UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"ESPECIFICACIÓN FORMAL EN SDL DE UNA PASARELA CAN/LIN"

TESIS

PARA OBTENER EL GRADO DE:

MAESTRO EN ELECTRÓNICA, OPCIÓN: SISTEMAS INTELIGENTES APLICADOS

PRESENTA:

ING. EDGAR SEGISMUNDO CRUZ GARCÍA

DIRECTOR DE TESIS:

DR. ENRIQUE GUZMÁN RAMÍREZ

CO-DIRECTOR DE TESIS:

M.E. HERIBERTO ILDEFONSO HERNÁNDEZ MARTÍNEZ

H. CD. DE HUAJUAPAN DE LEÓN, OAXACA, JULIO DE 2017

Tesis presentada el 13 Julio de 2017

ante los sinodales:

M.S.R.C. José Antonio Moreno Espinosa

M.C. Esteban Osvaldo Guerrero Ramírez

Dr. Moisés Homero Sánchez López

Dr. Antonio Orantes Molina

Director de tesis:

Dr. Enrique Guzmán Ramírez

Co-Director de tesis:

M.E. Heriberto Ildefonso Hernández Martínez

Dedicatoria

A mi familia, quienes me han apoyado siempre sin importar las dificultades que se presenten; gracias a su apoyo y sus consejos, he llegado a realizar una de mis más grandes metas.

A mi madre:

María García López

A mi padre:

Lorenzo Neptalí Cruz Caballero

A mi Esposa:

Victoria Patricia Santiago Sanchéz

Edgar.

Agradecimientos

"La vida no es justa, acostúmbrate a ello"

Bill Gates.

Agradezco a Heriberto Ildefonso Hernández Martínez y Enrique Guzmán Ramírez, por permitirme trabajar bajo su dirección, por brindarme su experiencia, conocimientos, amistad y confianza.

A mis profesores por el conocimiento que me brindaron durante mi estancia en la universidad.

A mis sinodales por el tiempo dedicado a la revisión de este trabajo, por sus observaciones y sugerencias realizadas.

A mis padres, aunque no existe forma de agradecer una vida de sacrificio, esfuerzo y dedicación, quiero que sientan que el objetivo logrado no es sólo mío, también es de ustedes porque la fuerza que me ayudó a conseguirlo fue su apoyo, cariño y comprensión.

A mi querida esposa, por ser mi razón, por luchar siempre a mi lado brindándome tu amor y tus consejos.

A mis hijos, gracias por su paciencia, apoyo y amor incondicional.

Edgar.

Índice General

Índice de Figuras

1. Introducción

1.1. Antecedentes

En décadas pasadas, la interconexión de sensores, actuadores y unidades de control electrónico (ECUs, *Electronic Control Units*)[1] ha incrementado la complejidad de la red de comunicaciones y con ello el tráfico de información en un automóvil. En la actualidad, la mayoría de fabricantes automotrices están usando múltiples protocolos de comunicaciones como son: CAN (*Controller Area Network*) (Robert Bosch GmbH, 1991), LIN (*Local Interconnect Network*) (LIN Consortium, 2010), FlexRay (FlexRay Consortium, 2005) y MOST (*Media Oriented Systems Transport*) (MOST Cooperation, 2010), entre otros, para satisfacer los requerimientos de las diferentes aplicaciones[2] (véase Figura 1.1).

Mientras que CAN es el protocolo de facto en el manejo del tren motriz[3], LIN provee una solución de red de bajo costo para aplicaciones de encendido y apagado (*on-off).*

En muchos casos, los datos tienen que ser compartidos entre varias redes usando diferentes protocolos, las pasarelas (GWs, *Gateways*) optimizan el

[1] Una unidad de control electrónico es un subsistema compuesto por un microcontrolador, un conjunto de sensores y de actuadores para implementar funciones en un vehículo.

[2] Generalmente, un vehículo de 25-45 ECUs se implementa mediante el uso de cuatro protocolos de comunicaciones para el intercambio de 100-300 señales aproximadamente (Nolte, Hansson y Lo Bello, 2005).

[3] El tren motriz es el conjunto de motor-transmisión-árbol de transmisión en un vehículo que generan energía mecánica para su movimiento.

intercambio de información de un protocolo a otro. Vivekanaandan, *et al.* enfatizan la importancia de una red configurable en un vehículo, donde CAN funciona como una red principal a la que se conectan redes LIN a través de una pasarela (Vivekanaandan, Sabane y Krishna, 2003). Hyan Seo, *et al.* diseñaron una pasarela entre los protocolos CAN y FlexRay para intercambiar información entre una ECU principal y ECUs secundarias con la finalidad de dividirse la carga de trabajo (Seo, Lee, Hwang y Wook Jeon, 2006). Cabe señalar que un error en la transferencia de datos de un protocolo a otro a través de las GWs y las ECUs es una tarea crítica, que puede repercutir en fallas de seguridad en el vehículo o inconvenientes al pasajero.

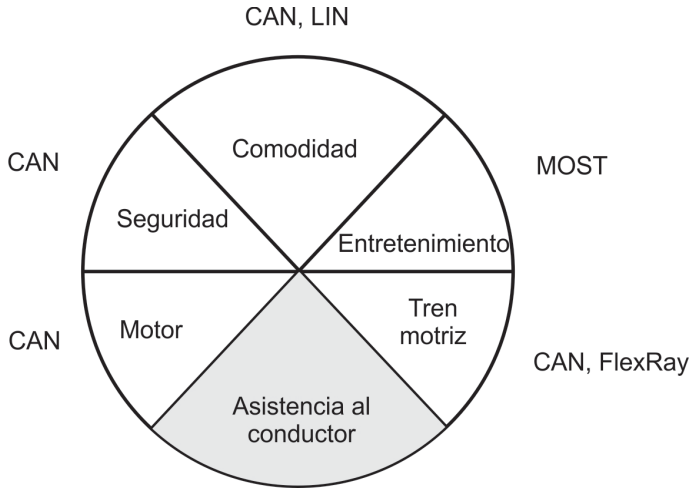

Figura 1.1. Principales áreas de aplicación de los buses de campo en automoción.

Se citan como antecedentes los siguientes trabajos de investigación realizados en la UTM:

- "Desarrollo de un sistema educativo para la enseñanza del protocolo de comunicaciones CAN", describe un sistema educativo cuyo propósito es la enseñanza del protocolo de comunicaciones CAN (SeeCAN). El SeeCAN se diseñó mediante una metodología de desarrollo de sistemas embebidos, y como resultado se obtuvo un sistema destinado a la enseñanza del protocolo CAN, ya que los sistemas de entrenamiento (*starter kit*) comerciales no están enfocados a la enseñanza general del protocolo y presentan costos elevados (Chamú Morales, 2005).

- "Especificación del protocolo FlexRay utilizando un lenguaje de descripción formal", en esta investigación se obtuvo la especificación formal en SDL de la capa de enlace de datos del protocolo de comunicaciones FlexRay utilizando Cinderella SDL (González Salinas, 2008).
- "Especificación del protocolo DNP3 utilizando un lenguaje de descripción formal", esta investigación se realizó en dos áreas importantes en el desarrollo tecnológico e industrial, por un lado, el protocolo de comunicaciones DNP3 y por otro la FDT SDL. En este trabajo se puso énfasis en el algoritmo para el cálculo del CRC, dado que éste no está definido de manera concreta en la especificación del protocolo de comunicaciones (Pérez Ortiz, 2011).
- "Especificación del protocolo CAN FD utilizando un lenguaje de descripción formal", este trabajo de investigación describe la capa de enlace de datos y la capa de supervisor en su totalidad de acuerdo a la especificación CAN FD 1.0 (López Pérez, 2015).

1.2. Técnicas de Descripción Formal

La descripción convencional de un sistema normalmente se realiza utilizando un lenguaje natural o mediante diagramas, lo que hace que presente ambigüedades y sea difícil de analizar. Generalmente, en la etapa de desarrollo físico del sistema se detectan errores y su rectificación es costosa, en tiempo y en dinero, debido a que se tiene que volver a la etapa de diseño para corregir errores y reiniciar la implementación física. Por otro lado, una descripción formal permite simular paso a paso la evolución de la especificación, realizar diferentes pruebas de comportamiento con la finalidad de detectar posibles errores u omisiones en la etapa de diseño y llegar a la etapa de implementación física solamente cuando se sabe con certeza que el sistema, equipo o protocolo descrito es correcto (ISO/IEC 9646, 1994). El uso de un método formal en el desarrollo de

una implementación contribuye a encontrar errores en la etapa inicial y no en la etapa de operación, lo que reduce el costo total del proyecto (Hall, 2005).

Las técnicas de descripción formal (FDTs, *Formal Description Techniques*) surgieron por la necesidad de contar con métodos rigurosos de diseño para el desarrollo de sistemas computacionales y de comunicaciones (Turner, 1993); a partir de la década de 1960, los protocolos de dichos sistemas presentaron un incremento en su complejidad, por lo que su diseño y su verificación representaba una tarea difícil y de altos costos, tanto en recursos humanos como económicos. Para resolver tal problema se planteó el desarrollo de lenguajes de especificación formal (FDL, *Formal Description Language*); en esencia estos lenguajes se fundamentan en una teoría matemática que asegura su precisión y manejabilidad para obtener diseños correctos de equipos, sistemas y protocolos.

Las principales áreas de aplicación de las FDTs son (Ellsberger, Hogrefe y Sarma, 1997):

- Sistemas concurrentes: Diseño de hardware, sistemas distribuidos, sistemas en tiempo real y procesamiento en paralelo, entre otros.
- Sistemas de calidad crítica: Aplicaciones financieras, telecomunicaciones y sistemas operativos.
- Sistemas de seguridad crítica: Defensa, medicina, equipos militares, señalización ferroviaria, telecomunicaciones y aeronaves.
- Sistemas de confidencialidad: De información y prevención de accesos no autorizados.
- Descripción de normas internacionales: De amplio uso y que deben ser interpretadas sin ambigüedad en todo el mundo.

Las principales organizaciones que se interesaron por la elaboración de FDTs, y que incentivaron su aparición y aplicación, fueron: CCITT (*Consultative Committee for International Telegraphy and Telephony*), actualmente ITU-T (*International Telecommunications Union, Telecommunications Standardization*), e ISO (*International Organization for Standardization*). El

CCITT inició su trabajo para la creación de normas mediante FDTs en el año 1972 y la ISO alrededor del año 1978; inicialmente la investigación del CCITT estuvo dirigida a la estandarización de aplicaciones en el campo de las telecomunicaciones, señalización y conmutación, mientras que la ISO se dedicó a la estandarización de aplicaciones para las comunicaciones y procesamiento de datos. Con el tiempo, las líneas de investigación de ambas organizaciones se fueron aproximando dando lugar a la aparición de las FDTs (Turner, 1993). La ISO normalizó ESTELLE (ISO/IEC 9074, 1989) y LOTOS (ISO/IEC 9907, 1989), mientras que el CCITT elaboró la norma SDL, detallada en la recomendación Z.100 (ITU-T, 2000).

Debido a que inicialmente las FDTs estaban orientadas a su aplicación en sistemas con distintos objetivos, es difícil encontrar un determinado FDL que presente una solución global y universal para todos los diseños. Por tal motivo, en 1985 el CCITT y la ISO decidieron trabajar conjuntamente en la elaboración de normas, así como su aplicación e introducción al mercado industrial de las FDTs. Esta colaboración consistió en ofrecer un conjunto de ejemplos con sus descripciones en ESTELLE, LOTOS y SDL, desde sistemas simples hasta sistemas complejos, con la finalidad de que los diseñadores de sistemas pudieran estudiar y comparar los FDLs para elegir el que mejor se adecue a su campo de aplicación.

1.2.1. Lenguaje de descripción formal

El lenguaje de descripción formal (SDL, S*pecification and Description Language*) utiliza una especificación con definiciones formales de semánticos (SDL-96 es la última versión). Se basa en un modelo de Máquina de Estado Finito Extendida (EFSM, *Extended Finite State Machine*) y se puede utilizar como un lenguaje de amplio espectro para describir un sistema desde sus requerimientos hasta su implementación.

La denominación SDL como lenguaje de especificación y descripción requiere de una diferenciación entre estos dos términos con respecto a un modelado:

- La especificación considera un sistema como una caja negra y está orientada a la captura de requerimientos y el ordenamiento de ideas, pone poco interés en su implementación.
- La descripción refleja la estructura de un sistema planificado o implementado.
- Un modelado representa las propiedades significativas del sistema y consecuentemente tiene sus limitaciones.

SDL por sí solo no diferencia entre especificación o descripción. En este trabajo de tesis, el término especificación se utiliza para denotar ambos, por lo tanto, una especificación en SDL puede o no reflejar la estructura de un diseño basado en la especificación (Belina, Hogrefe y Sarma, 1991). SDL provee una forma jerárquica de describir los sistemas permitiendo diferentes niveles de abstracción, como muestra la Figura 1.2 (González Salinas, 2008).

Un sistema SDL es visto como un conjunto de bloques conectados por canales (*channels*), proporcionando un panorama general de la estructura del sistema. Cada bloque puede ser dividido en otros bloques o directamente en procesos. Cada proceso representa una EFSM, y a este nivel, SDL describe un sistema como un conjunto de Máquinas de Estado Finito Extendidas Comunicantes (CEFSMs, *Communicating Extended Finite State Machines*). La comunicación entre procesos (CEFSMs), sistemas y entorno (*environment*) se realiza intercambiando mensajes (*messages*). Los mensajes son modelados usando el constructor de señal y, si es necesario, puede acarrear parámetros a través de una señal (*signal*).

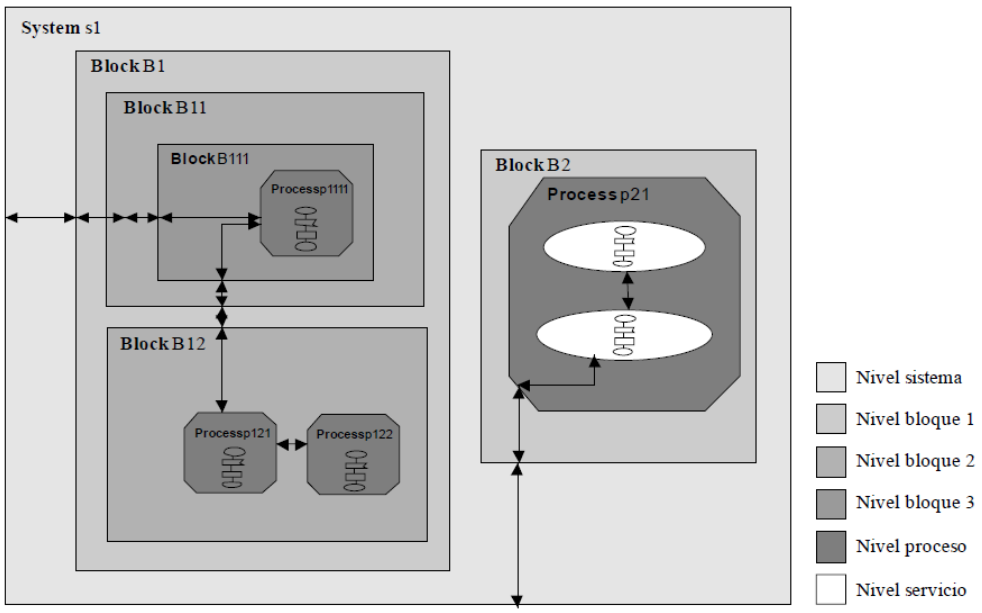

Figura 1.2. Niveles de abstracción jerárquica de un sistema SDL.

Una especificación SDL se divide en tres partes.

- Datos: Aún cuando el modelo de datos SDL se basa en ACT ONE, incluido en la recomendación Z.100, se apoya en el lenguaje de descripción de datos ASN.1 (*Abstract Sintaxis Notation One*), recogida en la recomendación Z.105 por la ITU y soportada en su totalidad por las herramientas existentes como TAU, ObjectGEODE, Melba96, Cinderella SDL y Progen. Algunos tipos de datos predefinidos son: *Integer, Boolean, Character* y *String*, con los que es posible construir tipos de datos más robustos mediante el uso de constructores como: *BewType*, *Struct*, *Array*, *Choice*, así como el uso de temporizadores (*timers*).
- Especificación estática: Descompone un sistema en varias partes, bloques o módulos, cada una de estas partes es susceptible de ser dividida en componentes más sencillos hasta alcanzar el nivel de abstracción deseado.
- Especificación dinámica: Los procesos contienen la información que describe cómo interacciona el sistema con su entorno, es decir, cómo ha de responder a las situaciones de operación para las que fue diseñado.

Esta información está implícita en el comportamiento de la máquina de estados que cada proceso tiene asociado.

SDL provee dos tipos de representación, representación textual (PR, *Phrase Representation*) y representación gráfica (GR, *Graphical Representation*); es posible la conversión entre PR y GR, y viceversa.

La especificación SDL que muestra la Figura 1.3 representa la estructura del sistema Example, el cual contiene dos bloques llamados Control y MachineTool conectados mediante el canal CommunicationServer, éste transmite las señales StartProgram, StopProgram, Operating y Stopped (R. M. & G. D., 1999).

En el siguiente nivel de abstracción se describe cada uno de los bloques. La Figura 1.4 muestra la estructura interna del bloque MachineTool. El bloque contiene sólo el proceso MachineControlUnit que se comunica con su entorno a través de la señal sr1 (R. M. & G. D., 1999).

El nivel de abstracción de la especificación se describe como una EFSM. En este caso el proceso MachineControlUnit se describe en la Figura 1.5 por una Máquina de Estado Finito (FSM, *Finite State Machine*) (R. M. & G. D., 1999).

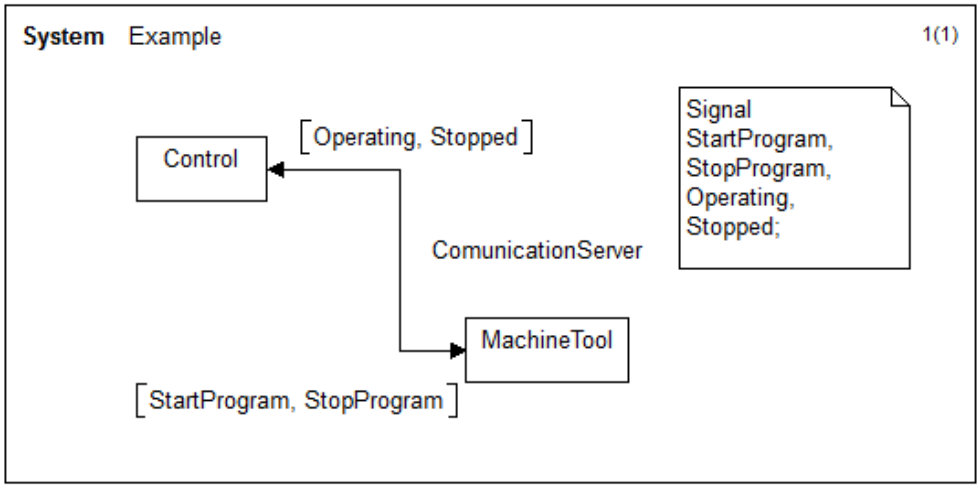

Figura 1.3. Especificación SDL del sistema Example *de la representación GR.*

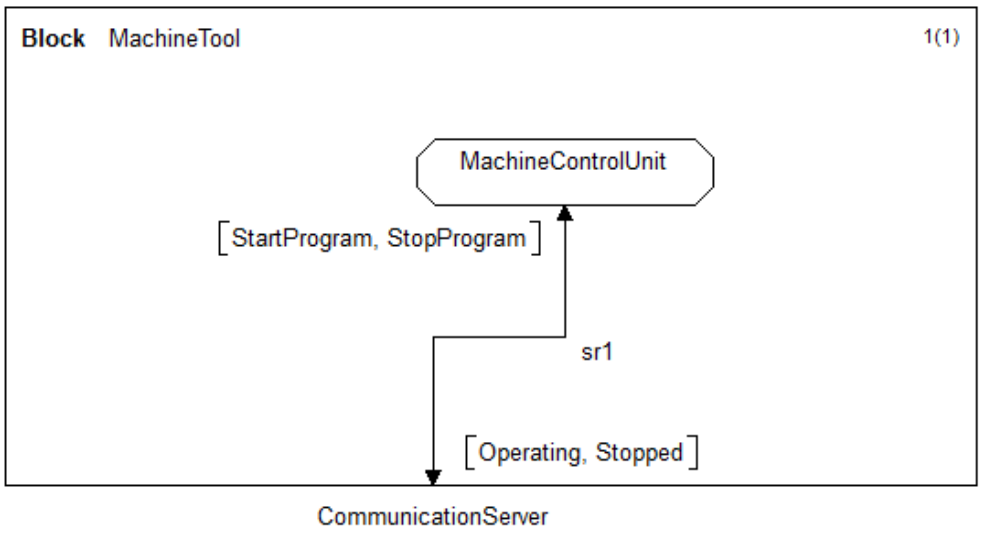

Figura 1.4. Especificación SDL del bloque MachineTool.

El proceso MachineControlUnit se muestra mediante diagramas de flujo, por ejemplo, si la FSM se encuentra en el estado *STOPPED* y llega un mensaje de *StartProgram*, se envía el mensaje *Operating* y la FSM cambia al estado *RUNNING*.

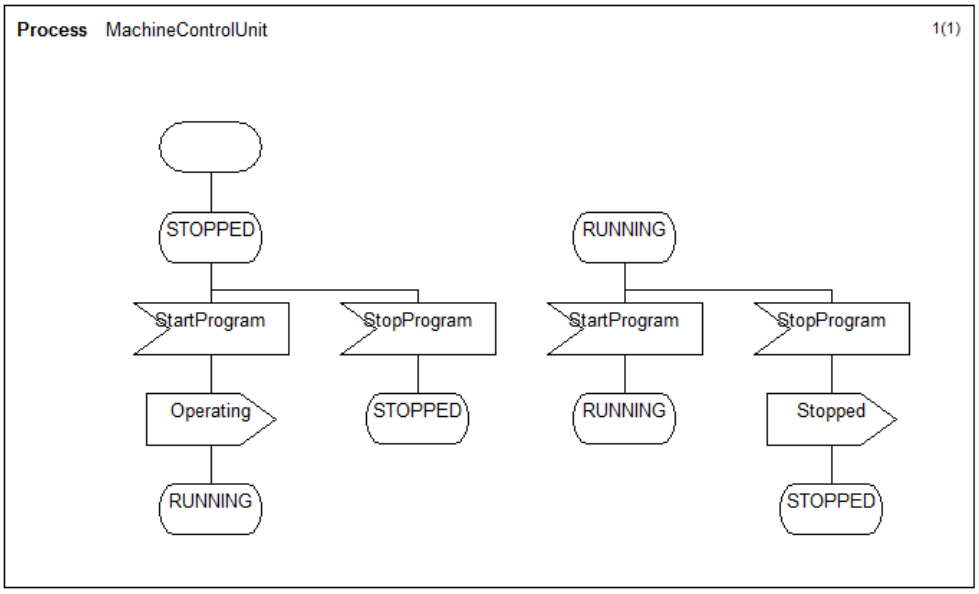

Figura 1.5. Especificación SDL del proceso MachineControlUnit.

1.2.2. FDL Cinderella SDL

Cinderella SDL es una FDL comercial que incorpora las modificaciones más recientes al estándar y que permite crear sistemas SDL de forma amigable; sus principales características son: análisis incremental, integración con el estándar

ASN.1, edición, análisis y simulación integral, importar y exportar especificaciones con otras herramientas de SDL, soporte con SDL de acuerdo al estándar Z.100 (Cinderella ApS, 2006).

La Figura 1.6 muestra la interfaz gráfica de usuario del FDL Cinderella SDL, en la que se puede observar que soporta las representaciones GR y PR, esta última es poco recomendable para especificar sistemas robustos. Una desventaja de Cinderella SDL es que no soporta macros en representación GR, los cuales son de gran ayuda en sistemas con muchos estados.

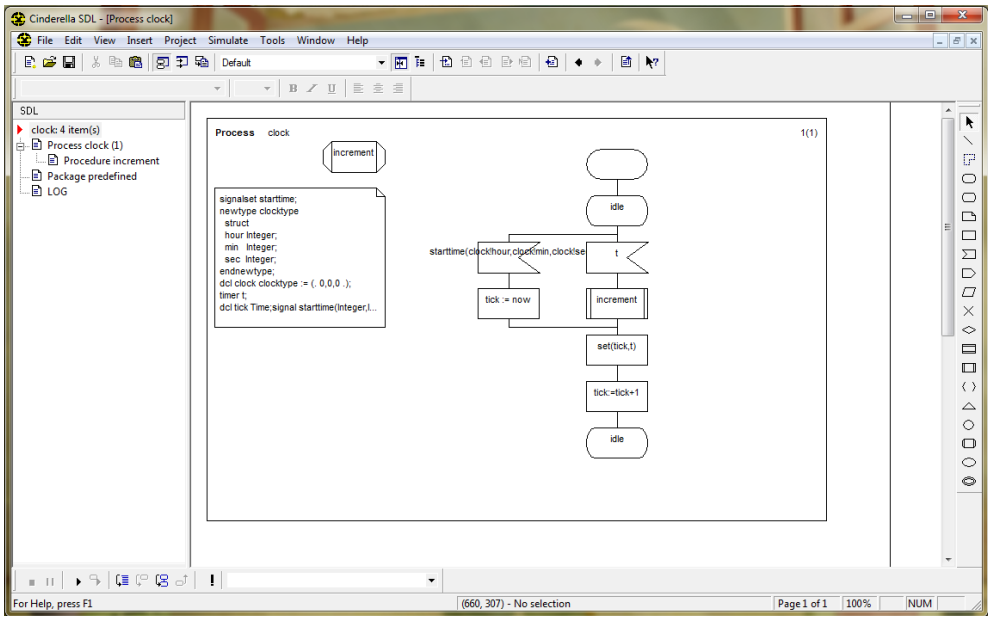

Figura 1.6. Interfaz gráfica de la herramienta Cinderella SDL.

