores de los controladores (microcontroladores ó stand alone) CAN presentes sobre la red,
- la calidad de las señales. (Cf. [Semiconductor, 1994a]).

Esto puede resumirse por: la distancia máxima de una red CAN está principalmente ligada a las características temporales del medio utilizado y a las consecuencias del principio adoptado para el funcionamiento del procedimiento de arbitraje definido en el protocolo (a nivel de bit) no destructivo presente sobre la línea (Cf. [Semiconductor, 1994a]).

La relación velocidad / longitud máxima para un soporte determinado.

Consultemos nuevamente la Figura 3.12. Como se puede ver esta Figura, es obligatorio, para que el protocolo CAN funcione correctamente, tener en cuenta no solamente una vez el tiempo de recorrido de la señal sobre la red (T_res) si no también de alrededor de dos veces este valor, para que el emisor del mensaje pueda tener un regreso de información durante la duración misma del bit (en caso de fase de arbitraje, por ejemplo) (Cf. [Semiconductor, 1994a]).

Así, el valor dicho de segmento de tiempo de propagación del nominal bit time de una red CAN debe ser superior (o, al menos igual) a dos veces T_res. Seg_T_prop>= 2 T_res

El segmento de tiempo de propagación no representa más que una parte del nominal bit time, T_bit (Cf. [Semiconductor, 1994a]).

Ó sea el porcentaje de T_bit, se puede escribir:

$$Seg\_T\_prog = XxT\_bit \text{o también,} Seg\_T\_prog = X/\text{baud rate} \qquad (3.6)$$

Fijándonos en los valores obtenidos anteriormente en las ecuaciones precedentes, se obtiene la relación entre la longitud de una red y la velocidad elegida, habiendo tomado por hipótesis conocidas de una parte los valores de T_elec y la velocidad del medio

(el medio es entonces elegido) y, por otra parte, habiendo elegido ó predeterminado el valor de X, o sea:

$$L(0{,}2m/ns)x(X/(2xbaudrate)) - T\_elec \tag{3.7}$$

La curva de la Figura 3.16 da el aspecto gráfico de esta ecuación para:

$$T\_elec = 100nsV\_prop = 0{,}2m/ns(\text{medio de hilos y óptico})\ X = 0{,}66 = 66\,\% \tag{3.8}$$

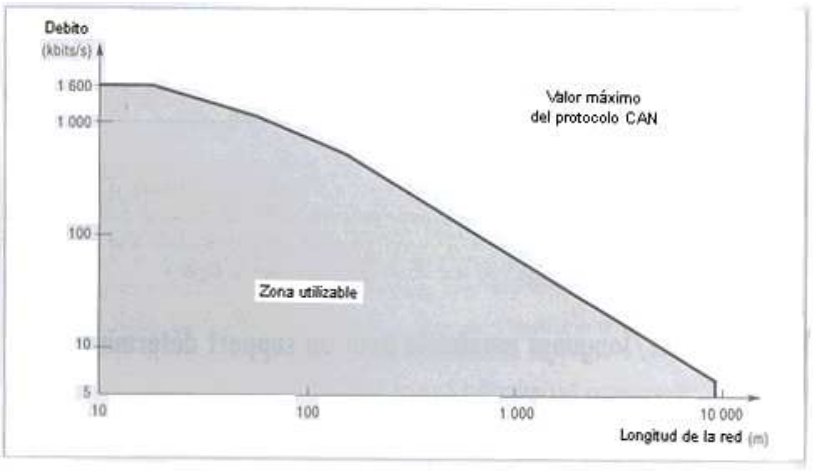

Figura 3.16: Representación de la ecuación 3.8

Esta curva permite estimar sin demasiadas sorpresas futuras los desempeños de una red (Cf. [Semiconductor, 1994a]).

Naturalmente, en el estudio riguroso de una red, es necesario calcular con precisión los diversos parámetros.

Es posible que en el estudio de su proyecto de red no se quiera utilizar parejas de hilos o bien tampoco las fibras ópticas y se quiera utilizar otro medio en este momento visto el cálculo que se ha realizado tenemos todos los elementos para efectuar cálculos aplicándolos a cualquier tipo de medio (Cf. [Semiconductor, 1994a]).

Ejemplo.

Ahora entonces nos encontramos en posesión de todos los elementos que sirven para cuantificar los tiempos de propagación y su compensación de una red con la ayuda de circuitos electrónicos apropiados en la circuitería de los microcontroladores CAN.

Para poder basar el estudio que acabamos de realizar, se verá en detalle algunos ejemplos que son basados en los microcontroladores 82 C 200 y 8x C 592, debido a que estos dos componentes poseen la misma arquitectura hardware CAN que tiene que ver con este tópico (consultar las hojas técnicas de estos microcontroladores para mayor detalle) (Cf. [Semiconductor, 1994a]).

Existen dos registros internos (el SFR, BTR0 y BTR1) en esta familia de microcontroladores, en los cuales los contenidos permiten manipular los valores para afectar a los diferentes segmentos del *bit time*. La Figura 3.17 representa la estructura del bit time y las definiciones de los parámetros para estos componentes (Cf. [Semiconductor, 1994a]).

Figura 3.17: Estructura del bit time

A continuación se verá, con las definiciones anteriormente presentadas, el estudio de los valores que deberían tener estos registros y el cálculo del tiempo de retardo máximo de propagación que podría soportar nuestra red, en los tiempos de propagación de soporte (cable) conocido, así como también la longitud máxima que podría soportar nuestra red.

Como una manera de ejemplificar, y sobre todo a fin de dar rápidamente algunas medidas, en el caso del 82 C 200 que se utilizará en este ejemplo, se obtiene la tabla de resultados que indica la Figura 3.18. En esta, fuera de la relación velocidad / distancia, se ha introducido directamente los valores a cargar en los registros BTR0 y BTR1 de forma tal que se puedan obtener estas velocidades con relación al cuarzo utilizado que en esta ocasión es de 16 MHz (Cf. [Semiconductor, 1994a]).

Debito	Distancia máxima	Bus timing	
		BTR0	BTR1
1,6 Mbits/s	10 m	00h	11h
1 Mbits/s	40 m	00h	14h
500 kbits/s	130 m	00h	1Ch
250 kbits/s	270 m	01h	1Ch
125 kbits/s	530 m	03h	1Ch
100 kbits/s	620 m	43h	2Fh
50 kbits/s	1,3 km	47h	2Fh
20 kbits/s	3,3 km	53h	2Fh
10 kbits/s	6,7 km	67h	2Fh
5 kbits/s	10 km	7Fh	7Fh

Figura 3.18: Tabla de resultados

Esta tabla no significaría nada si no se tiene mucho cuidado al indicar los parámetros incluidos en estos cálculos, a saber:

- la topología de la red es una simple línea, un bus por ejemplo, y no un anillo,
- las tolerancias del oscilador deben de ser inferiores a 0.1
- la línea, par trenzado en este caso, debe tener una velocidad de propagación inferior a 5 ns/m,
- la suma de los retardos, que introducen el conjunto de los circuitos integrados del emisor y del receptor, no deben exceder : 70 ns (en 1.6 Mb/s), 90 ns (de 250 kb/s a 1 Mb/s), 300 ns (de 5kb/s a 125 kb/s) (Cf. [Semiconductor, 1994a]).

3.7. Bit de sincronización (Synchronisation bit).

Así, es necesario considerar por una parte, las distancias importantes que se pueden recorrer con el bus y en consecuencia los retardos que no pueden ser omitidos debido a los tiempos de propagación de las señales transportadas y, por otra parte, las precisiones, derivas y tolerancias de las frecuencias de los osciladores de cada una de las estaciones. Lo anterior es para comprender las numerosas particularidades que acompañan a lo que uno sobrentiende por el vocablo de sincronización en el seno del protocolo CAN (Cf. [Bühring, 1991]).

Entonces teniendo en cuenta esto, se verá enseguida el synchronization bit.

Los párrafos siguientes describen a detalle todas las sutilezas que encierran estos procedimientos. Ellos definen en que momento el principio del bit ha llegado (o se espera que haya llegado con respecto al momento a donde se supone llegará), y reaccionar y muestrear el bit en consecuencia.

Entonces, aunque esto es un poco complejo se abordará esto en una forma ordenada en los párrafos siguientes.

3.7.1. Generalidades y notas importantes.

De entrada se examinarán algunos puntos preliminares que tienen su importancia para una buena comprensión de este tópico. Para bien entender los mecanismos y el funcionamiento de la sincronización del bus CAN, es necesario asimilar bien los elementos siguientes (Cf. [Bühring, 1991]).

Relojes internos (clock) de cada nodo.

Generalmente, cada nodo posee un oscilador interno que le es propio y que funciona permanentemente. Las frecuencias de reloj, *(clock)*, de estos osciladores permiten, después de una división, obtener la frecuencia y la duración correspondiente de la velocidad (única) elegida para la red CAN considerada, así como todos los parámetros de su nominal bit time individual, (revisar los párrafos precedentes) (Cf. [Bühring, 1991]).

Todas las frecuencias clock de estos nodos no tienen necesariamente - en la misma velocidad nominal teórica del CAN- exactamente los mismos valores. Esto es, debido al hecho de que los valores estrictos y tolerancias de los componentes, o sea debido entonces del hecho de que los relojes de inicio de los diferentes nodos no son rigurosamente idénticos (un nodo que funciona a 12 MHz, otro que funciona a 16 MHz, un último a 11.0592 MHz por algunas razones funcionales que les son propias) y las (pre) divisiones internas no dan estrictamente los mismos valores (Cf. [Bühring, 1991]).

Suponiendo que uno se las arregla para que todos los nodos posean el mismo valor de clock nominal, los mismos factores de división, etc., de nodo a nodo, los relojes locales no serán jamás estrictamente idénticos (tolerancias en los cuarzos o también derivas debidas al hecho de que cada nodo funciona con temperatura sensiblemente diferente) (Cf. [Bühring, 1991]).

Suponiendo incluso que todos los nodos puedan ser o sean puestos en la misma cadencia por una misma y única señal de reloj (clock), todas las señales de cadencia estarían síncronas pero no serían todas estrictamente síncronas por que ellas no tendrían necesariamente la misma fase temporal, debido a la distancia entre los nodos y

los fenómenos de propagación (Cf. [Bühring, 1991]).

En consecuencia a todo esto, cada nodo (vía el componente local que administra el protocolo) posee de forma interna todas las informaciones para elaborar su propio bit time. De hecho, cada circuito que posee su propio reloj interno (clock) es capaz - con el control del usuario y la ayuda de sus registros internos- de construir el corte (fase segmento 1) de su propio nominal bit time. De hecho, es deber del diseñador el haber definido o elegido estos (y sus) valores o haber aceptado aquellas propuestas dadas por definición por el fabricante del componente que administra el protocolo (Cf. [Bühring, 1991]).

Existirá siempre una primera puesta en tensión - un primer power on - de la red y en consecuencia los primeros power on reset de los componentes que constituyen las estaciones, que estarán acompañadas de sus inevitables secuencias de inicialización y, antes que el primer intercambio sobre la red no haya comenzado, un cierto lapso de tiempo transcurrirá durante el cual el bus estará en reposo (modo idle) (Cf. [Bühring, 1991]).