1.3. Planteamiento del Problema

A mediados de la década de los ochenta se dieron a conocer diversos protocolos de buses de campo (FBs, *Fieldbus*) que atendían a diferentes soluciones en el área industrial, por ejemplo: CAN, LIN, Profibus, Interbus-S, P-Net, LON y FIP, entre otros (Navet, Song, Simonot-Lion y Wilwert, 2005).

CAN es considerado uno de los protocolos de comunicaciones más avanzados y en la actualidad es el estándar *de facto* en la industria automotriz; desde su origen este protocolo ha sido objeto de diferentes estudios.

En los últimos años el protocolo de comunicaciones CAN ha ampliado su gama de aplicaciones al control electrónico en los automóviles (Xing, Chen y Ding, 1999) (Tsugawa, 2005), sistemas de transporte, sistemas inteligentes, técnicas de medición, equipo médico, construcciones automatizadas y nuevas tecnologías encaminadas al monitoreo remoto de sensores (Lawrenz, 1997), lo anterior se debe principalmente a su estandarización en ISO-11898 (ISO/IEC 11898, 1993) e ISO-11519-1 (ISO/IEC 11519, 1994).

Así mismo, el bus LIN se ha implementado en los automóviles para controlar dispositivos donde no es necesario un ancho de banda elevado y el costo de implementación es menor.

Los protocolos CAN y LIN se utilizan en la industria automotriz en aplicaciones de tiempo real críticas para interconectar sensores, actuadores y ECUs. La conversión de dichos protocolos se realiza por medio de una pasarela CAN/LIN, la cual puede generar fallas de seguridad en el automóvil. La creciente complejidad de los protocolos de comunicaciones requiere de la utilización de nuevas técnicas que permitan realizar una especificación de los mismos de una forma completa, consistente y no ambigua, estas nuevas técnicas se conocen de forma genérica como FDTs. Actualmente se cuentan con diversas soluciones, sin embargo, no existe una especificación formal de una pasarela CAN/LIN.

Con base en lo anterior se formula la siguiente pregunta de investigación: ¿Es posible que mediante la especificación formal en SDL de una pasarela CAN/LIN se puedan identificar errores y ambigüedades que permitan mejorar su implementación?

1.4. Justificación

En la actualidad la mayoría de las ECUs intercambian información usando buses de campo, en un vehículo los más utilizados son los protocolos CAN y LIN. Al utilizar diferentes protocolos de comunicaciones en las diferentes secciones del vehículo es necesario el uso de una pasarela para poder intercomunicar las

ECUs, ésta es una tarea crítica que puede repercutir en fallas de seguridad en el vehículo o inconvenientes al pasajero debidas a errores en el intercambio de la información, las principales fallas se han presentado en el sistema de frenado de los automóviles.

Estos protocolos se estudiaron en el curso Programación de Interfaces de la Maestría en Electrónica, Opción en Sistemas Inteligentes Aplicados de la Universidad Tecnológica de la Mixteca. El presente trabajo de tesis propone realizar la especificación de una pasarela de los protocolos de comunicaciones CAN/LIN con el uso de una técnica de descripción formal y la herramienta Cinderella SDL, ya que no existe información para fines de estudio e investigación.

1.4.1. Pertinencia

Realizar una especificación formal de una pasarela CAN/LIN es pertinente ya que en la Universidad Tecnológica de la Mixteca (UTM) se cuenta con licencias de la herramienta Cinderella SDL.

1.4.2. Relevancia

La especificación de una pasarela CAN/LIN es importante ya que su utilización ha sido la base de la creación de nuevas tecnologías aplicadas a los buses de campo, por tal motivo, es importante estudiar y especificar formalmente una pasarela CAN/LIN; además, no se cuenta con la suficiente información para entender su funcionamiento, desde la presentación de las especificaciones de los protocolos CAN y LIN no existen publicaciones con fines académicos e investigación de la especificación formal de una pasarela CAN/LIN.

1.5. Hipótesis

Mediante la especificación formal en SDL de una pasarela CAN/LIN se pueden identificar errores y ambigüedades que permitan mejorar su implementación.

1.6. Objetivos

El objetivo principal de este trabajo de tesis es realizar la especificación formal en SDL de una pasarela CAN/LIN.

1.6.1. Objetivos particulares

Para cumplir el objetivo principal, se plantean los siguientes objetivos particulares:

- Elaborar un estado del arte de los protocolos de comunicaciones en el campo de la automoción.
- Analizar el protocolo de comunicaciones CAN.
- Analizar el protocolo de comunicaciones LIN.
- Aprender la utilización de la herramienta Cinderella SDL.
- Realizar la especificación estática de la pasarela CAN/LIN.
- Realizar la especificación dinámica de la pasarela CAN/LIN.
- Realizar la verificación general a la pasarela CAN/LIN.

1.7. Metas

A continuación, se listan las metas a cumplir durante el proceso de investigación:

- Documentación de los protocolos de comunicaciones en el campo de la automoción.
- Construcción del estado del arte de los protocolos de comunicaciones en el campo de la automoción.
- Registro de la evolución de los protocolos CAN y LIN.
- Estudio de las normas del protocolo de comunicaciones CAN publicadas por Robert Bosch GmbH.
- Estudio de los estándares del protocolo de comunicaciones CAN publicadas por ISO-11898 e ISO-11519-1.
- Comprensión y análisis del protocolo de comunicaciones CAN.

- Comprensión y análisis el protocolo de comunicaciones LIN.
- Estudio del estándar Z.100 y la bibliografía sobre SDL.
- Conocimiento del funcionamiento del IDE (*Integrated Development Enviroment*) Cinderella SDL.
- Obtención de los requerimientos del sistema CAN.
- Obtención de los requerimientos del sistema LIN.
- Especificación de datos para el sistema CAN.
- Especificación de datos para el sistema LIN.
- Especificación de datos de la pasarela CAN/LIN.
- Especificación estática de la pasarela CAN/LIN.
- Especificación dinámica de la pasarela CAN/LIN.
- Simulación y validación de la pasarela CAN/LIN.
- Análisis de los resultados obtenidos.
- Documentación del trabajo de investigación.

1.8. Limitaciones

Se presentan las siguientes limitaciones al presente trabajo de tesis:

- La pasarela CAN/LIN no se implementará a nivel Hardware ya que no forma parte del estudio del presente trabajo.
- No se realizará la comparativa con otro software de especificación formal SDL.

1.9. Metodología de Desarrollo

Para realizar una especificación formal de la pasarela de los protocolos de comunicaciones CAN/LIN es necesario planificar el desarrollo en una serie de etapas. La metodología de desarrollo a usar será orientada a estados y a restricciones (González Salinas, 2008), dado que en la especificación de un sistema en SDL el aspecto dinámico (procesos) se realiza mediante máquinas de estados (orientada a estados), estados que estarán bien definidos y estructurados

en módulos (orientado a restricciones). Así las fases a seguir para el desarrollo de la especificación formal son las siguientes[4] (véase Figura 1.7):

- Requisitos de usuario: En esta etapa se definen, de forma clara y concisa, los requisitos que debe cumplir la especificación mediante el uso del lenguaje natural. En este caso los requisitos de usuario serán las especificaciones de los protocolos CAN y LIN.
- Requisitos de la especificación formal: Se definen los requisitos necesarios para escribir la especificación SDL, así como validar, verificar y simular su comportamiento.
- Diseño de arquitectura: En esta etapa se descompone el sistema global en módulos que interaccionan entre sí, el número de módulos y/o sub-módulos depende de la profundidad de abstracción con la que se desea realizar la especificación.
- Diseño detallado: Cada uno de los módulos que componen al sistema será especificado formalmente con SDL, incluyendo la depuración, la simulación y el refinamiento mediante la herramienta de desarrollo.
- Transferencia: Cuando finaliza la especificación se transfiere a los usuarios (no ejecutada en este trabajo de tesis).
- Operación y mantenimiento: Se usa la especificación formal obtenida y se hacen las modificaciones necesarias para adaptarla a los usuarios finales (no ejecutada en este trabajo de tesis).

La metodología descrita debe permitir escribir la especificación de la pasarela CAN/LIN sin ambigüedad, clara, precisa y concisa; se podrá analizar y corregir en su plenitud la especificación y determinar si un diseño cumple con los requerimientos solicitados mediante el uso de las herramientas para crear, mantener, analizar y simular especificaciones.

[4] Las dos últimas etapas no se ejecutan debido a que la presente tesis, como caso de estudio, no está desarrollada para una aplicación específica y las primeras cuatro etapas cubren los requerimientos de diseño formal con una FDT.

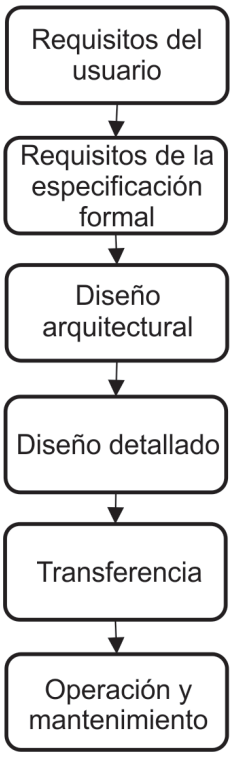

Figura 1.7. Metodología de desarrollo de la especificación de la pasarela CAN/LIN.

1.10. Estructura del Documento de Tesis

El presente trabajo de tesis se encuentra organizado en ocho capítulos:

- El Capítulo 1 presenta una introducción a la problemática tratada en el trabajo de tesis, el planteamiento para la resolución del problema y los objetivos a cubrir.
- El Capítulo 2 presenta el estado del arte sobre los protocolos de comunicaciones en el campo de la automoción.
- Los Capítulos 3 y 4 presentan el estudio de los protocolos de comunicaciones LIN y CAN respectivamente, basado en la arquitectura de protocolos OSI.
- La especificación formal de la pasarela CAN/LIN se presenta en el Capítulo 5.
- El Capítulo 6 presenta la especificación dinámica de la pasarela CAN/LIN.

- El Capítulo 7 presenta los resultados obtenidos de las pruebas realizadas a la especificación.
- Las conclusiones y trabajos futuros se presentan en el Capítulo 8 y por último las referencias bibliográficas utilizadas durante el presente trabajo de tesis.

2. Protocolos de Comunicaciones en el Campo de la Automoción

2.1. Introducción

La electrónica se empezó a utilizar en los automóviles a finales de la década de 1950 y principios de 1960, sin embargo, fue descontinuada tiempo después debido a que no fue bien aceptada por los clientes. Durante la década de 1970 ocurrieron dos eventos que reactivaron la tendencia hacia el uso moderno de la electrónica en los automóviles, el primero fue la regulación del gobierno de las emisiones y el consumo de combustible, lo que requería mejor control del motor, y el segundo fue el desarrollo de componentes electrónicos de estado sólido de bajo costo, que podían utilizarse para el control del motor y otras aplicaciones (Ribbens y Mansour, 2003). Desde entonces, se observó un incremento exponencial en el número de sistemas electrónicos que gradualmente reemplazaron a aquellos puramente mecánicos e hidráulicos.

Uno de los principales propósitos de los sistemas electrónicos en el vehículo era asistir al conductor en el control de funciones relacionadas con la dirección, la tracción y el frenado, entre otras; por lo tanto, para que los nuevos dispositivos electrónicos trabajarán de forma efectiva era necesario recuperar la información de los dispositivos y sensores colocados en el automóvil e interconectados de forma individual. Cuando incrementó el número de sensores también incrementó el cableado necesario para su interconexión; por ejemplo, los automóviles tenían

más de 4 kilómetros de cableado comparado con los 45 metros de los vehículos manufacturados en 1955 (Leen y Hefferman, 2002).

El aumento de cableado provocó problemas, uno de ellos fue el incremento del peso del vehículo, por ejemplo, cada 50 kilogramos de cable extra aumentaban 100 watts de potencia, lo que incrementaba el consumo de combustible en 0.2 litros por cada 100 kilómetros recorridos. El cableado extra también incrementó el número de conectores, ocasionando más puntos posibles de falla que afectaban directamente la confianza de los sistemas. También los arneses complejos ocupaban un espacio considerable en el vehículo, limitando su expansión; el cableado de arneses se convirtió en el componente más caro y complicado de un sistema eléctrico del automóvil (Leen y Hefferman, 2002). Cada nueva función era implementada con una ECU y esta estrategia requería una cantidad de canales de comunicación del orden de n^2, donde n es el número de ECUs; si cada nodo se interconectaba con todos los demás, el número de cables crecía al cuadrado de n demostrándose que este arreglo era insuficiente (véase Figura 2.1).

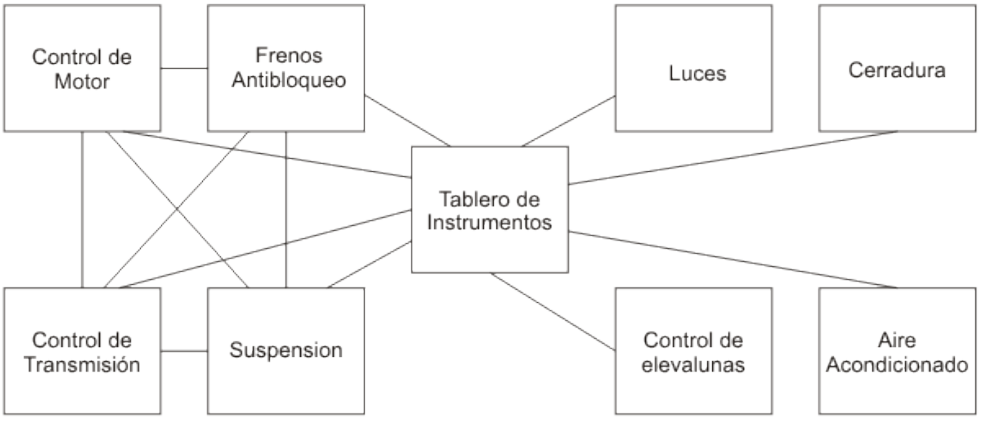

Figura 2.1. Sistema de cableado punto a punto.

Para reducir la cantidad de cableado se introdujo el concepto de buses de campo (FB), los cuales son sistemas de transferencia de información destinados a aplicaciones distribuidas de tiempo real, sistemas de automatización, sistemas inteligentes, sistemas de supervisión y control en el ámbito de celdas de

producción. Los FBs interconectan dispositivos electrónicos, tales como: sensores, actuadores y ECUs, que operan desde complicados procesos industriales hasta simples procesos en el hogar.

La evolución de las redes de comunicaciones en el automóvil, con la utilización de FBs, presenta tres ventajas: Reducir la cantidad de cableado en un automóvil, monitorear los datos generales a través de cualquier ECU y reducir el número de sensores al dotarlos de inteligencia (*smart sensors*), con lo que se reduce el peso del vehículo, se incrementa la flexibilidad y se reduce el costo total (Kumar, Sharma y Ramya, 2010). Su uso proviene de la industria donde se utilizaban para realizar control de tiempo distribuido y conectar instrumentos en una planta manufacturera. Los FBs trabajan en una estructura de red con topologías de estrella, anillo y árbol, entre otras. Previamente, las computadoras empleaban comunicación serial RS-232 para comunicarse con otros dispositivos, sin embargo, con el uso de los FBs se reduce la longitud y cantidad de cableado requerido.

El FB más antiguo en la industria fue Bitbus, desarrollado por la empresa Intel. Aunque la tecnología de FBs se remota al año 1988, el desarrollo de su estandarización tomó varios años más. Hubo varias tecnologías de buses de campo y la unificación de mecanismos de comunicación aún no se ha realizado. La tecnología de FBs necesita ser implementada en diferentes aplicaciones, por ejemplo: los FBs en la automoción son funcionalmente diferentes a los empleados en el control de procesos en una planta industrial. Por tal motivo, se traslada la ventaja de usar FBs en la automoción para reducir el cableado, reduciendo tanto el peso como el costo de los sistemas automotrices (véase Figura 2.2).

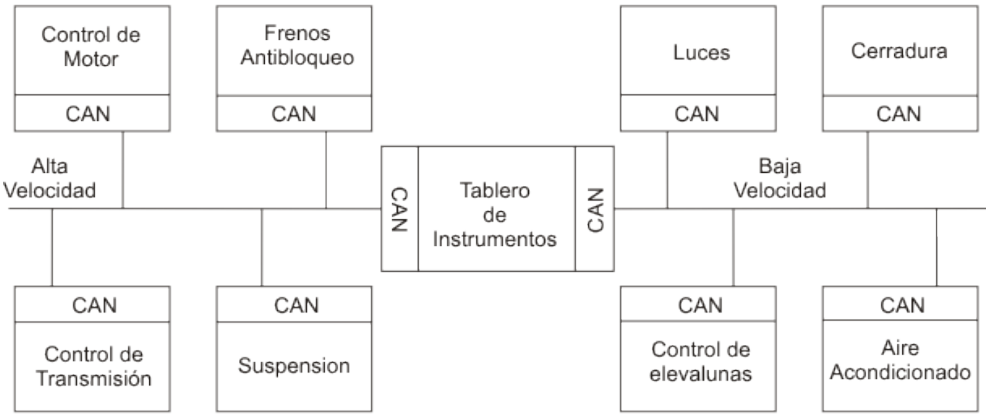

Figura 2.2. Red CAN.

Motorola reportó que reemplazando los arneses con una red CAN en las cuatro puertas de un vehículo BMW redujo el peso en 15 Kg y mejoró la funcionalidad. Para incorporar la red CAN, los fabricantes diseñaron sus propias tecnologías, algunos ejemplos son el protocolo CAN por Robert Bosch GmbH, el protocolo VAN (*Vehicle Area Network*) desarrollado por Renault y Peugeot, el protocolo Byteflight por BMW y J1850 por GM, Ford y Chrysler, entre otros. Sin embargo, como la mayoría de los fabricantes contrataban proveedores en común, hubo la necesidad de crear estándares. Uno de los protocolos que se hizo popular a inicios de la década de 1990 fue CAN y pronto se volvió la red más usada en la industria automotriz. Con el uso de este protocolo se redujo la longitud y el número de cables dedicados en un arnés, por ejemplo, en un Peugeot 307 que cuenta con dos buses CAN el número de cables se redujo en un 40%, de 635 a 370, en comparación con el Peugeot 306 que no tiene sistemas de multiplexación (Navet, *et al.*, 2005).

El control de tracción, necesario en los vehículos modernos, es un ejemplo de un sistema inteligente y seguro, donde se emplean cuatro sensores en el vehículo y cada uno mide la velocidad rotacional de las llantas enviando las lecturas al sistema de control principal, éste evalúa la información y determina, calculando las diferencias de la velocidad rotacional, si una o más llantas han perdido tracción. Si el control principal detecta que una llanta está girando más que las

otras, envía un mensaje para que aplique el freno a la llanta en cuestión, asegurando que el vehículo siempre esté en contacto con el piso y permitiendo un recorrido más seguro. Otro ejemplo es la velocidad del vehículo, estimada por el controlador del motor o por los sensores de rotación de la llanta, la cual se debe conocer para corregir el esfuerzo en la dirección o para controlar la suspensión.

La siguiente generación de automóviles está mostrando una tendencia de integración de los sistemas actuales con mejores rendimientos. Por ejemplo, se ha determinado que el sistema de frenos antibloqueo (ABS, *Antilock Brake System*) y el sistema de tracción proporcionan mejor desempeño si se conocen los valores de aceleración vertical (tradicionalmente necesarios sólo para el control de la suspensión) (Cena, Valenzano y Vitturi, 2005). Esto genera más tráfico a la red, que no puede ser resuelta por una red CAN convencional; una opción es utilizar múltiples redes CAN conectadas entre sí por una pasarela, sin embargo, no es la solución óptima debido tanto al costo de las pasarelas como a los retardos adicionales que generan, por ejemplo, en los vehículos de lujo se intercambian hasta 2500 señales entre aproximadamente 70 ECUs (Navet, et al., 2005). Otra solución es implementar diversos protocolos dentro del vehículo para repartir la carga de trabajo; por ejemplo, el Volvo XC90 contiene 40 ECUs interconectadas por un bus LIN, un bus MOST y dos buses CAN, uno de alta y otro de baja velocidad. Actualmente, se pretende reducir el número de redes diferentes que se utilizan en un vehículo, debido a que no es posible confiar en una sola red, especialmente en vehículos de alto costo (Cena, Valenzano y Vitturi, 2005). Todas estas nuevas funciones requieren de una gran cantidad de datos para ser intercambiados dentro del vehículo y con el mundo externo a través del uso de tecnologías inalámbricas.

Las diferentes innovaciones de sistemas en el vehículo han originado demandas de ancho de banda y un incremento en la necesidad de redes determinantes y tolerante a fallos. Los sistemas de diagnóstico se utilizan para proveer información de las condiciones del vehículo y juegan un papel importante

en el proceso de búsqueda de fallos, con lo cual se puede determinar fácilmente dónde se encuentra el problema. Los sensores en un vehículo pueden detectar en tiempo real cuando un componente funciona fuera de sus límites y alertar al usuario antes de que se genere un fallo crítico, evitando con ello una situación crítica con el automóvil (Axelsson, Fröberg, Hansson, Norström, Sandström y Villing, 2003). De acuerdo con Navet, et al. (2005) y el proyecto X-by-wire (1998), la probabilidad de encontrar una falla crítica de seguridad en el vehículo no debe exceder $5x10^{-10}$ por hora y por sistema, pero otros estudios consideran un valor menor a 10^{-9}. Una práctica común es la de transferir datos entre dos dominios diferentes a través de una pasarela, usualmente llamado 'cuerpo central electrónico', este subsistema es reconocido como crítico en un vehículo ya que constituye un simple punto de falla, su diseño es complejo y los problemas de desempeño se incrementan debido a la creciente carga de trabajo.

Los nuevos avances tecnológicos están reemplazando las partes hidráulicas de los sistemas automotrices por sistemas electrónicos, estas soluciones son llamadas sistemas *x-by-wire*. Hay muchas razones por las cuales los fabricantes de automóviles esperan sustituir los sistemas hidráulicos y mecánicos por sistemas electrónicos. Las principales razones son el costo y las limitaciones tecnológicas de los sistemas hidráulicos y mecánicos, ya que particularmente, es cada vez más complicado implementar nuevos sistemas avanzados hidráulicos. La Figura 2.3 muestra un ejemplo de un sistema electrónico que reemplazó a un sistema mecánico e hidráulico, conocido como *Steer-by-wire* (Waern, 2003).

Existen diferentes tipos de FB y protocolos utilizados por las compañías fabricantes de automóviles, la mayoría están convencidas en unificar un sólo protocolo, pero todavía no se ha logrado optar por uno.

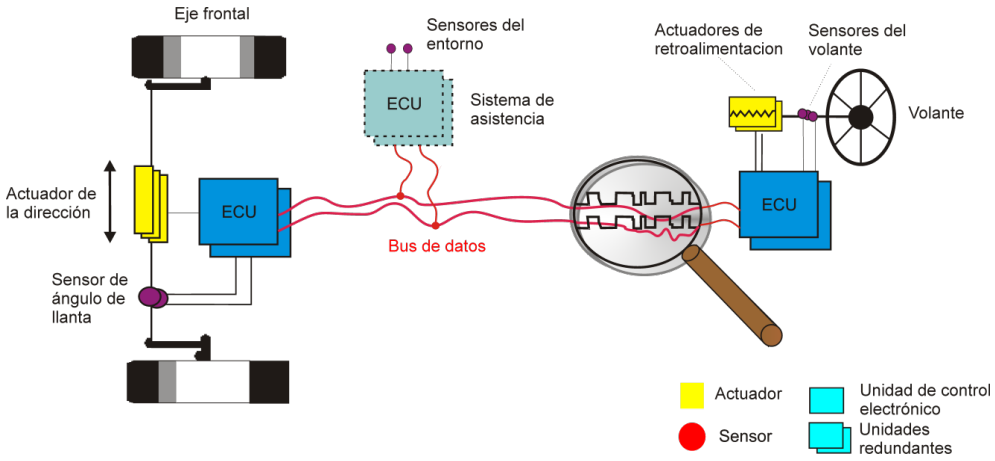

Figura 2.3. Sistema Steer-by-wire.

2.2. Clasificación de los Buses de Campo

En 1994, la SAE (*Society for Automotive Engineers*) estableció la siguiente clasificación formal de los protocolos de comunicaciones en el campo de la automoción, basada en las áreas de aplicación dentro del automóvil y la velocidad de transmisión de los datos (Navet, et al., 2005) (SAE J2056-2, 1994):

- Clase A: Redes de baja velocidad, con velocidades de transmisión de datos menores a 10 kbps, usadas para transmitir controles como interruptores (*switches*) de encendido/apagado, faros, luces de paro, posición de espejos, sensor de lluvia, control de puertas y ventanas, etc.; algunos ejemplos de protocolos de esta clase son LIN y TTP/A (*Time-Triggered Real-Time Fieldbus Protocol*).
- Clase B: Redes dedicadas al intercambio de datos entre las ECUs para reducir el número de sensores que comparten información. Las velocidades de transmisión de datos son de 10 a 125 kbps, sirven para aplicaciones como aire acondicionado y tablero de instrumentos, entre otros. Los protocolos J1850 y CAN de baja velocidad son los principales representantes de esta clase.
- Clase C: Redes dedicadas a aplicaciones en tiempo real, su velocidad de transmisión de datos oscila entre 125 kbps a 1 Mbps, sus principales aplicaciones son el control del motor y el control de estabilidad, entre

otros; el protocolo CAN de alta velocidad es el principal protocolo utilizado en esta clase.

- Clase D[5]: Redes dedicadas a altas velocidades de transmisión de datos, superiores a 1 Mbps; sus aplicaciones son orientadas a multimedia como son teléfonos y navegación basada en GPS, entre otros. Los protocolos más representativos son FlexRay y TTP/C (*Time-Triggered Protocol: A Safety-Critical System Protocol*).

2.3. Buses representativos en Automoción

A continuación, se presenta una descripción técnica de los buses de campo utilizados en automoción.

El protocolo IEBus (*Inter Equipment Bus*) pertenece a la clase A de la SAE, soporta una comunicación multimaestro utilizando CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) para el control de acceso al bus y soporta hasta 50 nodos conectados a una sola red. Utiliza un par de cables diferenciales para la transferencia de datos a una velocidad de operación que oscila entre 3.9 a 18 kbps sobre el bus con longitud máxima de 50 metros y como método de detección de errores se utiliza un bit de reconocimiento. Este protocolo es utilizado para aplicaciones de diagnóstico y control.

El protocolo MI (*Motorola Interconnect*) pertenece a la clase A, es un protocolo serial que cuenta con un nodo Maestro y soporta hasta 8 nodos Esclavo. Requiere de un sólo cable de comunicación y transmite datos a una velocidad de hasta 20 kbps sobre un bus de longitud máxima de 15 metros. Los nodos que utilizan este protocolo tienen serias limitantes en la cantidad de datos que se permiten transmitir y como método de detección de errores se utiliza un bit de reconocimiento. El protocolo MI es utilizado en aplicaciones de movimiento de

[5] La SAE todavía no ha definido estándares para redes de la clase D. Sin embargo, se consideran en esta clasificación aquellas redes que exceden una velocidad de 1 Mbps.

los espejos laterales, asientos, cerraduras de puertas, elevalunas y orientación de las luces frontales, entre otros (Burri y Renard, 1993).

El protocolo LIN (*Local Interconnect Network*) pertenece a la clase A de la SAE, es un protocolo serial que cuenta con un nodo maestro y hasta 15 nodos esclavo, la información se transmite por un único cable a una velocidad de hasta 20 kbps con una distancia máxima de 40 m. Su primera versión se realizó en el año 1999 y la última (revisión 2.2A) en el año 2010 por el consorcio LIN (Audi, BMW, Daimler-Chrysler, Volcano, Volvo, Volkswagen y Motorola). LIN fue estandarizado en el año 2000 y la producción de la primera serie de automóviles fue en el año 2001. Hoy sostiene un dominio en las aplicaciones automotrices donde coexiste con el protocolo CAN permitiendo la creación de redes jerárquicas con lo que se reduce el costo de desarrollo, producción, servicio y logística en vehículos electrónicos. La transferencia de datos es de 1 hasta 8 bytes, donde cada trama contiene un identificador de suma de verificación (*checksum*) como código de detección de errores. La especificación del protocolo LIN describe la capa física, la capa de enlace de datos y la capa de transporte conforme al modelo de referencia OSI. El protocolo LIN se utiliza en aplicaciones de cerraduras de puertas, control de luces, en sistemas de confort y control de clima, entre otros.

El protocolo J1850 pertenece a la clase B de la SAE, soporta una comunicación multimaestro ya que cada nodo tiene la capacidad de transmitir datos sobre el bus, para tener acceso al bus se utiliza CSMA/CR (*Carrier Sense Multiple Access with Collision Resolution*). La interface J1850 tiene dos formas de operación: modulación de ancho de pulso a 41.6 kbps con un par trenzado de cables diferenciales o ancho de pulso variable a 10.4 kbps con un sólo cable, soporta hasta 32 nodos con un bus de longitud máxima de 35 m. Este protocolo fue desarrollado en 1994 y no obtuvo popularidad ya que era el rival directo de CAN. Las dos principales razones de su poco uso fue su baja velocidad y el hecho que sólo se podía utilizar en Estados Unidos de América respaldado por Chrysler, GM y Ford, así como por importantes suministradores de componentes como

Intel y Motorola. Utiliza la comprobación de redundancia cíclica (CRC, *Cyclic Redundancy Check*) como método de detección de errores. La especificación del protocolo SAE J1850 describe la capa física y la capa de enlace de datos conforme al modelo de referencia OSI. Este protocolo es utilizado para aplicaciones compartidas de diagnóstico y datos en vehículos *on/off* (SAE J1850, 1995).

El protocolo DSI3 (*Third Generation Distributed System Interface*) pertenece a la clase B de la SAE, es uno de los primeros protocolos de seguridad crítica desarrollado por Motorola. Es un bus serial con un nodo Maestro y soporta hasta 15 nodos Esclavo, para tener acceso al medio se utiliza el método TDMA (*Time Division Multiple Access*). Este protocolo utiliza dos cables de transmisión con velocidades de transferencia de datos hasta 150 kbps y emplea la comprobación de redundancia cíclica como método de detección de errores. La especificación del protocolo DSI3 describe la capa física y la capa de enlace de datos conforme al modelo de referencia OSI y su uso principal es en aplicaciones de sistemas de bolsas de aire (DSI Consortium, 2011).

El protocolo VAN (*Vehicle Area Network*) fue desarrollado por un grupo de interés económico compuesto por PSA Peugeot Citröen y Renault a principios de la década de 1990. VAN está estandarizado en ISO 11519-3, es un protocolo serial que pertenece a la clase B de la SAE y opera en modo Multimaestro-Multiesclavo sobre un par de cables a una velocidad de hasta 125 kbps. Este protocolo puede ser utilizado en topologías en anillo, en árbol o estrella. El estándar ISO 11519-3 describe la capa física y la capa de enlace de datos conforme al modelo de referencia OSI. Para tener acceso al bus VAN utiliza la técnica CSMA/CD. VAN utiliza CRC de 15 bits como método de detección de errores y es empleado en aplicaciones de frenos ABS, tren motriz, luces frontales y limpia parabrisas, entre otros.

El protocolo Intellibus fue desarrollado por Boeing, inicialmente para aplicaciones militares aéreas, pero su uso también ha incurrido en el campo automotriz, pertenece a las clases B y C de la SAE. Este protocolo utiliza una

configuración de un nodo Maestro y hasta 63 nodos Esclavo sobre un par de cables trenzados, opera a velocidades de 1.5 Mbps y la máxima longitud del bus es de 30 metros, utiliza un código de detección de errores por paridad y CRC. La especificación del protocolo Intellibus describe la capa física, la capa de enlace de datos y la capa de presentación conforme al modelo de referencia OSI. Este protocolo es utilizado para controlar el motor y la transmisión, entre otros (Firlit, Sandoval y Ellerbrock, 2004).

El protocolo CAN (*Controller Area Network*) fue desarrollado a principios de la década de 1980 por Bosch GmbH, pertenece a las clases B y C de la SAE. Al día de hoy CAN es el protocolo más utilizado en redes automotrices. Utiliza una configuración donde todos los nodos son multimaestro, cualquier nodo puede tomar el control del bus, soporta hasta 32 nodos sobre un par trenzado de cables. CAN opera a velocidades de 10 kbps a una distancia de 1000 metros o 1 Mbps a una distancia de 40 metros. Al pasar de los años se han desarrollado diferentes estándares de CAN, por ejemplo, el ISO 11898 es el bus de campo más utilizado en la industria automotriz europea. Existe una versión de CAN tolerante a fallos descrita en ISO 11519-2. Todos los estándares CAN trabajan de forma similar y sus diferencias principales son las velocidades de transmisión y las capas que utilizan dependiendo de la aplicación CAN. Utiliza la comprobación de redundancia cíclica como método de detección de errores. La especificación del protocolo define la capa física y la capa de enlace de datos conforme al modelo de referencia OSI. CAN es utilizado en aplicaciones de control del motor, tren motriz, seguridad y confort, principalmente.

El protocolo Byteflight fue introducido por BMW en 1996 y posteriormente desarrollado por BMW, ELMOS, Infineon, Motorola y Tyco EC. Pertenece a la clase C de la SAE y su configuración permite que cualquier nodo pueda ser configurado como Maestro generando una señal de sincronización que proporciona una señal de reloj base para todos los demás nodos, el acceso al bus es regulado de acuerdo al Acceso Múltiple por División Flexible en el Tiempo

(FTDMA, *Flexible Time Division Multiple Access*). Este protocolo permite el uso de un bus completo utilizando dos o tres cables o la tecnología de fibra óptica empleando velocidades de transferencia de datos de hasta 10 Mbps. El protocolo une las ventajas de los métodos síncronos y asíncronos. La especificación de Byteflight describe la capa física y la capa de enlace de datos conforme al modelo de referencia OSI. La principal área de aplicación de este protocolo es en sistemas críticos de seguridad y utiliza la comprobación de redundancia cíclica como método de detección de errores. Su principal aplicación es en los tensores del cinturón de seguridad donde se requiere una rápida respuesta en tiempo de ejecución (BMW AG, 1999).

El protocolo FlexRay es un bus de comunicación serial creado por el consorcio FlexRay (BMW, Daimler-Chrysler, Philips Semiconductor y Freescale Semiconductor). Pertenece a la clase D de la SAE y no está restringido a ninguna topología. FlexRay está basado en una arquitectura de disparo por tiempo (TTA, *Time Triggered Architecture*) con tiempos de respuesta conocidos. Las tareas realizadas por el protocolo FlexRay se basan en un esquema de acceso estático (TDMA) y flexible (FTDMA), soporta hasta 64 nodos conectados y ofrece velocidades de hasta 10 Mbps por un par trenzado de cables o por fibra óptica. FlexRay es un sistema de comunicaciones síncrona y asíncrona, tolerante a fallos debido a su capacidad de soportar dos canales y el uso de algoritmos de sincronización FTM (*Fault Tolerant Midpoint Algorithm*). Utiliza CRC de 24 bits como método de detección de errores. La especificación del protocolo FlexRay describe la capa física y la capa de enlace de datos conforme al modelo de referencia OSI. La principal área de aplicación es en sistemas críticos de seguridad y aplicaciones *x-by-wire* en el automóvil.

El protocolo MOST (*Media Oriented Systems Transport*) fue creado a inicios del año 1998 por el consorcio MOST, pertenece a la clase D de la SAE. Este protocolo utiliza una configuración multimaestro que soporta hasta 64 nodos conectados a un bus de fibra óptica trabajando a velocidades de hasta 24.8 Mbps.

Para tener acceso al bus se utiliza el método TDMA. La especificación del protocolo MOST define las siete capas del modelo de referencia OSI para la comunicación de los datos. Se utiliza un CRC como método de detección de errores además de un reconocimiento con respuesta automática; las aplicaciones típicas son multimedia, entretenimiento, navegación GPS, audio y video.

El protocolo IDB-1394 fue desarrollado por el foro IDB y el 1394 *Trade Association*, pertenece a la clase C de la SAE y es utilizado para transmitir video, audio y datos multimedia sobre un par trenzado dentro del vehículo. Una red IDB-1394 se puede implementar de forma simple, de manera lineal o en anillo, y soporta velocidades de transmisión de datos de hasta 100 Mbps; puede configurarse como una red multiplexada que soporta video múltiple y audio de forma continua; toda esta información puede ser transmitida simultáneamente sobre la misma capa física reduciendo peso y cableado complejo. El protocolo IDB-1394 utiliza el método de CRC de 15 bits como detección de errores y es una versión automotriz del IEEE 1394. La especificación del protocolo IDB-1394 describe la capa física, la capa de enlace de datos, la capa de transporte y la capa de aplicación conforme al modelo de referencia OSI. Su uso es para aplicaciones de sistemas de visualización, cámaras en el automóvil y entretenimiento, con los cuales los pasajeros puedan reproducir DVDs, ver programas de televisión digital y accesar a la navegación del vehículo (Wiewesiek, 2007).

El protocolo D2B (*Domestic Digital Bus*) es un bus de datos óptico creado por Matsushita y Philips en 1992, y promocionado por el *Optical Chip Consortium* (C&C Electronics y Becker). El bus interconecta audio y video en una estructura de anillo y estrella dentro del vehículo. Pertenece a la clase D de la SAE y proporciona velocidades de transmisión de hasta 12 Mbps. En 1998, Daimler-Chrysler utilizó este protocolo en sus vehículos de lujo, el sistema se utilizaba para la conexión de dispositivos de audio y fue adoptado por Mercedes Benz en el año 2007. La máxima distancia con fibra óptica como medio físico es 10 metros, cuando se utiliza un sólo empalme, o hasta 7 metros, cuando se utilizan dos

empalmes. Se utiliza principalmente en aplicaciones de telemetría, sistemas de CD y reconocimiento de voz, entre otros (Leen y Hefferman, 2002).

MML (*Mobile Multimedia Link*) es un protocolo utilizado para aplicaciones multimedia, pertenece a la clase D de la SAE. Utiliza una arquitectura Maestro/Esclavo y soporta hasta 16 nodos sobre una red estrella de fibra óptica, su velocidad de operación es de 100 Mbps con una longitud máxima de cableado de 10 metros. La empresa Delphi Packard Electric Systems desarrolló el protocolo MML basado en una capa física de fibra óptica. Este protocolo utiliza el método CRC como detección de errores y es utilizado para aplicaciones de equipo de audio y video, dispositivos de visualización y navegación del vehículo, entre otros (Leen y Hefferman, 2002).

La Tabla 2.1 muestra las principales características de los buses de campo.

Tabla 2.1. Comparación de los protocolos de comunicaciones en el campo de la automoción.

Protocolo	Clase SAE	Control	Capa física	Acceso al medio	Velocidad	Capas OSI	Detección de errores	Aplicación
IEBus	A	Multimaestro	Par trenzado	CSMA/CD	3.9-18 kbps	1, 2	ACK	Control y diagnóstico
MI	A	Maestro/Esclavo	1 cable	NA	20 kbps	1, 2	ACK	Espejos laterales, asientos, cerraduras, luces frontales.
LIN	A	Maestro/Esclavo	Cable simple	Invitación a transmitir	20 kbps	1, 2, 5	Checksum; Bit de paridad	Elevalunas, Regulación del clima, Luces
J1850	B	Multimaestro	1 o 2 cables	CSMA/NDA	10.4-41.6 kbps	1, 2	CRC; Cheksum	Diagnóstico, Datos ON/OFF
DSI3	B	Maestro/Esclavo	Par trenzado	TDMA	150 kbps	1, 2	CRC	Bolsas de aire
VAN	B	Multimaestro Multiesclavo	Par trenzado	CSMA/CD	125 kbps	1, 2	CRC	Tren motriz, ABS, Limpia parabrisas
Intellibus	B y C	Maestro/Esclavo	Par de cables	NA	1.5 Mbps	1, 2, 6	CRC; Paridad	Motor, Transmisión, Confort
CAN	B y C	Multimaestro	Par trenzado	CSMA/CA	1 Mbps	1, 2	CRC; Bit de paridad	Frenos ABS, Control de motor, Tren motriz, Seguridad
Byteflight	C	Multimaestro	2 o 3 cables; Fibra óptica	FTDMA	10 Mbps	1, 2	CRC	Tensor cinturón seguridad
FlexRay	D	Multimaestro	Fibra óptica	TDMA; FTDMA	10 Mbps	1, 2	CRC; Bus guardián	Sistemas de emergencia, *Break by wire*, *Steer by wire*
MOST	D	Multimaestro	Fibra óptica	TDM; CSMA/CA	24 Mbps	1, 2, 3, 4, 5, 6, 7	CRC; Servicio de Sistema	Entretenimiento, Navegación, Servicios de información
IDB-1394	D	Maestro/Esclavo	Par de cables	NA	100 Mbps	1, 2, 4, 7	NA	Entretenimiento, Sistemas de visualización
D2B	D	Maestro/Esclavo	Fibra óptica	NA	12 Mbps	1, 2	Paridad	Telemetría, Sistemas CD, Reconocimiento de Voz
MML	D	Maestro/Esclavo	Fibra óptica	NA	100 Mbps	1, 2	CRC	Multimedia

3. Protocolo de Comunicaciones LIN

El protocolo de comunicaciones LIN, conocido como bus LIN, fue propuesto por el consorcio LIN con la finalidad de crear una red estándar para la comunicación multiplexada en redes automotrices de bajo costo. A pesar de que el protocolo de comunicaciones CAN cubre las necesidades de ancho de banda y detección avanzada de errores en aplicaciones de tiempo real crítico en redes automotrices, los costos de hardware y de software derivados de su implementación se han vuelto prohibitivos para dispositivos de menor rendimiento como son controladores de potencia de elevalunas y asientos, e interruptores de encendido/apagado, entre otros. El protocolo de comunicaciones LIN proporciona comunicación rentable en aplicaciones que no requieren gran ancho de banda ni la versatilidad de CAN; se puede implementar a un menor precio usando el transmisor/receptor estándar serial universal asincrónico (UART, *Universal Asynchronous Receiver-Transmitter*) que se encuentra incrustado en la mayoría de microcontroladores de ocho bits. Se puede decir que LIN fue diseñado para complementar el funcionamiento de los demás protocolos de comunicaciones en redes automotrices.

3.1. Desarrollo Histórico

En el año de 1998, durante la conferencia sobre vehículos organizada por la Asociación de Ingenieros Alemanes (VDI, *Verein Deutscher Ingenieure*), surgió la idea de diseñar un estándar de comunicaciones unificado y para ello se creó el

Consorcio LIN con la participación de Daimler Chrysler, BMW, Audi, Volkswagen, Volcano Technologies y Motorola.

En noviembre de 1998 se celebró la primera reunión del Consorcio LIN, en ella se definieron los requisitos básicos de la red y se desarrolló el sistema de datos de bus LIN. Para el verano de 1999 se dio a conocer la primera versión del protocolo LIN, versión 1.0, el protocolo consistía en la capa física operada por la capa de aplicación; esta versión se presentó oficialmente en Detroit durante la conferencia de la Sociedad de Ingenieros Automotrices (SAE) celebrada en marzo de 2000; durante este año se realizaron las versiones 1.1 y 1.2.

En el año 2001 se llevó a cabo la primera implementación de LIN en la producción en serie de vehículos clase SL de la empresa Daimler Chrysler (Müller, Sommer y Stegemann, 2009), durante este año se publicó la versión 1.3 del protocolo de comunicaciones, cuyas principales novedades fueron la revisión de la capa física y la capa de enlace de datos. En septiembre de 2003 se presentó la versión 2.0 con el objetivo de definir los requisitos para facilitar el desarrollo e implementación de redes LIN. En 2006 se presentó la versión 2.1, en la cual se añadieron las capas de transporte y de diagnóstico; hasta el momento la versión actual es la 2.2A que se publicó en el año 2010. La Figura 3.1 presenta la evolución del protocolo de comunicación LIN.

3.2. Sistemas Eléctricos Multiplexados

En un vehículo moderno coexisten diversos sistemas multiplexados[6] que realizan una gran variedad de funciones; los principales protocolos de comunicaciones son CAN y LIN, como se muestra en la Figura 3.2.

[6] Un sistema eléctrico multiplexado se puede definir como una red o bus de campo en la que los nodos activos comparten una misma infraestructura para distribuir información con la finalidad de controlar interruptores, sensores y pantallas, entre otros.

Figura 3.1. Evolución del protocolo de comunicaciones LIN.

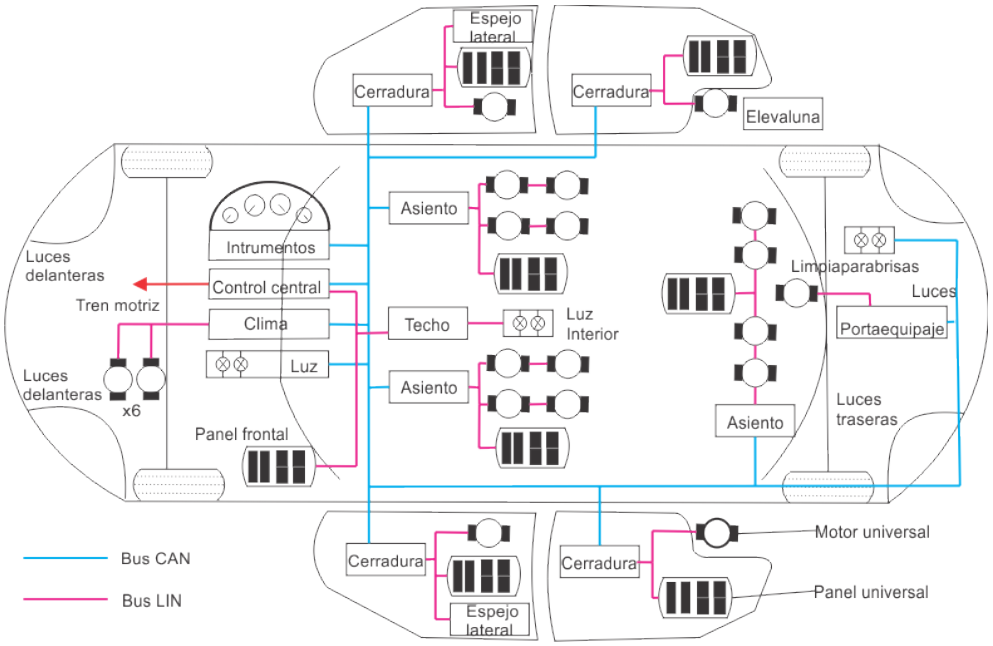

Figura 3.2. Vehículo con sistemas eléctricos multiplexados.

El protocolo de comunicaciones CAN es el protocolo *de facto* utilizado en aplicaciones de tiempo real crítico en una red automotriz, algunas de estas aplicaciones hacen referencia al tren motriz y redes centrales, entre otras. El protocolo de comunicaciones MOST se utiliza en aplicaciones multimedia debido

al ancho de banda que soporta. El protocolo de comunicaciones LIN, junto al protocolo de comunicaciones CAN de baja velocidad, se utilizan para aplicaciones simples como interruptores y sensores de baja velocidad de conmutación.

El bus LIN está diseñado para complementar al protocolo CAN en las siguientes aplicaciones:

- Chasis (Sensores de lluvia, de luz y de presencia de sol).
- Puertas (Espejos laterales y ventanas laterales).
- Motor (Sensores, pequeños motores, llantas, señal de encendido, radio y clima).
- Asientos (Motores de posición de asientos y sensor de ocupación).

En la actualidad existen cuatro áreas de aplicación de las redes automotrices, en donde cada una utiliza un protocolo de comunicaciones en particular ya que hasta la fecha no existe un sólo protocolo universal; además cada área de aplicación tiene sus requerimientos especiales, por ejemplo:

- Las aplicaciones de tren motriz requieren de protocolos de comunicaciones que operen en tiempo real.
- Las aplicaciones multimedia requieren grandes anchos de banda.
- Las aplicaciones críticas requieren de sistemas tolerantes a fallos y de respuesta en tiempo real.
- Los sensores y actuadores inteligentes han generado ECUs complejas con características específicas.

La Figura 3.3 muestra la relación entre la velocidad y el costo de los diferentes protocolos de comunicaciones en redes automotrices.

La arquitectura de protocolos LIN, de acuerdo al modelo de referencia OSI, incluye cuatro capas: física, transporte, enlace de datos y aplicación, e incorpora una capa especial para el control del nodo llamada capa de supervisor (véase Figura 3.4).