Con ayuda de todos estos elementos que ya han sido enunciados, puede ahora comenzar, con la nota final: ¿por que se quiere que cada nodo, del cual el reloj oscila de forma autónoma, construya desde su puesta en tensión, su *nominal bit time*?

De hecho, todos los predivisores internos están preprogramados y el nodo no espera más que la puesta en función de la red para sincronizar la inicialización de su primer *nominal bit time* sobre el primer flanco que va a presentarse (que normalmente será un flanco recesivo a dominante que señala el bit de Start Of Frame (SOF: inicio de trama) de toda la primer trama) (Cf. [Bühring, 1991]).

Enseguida, se verá ahora en detalle el mecanismo de sincronización.

3.7.2. Bit de sincronización (Synchronisation bit).

Problemas de sincronización del bit CAN.

De una forma general, cuando hay una transmisión de tipo asíncrono tal y como la que se utiliza para el CAN (ausencia de hilo dedicado a una señal permanente de reloj durante la transmisión) en el nivel de la estación receptora, la recuperación de un valor de un bit conlleva frecuentemente a problemas importantes debido a la ausencia de información de cadencia regular. Estos problemas entonces conllevan a muchas particularidades del protocolo CAN. La primera es que el bit CAN está codificado en NRZ y que es en principio, difícil de recuperar de las transiciones de un bit a otro cuando ellas están son algunas veces estructuralmente ausentes (Cf. [Bühring, 1991]).

¿Que hacer entonces para determinar si un bit de un mismo valor que el precedente esta en ese momento llegando? Una solución es inventar un reloj local, lo que ha sido hecho construyendo localmente en cada estación un nominal bit time. No hace falta entonces más que sincronizar el reloj local sobre los frentes de los bits incidentes (cuando están ellos presentes) y esperar que todo vaya bien enseguida. Es decir, esperar que no haya mucha deriva del reloj (debido a las tolerancias) entre la estación fuente del mensaje y la estación receptora. Sino, se podría producir catástrofes al tomar el valor de un bit por otro a causa de un cúmulo de retardos (Cf. [Bühring, 1991]).

Se podría pensar que una forma de resolver esto sería de resincronizar sobre el cúmulo de bits incidentes pero esto sería imposible, dado que puede ser que los numerosos bits tengan el mismo valor (binario y eléctrico).

Se puede pensar también en resincronizar de vez en cuando sobre los flancos presentes en el transcurso del intercambio e intentar de mantener el ritmo el tiempo más largo posible entre los relojes locales y el de la fuente emisora. Es en este nivel, que el protocolo CAN posee una segunda particularidad interesante.

Esta segunda particularidad proviene del principio de codificación adoptado (llamado de llenado de bits -*bit stuffing*) para asegurar la transmisión de los mensajes sobre el medio. Sobre este principio, cuando la trama mas grande CAN emitida, se recibe, solo una cantidad máxima de 5 bits consecutivos puede tener el mismo valor físico (eléctrico, óptico, etc. sobre el medio). Existe entonces una transición al menos todos los 5 bits. Esto ofrece la posibilidad de concebir un mecanismo mucho más fiable de resincroización de un bit unitario en el interior de un conjunto de bits durante una trama, liberándose de esta forma más fácilmente de las posibles derivas entre los relojes de emisión y de recepción (Cf. [Bühring, 1991]).

Nota: Para ser más precisos, el texto que describe el protocolo CAN indica que la distancia máxima entre dos transiciones para realizar correctamente la resincronización es de 29 bits (y no solamente de 5). De hecho, puede existir largas zonas de la trama CAN que no toman en cuenta los procedimientos de bit stuffing -citemos, por ejemplo, los finales de la tramas seguidas de la inter trama durante las cuales es necesario localizar más de 10 bits sucesivos idénticos (recesivos) seguido de la trama de sobrecarga. El valor de 29 bits permite tomar en cuenta la totalidad de los problemas de acumulación de errores y/o corrimientos de relojes, fases, etc. debidos a las tolerancias de los componentes.

3.7.3. Realización de la sincronización del bit CAN.

La lectura del protocolo CAN describe dos modos de sincronización, la sincronización por hardware y la resincronización. El análisis del texto impone a los fabricantes de los componentes realizar estos dispositivos de sincronización bajo la forma electrónica de máquina de estados. Estos circuitos tienen por misión la de comparar la posición del flanco antes de la señal incidente con el inicio del bit timing en ese momento procesado en la estación y readaptarlo a la necesidad de la duración de este último por un mecanismo de sincronización por hardware y/o de resincronización (Cf. [Bühring, 1991]).

Sincronización por hardware.
Se dice que hay un mecanismo de sincronización por hardware cuando el flanco antes (de la transición) del bit incidente se sitúa -o bien es forzado a situarse- en el interior del segmento de sincronización del *nominal bit time* que en ese momento se inicializa.

En una palabra, cuando se trata de un sincronización por hardware el nuevo *bit time* reinicia con el segmento de sincronización (synchronisation segment) (ver la Figura 3.19). De hecho esto es el equivalente de la sincronización directa de los osciloscopios de antaño (Cf. [Bühring, 1991]).

Bit de resincronización.
El bit de resincronización es un mecanismo que puede producirse durante la transmisión de un bit aislado o de una flotilla de bits de un mensaje para compensar:

- ya sea las variaciones de frecuencias individuales (instantáneas) de los controladores CAN presentes sobre la red,
- o bien, sea los cambios aparentes de duración de los nominal bit time que introducen las conmutaciones de las estaciones emisoras de una a otra (por ejemplo, durante las fases de arbitraje en donde los flancos no llegan estrictamente en el mismo instante debido al hecho de los tiempos de propagación o de las distancias diferentes de los candidatos que en ese momento son arbitrados).

Además, es posible o a veces deseable (por ejemplo cuando hay razones de funcionamiento que se llaman de tiempo real), de desear sincronizar la flotilla de bits reaccionando de forma dinámica, en todo instante, sobre los valores de los intervalos de tiempo phase_seg1 y phase_seg2 como se explicará enseguida (Cf. [Bühring, 1991]).

El modo de resincronización resalta de forma positiva las diferentes y nuevas especificaciones del CAN:

- los errores de fases y sus señales asociadas,
- los saltos de fase y las longitudes del salto de fase (Cf. [Bühring, 1991]).

A continuación se verá en detalle estas dos entidades.

Error de fase de la transición (o flanco) de una señal.

El error de fase (phase error), de un flanco que esta dada por la separación de la posición relativa que existe entre el flanco del bit incidente y la posición prevista del segmento de sincronización del bit time (Cf. [Bühring, 1991]) (ver Figura 3.19).

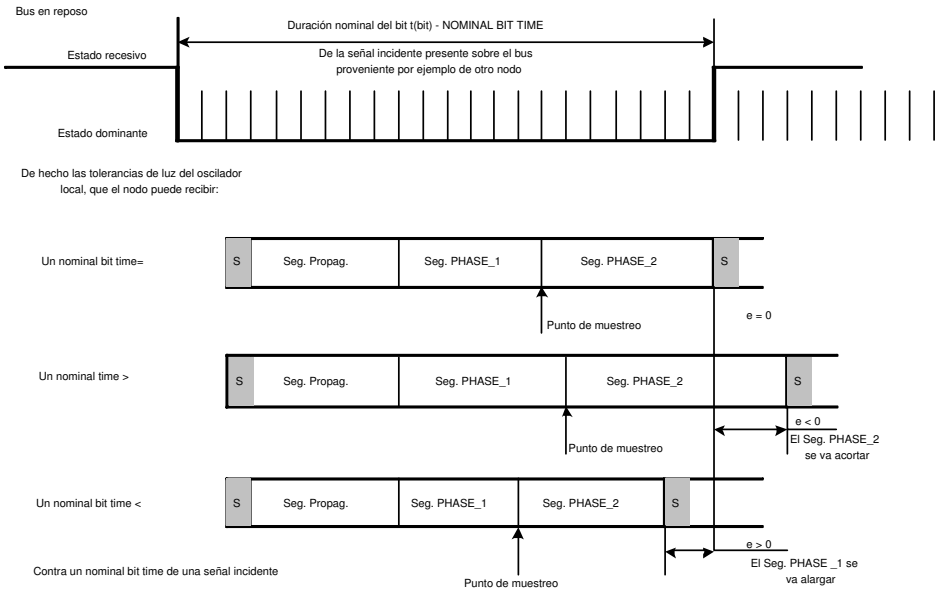

Figura 3.19: Error de fase

La medida del error de fase se caracteriza por:

- su intensidad (su valor), medido y expresado en unidades de time quanta,
- su señal, definida como sigue (ver la Figura 3.19):
  - e = 0 si el flanco se produce en el interior del segmento de sincronización,
  - e >0 si el flanco se produce antes del punto de muestreo del bit en proceso, es decir durante el segmento de propagación o el segmento de fase 1,
  - e <0 si el flanco tiene lugar después del punto de muestreo del bit precedente y antes del segmento de sincronización del bit que viene enseguida, es decir durante el segmento de fase 2 del bit precedente (Cf. [Bühring, 1991]).

Una vez que se han comprendido estas nociones, considere a continuación la forma en que se utilizan.

Salto de fase y longitud de salto de resincronización.

A fin de concebir tal dispositivo de resincronización de un bit (o de una flotilla de bits), es importante definir desde el inicio cual será su zona de validez y de acción imponiendo inmediatamente algunos topes.

El valor máximo de este tope se llama la longitud de salto de resincronización - resynchroni- sation jump width (RJW)- de la cual el valor, expresado en time quantum, puede ser programado entre el valor 1 y 4 (para el valor mínimo del intervalo) y el número de quanta del segmento de fase número 1 (Cf. [Bühring, 1991]).

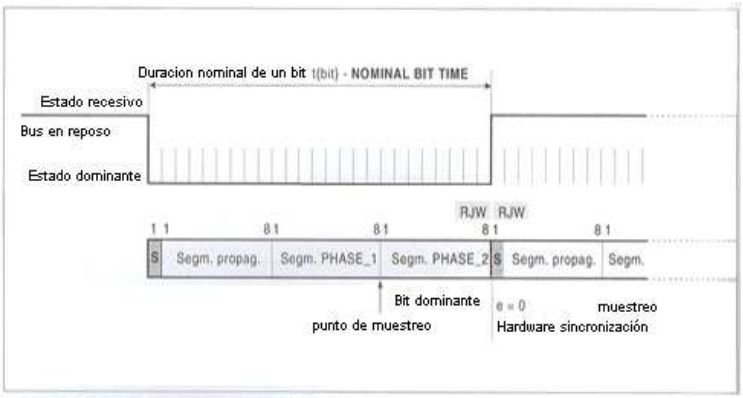

Figura 3.20: Efectos de la resincronización (parte A)

Los efectos de una resincronización están ligados a la relación entre amplitud del error de fase que provoca el fenómeno de resincronización y el valor programado de la *resynchronisation jump width*. Dos grandes categorías se distinguen (ver la Figura 3.20):

- | e | <= RJW; En este caso, aunque el valor e sea positivo o negativo (pero inferior o igual a RJW), el efecto producido será el mismo que aquel cuando hay una sincronización por hardware, (ver párrafo precedente).
- | e | >RJW;
  - si e>0 y e >RJW, entonces hay una resincronización del *nominal bit time* al prolongar el segmento de fase número 1 que es entonces aumentado en una cantidad igual al RJW,
  - si RJW<e, <0, hay una resincronización del *nominal bit time* por reducción del segmento de fase número 2 que es entonces disminuido en una cantidad igual al RJW (Cf. [Bühring, 1991]).

Como se acaba de demostrar, el efecto de la acción de la resincronización permite el componer de forma temporal el *punto de muestreo* en el momento más oportuno del bit incidente para medirlo con una mayor precisión.

Reglas de funcionamiento que tiene que ver con los dos modos de sincronización.

El protocolo CAN indica que los dos modos de sincronización hardware y de resincronización deben obedecer a las reglas siguientes:

- un solo tipo de sincronización es autorizado en el interior de un nominal bit time (sincronización por hardware o resincronización),

- cuando el bus esta en reposo (estado idle), todas la veces que una transición se produzca del estado recesivo al dominante, la sincronización del tipo hardware es efectuada; esto corresponde de hecho al inicio de una trama,

- una transición (flanco de la señal) será utilizada para asegurar la sincronización de esté, sí, y solamente sí, el valor leído sobre el bus inmediatamente después de la transición es diferente de aquel detectado en el punto de muestreo precedente (valor del bus precedentemente leído),

- todas las otras transiciones (de recesivo a dominante y eventualmente de dominante a recesivo en los casos de velocidades débiles) que reemplazan las condiciones de las reglas 1 y 2 serán utilizadas para la resincronización, con la excepción del caso en donde un nodo que transmite un bit dominante no efectué una resincronización como resultado de una transición de recesivo hacia dominante que conlleve a un error de fase positiva, si solamente las transiciones de recesivo a dominante son utilizadas para la resincronización (Cf. [Bühring, 1991]).

Esta última excepción asegura que los tiempos de retardo del *driver* de control y aquel de la entrada del comparador no conlleva a un aumento permanente (y/o acumulativo) del *bit time.*

Para completar y terminar las aplicaciones que tienen que ver con la utilidad de los segmentos de las fases 1 y 2, se verá ahora, con unas notas importantes, lo que tiene que ver con la incidencia de las derivas y tolerancias de las frecuencias de los relojes de las diferentes estaciones sobre la definición del valor del *nominal bit time* (Cf. [Bühring, 1991]).

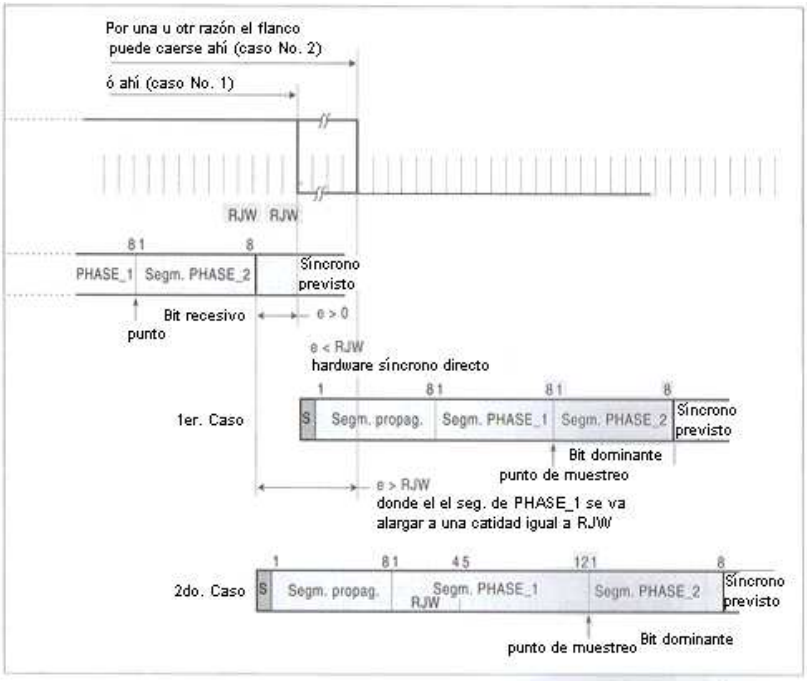

Figura 3.21: Efectos de la resincronización (parte B)

Incidencias de la precisión, derivas y tolerancias de las frecuencias de los osciladores.

Como seguramente ya habrá notado el lector durante estos últimos párrafos, la posición temporal del punto de muestreo es relativamente crítico si se desea tener una buena calidad de lectura del bit. Con esto se sobrentiende que si se desea asegurar una buen homogeneidad en la red, es necesario que todos los nodos dispongan de relojes (clock) de calidad y de tolerancia del mismo orden de magnitud, todos con la misma topología, velocidad y longitud (Cf. [Bühring, 1991]).

Es aquí cuando el diseñador da por hecho de que nadie es perfecto, los cuarzos y los resonadores poseen tolerancias, de derivas en temperatura y en el tiempo, y todo esto deben de tener en cuenta los participantes de la red para funcionar, incluso en el peor de los casos (Cf. [Bühring, 1991]).

Al englobar el conjunto de estos valores bajo el vocablo de variaciones del oscilador (oscillator drifts) - llamado **od** en lo que sigue, se puede directamente decir que en una red en la que todos los participantes están normalmente posicionados sobre una misma y única frecuencia de reloj (y por lo tanto sobre un mismo nominal bit time):

$$Tclkmin = Tnom(1 - od/100) < Tclknom < Tnom(1 + od/100) = Tclkmax \quad (3.9)$$

Se puede expresar también esto bajo la forma siguiente: para que una red funcione correctamente, hace falta que todos los participantes operen con los osciladores teniendo los mismos desempeños y tolerancias o bien muy cercanas (Cf. [Bühring, 1991]).

La conclusión es simple, emplear cuarzos. Pero esto conlleva al problema del costo de un nodo cuando la función que se supone asegurar debe ser simple, lo que es normalmente el caso en ciertos órganos pasivos de entrada/salida.

En este caso se desearía utilizar los resonadores cerámicos de los cuales, desafortunadamente, las tolerancias son más elevadas (del orden de 1.5). Para evitar este problema, algunas recomendaciones son propuestas a los fabricantes de los circuitos integrados en el protocolo CAN para de esta forma liberarse de estas tolerancias más importantes. Al respetar estas últimas, llega a ser posible hacer funcionar correctamente el bus con una velocidad de 125 Kb/s y utilizar los osciladores que tienen como tolerancias de alrededor de 1.58 % (Cf. [Bühring, 1991]).

A forma de conclusión, se debe de indicar al lector las importantes recomendaciones de uso:

- sí, en una red, se debe utilizar diversos administradores de protocolo de los cuales no se sabe si satisfacen o no las últimas recomendaciones, hay solamente una salida: utilizar osciladores de cuarzo,
- el nodo (la estación que posee las restricciones menos severas para su oscilador determina la precisión requerida para todos los otros nodos),
- los resonadores cerámicos pueden ser utilizados solamente si todos los demás nodos de la red utilizan el protocolo reforzado (enhanced) (Cf. [Bühring, 1991]).

3.8. Estimación rápida de RJW vía la tolerancia del oscilador.

El protocolo CAN especifíca que el sistema debe de ser capaz de resincronizarse en la presencia de un máximo de 29 bits sucesivos recesivos. Esto quiere decir que las derivas (ó corrimientos) acumulativos de frecuencia de los osciladores entre el emisor del mensaje y todas las derivas de los otros participantes de la red, deben ser calculados de forma tal que el punto de muestreo esté siempre en su lugar con respecto a la señal incidente y que el valor del bit detectado, leído, o medido, sea siempre bueno hasta el último bit, tomando incluso el caso más desfavorable (al tomar en cuenta para un nodo el Tclk min y para el otro nodo el Tclk max) (Cf. [Bühring, 1991]).

Para estimar el valor de RJW que debe ser cargado en los registros del microcontrolador, es necesario definir algunos valores suplementarios.

De inicio, hay que saber que el valor de RJW influye de la misma forma en los dos segmentos de fase 1 y 2 y que su misión es la de compensar las tolerancias de los diferentes relojes. Se puede entonces describir de forma estadística, para un bit, que:

$$T\_bitx(X/100) = RJW \tag{3.10}$$

En esta ecuación RJW es el valor de la longitud del salto de fase de resincronización, T_bit es el nominal bit time, y X representa la tolerancia máxima del oscilador de un nodo (Cf. [Bühring, 1991]).

De hecho, si se pretende en el caso más desfavorable (worst case), es necesario de tomar no solamente esté valor si no el doble de esté, el reloj de una estación que puede estar instantáneamente en (T_clock - X/100) y el reloj de otra en (T_clock + X/100), lo que da para la duración de un bit:

$$2[T\_bitx(X/100)] = RJW\_max \tag{3.11}$$

Sin embargo, hay que tener en cuenta la eventualidad de la presencia de 29 sucesivos del mismo valor. Esto da, al tener en cuenta el efecto acumulativo posible de los retardos:

$$29x2[T\_bitx(X/100)] = RJW\_max\_max \tag{3.12}$$

Es entonces este último valor que hay que cargar en el nominal bit time para esperar que la red funcione siempre correctamente (Cf. [Bühring, 1991]).

Por otro lado, sabemos que la estructura del nominal bit time será siempre de la forma:

$$T\_bit = (1tq + seg\_T\_prop + seg\_phase\_1 + seg\_phase2) \tag{3.13}$$

Sabemos también que, según el valor y la señal de error de fase en un instante dado, el segmento de fase 1 puede extenderse o bien el segmento de fase 2 puede reducirse (pero no los dos al mismo tiempo).

Sobre este principio, el segmento de fase 1 podría tener un valor nulo (mínimo de 1 tq obligatorio) dado que no se puede alargar. Por el contrario, el segmento de fase 2 no pudiéndose más que disminuir, debe ser al menos ser igual al valor de RJW_max_max (Cf. [Bühring, 1991]).