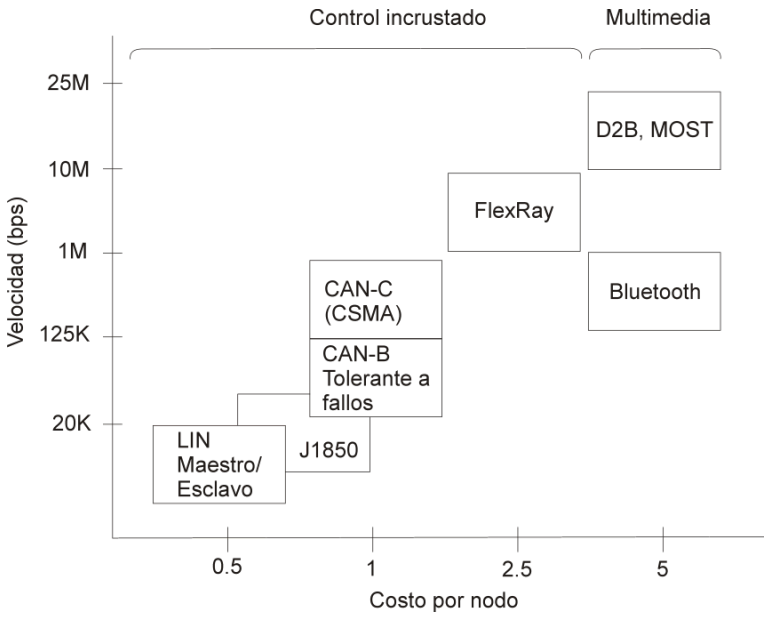

Figura 3.3. Relación entre la velocidad y el costo de varias redes automotrices de tiempo real.

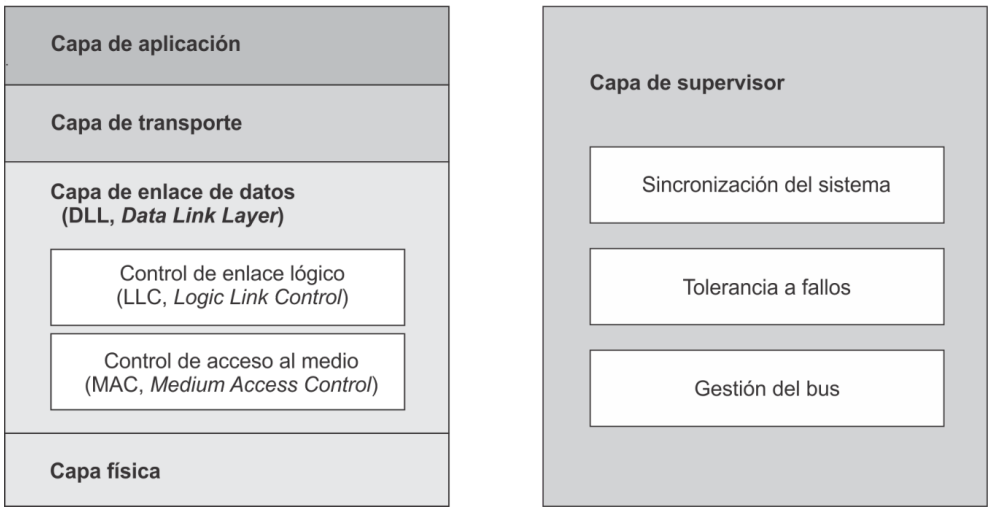

Figura 3.4. Arquitectura de protocolos LIN.

El protocolo LIN basa su funcionamiento en un *cluster*[7] LIN, el cual considera el modelo Maestro/Esclavo y consta de un nodo Maestro y uno o más nodos Esclavo; el nodo Maestro se compone de una tarea de Maestro y una tarea de Esclavo, mientras que los nodos Esclavo sólo contienen tareas de Esclavo; este protocolo pertenece a la clase A de la SAE dada su limitada velocidad de transferencia de datos (véase apartado 2.2).

[7] Un grupo o una red de grupos.

En la actualidad, una red automotriz utiliza una combinación de: a) una red LIN, para aplicaciones de bajo costo; b) una red CAN, para la comunicación del tren motriz; y c) una red FlexRay, para comunicaciones de datos de alta velocidad en aplicaciones multimedia y de suspensión activa.

Las redes LIN suelen ser subredes de una red principal CAN para interconectar con el nodo principal y su uso está destinado a aplicaciones no críticas. Las principales características de una red LIN son: a) la auto sincronización de los nodos Esclavo empleando una secuencia de reloj generada por el nodo Maestro, b) la existencia de una tabla de programación (*Schedule table*) para determinar cuándo y qué trama debe ser transmitida, y c) la capacidad de poner a los nodos Esclavo en modo descanso (*sleep*) para ahorrar energía cuando éstos no tienen nada que transmitir.

Las principales características de una red LIN son:

- El bus LIN se puede implementar utilizando una UART.
- La capa física se encuentra estandarizada en ISO-9141.
- El medio físico es monoalámbrico bidireccional.
- Las velocidades de transferencia de datos que soporta el bus LIN van desde 1 hasta 20 kbps.
- Permite la utilización de 16 identificadores LIN (LIN ID) y puede transferir hasta 8 bytes de datos.
- Modelo Maestro/Esclavo.
- Soporta una longitud máxima del bus de hasta 40 metros.

3.3. Arquitectura de Protocolos LIN

3.3.1. Capa física

La capa física de un sistema de comunicaciones define los aspectos del medio físico para la transmisión de datos entre los nodos de una red, los más importantes hacen referencia a los niveles de señal, representación, sincronización y tiempos en los que los bits se transfieren al bus (Lawrenz, 1997).

Para estudiar la capa física del protocolo LIN, a continuación, se describen sus especificaciones eléctricas y mecánicas.

3.3.1.1. Especificaciones eléctricas

En una red LIN se utiliza el tiempo de bit (T_{bit}) del nodo Maestro como referencia y se define como el tiempo requerido para transmitir un bit.

Durante el procedimiento de sincronicación, el Byte Sync consiste del dato 0x55 dentro del campo Sync, como se muestra en la Figura 3.5, el procedimiento para que un nodo Esclavo pueda sincronizarse se basa en la medición del tiempo entre los flancos de bajada del campo Sync. Estos flancos están disponibles en distancias de 2, 4, 6 y 8 T_{bit} lo que permite realizar el cálculo de T_{bit}.

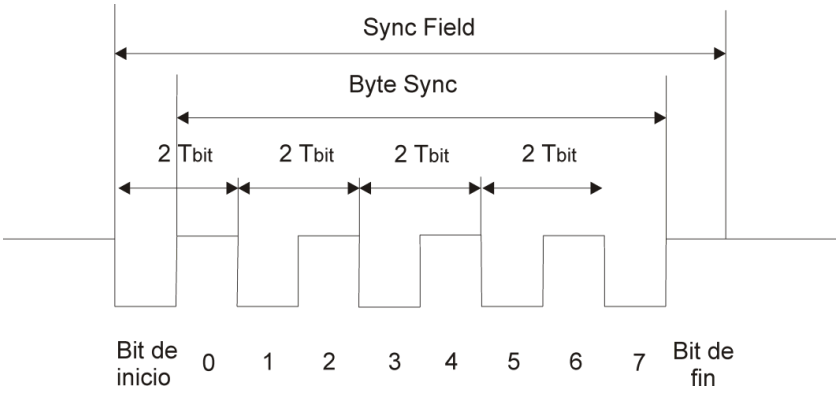

Figura 3.5. Campo Sync.

La Figura 3.6 muestra los tiempos de muestreo de bit. Para realizar el muestreo de un bit se utiliza el flanco de bajada del bit de inicio (BFS) y debe tener una exactitud de t_{BFS}.

A partir de la especificación LIN 2.x, en el mercado existen diferentes soluciones para el muestreo del bit de inicio, sin embargo, se permiten todos los métodos de muestreo del bit de inicio que cumplan con el tiempo t_{BFS}.

Después de la sincronización del bit de inicio, se debe muestrear el bit de datos dentro de la ventana conformada por el tiempo de muestreo de bit más cercano T_{EBS} (*earliest bit sample time*) y el tiempo de muestreo de bit más lejano T_{LBS} (*latest bit sample time*). La lectura de los siguientes bits se realiza dentro de la ventana de muestreo, desde el primer bit de datos hasta el último bit utilizando la ventana de repeticiones T_{SR}.

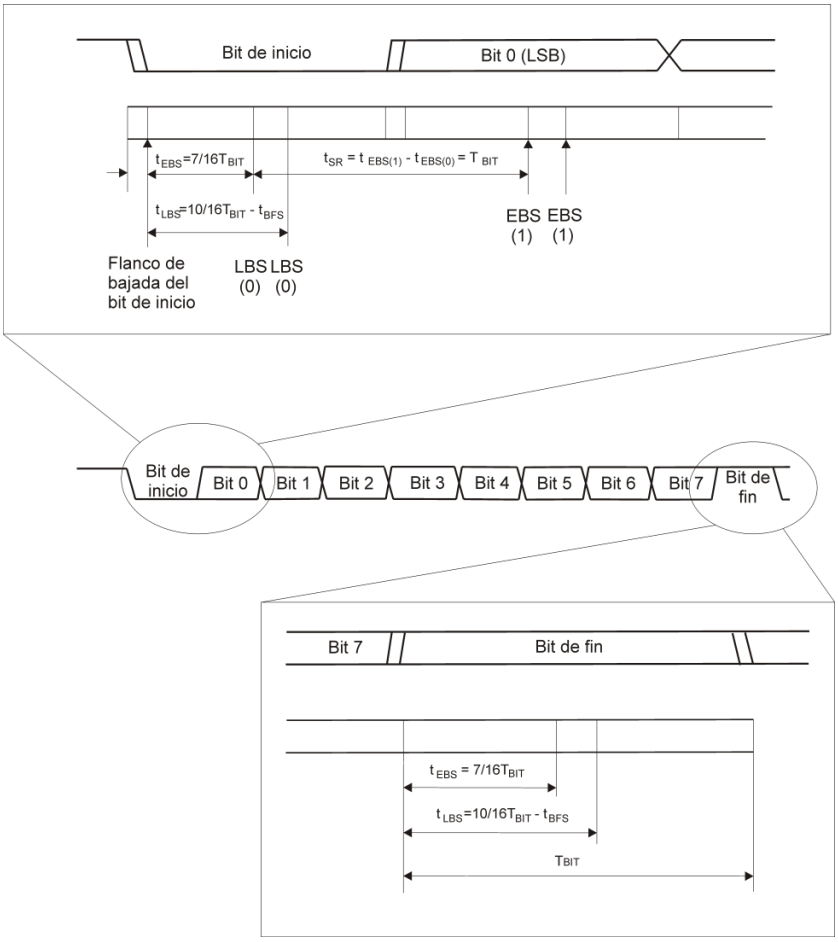

Figura 3.6. Tiempos de muestreo de bit.

El controlador de línea se basa en el estándar ISO 9141 (ISO/IEC 7498, 1989), y consiste en una línea de bus bidireccional que conecta al bus con cada uno de los nodos del *cluster* LIN; el estándar describe que el controlador de línea

se debe conectar con una resistencia de terminación y un diodo conectado al nodo positivo de la batería V_{BAT} (véase Figura 3.7). El diodo es necesario para prevenir un suministro sin control del bus a la ECU en caso de pérdida de voltaje de la batería.

Todas las ECUs en un sistema LIN obtienen el suministro de energía V_{BAT} a través de un conector, mientras que los componentes electrónicos dentro de esta unidad se alimentan del voltaje interno V_{SUP}, en la mayoría de casos es diferente a V_{BAT} (véase Figura 3.7). Esta configuración se utiliza para filtrar y proteger de los cambios repentinos de voltaje a los sistemas semiconductores que conforman la ECU.

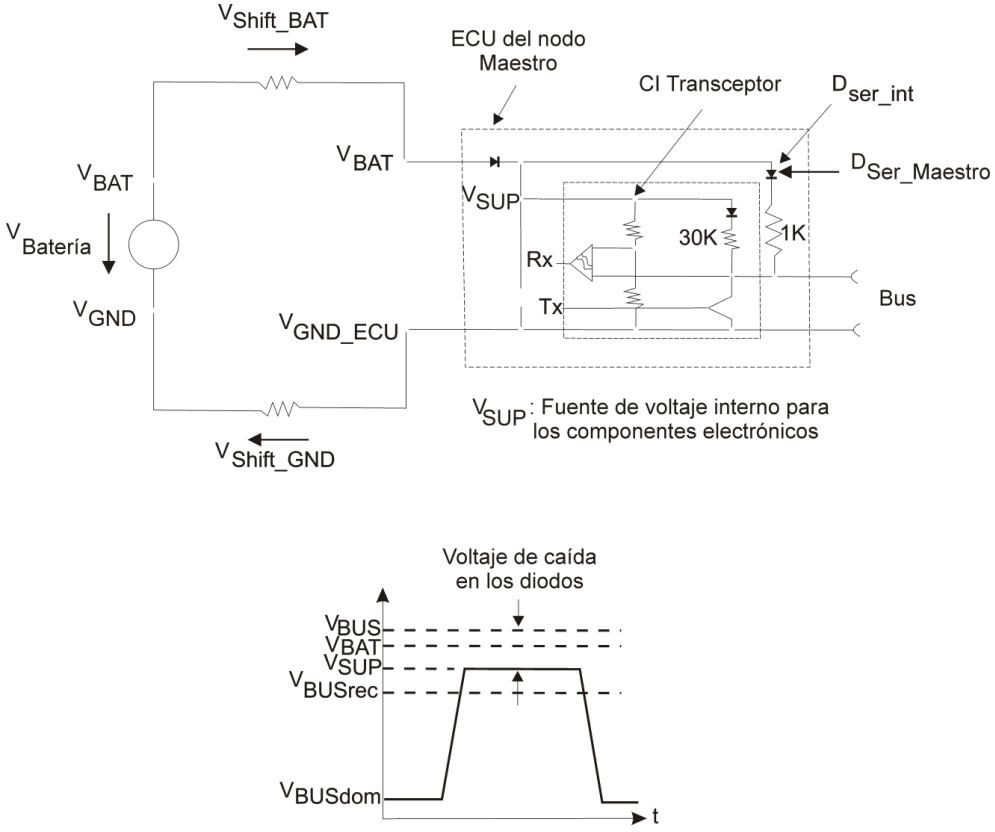

Figura 3.7. Diferencia entre la fuente de voltaje externa V_{BAT} y la fuente de voltaje interna V_{SUP}.

Para la correcta transmisión y recepción de un bit es necesario que la señal tenga un nivel de voltaje correcto (dominante o recesivo) al momento de muestrear el bit por parte del receptor, éste se tiene que asegurar que la señal esté disponible con un nivel de voltaje correcto en el momento que se toma la

muestra en el nodo Esclavo (véase Figura 3.8). La Figura 3.9 muestra los parámetros de los tiempos que impactan el comportamiento del bus LIN.

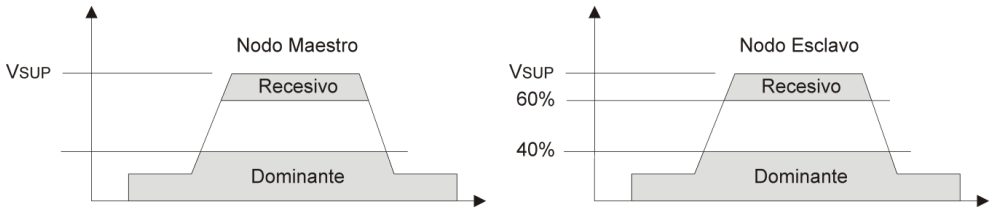

Figura 3.8. Niveles de voltaje sobre el bus LIN

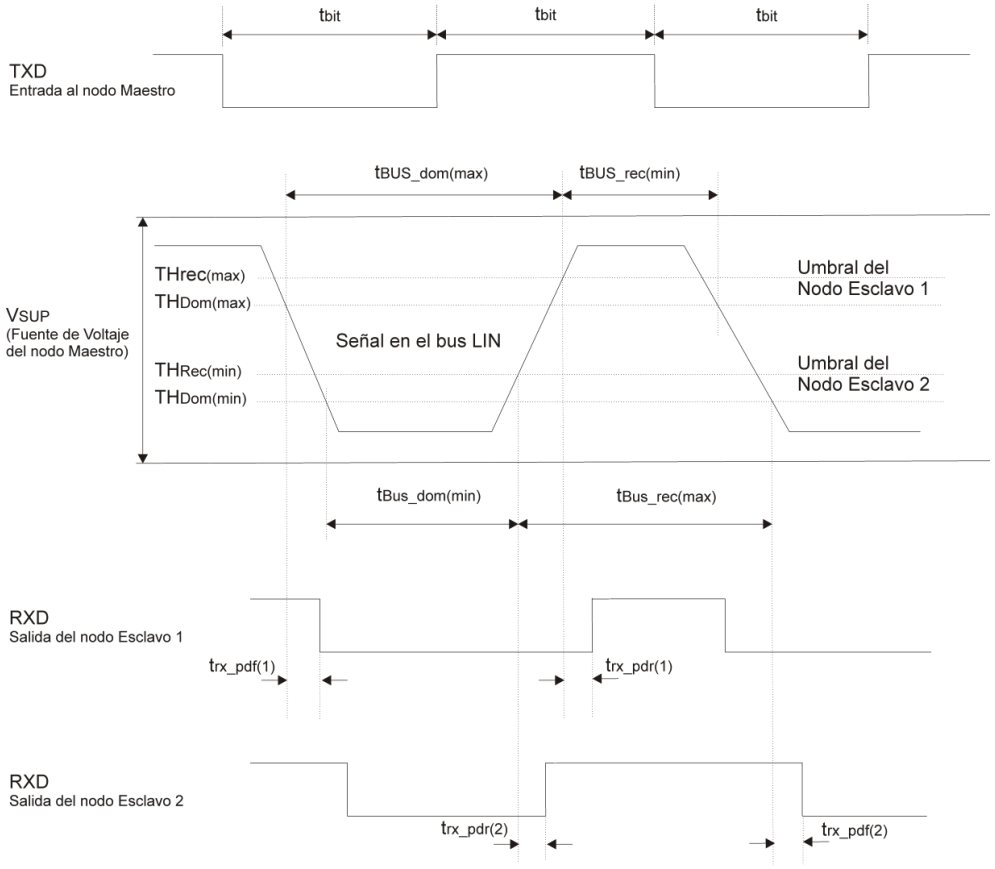

Figura 3.9. Definición de los tiempos del bus LIN.

La máxima velocidad de transferencia de datos del bus LIN está limitada por la velocidad de transferencia de datos que soportan los transceptores. Para lograr que la señal que se propaga por el bus sea lo más simétrica posible, la capacitancia del bus debe mantenerse en un valor bajo, la cual se puede controlar mediante la constante de tiempo RC. La Tabla 3.1 muestra las características y parámetros principales del bus LIN.

Tabla 3.1. Características y parámetros del bus LIN.

Descripción	Parámetro	Mínimo	Típico	Máximo	Unidad
Longitud total del Bus LIN	LEN_{Bus}	-	-	40	m
Capacitancia total del bus incluyendo la capacitancia del nodo Maestro y las capacitancias de los nodos Esclavo	C_{Bus}	1	4	10	nF
Constante de tiempo del sistema completo	τ	1	-	5	µS
Capacitancia del nodo Maestro	$C_{Maestro}$	-	220		pF
Capacitancia del nodo Esclavo	$C_{Esclavo}$	-	220	250	pF
Capacitancia de la línea	C'_{Linea}	-	100	150	pF/m

El número de nodos interconectados a un *cluster* no debe ser mayor a 16, debido a que la impedancia de la red no permite una comunicación libre de fallas cuando se excede este número de nodos.

Cuando una ECU pierde la conexión con la fuente de voltaje o con tierra, genera un fallo que no debe interferir con la comunicación con los demás nodos del *cluster* LIN, y una vez que se recupera del fallo, tanto la ECU como el bus LIN deben seguir operando con normalidad. En aplicaciones automotrices son comunes los cortos circuitos que se generan en la batería del automóvil y ocasionan dichas situaciones.

Los dispositivos semiconductores de la capa física deben cumplir los requerimientos de protección contra descarga electrostática del cuerpo humano de acuerdo al IEC 61000-4-2:1995. El mínimo voltaje de descarga es de ±2000 V. El nivel de descarga electrostática (ESD, *Electrostatic Discharge*) para aplicaciones automotrices puede ser de hasta ±8000 V en los conectores de una ECU.

3.3.1.2. Especificaciones mecánicas

En la actualidad el mercado ofrece soluciones LIN que van desde transceptores y microcontroladores de diferentes fabricantes (Infineon, Freescale y Microchip, entre otros) hasta módulos de software conocidos como núcleos FPGA.

En un vehículo, los transceptores tienen que funcionar en un ambiente hostil a los componentes electrónicos, ya que tienen que lidiar con factores de suciedad,

calor y humedad que pueden afectar su funcionalidad, así mismo, tienen que resistir interferencias electromagnéticas, cambios bruscos de voltaje debido a cargas, cortocircuitos y otras fluctuaciones generadas por otras partes eléctricas del vehículo.

Existen características especiales implementadas en algunos transceptores, por ejemplo, el transceptor Philips TJA1020 tiene la habilidad de reconocer si la señal *sleep* proviene de manera local o externa y notificar el evento al controlador (Philips Semiconductor, 2004); otros transceptores tienen un regulador de voltaje integrado para suministrar alimentación a un microcontrolador Microchip MCP201 (Microchip Technologies Inc., 2007), mientras otros sólo tienen la habilidad de controlar un regulador externo como es el caso del microcontrolador Freescale MC3339.

Los transceptores se construyen de acuerdo a la especificación LIN y son adaptados a la industria automotriz. La mayoría de los transceptores pueden manejar niveles de voltaje de hasta 18 V, el cual es el nivel máximo del bus LIN. Existen en el mercado transceptores que pueden operar con fuentes de alimentación de hasta 40 V, por ejemplo, Infineon TLE8458Gx (Infineon Technologies AG, 2009), estos transceptores podrían, en teoría, adaptarse al uso en un vehículo con sistemas de 24 V, sin embargo, si se suministra con 24 V, los transceptores no podrían operar dentro de los niveles requeridos por la especificación y como resultado sería difícil mencionar cómo afectarían al sistema en general las propiedades de velocidad e interferencias electromagnéticas.

La implementación de un nodo Maestro o un nodo Esclavo LIN no restringe los requerimientos de un tipo específico de hardware a utilizar, ya que es la aplicación en conjunto la que lo determina. En la actualidad el mercado ofrece soluciones que incorporan las características básicas de LIN, un ejemplo es el microcontrolador Freescale MC9S12P128RMV1, que mediante hardware habilita una bandera especial para identificar la llegada del campo de

sincronización de la cabecera del mensaje (Freescale Semiconductor, 2013); otros ejemplos son los microcontroladores PIC16C432, PIC16C433 y PIC16F1829LIN de Microchip (Microchip Technologies Inc., 2013), que incorporan un transceptor LIN; dichos ejemplos permiten la implementación de controladores LIN compactos, pequeños y fáciles de usar.

El mercado de productos LIN no sólo consiste de microcontroladores y transceptores, ya que muchas compañías se están enfocando en soluciones que son más efectivas en costo para grandes volúmenes de producción; es el caso de los módulos LIN diseñados en ASIC (*Aplication Specific Integrate Circuit*) o en FPGA (*Field Programmable Gate Array*). Un ejemplo es el núcleo LIN de la empresa *Cast*, el cual transmite y recibe tramas completas para llevar a cabo comunicación serial utilizando la especificación LIN 2.2A y se puede implementar como un nodo Maestro o como un nodo Esclavo (Cast Inc., 2013).

3.3.2. Capa de enlace de datos

En este apartado se describen los fundamentos y principios básicos de la capa de enlace de datos (DLL, *Data Link Layer*), la cual se divide en dos subcapas: Control de enlace lógico (LLC, *Logic Link Control*) y Control de acceso al medio (MAC, *Medium Access Control*), las cuales se describen a continuación.

La subcapa LLC describe la parte alta de la capa DLL y realiza las siguientes funciones:

- Filtrar mensajes (*frame acceptance filtering*): El identificador de una trama no indica la dirección destino, pero define el contenido del mensaje y mediante esta función todo receptor activo en la red determina si el mensaje es relevante o no para sus propósitos.
- Proceso de recuperación (*recovery management*): La subcapa LLC proporciona la capacidad de retransmisión automática de tramas cuando una trama presenta errores durante su transmisión, dicho servicio se confirma al usuario hasta que la transmisión se completa con éxito.

- Validación de mensajes (*message validation*): Una trama de datos es válida tanto para el nodo Maestro como para los nodos Esclavo si no se detectan errores hasta el fin de la trama.

La subcapa MAC describe la parte baja de la capa DLL. Esta subcapa representa el núcleo del protocolo de comunicaciones LIN ya que se encarga de presentar los datos provenientes de la subcapa LLC y de aceptar los mensajes a ser transmitidos a la subcapa LLC. La subcapa MAC es supervisada por la capa de Supervisor.

3.3.2.1. Transmisión de mensajes

Una trama LIN consiste de una cabecera y de una respuesta (véase Figura 3.10); la cabecera tiene una longitud fija mientras que la respuesta puede contener de 0 a 8 bytes de datos. El tiempo de respuesta entre trama es el tiempo que le toma a un nodo Esclavo responder a la orden del nodo Maestro; este tiempo puede variar entre los nodos de la red ya que depende de la implementación del hardware y del software en cada nodo. Al final del campo de respuesta se encuentra la suma de verificación (*checksum*).

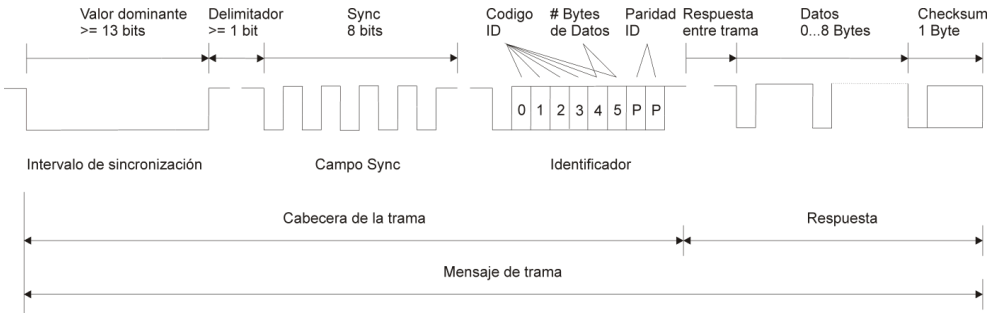

Figura 3.10. Trama de datos LIN.

La cabecera se divide en tres campos:

- Intervalo de sincronización: Consiste en un bit a nivel 'cero lógico'. Este intervalo permite a los nodos Esclavo detectar el mensaje a transmitir sobre el bus. De acuerdo a la especificación LIN se permite a los nodos

Esclavo tener una frecuencia de reloj que difiera en un 15% con la frecuencia del nodo Maestro.

- Campo de sincronización: El nodo Maestro inicia una transmisión enviando una cabecera, los nodos Esclavo son capaces de sincronizar sus relojes cada vez que se recibe un nuevo mensaje. Con lo anterior no es necesario implementar osciladores y resonadores costosos en los nodos Esclavo y solamente es necesario implementar un resonador exacto en el nodo Maestro como referencia de tiempo. Los nodos Esclavo sincronizan su reloj midiendo el tiempo que toma desde el primer flanco de bajada del campo *Sync* (flanco de bajada del bit de inicio) hasta el quinto flanco de bajada (bit 7 del campo *Sync*) y lo divide entre 8 para obtener el tiempo de bit o la velocidad de transferencia del nodo Maestro.
- Identificador (ID): Se encarga de identificar el contenido de la parte de datos de la trama y está protegido con dos bits de paridad (véase Figura 3.11). Los nodos Esclavo son direccionados por el campo ID. Cabe señalar que en una red LIN los nodos no tienen una dirección física por lo que utilizan una lista previamente programada de identificadores (IDs) válidos en su memoria para filtrar los mensajes de su interés.

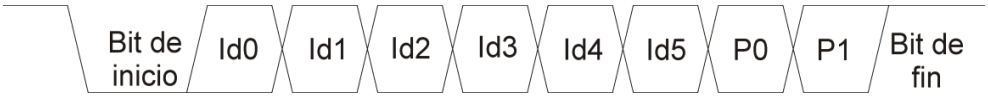

Figura 3.11. Mapeo del campo del byte del identificador y del protector del identificador.

El último byte de la trama de datos es la suma de verificación (*checksum*) y se obtiene de la suma invertida de todos los bytes de datos con el bit de acarreo en cada operación. La especificación LIN 1.x define esta suma de verificación conocida como clásica y que el nodo Maestro utiliza para la comunicación con los nodos Esclavo.

La especificación LIN 2.x define el cálculo de la suma de verificación utilizando todos los bytes, el identificador protegido y el bit de acarreo en cada operación, procedimiento conocido como suma de verificación mejorada y se

utiliza para la comunicación del nodo Maestro con sus nodos Esclavo. El nodo Maestro es el responsable de elegir entre la suma de verificación clásica o mejorada dependiendo con que nodo Esclavo se comunicará. En este trabajo de tésis se utiliza la suma de verificación mejorada.

El valor mínimo para la transmisión de una trama corresponde al número de bits enviados por el bus (sin espacios entre bytes y respuesta). El intervalo de sincronización tiene una longitud mínima de 14 bits (13 bits nominales más un bit delimitador), por lo tanto:

$$t_{Cabecera\_Mínima} = 34 * t_{bit} \qquad \textit{Ecuación 3.1}$$

$$t_{Respuesta\_Mínima} = 10(n_{datos} + 1) * t_{bit} \qquad \textit{Ecuación 3.2}$$

$$t_{Trama\_Mínima} = t_{Cabecera\_Nominal} + t_{Respuesta\_Nominal} \qquad \textit{Ecuación 3.3}$$

Donde t_{bit} es el tiempo requerido para transmitir un bit, como se definió en el apartado 3.3.1.1.

La máxima longitud de bit de la trama proporciona el tiempo máximo en el cual la transmisión de la trama se debe completar. Se calcula de acuerdo a la especificación LIN como:

$$t_{Cabecera\_Máxima} = 1.4 * t_{Cabecera\_Mínima} \qquad \textit{Ecuación 3.4}$$

$$t_{Respuesta\_Máxima} = 1.4 * t_{Respuesta\_Mínima} \qquad \textit{Ecuación 3.5}$$

$$t_{Trama\_Máxima} = t_{Cabecera\_Máxima} + t_{Respuesta\_Máxima} \qquad \textit{Ecuación 3.6}$$

Como se puede visualizar, la mínima longitud de una trama se extiende al 40% para proporcionar la máxima longitud permitida. Esto se lleva a cabo de tal manera que las tramas suscritas tendrán tiempo entre tramas para realizar cálculos y otras tareas.

3.3.2.2. Tipos de tramas

Los tipos de trama hacen referencia a las condiciones que deben ser válidas para transmitir una trama. Algunos tipos de tramas se utilizan para propósitos específicos, como se describe a continuación.

3.3.2.2.1. Trama incondicional

La trama incondicional contiene identificadores con valor desde cero hasta 59. La cabecera de una trama incondicional se transmite por el nodo Maestro y los nodos Esclavo responden a la cabecera. Todos los nodos Esclavo reciben la trama incondicional.

El identificador puede tener diferentes significados para los nodos y puede habilitar tres maneras de transferir los datos entre el nodo Maestro y los nodos Esclavo: nodo Maestro a nodos Esclavo, nodo Esclavo a nodo Maestro y nodo Esclavo a nodo Esclavo.

La Figura 3.12 muestra el proceso mediante el cual el nodo Maestro transmite una trama completa que contiene una orden (*command frame*), dicha trama consta tanto de la cabecera como de la respuesta para uno o más nodos Esclavo.

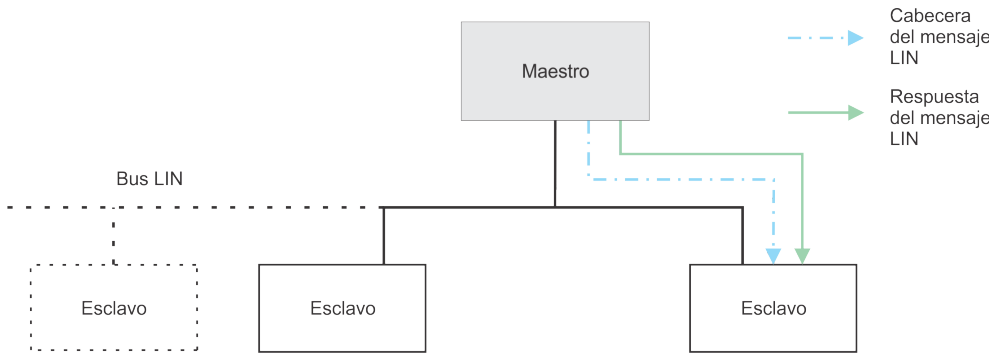

Figura 3.12. Transmisión de datos entre el nodo Maestro y los nodos Esclavo.

La Figura 3.13 muestra el proceso mediante el cual el nodo Maestro solicita una respuesta de un nodo Esclavo en específico (*polling*), dicha solicitud se realiza a través de la petición (*request frame*).

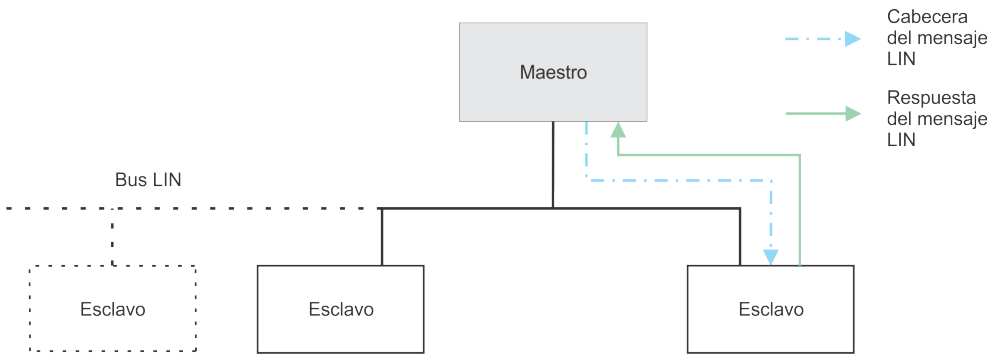

Figura 3.13. Transmisión de datos de un nodo Esclavo al nodo Maestro.

La comunicación entre nodos Esclavo se ilustra en la Figura 3.14, y se lleva a cabo cuando un nodo Esclavo transmite su respuesta a uno o más nodos Esclavo; este proceso sólo puede llevarse a cabo cuando el nodo Maestro no tiene datos a transmitir y comúnmente se envía información entre un sensor y un actuador.

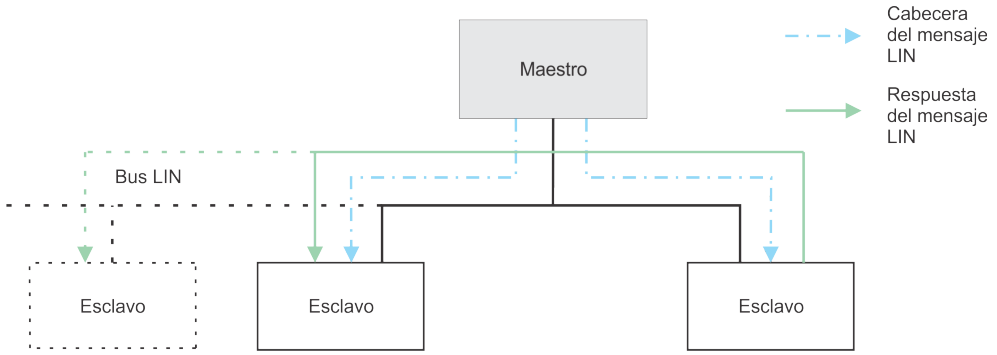

Figura 3.14. Transmisión de datos entre nodos Esclavo.

3.3.2.2.2. Trama activada por evento

El objetivo de una trama activada por evento es proporcionar a la red LIN responsabilidad sin asignarle un ancho de banda elevado, esto debido al múltiple sondeo de los nodos Esclavo que tienen eventos esporádicos.

Todos los nodos Esclavo que reciben la trama activada por evento utilizan los datos como si recibieran una trama incondicional. Si se presenta la trama incondicional asociada a la trama activada por evento, se proporcionará una respuesta.

3.3.2.2.3. Tramas esporádicas

Una trama esporádica es un grupo de tramas incondicionales que comparten el mismo espacio de trama. Cuando una señal se tiene que actualizar en los nodos Esclavo, ésta se transmitirá cuando la trama esporádica esté lista para transmitirse, si no hay señales por actualizar el espacio de trama permanecerá vacío.

Si una o más señales tienen que actualizarse, la trama correspondiente se transmitirá. Si existe más de una señal por actualizar, la trama prioriza la que se transmitirá primero y las tramas restantes no se perderán, se transmitirán cada vez que una trama esporádica esté lista para transmitir.

Si la trama incondicional se transmitió correctamente, no habrá tramas pendientes hasta que una señal se actualice nuevamente.

El nodo Maestro es el único que puede transmitir tramas incondicionales en una trama esporádica. Por lo tanto, sólo el nodo Maestro conoce cuando una trama incondicional está pendiente por transmitirse.

3.3.2.2.4. Tramas de diagnóstico

Las tramas de diagnóstico llevan datos de la capa de transporte y siempre contienen ocho bytes de datos. El identificador 60 corresponde a la petición de trama del nodo Maestro, el identificador 61 corresponde a la respuesta de trama del nodo Esclavo.

Antes de transmitir una petición de trama por parte del nodo Maestro, el nodo verifica su módulo de diagnóstico para ver si debe transmitirse o debe permanecer en modo de espera. El nodo Esclavo transmite una trama incondicional. Los nodos Esclavo transmiten y reciben respuestas de acuerdo a sus módulos de diagnóstico.

3.3.2.2.5. Tramas reservadas

Las tramas reservadas no se deben usar en un cluster LIN 2.x. Su identificadores son 62 y 63.

3.3.2.3. Tablas de planificación

En una red LIN el nodo Maestro es quien controla la transmisión de las tramas de acuerdo al uso de tablas de planificación (*Schedule tables*); las tablas de planificación aseguran que el bus nunca se sobrecargue. También es el componente clave para garantizar la periodicidad de las señales.

El comportamiento determinante es posible debido a que todas las transferencias en un *cluster* LIN se inicializan por el nodo Maestro. Es responsabilidad del nodo Maestro asegurarse de que a todas las tramas relevantes se les proporcione el tiempo necesario para su transmisión

La especificación LIN soporta el uso de múltiples planificadores que pueden ser útiles en la mayoría de los casos, como por ejemplo, en un vehículo el módulo de la puerta que contiene varias funciones tales como elevalunas motorizadas, espejos laterales motorizados y cerraduras, entre otros; en este ejemplo es necesario realizar una planificación extra que considere la activación del botón para subir el elevalunas. En este caso la planificación prioriza la transmisión del mensaje desde el nodo Maestro hacia el motor del elevalunas para disminuir el tiempo de respuesta cuando se desactive el botón. Lo anterior es importante, ya que la planificación debe considerar estas situaciones para evitar retardos en la transmisión de tramas.

3.3.3. Definiciones de tiempo

La unidad mínima que se utiliza en un cluster LIN es el tiempo base (T_{base}). El tiempo base se implementa en el nodo Maestro y se utiliza para controlar los tiempos de las tablas de planificación. Esto significa que los tiempos de las

tramas en una tabla de planificación se basa en el tiempo base. Usualmente un tiempo base oscila entre 5 y 10 ms.

El punto de inicio de un tiempo base se define como una marca de tiempo base. Por lo tanto, un espacio de trama siempre empieza en la misma marca de tiempo base.

El *jitter* (véase Figura 3.15) ocasionado por el nodo Maestro puede ser definido como la diferencia entre los retardos máximo y mínimo del comienzo del tiempo base hasta el comienzo de la cabecera de la nueva trama (flanco de bajada del intervalo de sincronización).

El tiempo entre tramas es el tiempo del fin de la trama hasta el inicio de la nueva trama (véase Figura 3.15). El tiempo entre tramas no es negativo.

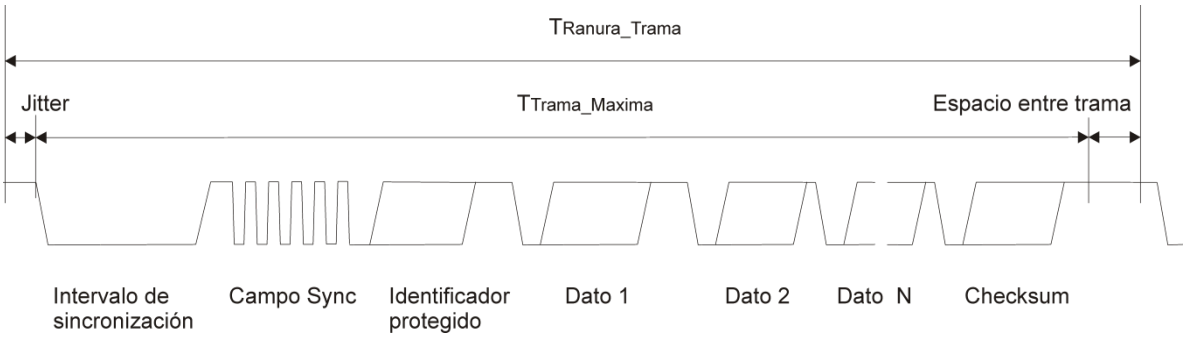

Figura 3.15. Ranura de trama.

La ranura de trama (T_{Frame_slot}) es el tiempo que está controlando la tabla de planificación. Es el tiempo transcurrido en que una tabla de planificación se ejecuta hasta que se ejecute la siguiente tabla. Se define como el múltiplo del tiempo base. El múltiplo entero normalmente es distinto para cada ranura de trama.

Una ranura debe tener una duración suficiente para permitir que el *jitter* se presente por el nodo Maestro.

La tabla de planificación se procesa hasta que otra tabla de planificación sea seleccionada. Cuando se ejecuta el fin de la actual tabla de planificación, se vuelve a ejecutar otra vez al inicio de la siguiente tabla. El cambio de una tabla

de planificación a otra se ejecuta al comienzo de la ranura de la trama, esto significa que el cambio de tabla de planificación no interrumpe la transmisión actual en el bus.

3.4. Modelo de comportamiento de las tareas en el bus

A continuación, se define el modelo del comportamiento de un nodo LIN. El modelo de comportamiento está basado en el concepto de tareas del nodo Maestro y del nodo Esclavo.

El nodo Maestro es responsable de la generación de las cabeceras, decidiendo qué trama se enviará, para mantener los tiempos correctos entre tramas, todo de acuerdo a la tabla de planificación. La Figura 3.16 muestra el diagrama de estados del nodo Maestro.

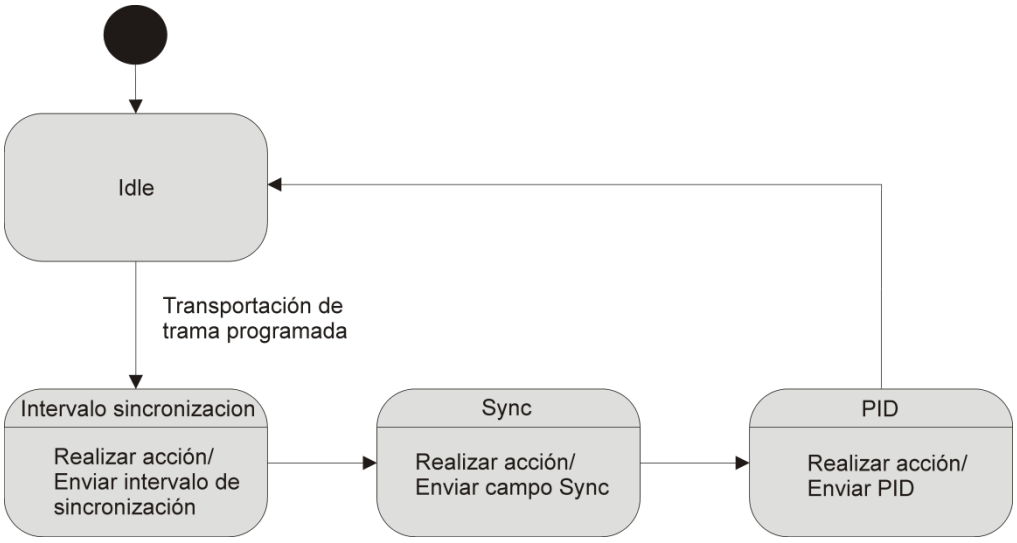

Figura 3.16. Máquina de estados de las tareas del nodo Maestro.

El nodo Esclavo es responsable de transmitir la respuesta de trama cuando se le solicita y de recibir la respuesta de trama. Las tareas del nodo Esclavo se modelan con dos máquinas de estados:

- Detector de secuencias del intervalo y campo de sincronización.
- Procesador de tramas

Un nodo Esclavo debe estar sincronizado al comienzo del identificador protegido de una trama ya que debe ser capaz de recibir correctamente el campo del identificador. Debe estar sincronizado con la velocidad de transferencia a través del resto de la trama. Para este propósito cada trama comienza con una secuencia donde se transmite el intervalo de sincronización seguido del campo Sync. Esta secuencia es única en toda la comunicación y suministra suficiente información para que cualquier nodo Esclavo pueda detectar una nueva trama y estar sincronizado al comienzo del campo del identificador.

El proceso de las tramas consiste en dos estados: *Idle* y Activo. El estado Activo contiene cinco estados internos. Tan pronto como se recibe el intervalo y el campo de sincronización se pasa del estado activo al estado PID. Esto significa que, si se recibe otro intervalo y campo de sincronización, el procesamiento de la señal actual se suspende como muestra la Figura 3.17. Si existiera una diferencia entre el valor del bit transmitido y el valor del bit leído en el bus provocará que la transmisión se suspenda.

3.5. Gestión de la red

La gestión de la red en un *cluster* LIN se refiere al encendido del *cluster* y solamente al modo de bajo consumo de energía. Otras funciones de la gestión de la red como la detección de la configuración y la gestión del funcionamiento de emergencia se dejan a la capa de aplicación.

3.5.1. Diagrama de estados de la comunicación del nodo Esclavo

El diagrama de estados de la Figura 3.18 muestra el modelo del comportamiento de un nodo Esclavo. Se presenta el estado de inicialización inmediatamente cuando un nodo Esclavo se conecta a la fuente de voltaje, reinicio o en modo de despertarse. El nodo Esclavo necesariamente se tiene que inicializar y posteriormente se va al estado operacional. La inicialización hace referencia a la inicialización del nodo LIN. La inicialización es diferente cuando sucede un reinicio o se despierta el nodo Esclavo.

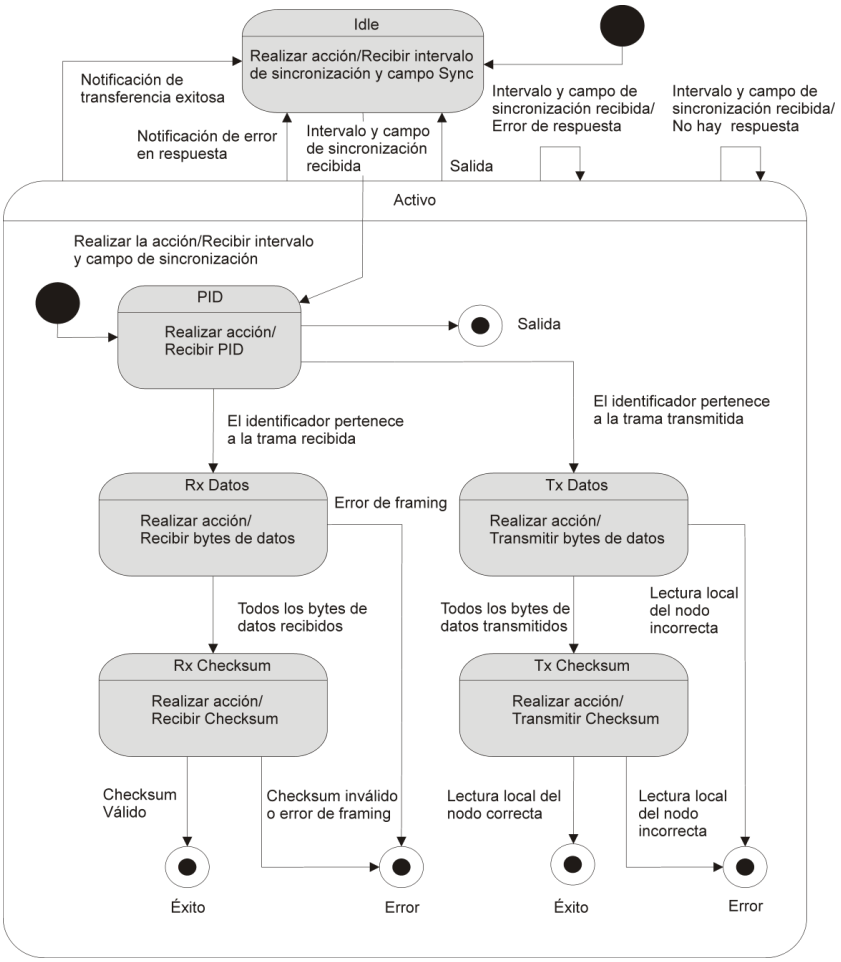

Figura 3.17. Máquina de estados del procesador de trama.

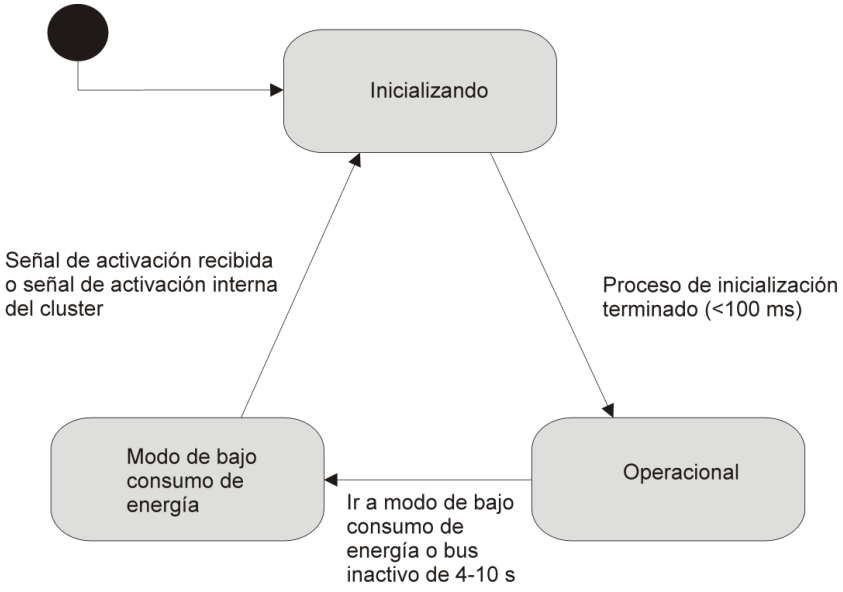

Figura 3.18. Diagrama de estados de la comunicación de los nodos Esclavo.