Se puede entonces estimar el valor de T_bit_min:

$$T\_bit\_min = [1tq + seg\_T\_prop + 1tq + (RJW\_max\_max)] \tag{3.14}$$

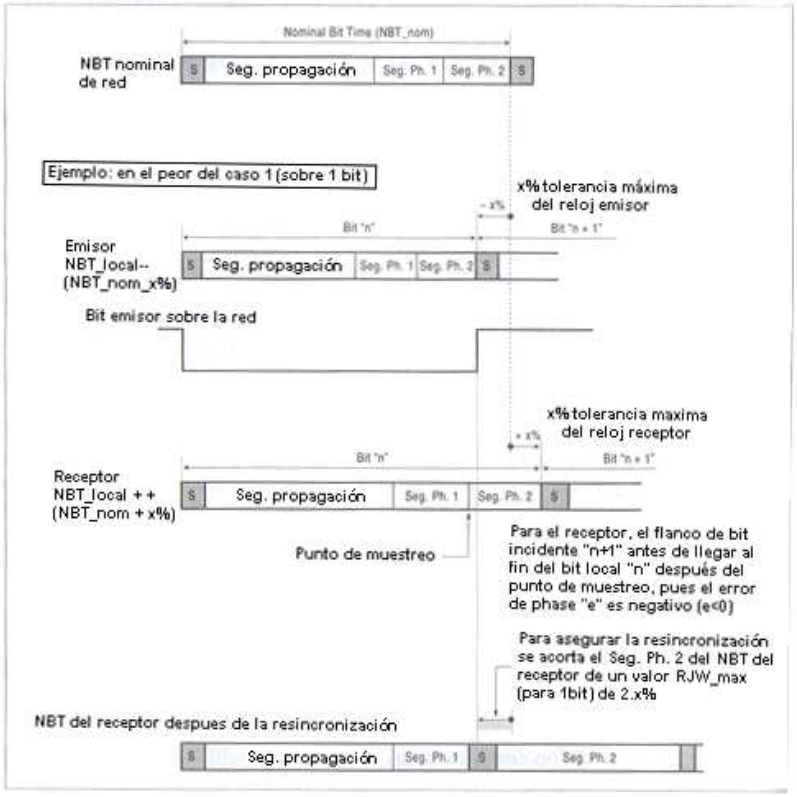

Figura 3.22: Mecanismo de resincronización (parte A)

Con, tq = time quantum (el segmento de sincronización que debe ser siempre igual al mínimo de 1 time quantum).

Relacionando con el valor de T_bit_min en la ecuación precedente:

$$[58x(X/100)x(2tq + seg\_T\_prop)] = RJW\_max\_max[(1 - (58x(X/100))] \quad (3.15)$$

Identificador en CAN 2.0A, sobre 11 bits			
Partie ID_Haute Sujets		Partie ID_basse Sources	
Sur 6 bits (64 valeurs) (64 paramétres)		sur 5 bits (32 valeurs) (32 noeuds)	
Sujeto principal	Sous - sujet	Source principale	Sous - source
Temperatura Sur 4 bits	Exterieure Sur 2 bits	D nord – est Sur 3 bits	3°étage sur 2 bits

Figura 3.23: Mecanismo de resincronización (parte B)

El valor del segmento de propagación que ha sido precedentemente estimado en función del medio, de los componentes de interconexión y de la calidad de la señal, se obtiene el valor de RJW_max_max a ser tomado en cuenta para calcular el nominal bit time [6] . (Cf. [Bühring, 1991])

La Figura 3.22 resume el mecanismo de resincronización descrito a lo largo de los párrafos anteriores.

3.9. Velocidad de la red.

3.9.1. Velocidad binaria (bit rate).

El protocolo CAN indica claramente que para una red específica la velocidad debe ser fija y uniforme (Cf. [Bühring, 1991]).

Velocidad bruta.

En el nivel del bit de transmisión, la norma define el termino *nominal bit rate* como el número de bits transmitidos por segundo por un emisor ideal (en la ausencia del

[6] La ecuación precedente no es valida más que cuando [1 - (58 x (X/100))]>0,es decir cuando X es inferior alrededor de 1.7. En conclusión, la suma de las tolerancias que soporta el CAN en lo que tiene que ver con los relojes es entonces de alrededor de 1.7 % a lo máximo.

mecanismo de resincronización) (Cf. [Bühring, 1991]).

Esta velocidad, bruta, puede extenderse desde algunos Kb/s a 1 Mb/s.

Nota: formalmente hay dos clases de velocidades que son igualmente normalizadas con el ISO:

- el modo llamado low speed en la cual la velocidad máxima está limitada hasta 125 Kb/s,
- el modo llamado high speed, de 125 Kb/s a 1 Mb/s a lo más.

Velocidad neta.

Para estimar la velocidad neta -cantidad de bits útiles transportados por segundo- es necesario proceder con el inventario del contenido completo de una trama (de datos, por ejemplo) teniendo en cuenta las particularidades tanto a nivel del número de bytes que transporta así como el número de bits rellenado (stuff bits) asociados que ella contiene (Cf. [Bühring, 1991]).

El protocolo CAN que soporta las versiones 2.0 A y B, nos da precisamente estos dos tipos de tramas como se puede ver a continuación.

Estimación en número de bits de una trama estándar (2.0 A), en el orden (Cf. [Bühring, 1991]).

Estimación en número de bits de una trama extendida (2.0 B).

Retirando lo superficial, ¿qué queda?

Retomemos por ejemplo el caso de la trama extendida (CAN 2.0 B). Para un bit rate (velocidad bruta) de 1 Mb/s (1 bit = 1 s), la trama extendida dura a lo más 154 s para 64 s máximo de informaciones útiles, ósea alrededor de una velocidad neta de 64 /154= 0.415 para transportar 8 bytes y de los cuales los valores de los bits de identificadores, de los datos, etc., han sido estudiados para que el número de bits de stuff sea máximo (Cf. [Bühring, 1991]).

Debido que hemos asociado 154 bits a 1 Mb/s a los datos útiles del mensaje, ellos

1 bit de inicio- SOF
11 bits de identificación
1 bit RTR
6 bits de control
De 0 a 64 bits de datos (a lo más)
15 bits de CRC
1 bit de CRC delimitador
19 bits de stuff (a lo más, en el caso más desfavorable)
1 bit de ACK slot
1 bit de ACK delimitador
7 bits de end of frame
3 bits de inter frame space
Ósea 130 bits a lo más (fuera del overload frame).

Tabla 3.1: Trama estándar

1 bit de inicio- SOF
11 bits de identificación
1 bit SRR
1 bit de IDE
18 bits de identificación
1 bit de RTR
6 bits de control
de 0 a 64 bits de datos (a lo más)
15 bits de CRC
1 bit de CRC delimitador
23 bits de *stuff* (a lo máximo en el caso más desfavorable)
1 bit de ACK slot
1 bit de ACK delimitador
7 bits de end of frame
3 bits de inter frame space
Ósea 154 bits a lo máximo (fuera del overload frame)

Tabla 3.2: Trama Extendida

serán transportados a una velocidad neta de 415 Kb/s [7] . Este cálculo, un poco simplista, tiene el mérito de dar rápidamente un orden de magnitud de la velocidad neta, sin embargo hace falta precisarlo cuando existen aplicaciones concretas, teniendo en cuenta el número real de los datos transportados por tramas (el identificador que hace parte integral de los datos transportados, de la perdidas de arbitraje, y de los errores posibles) (Cf. [Bühring, 1991]).

Los detalles de estos cálculos salen del cuadro de esta monografía y damos en la tabla de la Figura 3.24 una mejor idea de las velocidades obtenidas con respecto a la función de los números de bytes transportados al funcionar a 1 Mb/s y haciendo de nuevo algunas abstracciones menores en la influencia de sus contenidos sobre la presencia o no de bits de stuff (Cf. [Bühring, 1991]).

Débito neto del CAN (en kilo bits por segundo)
Para un débito bruto de 1Mb/s, y en suma para un mensaje (un objeto CAN) el número de bits útiles igual a: * para el CAN 2.0A : 11(D) + 1 (RTR) + 4 (DLC) + número de bits de datos transmitidos * para el CAN 2.0B : 29(D) + 1 (RTR) + 4 (DLC) + número de bits de datos transmitidos Ejemplo: La trama completa CAN 2.0A (y comprendidos del bit stuff) contiene cerca de 138 bits para 8 bytes de datos ó sea, un débito neto de: (11 + 1 + 4 + 64) / 138 x 1 Mb/s = 579 Kb/s

Débito neto (en kilo bits por segundo)

Número de bytes transmitidos para trama CAN	Trama estándar CAN 2.0A	Trama extendida CAN 2.0B
1	72.1	61.1
2	144.1	122.1
3	216.2	183.2
4	288.3	244.3
5	360.4	305.3
6	432.4	366.4
7	504.5	427.5
8	576.6	488.5

Figura 3.24: Débito neto del CAN

[7] para ser más rigurosos, hay que tener en cuenta el identificador (11 ó 29 bits) del RTR (1 bit) y el DLC (4 bits) que también forman parte del mensaje y dan por lo tanto un rendimiento elevado.

3.10. Valor minimal de la relación (velocidad neta/-velocidad bruta).

La Figura 3.25 nos recuerda una vez más el principio de bit stuffing presentándose en un mensaje ordinario en donde, en todos los 5 bits del mismo valor, se introduce un bit de llenado.

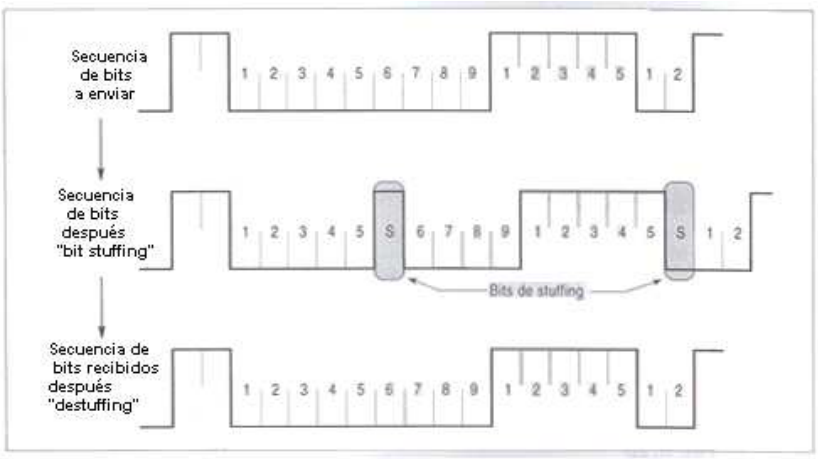

Figura 3.25: Bit Stuffing

La Figura 3.26 indica la zona de una trama de datos (CAN 2.0 A) en la cual pueden existir los bits de llenado (Cf. [Bühring, 1991]).

Figura 3.26: Zona de una trama de datos

La primera nota que se puede hacer sobre la señal que comporta el máximo de bits de llenado nos recuerda sobre un mensaje de la forma de la Figura 3.27 que da una relación de 5:1.

Sin embargo este no es el peor de los casos. El mensaje el más particular es representado en la Figura 3.28 la cual, con la excepción de los 5 primeros bits, el mensaje

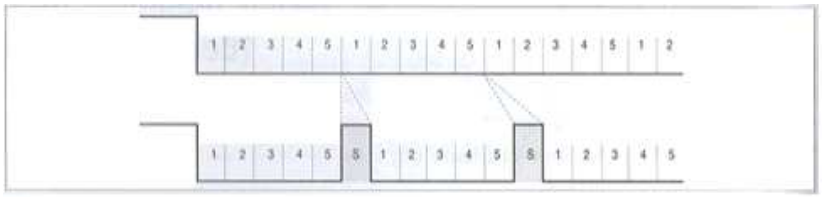

Figura 3.27: Relación 5:1

original comporta las sucesiones de 4 bits idénticos. De hecho, esto obliga a fabricar frecuentemente los bits de llenado, a cada secuencia de 4 bits útiles, del hecho de la introducción del primer bit de llenado. La relación se convierte entonces de 5:1 a 4:1 (Cf. [Bühring, 1991]).