El comportamiento del protocolo LIN especificado en este trabajo de tesis aplica al estado operacional (transmisión y recepción de tramas).

El nivel del bus se fija en estado recesivo y bajo estas circunstancias sólo la señal de despertarse podrá transmitirse en el *cluster*.

Cualquier nodo en un *cluster* LIN en el modo de bajo consumo de energía puede solicitar despertar, transmitiendo una señal *wakeup*. La señal *wakeup* comienza forzando al bus a un estado dominante por 250 µs a 5 ms, y se valida con el retorno de la señal del bus a un estado recesivo. El nodo Maestro puede transmitir un intervalo de sincronización ya que éste actuaría como una señal *wakeup* (en este caso el nodo Maestro debe estar alerta que esta trama no sea procesada por los nodos Esclavo ya que todavía no despiertan y estén listos para las cabeceras).

Cada nodo Esclavo (conectado a la fuente de voltaje) puede detectar la señal de *wakeup* (un pulso dominante mayor a 150 µs seguido de un flanco de subida del bus) y estar listos para recibir las órdenes dentro de 100 ms, medidos a partir del fin del flanco del pulso dominante (véase Figura 3.19). Si el nodo que transmite la señal *wakeup* es un nodo Esclavo, estará listo para recibir y transmitir tramas inmediatamente. El nodo Maestro también despertará y cuando los nodos Esclavo estén listos, empezará a transmitir cabeceras para encontrar quién ocasionó la señal *wakeup*.

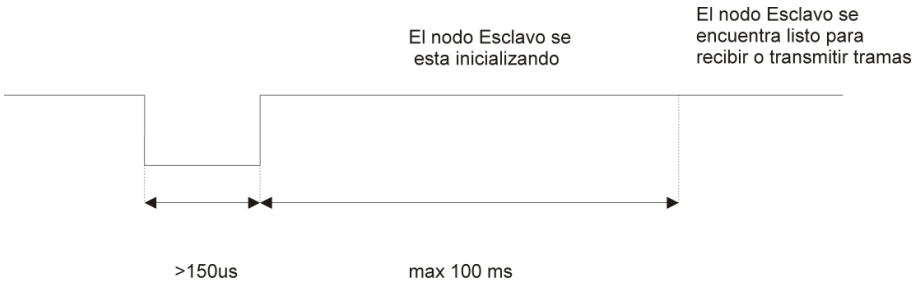

Figura 3.19. Recepción de la señal wakeup *en los nodos Esclavo.*

El nodo Maestro detectará la señal *wakeup* y estará listo para empezar la comunicación dentro del tiempo decidido por el diseñador o aplicación en específico.

Si el nodo Maestro no transmitió un intervalo de sincronización o si el nodo que emitió la señal *wakeup* no recibe otra señal *wakeup* de otro nodo dentro de 150 ms a 250 ms, el nodo que emitió la señal debe transmitir una nueva señal *wakeup* (véase Figura 3.20). En el caso que un nodo Esclavo transmita una señal *wakeup* al mismo instante que el nodo Maestro transmite un intervalo de sincronización, el nodo Esclavo recibirá y reconocerá este intervalo.

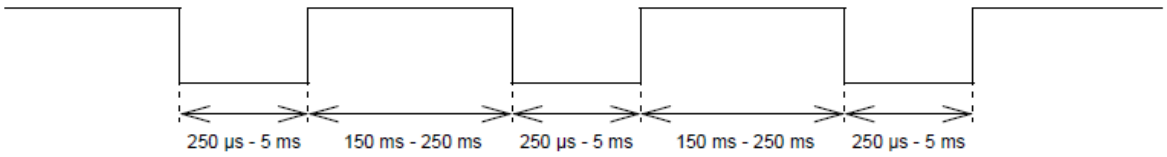

Figura 3.20. Bloque de señales wakeup.

Después de tres intentos fallidos, el nodo espera mínimo 1.5 segundos antes de emitir la cuarta señal de *wakeup*. La razón de esta espera es permitir al *cluster* comunicarse en caso que los nodos Esclavo tengan problemas. Si el nodo Esclavo tiene problemas al leer información del bus probablemente transmita indefinidamente la señal de *wakeup*.

La especificación LIN define un modo de operación de reposo para ahorrar energía y dar soporte a eventos esporádicos que necesitan hacer uso del bus. El nodo Maestro es el único que puede poner la red en modo reposo, sin embargo, todos los nodos son capaces de activar el bus nuevamente. Para llevar a cabo lo anterior, el nodo Maestro envía una orden con un formato de trama predefinido a todos los nodos Esclavo de la red, el ID=0x3C, el primer byte de datos = 0x00 y los bytes restantes con un valor de 0xFF (véase Tabla 3.2); los nodos Esclavo responden colocando sus transceptores en modo de bajo consumo de energía y esperan hasta que reciban alguna señal de operación en el bus o una activación directamente en el nodo.

Tabla 3.2. Órdenes de reposo.

Dato1	**Dato2**	**Dato3**	**Dato4**	**Dato5**	**Dato6**	**Dato7**	**Dato8**
0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

La señal de operación consiste del byte de datos “0x80”, y una vez que los nodos activos en el bus reciben dicha señal ejecutan sus procedimientos de inicialización y esperan a que el nodo Maestro transmita un intervalo de sincronización; el bus debe mantenerse en estado recesivo por un periodo de al menos 4 bits después de la señal de operación del bus.

3.6. Capa de transporte

La capa de transporte define la transportación de datos contenidos en una o más tramas. Los datos son transportados por tramas de diagnóstico como se mencionó en el párrafo 3.3.2.2.4.

El uso de la capa de transporte es elegido por los sistemas donde se tiene un bus principal (CAN, por ejemplo) y donde el diseñador del sistema quiere usar las mismas capacidades de diagnóstico en los *clusters* del sub-bus. Para este propósito se tienen dos identificadores que se utilizan: la solicitud de trama del nodo Maestro con el ID=60 y la respuesta de trama del nodo Esclavo con el ID=61. Los mensajes son idénticos al ISO 15765-2 (ISO/IEC 15765-2, 2002) y los PDUs (*packet data units*) que acarrean mensajes son similares. La configuración de un sistema típico se muestra en la Figura 3.21.

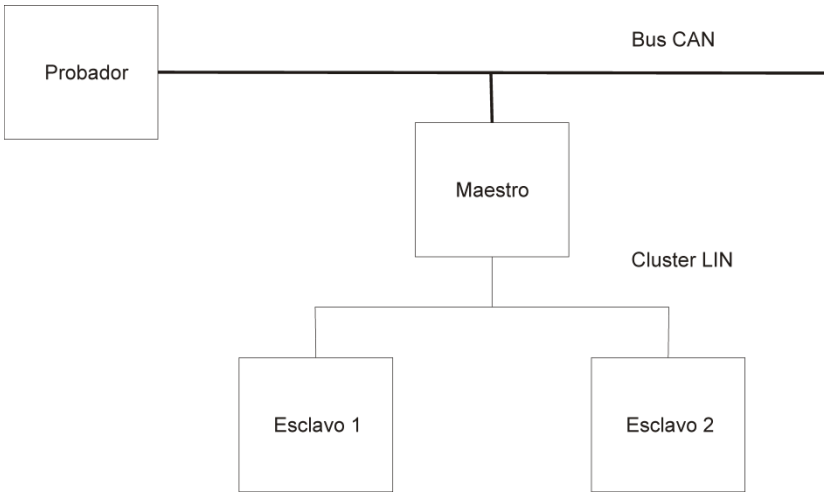

Figura 3.21. Sistema típico de un cluster *LIN usando la capa de transporte.*

Los objetivos de la capa de transporte son:

- Reducir el tráfico sobre la red.
- Proveer diagnóstico completo directamente en los nodos Esclavo.
- Poder construir nodos LIN complejos y robustos.

Una trama de diagnóstico recibe el nombre de PDU (*packet data unit*), un PDU puede ser un mensaje completo o parte del mensaje; en el último caso, múltiples PDUs concatenados forman un mensaje completo.

El primer byte de un PDU es un NAD (*node address for diagnostic*), después sigue el byte PCI (*protocol control information*) que maneja el control de flujo. En los bytes restantes se presentan el byte de identificador de servicio (SID, service identifier) que especifica la solicitud qué debe ejecutar el nodo Esclavo direccionado. El byte LEN especifica la longitud del mensaje.

Los PDUs de respuesta de diagnóstico del nodo Esclavo se construyen de forma similar a los PDUs del nodo Maestro. La respuesta del identificador de servicio (RSID, *response service identifier*) especifica el contenido de la respuesta.

La interpretación de los bytes de datos D1 a D6 depende del byte SID o del byte RSID. En múltiples mensajes PDU, todos los bytes del mensaje se concatenan en uno sólo antes de ser procesados.

3.7. Capa de supervisor

Las comunicaciones de bus serie presentan el problema de que una ECU defectuosa puede bloquear el funcionamiento del sistema completo. El propósito de esta capa es verificar el correcto funcionamiento de la capa DLL y de la capa física mediante los mecanismos autónomos conocidos como gestor de fallos (*Fault Confinement*), sincronización del sistema (*System Synchronization*) y manejo de fallos del bus (*Bus Failure Managment*).

El nodo Maestro es supervisado por la subcapa de gestión de fallos, con lo que tiene la posibilidad de solicitar la retransmisión de un mensaje. Los nodos

Esclavo no pueden señalizar errores directamente, ya que es responsabilidad del nodo Maestro sondear a los nodos Esclavo acerca de los errores ocurridos. Tanto el nodo Maestro como los nodos Esclavo tienen la capacidad de detectar errores de bit al comparar el bit transmitido en el bus con el bit recibido (efecto local).

El nodo Maestro da seguimiento a cualquier transmisión incrementando el contador *MasterTransmitErrorCounter*. Este contador se incrementa en ocho unidades cada vez que un campo *Sync* o un campo de identificador se corrompe, cuando la comparación del campo transmitido con el campo recibido es el mismo, el contador se decrementa en una unidad (no debajo de cero).

Si el contador se incrementa por encima de C_MASTER_TRANSMIT_ERROR_ THRESHOLD, se presenta un fallo masivo en el bus y la capa de aplicación se encarga del procedimiento de control de este error.

El nodo Maestro detecta un error cuando espera un dato proveniente del bus por parte del nodo Esclavo y éste no envía su respuesta.

Cuando un nodo Esclavo recibe una trama de datos, extrae la suma de verificación y la paridad del identificador de la trama de datos, posteriormente detecta los errores de suma de verificación y de paridad del identificador al compararlos respectivamente.

El nodo Maestro y los nodos Esclavo también pueden detectar errores en el campo *Sync* cuando los flancos de este campo se encuentran fuera de sus valores de tolerancia. Los nodos Esclavo pueden detectar errores mediante el uso de temporizadores cuando identifican que no pueden transmitir su respuesta y no existe actividad en el bus; si un nodo Esclavo detecta una inconsistencia en su funcionamiento, la almacena como información de diagnóstico y queda disponible para cuando el nodo Maestro lo requiera.

El propósito de la subcapa de sincronización del sistema es supervisar que el sistema se encuentre sincronizado y esté dentro de sus valores de tolerancia, en su defecto informa a la subcapa de gestión de fallos.

La subcapa de manejo de fallos del bus detecta cuando el bus presenta interferencia por parte del nodo Maestro o por parte de una respuesta de un nodo Esclavo, en tales circunstancias notifica al gestor de fallos para que tome acciones correctivas.

3.8. Capa de aplicación

El protocolo de comunicaciones LIN se encarga de las capas bajas (capa física y capa de enlace de datos) de acuerdo al modelo OSI, pero esto no es suficiente para una comunicación de red, la cual requiere de las capas de aplicación (capa superior de protocolos). Su función principal es transferir información de un nodo a otro nodo de la misma red. La capa de aplicación no forma parte de la especificación del protocolo LIN y no se implementa en este trabajo de tesis, a continuación, se presentan sus funciones más importantes:

- Transmisión de datos de un nodo Maestro a un nodo Esclavo.
- Disponibilidad de servicios de transmisión para bloques de datos mayores a 8 bytes.
- Soporte al modelo Cliente/Servidor.
- Funciones dedicadas a la gestión de red.
- Manejo de tablas de planificación especiales.
- Gestión del funcionamiento de emergencia de un nodo.
- Detección de la configuración de un nodo.

La capa de aplicación depende del diseñador y está relacionada con sus requerimientos particulares. Actualmente la mayoría de los sistemas LIN que se implementan utilizan protocolos de capa superior propietarios.

4. Protocolo de Comunicaciones CAN

CAN es un protocolo de comunicaciones serie, desarrollado por la firma alemana Robert Bosch GmbH, que soporta control distribuido en tiempo real con un alto nivel de seguridad y multiplexación (Robert Bosch GmbH, 1991); está basado en una topología de bus para la transmisión de mensajes y la gestión de la comunicación entre múltiples unidades centrales de proceso en ambientes distribuidos (Lawrenz, 1997).

El protocolo CAN se encuentra estandarizado en la norma ISO 11898-1, comprendiendo la capa de enlace de datos, los componentes de la capa física y los servicios de comunicación para el envío de mensajes (transmisión de trama de datos) y la solicitud de mensajes (solicitud de transmisión remota) (CAN in Automation, 2014). Actualmente CAN cuenta con el apoyo de más de 50 fabricantes de semiconductores, quienes atienden dicha norma en el diseño e implementación de soluciones hardware.

Un vehículo contiene una red CAN operando a diferentes velocidades de transmisión (Leen y Hefferman, 2002):

- Las aplicaciones de control no críticas y de confort utilizan una red CAN operando a 125 kbps, llamada red CAN de baja velocidad; los nodos CAN que implementan este tipo de aplicaciones incorporan un algoritmo de ahorro de energía que detiene sus osciladores hasta que un mensaje los

habilita con la finalidad de no afectar significativamente la carga de la batería.

- Las aplicaciones de gestión y control en tiempo real crítico, tal como frenos ABS y transmisión, utilizan una red CAN operando a 1 Mbps, llamada red CAN de alta velocidad; respecto al ahorro de energía, los nodos que implementan estas aplicaciones deben mantenerse activos durante el funcionamiento del vehículo. Cabe señalar que a velocidades mayores a 500 kbps las señales transmitidas mediante par trenzado se ven afectadas por interferencias electromagnéticas (EMI), por lo que el cable debe tener un recubrimiento especial, lo cual incrementa los costos de la aplicación.

CAN es un protocolo basado en mensajes, esto significa que los dispositivos conectados a la red CAN no tienen direcciones únicas, pero el mensaje que un dispositivo envía a través de la red posee un identificador único, de acuerdo a su contenido. Como resultado, cada dispositivo en la red recibe todos los mensajes transmitidos sobre el bus y determina qué acción necesita tomar.

CAN ha sido ampliamente aceptado en aplicaciones de redes automotrices debido a su bajo costo, alto rendimiento y disponibilidad de diversas implementaciones del protocolo en circuito integrado (Mariño, Hernández, Domínguez, Poza y Machado, 2003) (Mariño, Hernández, Domínguez, Poza, Machado y Vázquez, 2003) (Chamú Morales, 2005). El protocolo CAN proporciona los siguientes beneficios:

- Es un protocolo de comunicaciones estandarizado en ISO-11898 (ISO/IEC 11898, 1993) e ISO-11519 (ISO/IEC 11519, 1994), con lo que se simplifica y economiza la tarea de comunicar subsistemas de diferentes fabricantes sobre una red común.
- El CPU anfitrión (*host*) delega la carga de comunicaciones a un nodo inteligente, por lo tanto, el CPU anfitrión tiene más tiempo para ejecutar sus propias tareas.

- Al ser una red multiplexada, reduce considerablemente el cableado y elimina las conexiones punto a punto.

Un nodo CAN tiene la finalidad de sustituir o eliminar el cableado para interconectar los dispositivos electrónicos internos de un vehículo (Thompson, 1996). Las ECUs, sensores, sistemas antideslizantes, etc. se conectan mediante una red CAN a velocidades de transferencia de datos de hasta 1 Mbps.

La arquitectura de protocolos CAN, de acuerdo al modelo de referencia OSI (*Open Systems Interconnection*) (ISO/IEC 7498, 1989) incluye tres capas: física, enlace de datos y aplicación, además establece una capa especial para la gestión y control del nodo llamada capa de supervisor (véase Figura 4.1) (Chamú Morales, 2005).

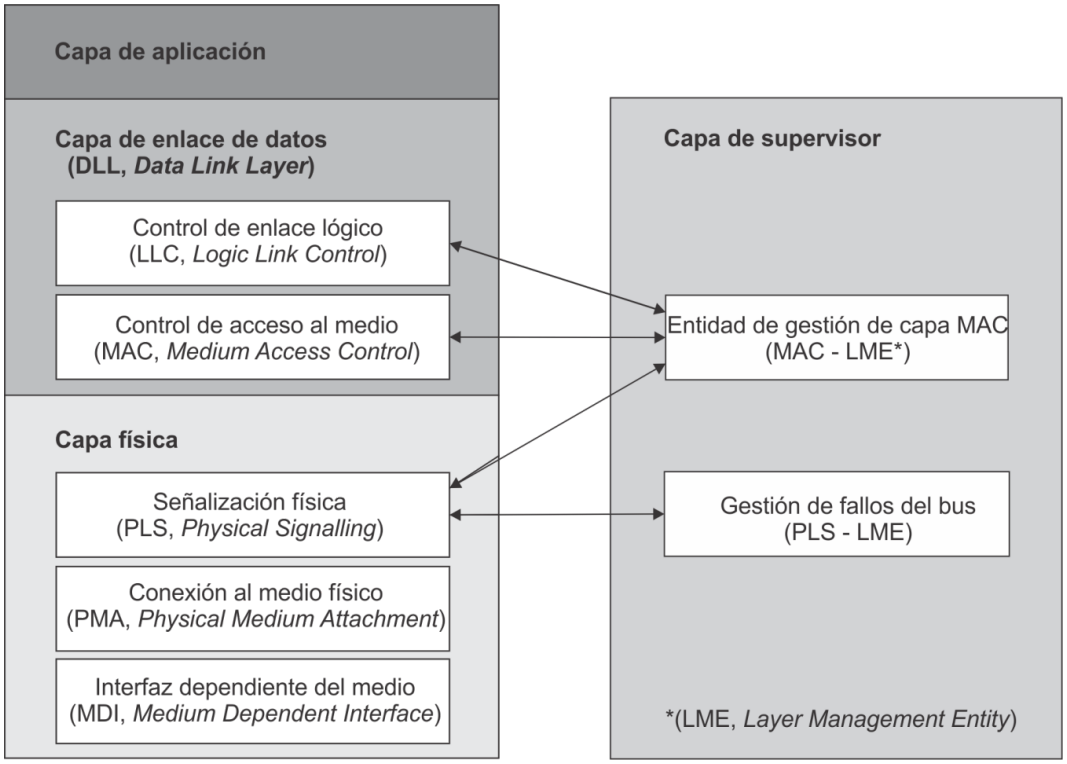

Figura 4.1. Arquitectura de protocolos CAN.

El protocolo de comunicaciones CAN se basa en el modelo Productor/Consumidor, el cual es un concepto o paradigma de comunicaciones de datos, que describe una relación entre un productor y uno o más consumidores.

Este modelo describe un servicio de comunicaciones en el que un proceso proporciona datos, por difusión de mensajes, a otros procesos sin que éstos los soliciten. El productor proporciona servicios a otros procesos mediante un identificador de servicio; los consumidores (procesos) pueden aceptar o ignorar dichos servicios; y debido a que el productor no conoce a los consumidores, no se requiere una confirmación del servicio.

Por otro lado, los protocolos de la capa de aplicación dan soporte al modelo Cliente/Servidor, entre ellos los más importantes son CAL (*CAN application layer*), CANopen y DeviceNet (Etschberger, 2001). El modelo Cliente/Servidor define un servidor que entrega y maneja la mayoría de los recursos y servicios que son consumidos por uno o más clientes.

CAN es un protocolo orientado a mensajes, y dentro de sus principales características se encuentran (Robert Bosch GmbH, 1991) (Lawrenz, 1997) (Etschberger, 2001):

- Prioridad de mensajes.
- Garantía de tiempos de latencia.
- Flexibilidad en la configuración.
- Recepción por multidifusión (*multicast*) con sincronización de tiempos.
- Sistema robusto en cuanto a consistencia de datos.
- Sistema multimaestro.
- Detección y señalización de errores.
- Retransmisión automática de tramas erróneas tan pronto como el bus esté libre.
- Distinción entre errores temporales y fallas permanentes de los nodos de la red, y desconexión automática de los nodos defectuosos.

4.1. Capa física

El diseño de una red CAN varía de acuerdo a las necesidades de desempeño y para ello se deben considerar los requisitos de la capa física.

La especificación del protocolo CAN no define una capa física, sin embargo, los estándares ISO 11898 e ISO 11519 establecen las características que deben cumplir las aplicaciones para la transferencia en alta y baja velocidad respectivamente. Cabe mencionar que las características definidas para la capa física deben implementarse en todos los nodos que se encuentren conectados dentro de una red CAN. La capa física se divide en tres secciones o subcapas las cuales se describen a continuación.

4.1.1. Subcapa de señalización física

La subcapa de señalización física (PLS, *Physical Signalling*) define las funciones relacionadas con la representación, tiempo y sincronización de los bits, y está implementada en los controladores del protocolo CAN.

4.1.1.1. Representación de bits

Una trama CAN está codificada de acuerdo con el método NRZ (*Non Return to Zero*), el cual establece que durante todo el tiempo de bit se genera un nivel de señal que puede ser dominante (d) o recesivo (r). Una ventaja de dicho método, en comparación con la codificación Manchester, es que produce una frecuencia menor de operación, ya que la codificación Manchester requiere de flancos en la mitad del tiempo de bit. Sin embargo, en el caso de transmitir una gran cantidad de bits con la misma polaridad, la codificación NRZ no proporciona flancos que puedan utilizarse en la sincronización y por ello se implementa el procedimiento de inserción de bit (*bit-stuffing*), el cual asegura que en la transmisión de una trama sólo puede haber un máximo de cinco bits consecutivos con la misma polaridad como se muestra en la Figura 4.2.

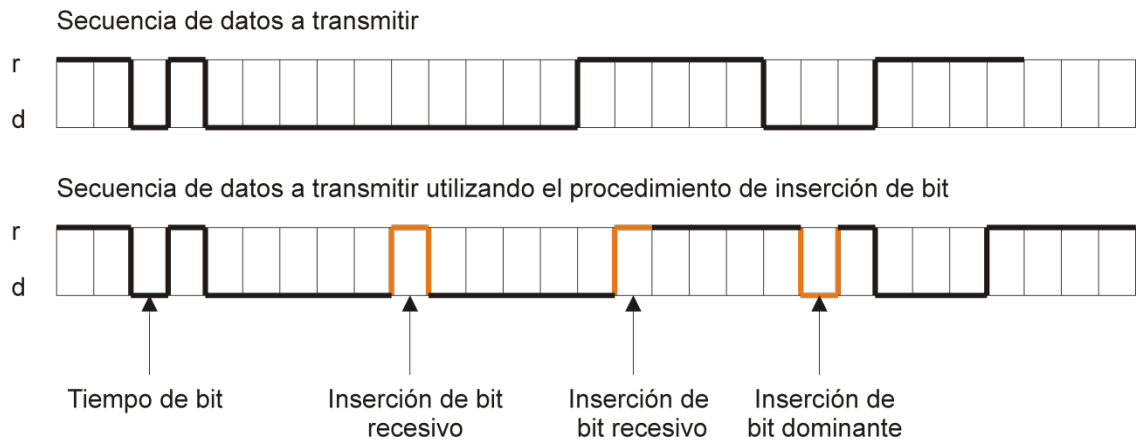

Figura 4.2. Ejemplo del procedimiento de inserción de bit.

En una trama CAN, de datos o remota (véase apartado 4.3.2), la codificación NRZ junto con el procedimiento de inserción de bit se aplican a los campos de inicio de trama (SOF, *Start of Frame*), arbitraje, control, datos[8] y código de redundancia cíclica (CRC, *Cyclic Redundant Check*). Los campos restantes, delimitador CRC, aceptación (ACK, *Acknowledgement*) y fin de trama (EOF, *End of Frame*), tienen un formato fijo y son transmitidos sin emplear el procedimiento de inserción de bit. Lo anterior también se aplica a las tramas de error y de sobrecarga (Etschberger, 2001).

4.1.1.2. Temporización de bit y sincronización

El protocolo CAN utiliza transferencia de datos sincrónica, lo cual incrementa la capacidad de transmisión, pero requiere un método sofisticado de sincronización de bits debido a que sólo está disponible un bit de inicio al comienzo de la trama. Lo anterior no es suficiente para mantener sincronizado el muestreo de bit del receptor con el del transmisor, y para lograrlo se requiere realizar una resincronización continua del receptor (Etschberger, 2001). Se insertan segmentos de memoria temporal de fase antes y después del punto de muestra dentro del intervalo de bit.