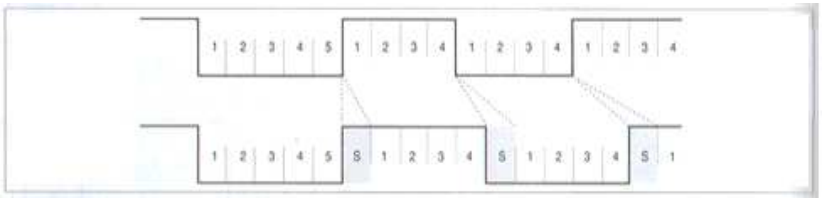

Figura 3.28: Relación 4:1

Hay que prestar atención entonces a los cálculos en la elaboración de los mensajes, los cuales se requieren que sean rápidos. Para la información, el mensaje que contiene el máximo de los bits de llenado, tiene en cuenta de bit RTR, el de DLC, el cálculo de CRC, etc., el cual es dado en la Figura 3.29 (Cf. [Bühring, 1991]).

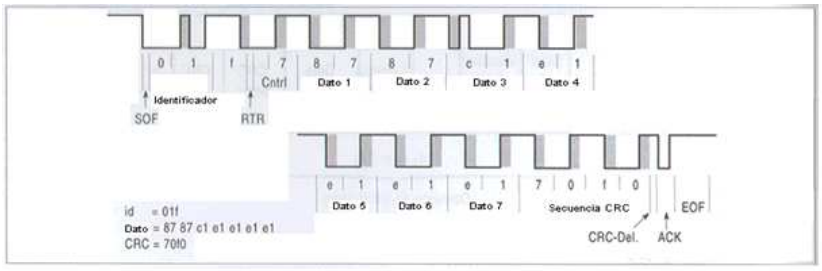

Figura 3.29: Cálculo de CRC

Después de este intermedio, se pasará a las consideraciones que tienen que ver con las topologías y tipos de medio que se pueden utilizar para transportar las tramas CAN.

3.11. Tiempo de latencia.

El tiempo de latencia es el intervalo de tiempo que existe entre el instante de inicialización (transmission request) y el inicio real de la transmisión sobre el bus (todos los parámetros incluidos). Su valor depende de una gran variedad de condiciones presentes sobre el bus, los mensajes en curso de intercambio (número de bytes de datos transmitidos, presencia o no de trama de error) (Cf. [Bühring, 1991]).

Examine ahora el caso más desfavorable que puede producirse e intente descifrar el valor de tiempo que se debería esperar para acceder a la red en el caso en donde este no esté perturbado (ausencia de trama de error) (Cf. [Bühring, 1991]).

En el protocolo CAN, la longitud del mensaje puede ser expresado de la forma siguiente.

Tipo de trama	**Estándar CAN 2.0A**	**Extendido CAN 2.0B**
Número de bits de las tramas sometidas en el bit de llenado n. Es decir	1+11+1+6+8x(DLC)+15 34+8x(DLC)	1+11+1+1+18+1+6+ 8x(DLC)+15 54+8x(DLC)
Número de bits bit stuffing Es decir	1+1+1+7 10	1+1+1+7 10
Número de bits bit stuffing	3	3
Longitud total de la trama CAN		
sin presencia de bits de llenado o sea para 8 octetos de datos Es decir	34+8x(DLC)+10+3 111+3 114	54+8x(DLC)+10+3 128+3 131
En presencia de bits de llenado. O sea para 8 octetos de transporte Es decir	(34+8x8-1)/4 24	(54+8x8-1)/4 29
Longitud máxima de la trama		
Para 8 octetos de datos Es decir	114+24 138	131+29 160

Tabla 3.3: Longitud de un mensaje

En este caso (caso extremo, el más desfavorable y fuertemente irrealista debido a que este mensaje tan particular que contiene un máximo *maximorum* de bits de llenado es ahora totalmente improbable, puesto que como se sabe puede existir y hay que evitarlo), el tiempo de latencia máximos garantizados por el protocolo para el identificador de más alta prioridad sería respectivamente de 137 y 159 bit times para el CAN 2.0 A y 2.0 B respectivamente (Cf. [Bühring, 1991]).

Nota: como habrá seguramente notado este valor comprende un bit time menor que la longitud máxima de la trama la más larga que puede existir (estadísticamente) durante la transmisión. En efecto, no teniendo verdaderamente oportunidad, se hubiera deseado iniciar un intercambio justo después del primer bit de start of frame del mensaje durante la transmisión (por lo tanto un bit de menos) y en donde se está en un estado de latencia. Este estado no hubiera existido si se hubiera iniciado el mensaje estrictamente en el mismo tiempo que el mensaje durante este tiempo debido a que, en principio, se hubiera ganado el arbitraje del bus. Esto es sorprendente, dado que en el caso de cifrado de un tiempo de latencia, se supone implícitamente que el mensaje que se desea transmitir posee, en esa zona de tiempo de funcionamiento del sistema, una prioridad superior al conjunto de todos los mensajes que pueden ser transmitidos. Para el cálculo completo de los tiempos de latencia de una red, es necesario aportar un juicio estadístico al conjunto del sistema, lo que sale del marco de trabajo de esta monografía sobre el CAN.

3.12. Conclusión y resumen.

Finalmente, se concluye con las partes más importantes que describen el bit CAN, es decir, los *nominal bit time*, los mecanismos de resincronización y las velocidades. El conocimiento y el dominio de todas estas entidades son fundamentales para definir el conjunto tipo de medio/longitud máxima/ velocidad máxima y las tolerancias de los componentes propios a una red determinada.

Las propiedades, calidades y compromisos que llevan a estos temas se reducen con la definición (ó elección) de la duración del nominal bit time y con las elecciones de los valores relativos de los cortes específicos de los diferentes segmentos de tiempo que lo componen.

Tratando de ser concisos (ver Figura 3.30):

- la elección de un tipo de medio implica el conocimiento del valor de su velocidad de propagación de la señal,
- la longitud deseada del medio y los desempeños temporales de los componentes de acoplamiento al medio permiten obtener el valor temporal mínimo necesario en el segmento de tiempo de propagación del nominal bit time (alrededor de 2 veces el tiempo para ir de un extremo al otro de la red),

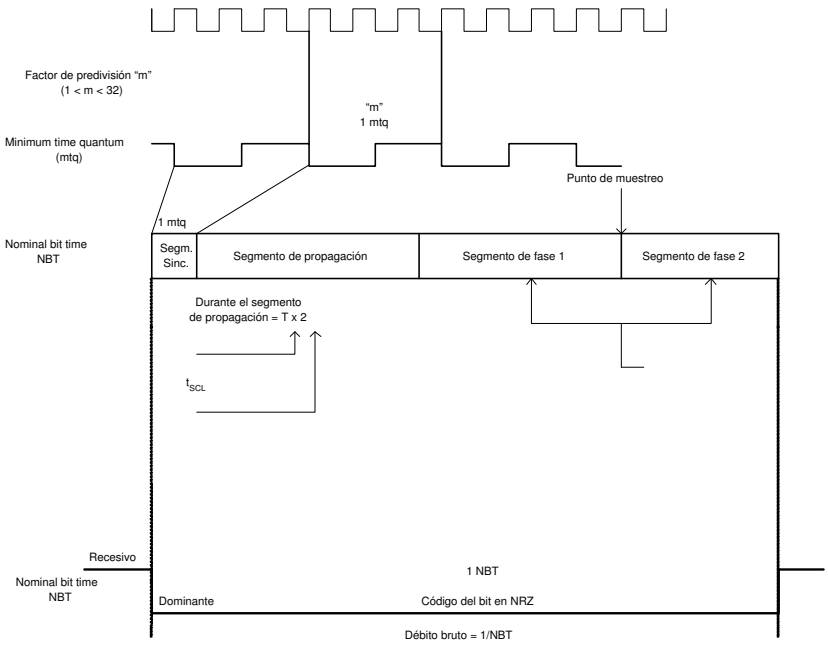

Figura 3.30: Duración de los diferentes segmentos de tiempo

- las tolerancias y derivas de los osciladores de los diferentes participantes permiten estimar los valores máximos de las longitudes de salto de resincronización (RJW), y en consecuencia, los valores mínimos de los segmentos de fase 1 y 2 de cada uno de estos participantes,
- el conjunto de los valores mínimos segmento de propagación aumentado de los segmentos de fase 1 y segmentos de fase 2 permiten obtener una primera idea de la duración mínima del *nominal bit time*, por lo tanto también de la velocidad bruta máxima que sería apta a soportar teóricamente la red CAN proyectada (fuera de todas las alteraciones de la señal debido al medio utilizado y llevándolo de esta forma a tener un funcionamiento regular sino es que mediocre,
- en función del tipo de mensajería deseada (número de bytes transportados por trama, etc.), la velocidad bruta que permite estimar la velocidad neta (media y máxima) y en consecuencia, los tiempos de latencia máxima de un mensaje en los cuales la prioridad (elección de los identificadores está determinada) (Cf. [Bühring, 1991]).

Ejemplo de Síntesis de este capítulo

Frecuencia nominal del Reloj de un nodoF_clock_nom = 10 MHz
O sea un Periodo nominal deT_clock_nom = 100 ns
Tolerancia máxima de de los relojes de la red % _F_clock = 0.5 %
O sea un error de periodo posible de % _T_clock = 0.5ns
Mínimo time quantum de un nodo= 100ns
Valor elegido del predivisor m (1<m<32) para el nodo considerado m = 10
Time quantum de la redtq = (100) x 10= 1000ns = 1μs
Medio elegido para la aplicación planeada = por hilos, par diferencial
Velocidad de propagación del mediov_prop = 200000 km/s
Longitud deseada de la redL = 600m
tiempos de propagación del medioT_prop_med = 3μs
Suma de tiempos eléctricos (T_sd,T_tx,T_rx,T_qual_sign) T_elec = 100ns = 0.1μs
Valor mínimo necesario del segmento de tiempos de propagación= 2x(3+0.1)=6.2μs
Número de time quantums necesarios 1<t-q-m<8 = 6.2/1=6.2 o sea 7 quanta
Valor considerado del tiempo de propagación seg_T_prop = 7μs
Valor mínimo de RJW_max_max (ver formula en el texto) RJW_max_max = 3.39μs
Número de quanta considerados para el segmento de fase 2 seg_phase_2 = 4 mínimos, 4 considerados, o sea 4μs
Número de quantas considerados para el segmento de fase 1 seg_phase_1 = 1 mínimo, 4 considerados, o sea 4μs
Duración del *nominal bit time* (1tq+seg_T_prop+seg_1+seg_2) = (1+7+4+4)=16μs
velocidad neta de la red dada las condiciones anteriores= 62.5kb/s

Tabla 3.4: Ejemplo de un caso de estudio particular

Capítulo 4

Prototipo con el bus CAN para un sensor de presión

Introducción

En este capítulo se analizará brevemente una prototipo con el bus CAN en la construcción de un sensor de presión usando el microcontrolador de la familia microchip PIC16F876 y el controlador CAN MCP2510.

En este prototipo se mide la presión por medio del sensor MPX2100AP (Motorola) que nos entrega una señal analógica de entre 0 y 40 mV (0 a 100 Kpa). Esta señal es convertida a una señal digital por medio del integrado MCP3201, que a su vez envía a través de un puerto de interfaz de periférico serial (SPI por sus siglas en inglés) el cual esta disponible en el microcontrolador PIC16F876. El microcontrolador PIC16F876 lee en consecuencia del sensor de presión, la variable de presión presente en ese instante de tiempo cuando se le requiere a través del bus CAN formado principalmente por el controlador CAN (MCP2510) y el manejador CAN (MCP2551). Para aprovechar otras líneas disponibles del PIC16F876, se lee también la configuración de algunos interruptores de entrada. Este prototipo puede encontrarse completo en el manual de Microchip DS00212B (Cf. [Technology, 2002]).

4.1. Sensor de presión CAN usando el MCP2510 y PIC16F876.

Introducción.

Los avances en las comunicaciones de datos han creado eficientes métodos para varios dispositivos de comunicaciones sobre un número mínimo de sistemas de cableado. La red de controladores de área (CAN) es uno de estos métodos. CAN envía y recibe

mensajes sobre dos alambres (bus CAN). Los nodos emiten sus mensajes individuales sobre el bus CAN, mientras los receptores aceptan el mensaje y anticipadamente un reconocimiento (ACK) de la señal indicando la recepción o no del mensaje. El protocolo de CAN utiliza como se ha visto anteriormente dos estados y los bits son cualquiera de estos dos, dominante (lógico 0) ó recesivo (lógico 1). Los nodos pueden intentar la transmisión de un mensaje al mismo tiempo. Para asegurar la reducción de estas colisiones a lo largo del bus, se utiliza el plan de arbitraje en el cual un nodo continua la transmisión hasta que un bit dominante es detectado, mientras este nodo está esperando un bit recesivo (en el campo ID) sobre el bus CAN. El nodo pierde el arbitraje y terminará su transmisión automáticamente y cambia al modo de receptor. Una vez que el bus CAN entra en modo idle (parado), estos nodos intentan la retransmisión. Si el nodo no pierde el arbitraje, la transmisión es completada (Cf. [Technology, 2002]).

La configuración del bus opera por el principio de multi-maestro, este permite conectar varios nodos directamente al bus. Si un nodo es suspendido en el sistema, el otro nodo no es afectado. La probabilidad de que la red entera esté suspendida es extremadamente baja comparado con la redes de tipo anillo. Las redes de tipo anillo tienen una alta probabilidad de falla en la velocidad de transmisión, debido a que si un nodo funciona mal toda la red llega a ser inoperante. El controlador CAN es la solución a este problema (Cf. [Technology, 2002]).

4.1.1. Beneficios del controlador CAN MCP2510.

- Monitoreo de varios dispositivos.
- Programación individual de un nodo.
- Sustitución de largas instalaciones de alambrado.

4.1.2. Revisión del módulo.

El módulo hardware puede ser dividido en 2 componentes:

- Tarjeta de nodo CAN-NET.
- Tarjeta de entrada analógica CAN-NET.

Estas tarjetas pueden ser directamente adquiridas a través de la empresa Diversified Engineering [Dave Ross, 2005], ordenando el kit de tarjeta de entrada analógica CAN-NET. La tarjeta CAN-NET de entrada analógica también requiere de algunas opciones instaladas por el cliente. Dos componentes adicionales son: un 14.5-PSI traductor de presión y un LED. La Tabla 4.1 muestra los números de parte para estos componentes.

Constructor	Componente	Número de parte
Diversified Engineering	Kit tarjeta CAN-NET de entrada analógica.	905190
Motorola	Traductor de presión.	MPX2010DP

Tabla 4.1: Números de parte de los Componentes

Este módulo tiene varias características. Estas incluyen:

- Interfaz SPI de alta velocidad.
- Herramienta de sustitución MPLAB ICD.
- Tecnología de baja potencia CMOS.
- Salida PWM para manejo de una lámpara.
- Soportes SPI en modos 0,0 y 1,1.

Tarjetas del nodo CAN-NET.

La tarjeta del nodo CAN-NET consiste de dispositivos de hardware que son usados en conjunción con técnicas de programación del software para una óptima red de controladores de área. La versatilidad del controlador CAN permite una amplia variedad de aplicaciones que pueden ser creadas bajo el presente concepto utilizando este diseño en particular.

El controlador CAN MCP2510 es el corazón de la interfaz CAN. Este se encarga de toda la transmisión y recepción de los paquetes de mensajes que contienen información útil para los otros nodos sobre la red vía el bus CAN. El controlador CAN MCP2510 está también diseñado para ser una interfaz con el puerto de interfaz del periférico serial (SPI). El puerto SPI está disponible en el microcontrolador PIC16F876 y el convertidor analógico / digital MCP3201 (ADC).

El controlador PIC16F876 almacena el programa en la memoria y las configuraciones del switch DIP envío y recepción de mensajes. Este controla la salida del PWM y al MPLAB ICD ser usado como una herramienta de sustitución.

Tarjeta de entrada analógica CAN-NET.

EL MCP3201 ADC acepta las señales de entrada del sensor de presión, utilizando una configuración de amplificador diferencial. El amplificador MCP602 utiliza una tecnología de amplificador operacional CMOS unitario (un solo amplificador en el encapsulado).

4.1.3. Revisión del hardware.

Esta sección describe el hardware de la tarjeta nodo CAN-NET y como funciona el CAN en el sistema de este nodo.

Controlador CAN MCP2510.

El diagrama de flujo de este sistema se muestra en la Figura 4.1, el concepto básico es habilitar al controlador CAN MCP2510, al microcontrolador PIC16F876 y al convertidor ADC MCP3201, para comunicarse entre ellos de forma eficiente utilizando el SPI. El MCP2510 maneja los protocolos de bajo nivel.

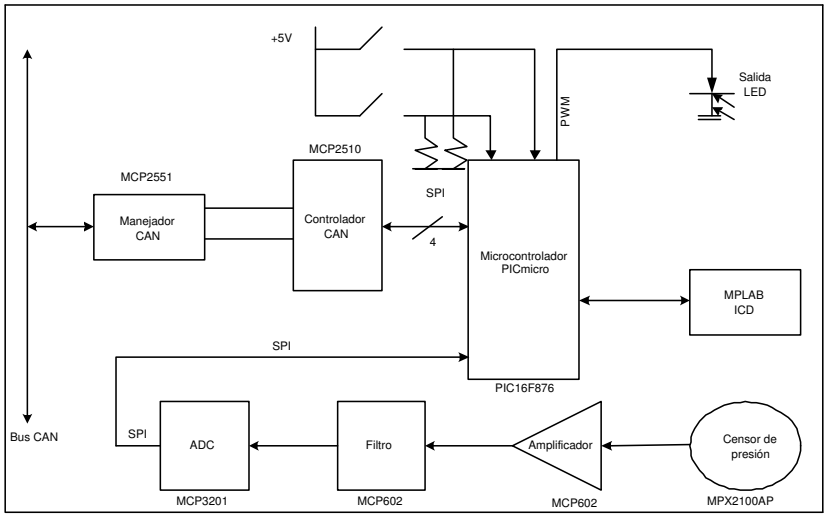

Figura 4.1: Diagrama de bloques de la tarjeta del nodo CAN

El microcontrolador PIC16F876 almacena el programa en memoria y constantemente lee del convertidor MCP3201ADC, junto con la referencia A/D.

En el lazo principal del programa, una variable es conmutada de la forma siguiente. Cuando el valor de la variable es un "0" lógico, el dispositivo PICmicro lee del sensor la presión y una vez que el valor de la variable es un "1" lógico, el dispositivo PICmicro lee

la referencia A/D. El microcontrolador también lee las configuraciones de los interruptores de entrada. Los 2 primeros interruptores (de 4) le dicen al microcontrolador cual mensaje y al nodo que está disponible para ser recibido. Los últimos 2 interruptores (de 4) le dicen al microcontrolador la dirección a transmitir del nodo. La configuración que se muestra en la Figura 4.2 ilustra las conexiones de los tres nodos en el bus CAN puestos para transmitir y recibir ciertos mensajes.

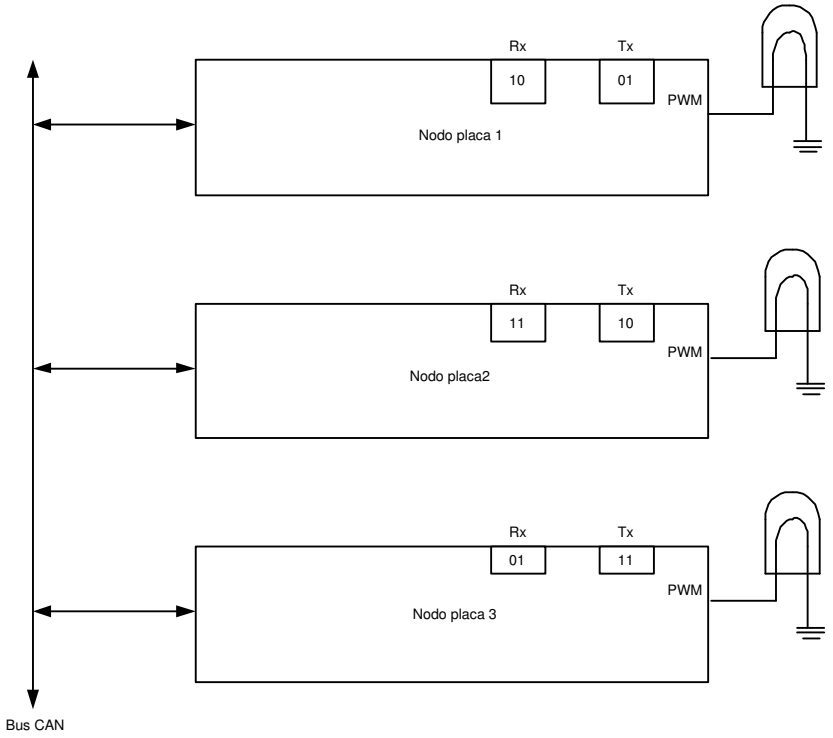

Figura 4.2: Tres tarjetas conectadas al bus CAN

En este caso, cada nodo transmite su propio valor del sensor de presión y es puesto para recibir un valor del sensor de presión de un nodo diferente. La identificación para cada conexión del nodo es 01, 10 y 11 estas configuraciones son los identificadores para transmisión y recepción de identificadores. La conexión del nodo 1 es puesta para recibir el valor del sensor de presión de la conexión del nodo 2. La conexión del nodo 2 es puesta para recibir el valor del sensor de presión de la conexión 3, mientras que la conexión del nodo 3 es puesta para recibir el valor del sensor de presión de la tabla del nodo 1. El valor del sensor de presión para la conexión del nodo es directamente proporcional a la salida PWM del microcontrolador correspondiente.

El manejador del chip CAN convierte la entrada y salida de los voltajes del bus CAN que van de 0 a 5 volts, a un cambio de $\pm$ 12V.