[8] El campo de datos no se incluye en una trama remota.

En una red CAN cada nodo se sincroniza con su propio oscilador, debido a ello pueden ocurrir desplazamientos de fase en los diferentes nodos y por lo tanto los controladores CAN proporcionan un mecanismo de sincronización para compensar dichos desplazamientos mientras reciben una trama CAN.

El protocolo CAN regula el acceso al bus utilizando la arbitrariedad de ancho de bit, así que la propagación de la señal del transmisor al receptor y de vuelta al transmisor se debe completar dentro de un tiempo de bit. Para propósitos de sincronización se necesita un tiempo de segmento mayor, el segmento de retardo de propagación, además del tiempo reservado para la sincronización. El segmento de propagación de retardo también considera un retardo de señal causado por los nodos transmisores y receptores (CAN in Automation, 2014).

Se emplean dos tipos de sincronización (Robert Bosch GmbH, 1991):

- Sincronización al comienzo de la trama (*Hard synchronization*): Después de una sincronización, se reinicia el tiempo de bit al finalizar el segmento de sincronización, sin tener cuidado de un error de fase, con ello la sincronización obliga al flanco a caer dentro del segmento de sincronización del bit reiniciado.
- Resincronización dentro de la trama (*Soft synchronization*): Éste acorta o alarga el tiempo de bit por lo que el muestreo del punto se cambia de acuerdo al flanco detectado. La cantidad que puede cambiar depende del parámetro SJW (*Synchronization Jump Width*).

Estos parámetros de tiempos pueden ser ajustados por el diseñador para satisfacer la aplicación actual. Los segmentos de tiempo de bit se muestran en la Figura 4.3.

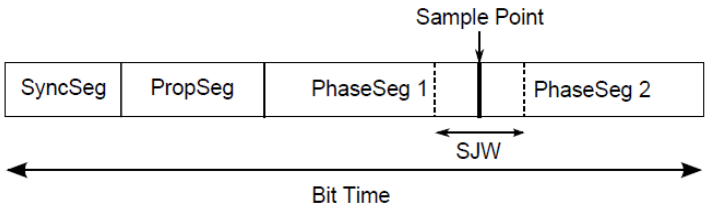

Figura 4.3. Segmentos de tiempo de bit CAN.

4.1.1.3. Velocidad de transferencia vs Longitud del bus

Es posible determinar la máxima longitud posible de un nodo CAN a una velocidad de transferencia en específico si se conoce la duración del retardo de propagación de la señal (véase Tabla 4.1). La propagación de la señal se determina por la separación de los nodos del sistema que estén más alejados uno del otro, es el tiempo que le lleva a una señal viajar de un nodo al otro (tomando en consideración el retardo causado por los nodos transmisor y receptor), sincronización y el viaje de la señal del segundo nodo de regreso al primero. El primer nodo decide si su mismo nivel de señal está en el actual nivel sobre el bus o si ha sido reemplazado por otro nodo. Esto es importante para el arbitraje del bus (Richards, 2001).

Tabla 4.1. Velocidad de transferencia vs Longitud del bus.

Velocidad (Kbps)	Longitud del bus (m)
1000	30
500	100
250	250
125	500
62.5	1000

4.1.2. Subcapa de unión al medio físico

La conexión entre el controlador CAN y el medio físico se define en la subcapa de unión al medio físico (PMA, *Physical Medium Attachment*). Esta subcapa describe las características funcionales y eléctricas que debe cumplir el transceptor para hacer posible la transmisión/recepción entre un nodo y el bus, además algunas de sus implementaciones en circuitos integrados proporcionan los medios para detectar fallos en el bus.

La interfaz eléctrica con el bus consiste principalmente de un transmisor y un receptor. La Figura 4.4 ilustra el diagrama a bloques básico de un transmisor-receptor en un nodo CAN, el controlador CAN provee funcionalidad básica de transmisor-receptor, que consiste de un comparador en la recepción y un amplificador en la salida.

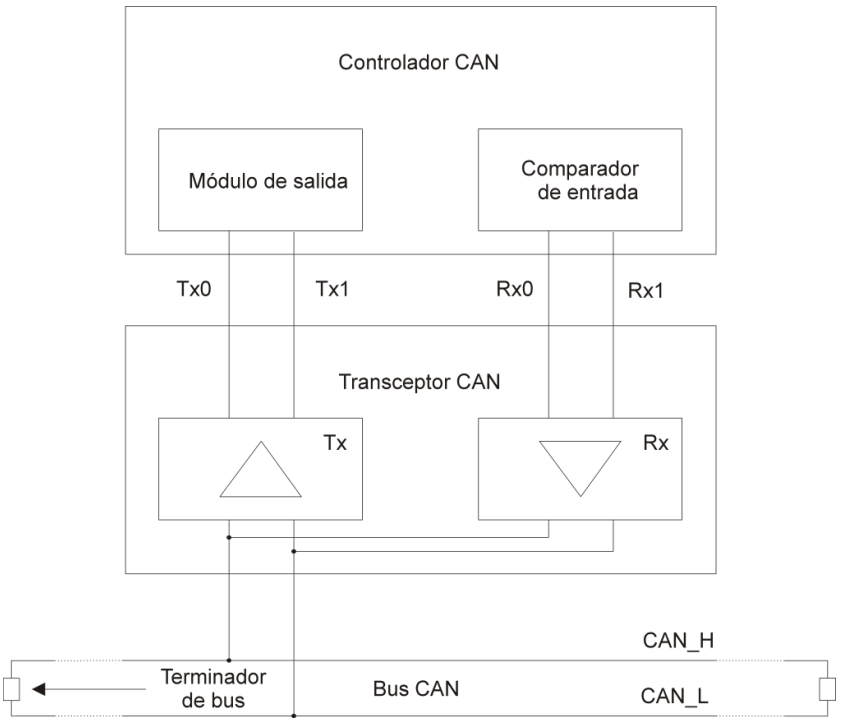

Figura 4.4. Arquitectura de un nodo CAN con transceptor.

Además de la adaptación de señales entre el controlador CAN y el bus, el transceptor tiene que cumplir con las siguientes funciones:

- Suministrar la capacidad de rendimiento al controlador CAN.
- Proteger al controlador CAN contra sobrecargas.
- Reducir la radiación electromagnética.

Como receptor realiza las siguientes funciones:

- Suministrar un nivel de señal recesivo.
- Proteger el comparador de entrada del controlador CAN contra sobrevoltajes de las líneas del bus.

- Suministrar suficiente sensibilidad para un mejor reconocimiento de la señal de entrada.
- Detectar errores en el bus tales como: ruptura de la línea, corto-circuito y conmutación a operación asimétrica de una sola línea.

Una función opcional del transmisor-receptor es proporcionar aislamiento galvánico entre el nodo CAN y las líneas de bus, mediante el empleo de optoacopladores.

4.1.3. Subcapa de interfaz dependiente del medio

La interfaz dependiente del medio (MDI, *Medium Dependent Interfaz*) define los aspectos eléctricos y mecánicos para la conexión entre el medio físico y el nodo.

Las especificaciones mecánicas de la capa física definen las características de la interfaz respecto al medio de transmisión y al tipo de conector.

4.1.3.1. Medio físico

Se debe elegir un medio con el que se puedan transmitir dos posibles estados de bit, “dominante” y “recesivo”. Una de las formas más comunes y económicas es utilizar un par de cables trenzados. Las líneas del bus se conocen como “CAN alto” y “CAN bajo”. Las dos líneas son controladas por los nodos por una señal diferencial la cual es definida por el protocolo (Véase Figura 4.5). Un bit recesivo se representa con ambas líneas llevadas a un nivel de voltaje de 2.5 V, resultando un voltaje diferencial de 0 V. Un bit dominante se representa con CAN alto en un nivel de 3.5 V y CAN bajo en un nivel de 1.5 V, resultando un voltaje diferencial de 2 V (ISO/IEC 11898, 1993). Para que el nodo lea correctamente el nivel de voltaje del bus es importante evitar la reflexión de la señal, esto se logra colocando una resistencia de 120 Ω en las terminaciones de ambos extremos del bus.

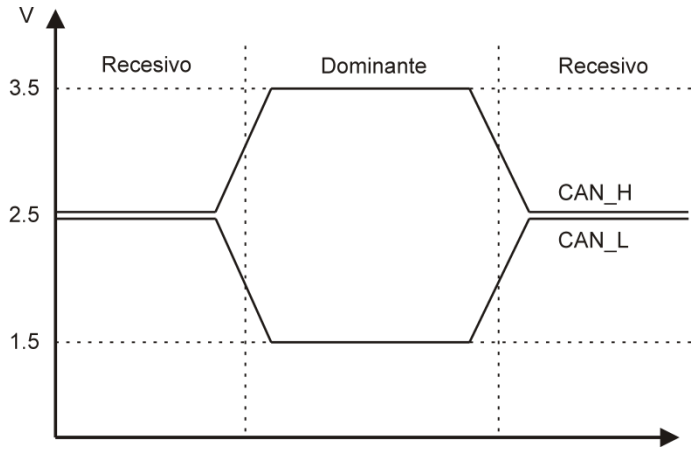

Figura 4.5. Niveles del voltaje del bus CAN.

4.2. Capa de enlace de datos

En este apartado se describen los fundamentos y principios básicos de la capa de enlace de datos (DLL, *Data Link Layer*), la cual se divide en dos subcapas: control de enlace lógico (LLC, *Logic Link Control*) y control de acceso al medio (MAC, *Medium Access Control*).

4.2.1. Control de enlace lógico

La subcapa LLC describe la parte alta de la capa DLL y define las tareas independientes del método de acceso al medio, así mismo proporciona dos tipos de servicios de transmisión sin conexión al usuario LLC (*LLC user*):

- Servicio de transmisión de datos sin reconocimiento: Proporciona al usuario LLC los medios para intercambiar unidades de datos de servicio de enlace (LSDU*, Link Service Data Units*) sin establecer una conexión de enlace de datos. La transmisión de datos puede ser punto a punto, multidifusión o difusión.
- Servicio de petición de datos remota sin reconocimiento: Proporciona al usuario LLC los medios para solicitar que un nodo remoto transmita sus LSDUs sin establecer una conexión de enlace de datos.

De acuerdo con los tipos de servicios, se definen dos formatos de trama, de datos LLC y remota LLC. Ambos formatos definen identificadores de 11 bits (estándar) y de 29 bits (extendida), los cuales se describen en el apartado 4.3.

La subcapa LLC realiza las siguientes funciones:

- Filtrar mensajes (*frame acceptance filtering*): El identificador de una trama no indica la dirección destino, pero define el contenido del mensaje, y mediante esta función todo receptor activo en la red determina si el mensaje es relevante o no para sus propósitos.
- Notificar sobrecarga (*overload notification*): Si las condiciones internas de un receptor requieren un retraso en la transmisión de la siguiente trama de datos o remota, la subcapa LLC transmite una trama de sobrecarga. Solamente se pueden generar dos tramas de sobrecarga como máximo.
- Proceso de recuperación (*recovery management*): La subcapa LLC proporciona la capacidad de retransmisión automática de tramas cuando una trama pierde el arbitraje o presenta errores durante su transmisión, dicho servicio se confirma al usuario hasta que la transmisión se completa con éxito.

4.2.2. Control de acceso al medio

Una red CAN brinda soporte para procesamiento en tiempo real a todos los sistemas que la integran. El intercambio de mensajes que demanda dicho procesamiento requiere de un sistema de transmisión a frecuencias altas y retrasos mínimos (Etschberger, 2001). En redes multimaestro, la técnica de acceso al medio es muy importante ya que todo nodo activo tiene los derechos para controlar la red y acaparar sus recursos.

Para acceder al medio, los nodos CAN utilizan el siguiente mecanismo de arbitraje:

- Cuando un nodo necesita enviar información a través de una red CAN, la capa de aplicación realiza una petición, de forma asíncrona, para

transmitir una trama. Puede ocurrir que varios nodos inicien el mismo procedimiento simultáneamente. CAN resuelve lo anterior al asignar prioridades mediante el identificador de cada mensaje, donde dicha asignación se realiza durante el diseño del sistema en forma de números binarios y no puede modificarse dinámicamente. El identificador con el menor número binario es el que tiene mayor prioridad.

- El método de acceso al medio utilizado es el de acceso múltiple por detección de portadora, con detección de colisiones y arbitraje por prioridad del mensaje (CSMA/CD+AMP, *Carrier Sense Multiple Access with Collision Detection and Arbitration Message Priority*). De acuerdo con este método, los nodos en la red que necesitan transmitir información deben esperar a que el bus esté libre (detección de portadora); cuando se cumple esta condición, dichos nodos transmiten un bit de inicio (acceso múltiple). Cada nodo lee el bus bit a bit durante la transmisión de la trama y compara el valor transmitido con el valor recibido; mientras los valores sean idénticos, el nodo continúa con la transmisión; si se detecta una diferencia en los valores de los bits, se lleva a cabo el mecanismo de arbitraje.
- De acuerdo a la Figura 4.6 durante la transmisión del bit 5, el nodo 1 transmite un bit dominante y el nodo 2 transmite un bit recesivo; el nivel dominante sobrescribe al recesivo; ambos nodos comparan los valores transmitidos con los que reciben; el nodo 2 detecta una diferencia (detección de colisión), por lo tanto, ha perdido el arbitraje e inmediatamente deja de transmitir para conmutar a modo de recepción; mientras tanto el nodo 1 continua transmitiendo (AMP) hasta que se presenta la misma situación en la transmisión del bit 2. Al final, el nodo 3 es quién gana el arbitraje y transmite su trama. Lo anterior se conoce como arbitraje no destructivo bit a bit, es decir que a pesar de que varios nodos inicien la transmisión de sus tramas al mismo tiempo, el nodo ganador del arbitraje, que tiene el mensaje con la mayor prioridad,

continúa con la transmisión de su trama sin necesidad de retransmitirla desde el principio.

- Si lo anterior se detecta en un campo distinto al bit de inicio, campo de arbitraje y ranura ACK, se activa el proceso de control de errores (véase apartado 4.3.7) por parte del administrador del nodo, quien lo considera un error de bit (Lawrenz, 1997).
- Si se inicia simultáneamente la transmisión de una trama de datos (véase apartado 4.3.1) y una trama remota (véase apartado 4.3.2), solicitada por un receptor, el proceso de arbitraje no puede resolverse únicamente con el identificador de la trama, sino que tiene que utilizarse el bit RTR.

El principio de arbitraje utilizado en CAN limita la extensión máxima de la red a una velocidad de transferencia de datos específica (Etschberger, 2001).

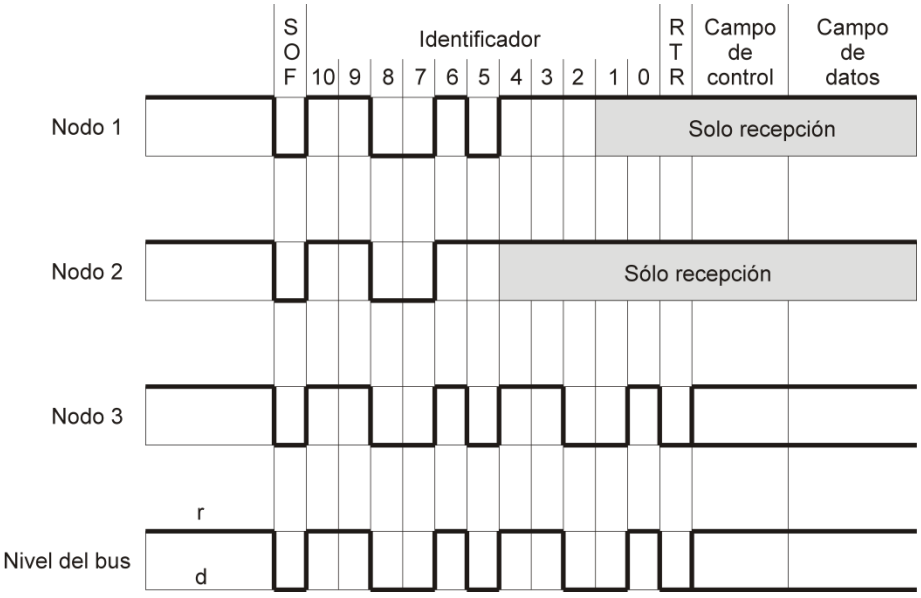

Figura 4.6. Lógica de arbitraje del bus CAN.

4.3. Transmisión de mensajes

CAN establece dos formatos de tramas de datos (*data frame*) que difieren en la longitud del campo del identificador, las tramas estándares (*standard frame*) con un identificador de 11 bits definidas en la especificación CAN 2.0A, y las

tramas extendidas (*extended frame*) con un identificador de 29 bits definidas en la especificación CAN 2.0B.

Para la transmisión y control de mensajes CAN, se definen cuatro tipos de tramas: de datos, remota (*remote frame*), de error (*error frame*) y de sobrecarga (*overload frame*).

Las tramas remotas también se establecen en ambos formatos, estándar y extendido, y tanto las tramas de datos como las remotas se separan de tramas precedentes mediante espacios entre tramas (*interframe space*).

4.3.1. Trama de datos

Una trama de datos se compone de siete campos (véase Figura 4.7):

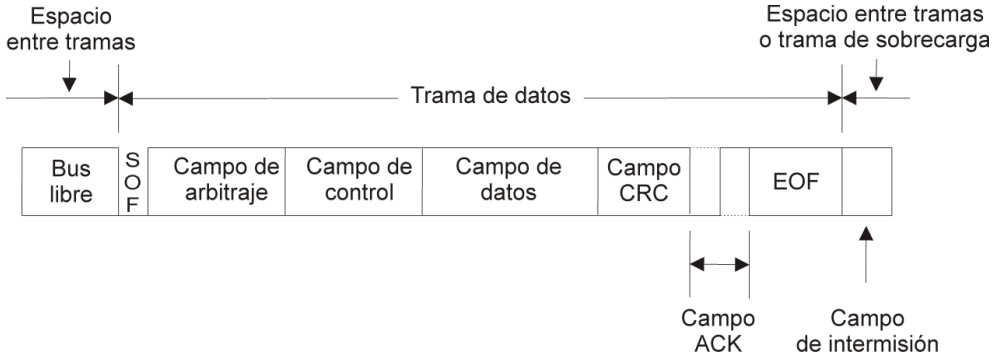

Figura 4.7. Formato de trama de datos CAN.

- Inicio de trama (SOF, *Start of Frame*): Indica el inicio de una trama de datos o una trama remota, y consiste en un bit dominante que sincroniza a todos los nodos activos en la red. Es el mismo para las tramas estándar y extendida.
- Campo de arbitraje (Arbitration field): Difiere según el formato de trama (véase Figura 4.8):
  - En el formato estándar está constituido por un identificador de 11 bits y el bit de petición de transmisión remota (RTR, *Remote Transmission Request*). El bit menos significativo del identificador se transmite al último y los 7 bits más significativos no pueden ser todos recesivos.

- En el formato extendido está formado por un identificador de 29 bits, el bit de petición remota substituta (SRR, *Substitute Remote Request*), el bit de extensión del identificador (IDE, *Identifier Extension*) y el bit RTR. El identificador se divide en dos secciones, la primera de 11 bits denominada base (*Base ID*) que corresponde al identificador del formato estándar, y la segunda sección de 18 bits conocida como extendida (*Extended ID*). Los bits de ambas secciones del identificador se transmiten en orden de mayor a menor prioridad.

El bit RTR debe ser dominante para ambos formatos de trama de datos y el bit SRR es un bit recesivo, por lo tanto, las posibles colisiones entre ambos tipos de formatos de trama que tengan el mismo valor en el campo Base ID se resuelven de manera que el formato de trama estándar predomina sobre el formato de trama extendida. En dicha resolución también se involucra al bit IDE, el cual pertenece al campo de arbitraje en el caso de un formato extendido y se encuentra en el campo de control para el caso de un formato de trama estándar. La transmisión del bit IDE es dominante para el formato estándar y recesivo para el extendido.

- Campo de control (*Control field*): Está compuesto de seis bits, IDE/r1, r0 y cuatro bits que forman el código de longitud de datos (DLC, *Data Length Code*). El primer bit que se transmite es IDE, el cual distingue entre los dos tipos de tramas; en seguida r0, en nivel dominante, que está reservado para futuras aplicaciones del protocolo CAN; finalmente se transmite el DLC para indicar el número de octetos contenidos en el campo de datos.
- Campo de datos (*Data field*): Contiene el mensaje a transmitir dentro de una trama CAN y puede tener una longitud de 0 a 8 octetos. Se transfiere primero el bit con mayor prioridad.
- Campo CRC (*CRC field*): Está constituido por una secuencia de 15 bits de verificación y un bit delimitador CRC (*CRC delimiter*), este último se transmite en un nivel recesivo. Mediante este campo, el receptor verifica

si la secuencia de bits recibidos fue alterada. Se utiliza el polinomio generador $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$.

- Campo de aceptación (*ACK field*): Está constituido por dos bits, ranura ACK (*ACK slot*) y delimitador ACK (*ACK delimiter*), este último siempre se transmite en un nivel recesivo. Todo nodo activo en la red CAN que recibe una trama válida, sobrescribe la ranura ACK con un nivel dominante, y con ello el transmisor verifica que su mensaje se envió correctamente; si por el contrario ningún nodo sobrescribe dicha ranura, el transmisor considera un error de transmisión.
- Fin de trama (EOF, *End of Frame*): Tanto la trama de datos como la trama remota están delimitadas por una secuencia de 7 bits recesivos que indican el fin de trama CAN. Cuando EOF está activo se realiza una violación al procedimiento de inserción de bit (véase apartado 4.1.1.1), por ello dicho procedimiento no se aplica a este campo.

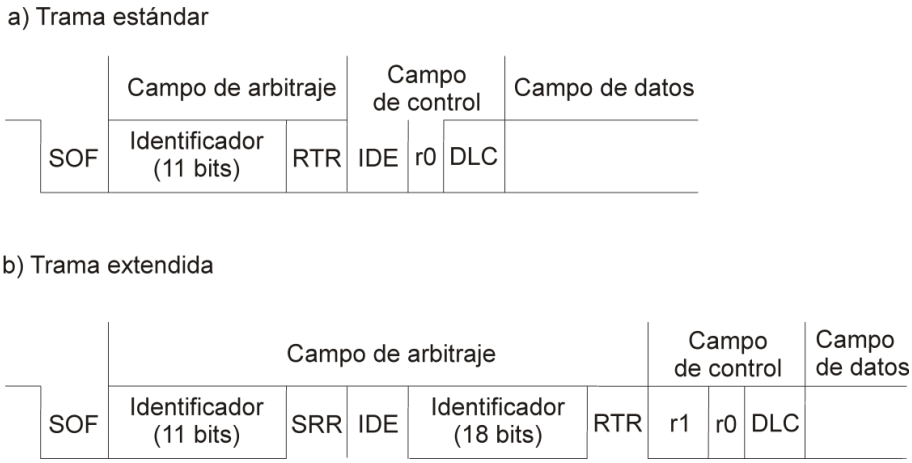

Figura 4.8. Formatos de tramas CAN, estándar y extendida.

4.3.2. Trama remota

Un nodo CAN, en modo receptor, puede iniciar la transmisión de su información mediante el envío de una trama remota, la cual se compone de seis campos tanto en formato estándar o extendido (véase Figura 4.9). Los campos de una trama remota son los mismos que los de una trama de datos, a excepción

que la trama remota no contiene el campo de datos y el bit RTR es recesivo. El valor del DLC debe coincidir con el de la trama de datos correspondiente.

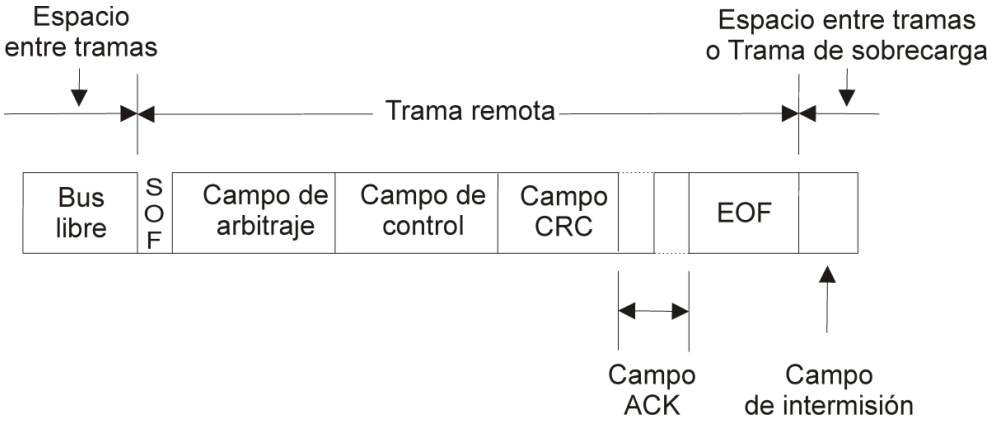

Figura 4.9. Formato de trama remota CAN.