El MCP3201 es un convertidor ADC de 12 bits, con una circuitería de muestreador y retenedor integrados. La entrada al dispositivo viene de un circuito de amplificación diferencial que se comunica con la interfaz de serie usando el protocolo SPI. El MCP602 se usa para diseñar un amplificador diferencial adecuado.

La ganancia del amplificador es determinada por la ecuación siguiente:

$$Ganancia = 1 + (R14/R13) + 2(R13/RP1) \tag{4.1}$$

La Figura 4.3 muestra el circuito del amplificador diferencial. La entrada a este amplificador va desde 0 a 5 volts y es útil para las aplicaciones de presión. La presión puede ser llamada como *presión cero*. La configuración de presión normalizada consiste en presión negativa y presión positiva. El sensor de presión produce un voltaje negativo cuando hay una presión negativa y un voltaje positivo cuando hay una presión positiva. La referencia para el amplificador diferencial es de 2.5 volts. Arriba de 2.5 volts, este indica una presión positiva. Debajo de 2.5 volts, este indica una presión negativa. La tarjeta del nodo CAN-NET con la conexión de entrada y salida analógica esta diseñada específicamente para la presión, pero puede alterarse fácilmente para realizar ambos.

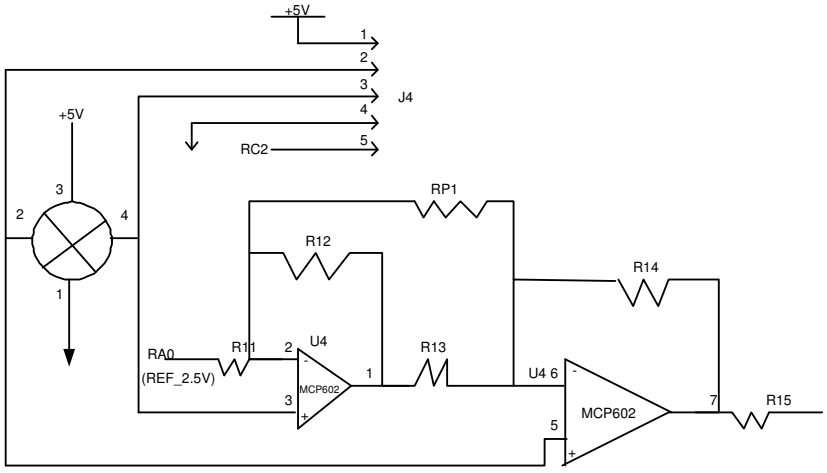

Figura 4.3: Circuito del amplificador diferencial

4.1.4. Herramientas de hardware.

El MPLAB IDC es una herramienta la cual ayuda al proceso de depuración y desarrollo de código y hardware. La depuración usa un dispositivo PIC16F876 y opera en *tiempo real*. Esta herramienta de bajo costo ahorra tiempo de diseño y dinero permitiendo evaluar el programa de aplicación del circuito en tiempo real. La interfaz

Componente	Valor	Tolerancia
R11	30.1 KΩ	1 %
R12	10.0 KΩ	1 %
R13	10.0 KΩ	1 %
R14	30.1 KΩ	1 %
R15	1.0 KΩ	1 %
RP1	0.0 - 50.0 KΩ	NA

Tabla 4.2: Valores de los componentes del Circuito Amplificador Diferencial

ICD también permite a los dispositivos PIC16F87X ser programados después de que la tarjeta ha sido fabricada. Esto permite al software hacer actualizaciones para ser programados en el dispositivo. El ICD usa los pines RB6 y RB7 del PIC16F87X para esto. Por esta razón, estos pines no son usados para ningún otro propósito en este sistema.

4.1.5. Revisión del software.

Estilo de programación.

El código para la tarjeta del nodo está escrito en el conjunto de instrucciones del dispositivo PICmicro para ser ensamblado usando el medio ambiente de microchips MPLAB. Hay un uso significativo de macros para hacer el código más fácil de leer y con menos errores. Las macros son definidas en tres archivos:

- Cerca del inicio del archivo principal.
- CANLIB.ASM (archivo que contiene macros CAN)
- MACROS16.INC

Si una instrucción poco familiar se encuentra, esta probablemente estará construida por un conjunto de instrucciones comunes en una de las macros. Las macros en el archivo MACROS16.INC son usadas en el código extensivamente en la escritura del código para la familia del microcontrolador PICmicro, debido a que ellas aumentan la legibilidad y reducen los errores de programación.

Código Común.

La tarjeta del nodo usa archivos de software comunes para maximizar la eficiencia del programa. Las rutinas que habilitan la comunicación con el chip CAN MCP2510 están en el archivo CANLIB.ASM, mientras las definiciones para los registros del MCP2510 están en MCP2510.INC las macros comunes están en MACROS16.INC.

Comunicaciones SPI.

La comunicación de un dispositivo en el nodo (tal como el de un microcontrolador) al MCP2510 es realizada a través del bus SPI. El dispositivo PICmicro usado en la tarjeta del nodo soporta completamente al SPI en el nodo maestro. Las cadenas de comando son enviados y recibidas usando un solo buffer de software. Para enviar una cadena, el buffer de software (llamado pSPIBuf Base) está cargado con los bytes a enviar y el interruptor SPI es encendido. El manejador de interrupción intercambia los bytes con el MCP2510. Los bytes recibidos por el MCP2510 reemplazan a los bytes que se enviaron del buffer de software para que, una vez que la cadena ha sido enviada, el buffer contenga los bytes recibidos del MCP2510. Toda la comunicación con el MCP2510 es manejada de esta manera y se encapsula en las rutinas en el CANLIB.ASM.

Estructura general del ID.

La estructura del ID usada por las tarjetas del nodo es determinado por las configuraciones en los interruptores DIP al ser encendido después de una inicialización. Cambiando los interruptores del DIP mientras esta corriendo no tiene efectos alguno en la estructura del ID.

Estructura de recepción del ID.

La tarjeta del nodo usa las configuraciones de la tarjeta 2 para recepción.

Registro	**Valor**
RxMask0	0xFFF
RxMask1	0xFFE
RxFilter0	0xFFF
RxFilter1	0xFFF
RxFilter2	$0xn00^{(1)}$
RxFilter3	0xn10
RxFilter4	0xFFF
RxFilter5	0xFFF

Tabla 4.3: Configuraciones de recepción

Nota: Este valor es el ID de recepción base para la recepción de las configuraciones del DIP#1 y el DIP#2 son usados para determinar este valor.

Las configuraciones del DIP para recepción son mostradas en la Tabla 4.3.

Un mensaje recibido por el Rxfilter2 (ID de recepción base) es asumido para ser un entero de 2 bytes que contiene un valor de 12 bits entre 0 y 4095. El dato de 12 bits

DIP#1	DIP#2	ID
0	0	0x000
0	1	0x100
1	0	0x200
1	1	0x300

Tabla 4.4: Configuración de ID en el interruptor DIP para recepción

es usado para generar una salida de PWM, donde un "0"genera un 0 % del ciclo de trabajo y 0xFFF genera un 100 % de ciclo de trabajo.

Estructura del ID de transmisión.

La tarjeta del nodo transmite un mensaje CAN cada 131ms. Un mensaje contiene 2 bytes de datos eso representa un valor de 12 bits con el byte menos significativo (LSB) enviado primero.

El interruptor de presión es asignado al ID de transmisión base y es medido y transmitido con esa ID cada 393ms como un entero de 2 bytes en un rango de 0 a 4095.

El lector debe notar lo siguiente, mientras que la medición de A/D es de 8 bits, este es recorrido en 4 bits antes de la transmisión. Así, su rango actual es de 0x0000 a 0x0FF0.

Cada fuente de datos tiene su propia ID base de transmisión obtenido de las configuraciones del DIP#3 y DIP#4. Estas configuraciones se muestran en la Tabla 4.

DIP#1	DIP#2	ID
0	0	Todas las transmisiones son deshabilitadas
0	1	0x100
1	0	0x200
1	1	0x300

Tabla 4.5: Configuración de ID en DIP SWITCH para transmisión.

El controlador CAN MCP2510 tiene una tasa de 125Kbit y el método de poleo es usado. El uso de interrupciones podría ser más fácil en el sistema, pero el poleo permite que los pines de interrupción permanezcan libres para otras funciones potenciales del sistema.

Hay tres métodos para transmitir información:

- Responder a un evento externo (controla por eventos).
- Enviar mensajes a intervalos regulares (transmisión temporizada). El tiempo del evento puede ser desconocido.
- Una combinación de los primeros dos. El receptor puede esperar los mensajes en un intervalo máximo conocido.

A continuación se presenta el programa principal representado en diagramas de bloques con una breve descripción de las funciones realizadas. Para obtener el código así como el diagrama de flujo en detalle consulte (Cf. [Technology, 2002]).

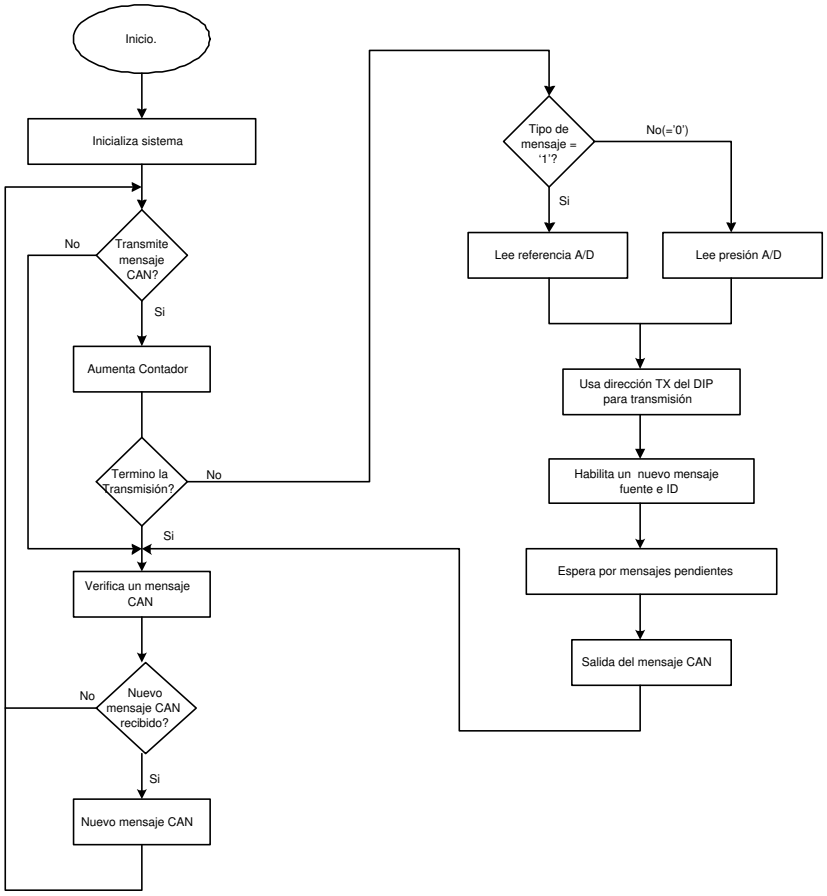

Figura 4.4: Lazo del programa principal

Nombre de la Función	Descripción de la Función
INI	Inicializa el sistema
VERCAN	Verifica un mensaje CAN
LEEAD	Lee variables analógicas convertidas a discretas
LEE3201	Lee la variable de presión
ESPERA	Espera los mensajes pendientes

Tabla 4.6: Funciones del programa principal

4.1.6. Conclusión.

El MCP2510 ofrece un método sencillo para comunicar una red CAN a fin de maximizar la transmisión y recepción de datos vía el bus CAN. Este eficiente método permite que una amplia variedad de dispositivos de entrada y salida puedan ser conectados a la red usando la tarjeta del nodo. Una ventaja al utilizar este tipo de sistema es la habilidad de supervisar varias tarjetas nodo en cualquier tiempo dado. Si ocurre un error, este es detectado y retransmitido sobre la línea del bus hasta que el receptor reconozca el mensaje. Otra ventaja es que varios nodos puedan trabajar en una línea de bus en vez de utilizar grandes extensiones de cableado que conectan al panel de control. Este diseño muestra una forma de implementar un sencillo interruptor de presión conectado a una tarjeta nodo, junto con una fuente de luz para mostrar el valor en términos de brillo. Por medio de este ejemplo, diferentes usos pueden ser implementados para muchos tipos de entradas y salidas al usar las técnicas básicas vistas en éste diseño.

Conclusiones Generales.

Con el fin de asegurar la comunicación e intercambio de datos entre diferentes unidades que están compartiendo datos, componentes de control y supervision, es necesario contar con un protocolo de comunicación robusto. Para esta tarea, se han diseñado varios protocolos de comunicación como: BATIBUS, BITBUS, EIB, FIP bus, J1850, LON, PROFIBUS, VAN, CAN. Sin embargo, como hemos visto, en la actualidad un protocolo interesante es el Bus-CAN, principalmente porque este protocolo permite el intercambio de información entre varios participantes en tiempo real y de manera inteligente.

Esto lo realiza de manera inteligente gracias a su forma de arbitrar, es decir, todos los nodos de la red son considerados como maestros y cada uno puede enviar mensajes en cualquier momento. Cada mensaje tiene una sección de identificador y este determina la prioridad del mensaje. Los identificadores con valor más pequeño tienen la más alta prioridad. Aunque varios mensajes sean enviados al mismo tiempo, este sistema de prioridad provocará que un mensaje permanezca en el bus y no que sean destruidos.

En esta monografía se estudió al protocolo y el bus CAN por sus siglas en inglés de Controller Area Network (Red de controlador de área). Tal como lo describe en las normas ISO, el bus CAN es un protocolo de comunicación serial que soporta de forma eficiente la distribución de comandos en tiempo real con un alto nivel de seguridad.

Su dominio de predilección cubre generalmente las aplicaciones de la red a una alta tasa de transmisión, alta fiabilidad de transmisión y con un concepto de cableado multiplexado de bajo costo.

Como se analizó en esta monografía , cada vez que se aborda el diseño de un bus los problemas a resolver son numerosos y las diferentes características de los campos de aplicación tomados como objetivos modifican el orden de los parámetros a ser tomados en cuenta e implican el desarrollo de nuevas ideas y conceptos para resolver dichos problemas.

Los problemas que se analizaron y se vieron algunas posibles soluciones fueron entre los más importantes:

a) Las nociones de acceso a las redes, incluyendo evidentemente los problemas de conflictos y de arbitraje, los tiempos de latencia, etc.,

b) Las nociones de elasticidad de una red,

c) La seguridad de la información transportadas o en otros términos, la estrategia de gestión de los errores (la detección, señalización y corrección de errores),

d) Las cuestiones que tienen que ver con las topologías, longitudes y débitos,

e) Las cuestiones de soporte físico, etc.

En esta monografía se ha intentado introducir al lector para que a la pregunta de ¿Por qué el CAN y por que no otro protocolo?, se le proporcione una ayuda para comprender y abordar rápidamente lo que es el terreno de este bus.

Se abordaron también algunos enfoques conjugados de técnicas y de marketing que permitieron hacer de este concepto una exitosa historia industrial.

Se abordaron también los principios básicos sobre los que descansa este protocolo de comunicación serial distribuida con sus diferentes capas. También se abordaron los diferentes problemas a resolver al diseñar e implementar una red como son: las nociones de acceso a las redes, incluyendo evidentemente los problemas de conflictos y de arbitraje, los tiempos de latencia, las nociones de elasticidad de una red, la seguridad de la información transportada, i.e la estrategia de gestión de los errores (la detección, señalización y corrección de errores), las cuestiones de la topología de la red, longitudes y débitos, de soporte físico, etc., así como las principales herramientas para resolver dichos problemas.

También se abordó como trabaja su mecanismo de arbitraje, se explicó que son los tiempos de latencia, la contención a nivel de bits, que tienen que ver con el principio de arbitraje que utiliza este bus; las consecuencias y capacidades que tiene una red con el uso del bus CAN, así como las nociones de elasticidad de un sistema, y las implicaciones que tiene la elasticidad de un sistema sobre la elección de un tipo de direccionamiento.

Algo de mucha importancia para completar este estudio es como se lleva a cabo en el bus CAN, el tratamiento, detección y gestión de los errores. De esta forma se intentó llegar al punto de visualizar desde la concepción hasta su implementación, y el lugar que ocupa este bus especial en el mundo automotriz, aeronáutica y marítimo.

Se abordó también los principios adoptados para la comunicación interna, describiendo de una forma simple y comprensiva compuesta de 5 partes: presentación general, conformidad ISO/OSI, vocabulario, definiciones, circulación de las tramas sobre el bus, los casos de errores, sus detecciones, sus tratamientos, el resto del protocolo (sobre la calidad del bus), y para terminar, el CAN 2.0B.

Se discutió ampliamente sobre la capa física sobre la cual se transmiten las tramas CAN. I.e. la capa física descrita en el documento de referencia y el de ISO, las propiedades del bit, los tipos y estructuras de redes, los problemas de propagación de señal, de distancia y de caudal de información.

Finalmente, el objetivo general de esta monografía fue la de analizar en detalle el protocolo CAN y que con el presente trabajo, se ha buscado que el lector conozca, a través del estudio de este protocolo, un bus de comunicación serial distribuida, para

comprender las bases del diseño e implementación de uno de los protocolos más utilizados en la industria.

Perspectivas de trabajo futuro.

Las perspectivas de la presente Monografía serían:

- una vez que se conocen las bases sobre las cuales esta fundamentado el protocolo CAN, es posible desarrollar un protocolo propio, implementando solamente algunas partes de este protocolo, si se pretende tener un desarrollo mas ligero cuando se cuenta con pocos recursos computacionales
- A través del estudio del presente trabajo, es posible diseñar e implementar una red de comunicación basada en el protocolo CAN con algunas ligeras modificaciones debidas al tipo de medio de transporte.
- Basado en el diseño presentado en el Capítulo 4, es posible hacer una implementación real con este protocolo, haciendo uso solamente de las partes que ya se han analizado.

SIGLARIO

ACK. - Acknowledgment.

CAL. - CAN Application Layer.

CAN. - Controller Area Network.

CRC. - Cyclic Redundancy Check.

CSMS/CA. - Carrier Sensor Multiple Access / Collision Avoidance

CSMS/CD. - Carrier Sensor Multiple Access / Collision Detection.

CiA. - CAN in Automation.

DLC. - Data Length Code.

DLL. - Data Link Layer.

ECU. - Electronic Central Units

EOF. - End of Frame.

GIE. - Agrupación de Interés Económico

IDE. - Identifier Extension.

ISO. - International Standards Organization.

I2C. - Inter Integrated Circuits.

LLC. - Logical Link Control

MAC. - Medium Access Control.

MDI. - Medium Depend Interface.

MSB. - Most Significant Bit.

NRZ. - Non Return to Zero.

OD.- Oscillator Drifts.

OLF. - Overload Flag.

OSEK. - Sistemas Abiertos e Interfaces para los dispositivos Electrónicos Distribuidos en un Automóvil

OSI. - Open Systems Interconnection.

PL. - Physical Layer.

PLS. - Physical Signaling Layer.

PMA. - Physical Medium Access.

PSA.- Peugeot Citroën. - Grupo Automotriz de estas marcas registradas

RJW.- Resynchronization Jump Width.

RTR. - Remote Transmission Request.

SAE. - Sociedad de Ingenieros Automotrices

SOF.- Start of Frame.

SRR. - Substitute Remote Request.

TSEG. - Time Segment.

VAN. - Vehicle Area Network.

Bibliografía

[Bühring, 1991] Bühring, P. (1991). *Bit timing parameters for CAN networks*. Application note KIE 07/91 ME - Philips Semiconductors.

[Charzinski, 1994] Charzinski, J. (1994). *Performance of the error detection mechanisms in CAN, Universidad de Stuttgart*. CiA publications.

[Dave Ross, 2005] Dave Ross, P. (2005). Diversified engineering & manufacturing, inc. Internet site, DVM, inc.

[Decotignie and Pleinevaux, 1993] Decotignie, J. and Pleinevaux, P. (1993). *A Survey on Industrial Communication Networks*, pages 435–448. Annals Des Telecommunications.

[Dominique, 1996] Dominique, P. (1996). *Réseaux Locaux, Le bus CAN*. DUNOD.

[Etschberger, 2000] Etschberger, K. (2000). *CAN Controller Area Network*. Éditeur Hanser.

[Kiencke, 1994] Kiencke, U. (1994). *Controller Area Network - from concept to reality -Universidad de Karlsruhe (IIIT)*. CiA publications.

[Kiencke et al., 1986] Kiencke, U., Dais, S., Litschel, I., and Unruh, J. (1986). *Minutes de meeting des SAE international Congress 1986 et 1988*. CiA publications.

[Organization, 1984] Organization, I. S. (1984). *Open Systems Interconnection - Basic reference Model*. ISO 7498.

[Peterson and Davie, 1996] Peterson, L. and Davie, B. (1996). *Computer Networks a Systems Approach*. MK Publishers.

[Reuss, 1988] Reuss, C. (1988). *CAN - A powerful solution for fieldbus applications - Universidad de Dresden*. CiA publications.

[Robert BOSCH, 1991] Robert BOSCH, S. (1991). *CAN protocol Specifications V 2.0 (A and B)*. Société BOSCH.

[Semiconductor, 1994a] Semiconductor, P. (1994a). *DATA SHEET, PCA82C250, CAN controller interface*. Objetive specification.

[Semiconductor, 1994b] Semiconductor, P. (1994b). *DATA SHEET, SJA 1000 Stand-alone CAN controller*. Preliminary specification.

[Site, 1999] Site, C. W. (1999). *http://www.can-cia.de/, CAN in Automation*. CiA.

[Technology, 2002] Technology, M. (2002). *Reference Manual of CAN applications*. Microchip Technology No. DS00212B.

[Unruh et al., 1989] Unruh, J., Mathony, H., Kaiser, K., and Bosch, R. (1989). *Error detection capabilities of the CAN protocol*. CiA publications.