4.3.3. Trama de error

La trama de error señaliza la detección de cualquier error durante la transmisión o recepción de una trama de datos o remota, la cual viola el procedimiento de inserción de bit y ocasiona que el transmisor reenvíe la trama. Así mismo, la detección de un error, durante la transmisión o recepción de una trama de sobrecarga o error, genera la transmisión de una nueva trama de error.

La trama de error está formada por dos campos (véase Figura 4.10):

- Bandera de error (*Error flag*): Existen dos formas de representar una bandera de error:
  - Bandera de error activo (*Active error flag*): Consiste en seis bits dominantes consecutivos.
  - Bandera de error pasivo (*Passive error flag*): Está formada por seis bits recesivos consecutivos, a menos que sean sobrescritos por bits dominantes de otros nodos.
- Delimitador de error (*Error delimiter*): Una trama de error termina con una secuencia de ocho bits recesivos. Posterior a la transmisión de una bandera de error, el nodo transmite bits recesivos y verifica el nivel del bus hasta que reconozca un bit recesivo, entonces comienza la

transmisión de otros siete bits recesivos. Con este mecanismo, el nodo puede determinar si fue el primero en transmitir una bandera de error y con ello detectar una condición de error.

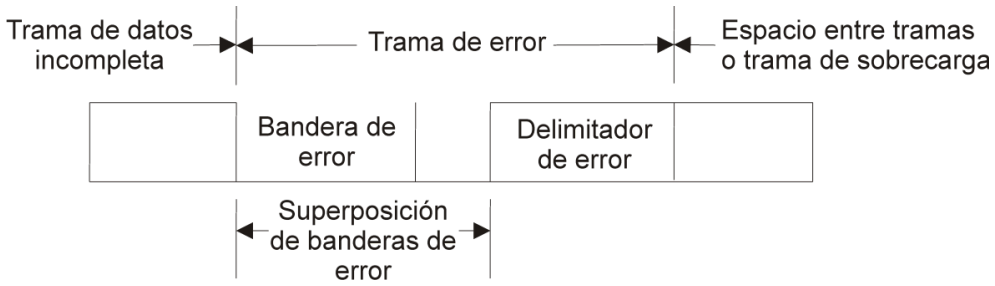

Figura 4.10. Formato de trama de error CAN.

4.3.4. Trama de sobrecarga

La trama de sobrecarga se utiliza para que un receptor solicite un retraso en la transmisión de la trama siguiente, ya sea de datos o remota, o para señalizar condiciones de error relacionadas con el campo de intermisión. El protocolo CAN permite la generación de dos tramas de sobrecarga como máximo para retrasar la transmisión de la siguiente trama.

Las tramas de sobrecarga se transmiten después de detectar las siguientes condiciones de error:

- Detección de un bit dominante durante los primeros dos bits del campo de intermisión. La detección de un bit dominante en el tercer bit del campo de intermisión se interpreta como un SOF.
- Cuando un receptor detecta un bit dominante en el último bit del campo EOF; o cuando un nodo, receptor o transmisor, detecta un bit dominante en el último bit del delimitador de una trama de error o de sobrecarga.

Una trama de sobrecarga se considera una forma especial de trama de error y consta de los siguientes campos (véase Figura 4.11):

- Bandera de sobrecarga (*Overload flag*): Se constituye por seis bits dominantes. La forma completa corresponde a la bandera de error activa.

- Delimitador de sobrecarga (*Overload delimiter*): Está formado por ocho bits recesivos.

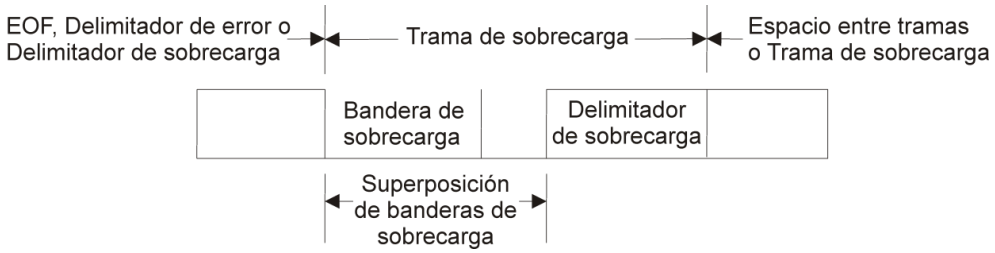

Figura 4.11. Formato de trama de sobrecarga CAN.

4.3.5. Espacio entre tramas

Las tramas de datos y remotas están separadas de tramas precedentes por un espacio entre tramas, a diferencia de las tramas de error y de sobrecarga que se transmiten en forma sucesiva, es decir sin un espacio entre tramas.

El espacio entre tramas está formado por tres campos:

- Intermisión (*Intermission*): Consiste en tres bits recesivos. Durante su transmisión la única acción que puede realizarse es señalar una condición de sobrecarga, y no se permite que ningún nodo inicie la transmisión de una trama de datos o remota.
- Bus libre (*Bus idle*): Este periodo es de longitud arbitraria y tiene un nivel recesivo hasta que algún nodo inicie la transmisión de una nueva trama.
- Suspender transmisión (*Suspend transmission*): Adicionalmente, el espacio entre tramas contiene un tiempo de inhibición de transmisión de ocho bits para nodos que se encuentren en estado de error pasivo.

4.3.6. Codificación de tramas

Los campos inicio de trama, arbitraje, control, datos y secuencia CRC, están codificados de acuerdo al procedimiento de inserción de bit que se describió en el inciso 4.1.1.1. Los campos restantes, Delimitador CRC, ACK y EOF, tienen un formato fijo y no siguen el procedimiento de inserción de bit; de igual forma, las

tramas de error y de sobrecarga tienen un formato fijo y no se codifican por dicho procedimiento (ISO/IEC 11898, 1993).

El instante de tiempo en el que se valida una trama difiere según:

- Transmisor: La trama es válida si no existen errores hasta el final del campo EOF. Si existe un error, en la trama se activa el proceso de recuperación.
- Receptor: La trama es válida si no existen errores hasta el siguiente bit después del campo EOF.

4.3.7. Detección y manejo de errores

Un controlador CAN cuenta con la capacidad de detectar y manejar los errores que surjan en una red CAN. Todo error detectado por un nodo, se notifica inmediatamente al resto de los nodos.

Un nodo puede tener una alteración local permanente, lo que provoca el envío continuo de tramas de error. Para prevenir dicho comportamiento, el protocolo CAN describe un algoritmo basado en la actividad del bus, que obliga a los nodos con errores permanentes a desconectarse de la red (*bus off*), para que los demás nodos no sean perturbados por los nodos defectuosos (Lawrenz, 1997).

4.3.7.1. Mecanismos de detección de errores

Para cumplir con las demandas relativas a la seguridad en la transmisión de datos, el protocolo CAN define los siguientes mecanismos para detección de errores:

- Monitoreo de bits: Todo nodo verifica que el nivel del bus transmitido sea el mismo nivel en el bus, y cuando dichos valores difieren se detecta un error de bit (*bit error*). El monitoreo de bits representa un mecanismo de seguridad global para la detección de todos los errores efectivos.
- Verificación del procedimiento de inserción de bit: Hace referencia al hecho de detectar un error de inserción de bit cuando ocurren seis niveles

consecutivos de bits con el mismo valor en un campo de trama codificado por el procedimiento de inserción de bit (*stuff error*).

- Verificación de redundancia cíclica de 15 bits: Se presenta un error de CRC (*CRC error*) cuando la secuencia CRC recibida no se corresponde con la secuencia CRC calculada.
- Verificación de trama: Detecta un error cuando un campo de bit de formato fijo contiene uno o más bits no válidos (*form error*).
- Verificación de aceptación: Un transmisor detecta un error de aceptación (ACK error) cuando la ranura ACK (*ACK slot*) no cambia a estado dominante (véase apartado 4.3.1).

4.3.7.2. Manejo de errores

Cuando un nodo detecta algún tipo de error, (de bit, de inserción de bit, de forma o de aceptación), inicia la transmisión de una bandera de error (*error flag*) en el siguiente bit. Cuando se detecta un error de CRC, se inicia la transmisión de una trama de error después del delimitador ACK, a excepción de que previamente se haya transmitido otra trama de error.

El manejo de errores se lleva a cabo de acuerdo con el diagrama de flujo de la Figura 4.12.

4.3.7.3. Capacidad de detección de errores

Los protocolos de comunicaciones de bus seriales emplean diferentes mecanismos de detección de error, los cuales proporcionan al receptor la capacidad de detectar si la trama recibida es errónea o no. Por otro lado, la capacidad de detectar errores de transmisión depende de los mecanismos de error y del protocolo utilizado.

Para valorar la integridad de los datos en un sistema de transmisión se deben considerar las interferencias electromagnéticas y la capacidad para detectar errores de transmisión. En un sistema de transmisión de datos existe una medida estadística que representa la integridad de datos transferidos conocida

como probabilidad de error residual, la cual especifica la probabilidad de que existan tramas erróneas sin detectar y está dada por la Ecuación 4.1.

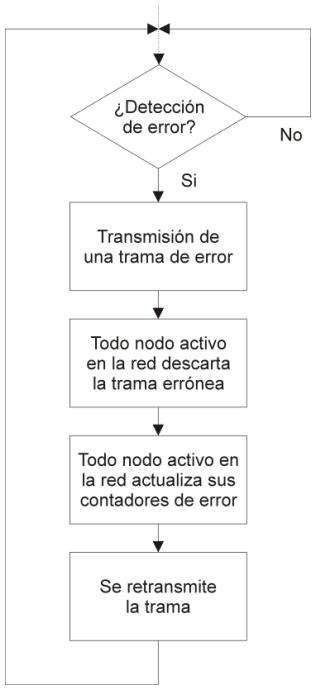

Figura 4.12. Diagrama de flujo para el manejo de errores.

$$Probabilidad\ de\ error\ residual < (4.7x10^{-11})(Velocidad\ de\ error\ de\ trama)$$ Ecuacion 4.1

La cual define el número de errores de transmisión sin detectar durante el tiempo de operación de una red CAN, que puede ser estimado por la velocidad de error de trama, el porcentaje de carga del bus, la velocidad de transferencia de datos y la longitud de la trama.

La mayoría de errores de transmisión se deben a interferencias electromagnéticas. La susceptibilidad de error de un sistema de transmisión de datos se puede caracterizar mediante parámetros estadísticos tales como velocidad de error de trama (*frame error rate*) y probabilidad de error de bit (*bit error probability*). La velocidad de error (*error rate*) está dada por la relación entre el número de tramas erróneas y el número total de tramas transmitidas durante un periodo de tiempo, con lo cual se describe la probabilidad de que exista una trama errónea, por otro lado, la probabilidad de error de bit especifica la probabilidad de que exista un bit erróneo en una trama transmitida.

Los errores de transmisión no ocurren en todos los nodos de una red, sino que aparecen en nodos individuales con diversos patrones de error. El protocolo CAN tiene la capacidad de señalizar los errores detectados mediante la transmisión de una bandera de error, sin embargo, existe la posibilidad de no detectar errores locales cuando se cumplen las siguientes condiciones en la distribución del error de bit:

- El nodo que transmite funciona adecuadamente, puede detectar un error al monitorear el bit y señalizarlo a todo el sistema.
- Todos los nodos receptores que reciben una trama errónea detectan un patrón de error no definido (indetectable). Debido a que estos patrones son muy raros, todos los nodos que recibieron la trama errónea deben mostrar un patrón de error idéntico. Esta condición es cada vez más improbable en un sistema distribuido con un gran número de nodos.

La DLL del protocolo de comunicaciones CAN garantiza un alto grado de detección de errores.

Todos los errores se detectan mediante el proceso de monitoreo de bit realizado por el transmisor de la trama, mediante dicho proceso el transmisor puede detectar errores a nivel global ocasionados por interferencia electromagnética, a nivel local, estos últimos ocurren en receptores individuales y se detectan mediante la verificación de la secuencia CRC de 15 bits, realizada por el mismo receptor.

4.4. Capa de supervisor

La sustitución del cableado convencional por un sistema de bus serie presenta el problema de que un nodo defectuoso puede bloquear el funcionamiento del sistema completo. Como se ha descrito con anterioridad, cada nodo activo transmite una bandera de error cuando detecta algún tipo de error (principio de señalización de errores) y puede ocasionar que un nodo defectuoso pueda acaparar el medio físico; para eliminar este riesgo, el protocolo CAN define

un mecanismo autónomo para detectar y desconectar un nodo defectuoso del bus, dicho mecanismo se conoce como aislamiento de fallos (*fault confinement*) (ISO/IEC 11898, 1993) y (Robert Bosch GmbH, 1991).

4.4.1. Aislamiento de fallos

El objetivo del aislamiento de fallos es preservar la disponibilidad del sistema de transmisión de datos, incluso en el caso de que existan nodos defectuosos, para ello se definen estrategias que incrementan la seguridad en los siguientes aspectos:

- Distinguir entre errores temporales (*short disturbances*) y fallas permanentes (*permanent failures*).
- Localizar y desconectar (*switched off*) nodos defectuosos (*defective nodes*).

Por ello, cada nodo tiene un contador de errores de transmisión (TEC, *Transmit Error Counter*) y un contador de errores de recepción (REC, *Receive Error Counter*) para registrar el número de errores respectivamente. Si las tramas se transmiten y reciben correctamente, dichos contadores decrementan sus valores, y en caso de ocurrir errores, los contadores se incrementan en proporciones mayores a los que se decrementan.

Los contadores incrementan o decrementan sus valores dependiendo de la relación existente entre las tramas enviadas y recibidas correctamente y aquellas con errores. Los valores en los contadores indican la frecuencia relativa de perturbaciones previas.

El comportamiento de los nodos se modifica, en relación a los errores, dependiendo de los valores del contador correspondiente. El incremento o decremento de los contadores se modifica de acuerdo a las reglas establecidas en (Robert Bosch GmbH, 1991) (ISO/IEC 11898, 1993).

Dependiendo del valor de sus contadores de error, un nodo puede estar en uno de los siguientes estados (véase Figura 4.13):

- Error activo (*Active error*): Un nodo toma parte en la comunicación de bus y cuando detecta un error envía una bandera de error activo; destruye la trama que transmitía, viola el procedimiento de inserción de bit y previene con ello a los demás nodos de la presencia de una trama errónea.
- Error pasivo (*Passive error*): Un nodo no puede enviar una bandera de error activo pero toma parte en la comunicación del bus; cuando detecta un error, sólo puede enviar una bandera de error pasivo, la cual no interfiere en la comunicación del bus.
- Desconectado (*Bus off*): En este estado, un nodo no tiene ninguna influencia en el bus y por ello no puede transmitir tramas, aceptación ACK, tramas de error o de sobrecarga.

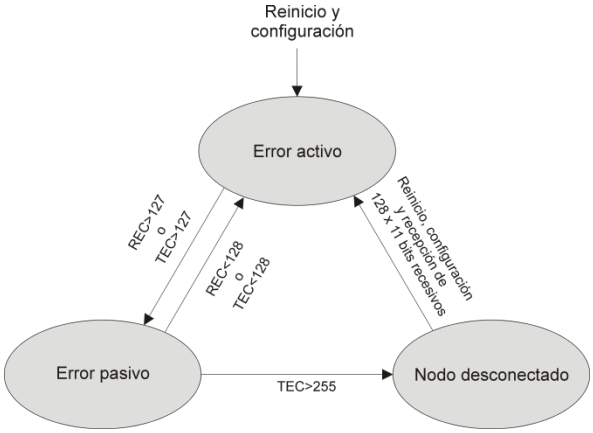

Figura 4.13. Diagrama de estados de error de un nodo CAN.

4.5. Capa de aplicación

CAN se utiliza en aplicaciones en las que no está determinada una estructura de capas entre el nivel de enlace proporcionado por el controlador de protocolo y la aplicación en el nodo. Actualmente, con la implementación de sistemas distribuidos basados en CAN han surgido nuevos requerimientos que no han sido considerados en el estándar ISO 11898-2, siendo los más importantes:

- Disponibilidad de servicios de transmisión para bloques de datos mayores a 8 bytes.

- Soporte al modelo Cliente/Servidor.
- Funciones dedicadas a la gestión de red (*network management*).
- Métodos para asignar identificadores de mensaje y configuración de parámetros específicos del nodo, de forma transparente al usuario.
- Interoperabilidad e intercambio de dispositivos de diferentes fabricantes.
- Estandarizar la funcionalidad y definir nuevos perfiles para dispositivos.

La necesidad de estandarizar las capas de aplicación ha surgido sobre todo en el sector de los FBs industriales, donde la comunicación entre dispositivos de diferentes fabricantes es una característica.

Respecto al protocolo CAN, existen diferentes estándares que definen su capa de aplicación; algunos son muy específicos y están relacionados con sus campos de aplicación. Las capas de aplicación más utilizadas son las siguientes:

- CAL: Define un amplio conjunto de funciones para la comunicación y gestión de una red CAN.
- CANopen: Protocolo de ámbito europeo, respaldado por CiA. Utiliza parte de CAL y le agrega nuevos perfiles de protocolo y dispositivos.
- DeviceNet: Desarrollado por la firma Allen-Bradley y de mayor utilización en E.U.A. Es un enlace de bajo costo que conecta dispositivos industriales a una red CAN.
- SDS (*Smart Distributed System*): Sistema de bus desarrollado por la firma Honeywell para la interconexión de sensores y actuadores inteligentes.
- OSEK (*Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug*): En inglés "*Open Systems and the Corresponding Interfaces for Automotive Electronics*", desarrollado por la firma Siemens como resultado de la cooperación entre fabricantes de automóviles alemanes para desarrollar una arquitectura abierta para unidades de control distribuido (OSEK-VDX, 2004).

- CANKingdom: Desarrollado por la firma Kvaser y compatible con DeviceNet, protocolo reconocido a nivel mundial en el control de maquinaria (CANkingdom International Inc., 2003).

4.6. Componentes CAN en el mercado

En este apartado se presentan algunos de los componentes encontrados en el mercado que se utilizan para comunicar datos sobre un bus CAN. Éstos son transceptores y microcontroladores de los diferentes fabricantes. Debido a la aceptación que ha logrado el protocolo de comunicaciones CAN, apoyado por fabricantes de semiconductores, organizaciones de estandarización y la SAE, actualmente existe una gran variedad de dispositivos que implementan dicho protocolo para diversas aplicaciones.

4.6.1. Manipuladores de protocolo (*Protocol handlers*)

Los manipuladores de protocolo también son conocidos como controladores de protocolo. Éstos se encargan de generar y decodificar el protocolo.

Los manipuladores de protocolo no pueden operar por sí sólos, sino que tienen que ser controlados por un microcontrolador. Por consiguiente, cuentan con una interfaz de comunicación paralela o serial. El controlador de protocolo de la marca Microchip MCP2510 implementa las especificaciones 2.0 A y 2.0 B. Soporta las versiones del protocolo CAN 1.2, CAN 2.0A, CAN 2.0 B pasiva y CAN 2.0 B activa. Es capaz de transmitir y recibir mensajes estándares y extendidos. También tiene la capacidad de aceptar y manipular los mensajes. Incluye tres registros de almacenamiento temporal para transmitir y dos registros de almacenamiento temporal para recibir los mensajes, los cuales reducen tráfico al microcontrolador. La comunicación entre el controlador y el microcontrolador es vía el estándar SPI (*Serial Peripheral Interface*) con velocidades de hasta 5 Mbps.

4.6.2. *Line drivers*

El propósito de CAN es el intercambio de información, en otras palabras, el uso de un medio para el transporte de mensajes. Debido a esto, existen diferentes drivers que llevan a cabo el control del medio llamado transmisor-receptor.

El transmisor-receptor CAN MCP2551 de alta velocidad es tolerante a fallos y funciona como una interface entre el controlador CAN y el bus físico. El MCP2551 provee la capacidad de transmitir y recibir voltajes diferenciales al controlador CAN y es compatible con el estándar ISO-11898, incluyendo los requerimientos de 24 V. Opera a velocidades de transmisión de hasta 1 Mbps. Típicamente, cada nodo en un sistema CAN debe tener un dispositivo que convierta las señales digitales generadas por un controlador a señales adaptadas para la transmisión sobre el bus. Otro transmisor-receptor es el TJA1054 cuya función es interfazar el controlador del protocolo y el medio físico. Su uso principal es en redes CAN de baja velocidad con velocidades de hasta 125 Kbps en automóviles. El dispositivo provee un voltaje diferencial de transmisión y recepción, pero cambia a transmisión y recepción de un sólo hilo en condiciones de errores.

4.6.3. Microcontroladores con CAN integrado (*on-board CAN handlers*)

Con el objetivo de reducir los costos de diseño, reducir el tamaño de la circuitería, evitar problemas de velocidad, conexión al CPU, la mayoría de las industrias manufactureras incluyen un controlador CAN en sus microcontroladores, como por ejemplo el PIC8F4680 de la marca Microchip, contiene un módulo CAN empotrado que consiste en manejar todas las funciones de transmitir y recibir mensajes a través del bus. Los mensajes son transmitidos primero almacenando los datos en los registros correspondientes. Se puede verificar la señalización del estado y errores en los registros destinados para ello. Se verifica cualquier mensaje detectado en el bus si hay errores y si coincide la

dirección con el nodo se recibe, se almacena en uno de los dos registros de recepción.

5. Especificación Formal de la Pasarela CAN/LIN

Los desarrolladores de aplicaciones para sistemas de comunicaciones en el automóvil evalúan las alternativas y soluciones de los protocolos de comunicaciones mediante su análisis, simulación y validación por diferentes métodos mientras se encuentran en la fase de diseño, en el proceso de producción y antes de ser lanzados al mercado o ser estandarizados.

Para evaluar las alternativas y soluciones en el desarrollo del sistema, el primer paso es obtener los requerimientos, con base a esto se realiza la especificación de los componentes y se da la posibilidad de mejorar la solución; posteriormente, teniendo la especificación se procede a realizar la implementación. Los requerimientos deben ser expresados de tal manera que las definiciones y descripciones de las funcionalidades de los componentes sean únicas.

La dificultad de la especificación de los requerimientos, propiedades y funciones se presenta cuando los recursos humanos no cuentan con una capacidad lingüística especializada, por lo tanto, la descripción de significados presenta ambigüedades y es casi imposible describir requerimientos de una manera arbitraria. Además, ningún diseñador de productos o sistemas expresa todas sus ideas y la describe adecuadamente en un lenguaje formal. En la mayoría de los casos las especificaciones son realizadas en documentos textuales, esto se debe a que los lenguajes formales no son ampliamente conocidos o no se cuenta con las herramientas disponibles.

Por otro lado, con la cantidad cada vez mayor de datos que necesitan comunicar las ECUs en los vehículos, el protocolo CAN y el protocolo LIN fueron creados para satisfacer las demandas de la industria y de los consumidores. Como resultado, los ingenieros en la industria del automóvil tienen que conocer y comprender a detalle los protocolos con el fin de ser capaces de tomar ventaja de sus capacidades. Debido a que el intercambio de la información de un protocolo a otro es crítico, en este capítulo se describe la especificación de la pasarela CAN/LIN, considerando inicialmente las especificaciones formales de los protocolos de comunicaciones LIN y CAN respectivamente, utilizando el FDL Cinderella SDL.

5.1. Requerimientos de la pasarela CAN/LIN

5.1.1. Requerimientos del protocolo de comunicaciones LIN

Con base en las carácteríticas del protocolo de comunicaciones LIN (Capítulo 3) se definen las siguientes restricciones respecto a su especificación:

- Este trabajo se enfoca en la especificación de las capas DLL y de supervisor, de acuerdo al modelo de referencia OSI (véase Figura 3.4) utilizando la especificiación LIN rev 2.2A.
- Debido a que el medio físico no está definido en la especificación del protocolo de comunicaciones LIN, se realiza una especificación mínima para ejecutar la especificación.
- No se considera la capa de aplicación, ya que ésta no es parte de la especificación del protocolo.
- Se asumirá que los datos que recibe la capa DLL, provenientes de la capa de aplicación, son correctos.
- El sistema LIN está formado por un sólo nodo activo en el bus, ya que es necesario para establecer un enlace de comunicación LIN.

5.1.2. Objetos de la especificación del protocolo de comunicaciones LIN

Una vez estudiado y analizado la especificación LIN rev 2.2A, el siguiente paso fue realizar la especificación formal, la definición exacta de la configuración del sistema que se va a formalizar y la identificación de las diferentes partes u objetos que lo constituyen. Con esta información se construyó la especificación estática del sistema utilizando esquemas en lenguaje SDL.

Posteriormente, se describieron los objetos creados para el desarrollo de la especificación del sistema, éstos consisten en bloques, sub-bloques y procesos que constituyen la especificación estática del protocolo de comunicaciones LIN. Los nombres asociados a cada objeto describen la funcion que éstos desempeñan y se presentan en orden descendente (arquitectura del sistema, arquitectura de bloque y nivel de procesos), definiendo el diseño arquitectural de la metodología de diseño. La Figura 5.1 muestra la representación de niveles jerárquicos en un diagrama de árbol de los elementos que constituyen al sistema, representados como objetos SDL, los cuales se describen a continuación:

- La unidad que realiza la función del medio físico, representando una topología Maestro/Esclavo.
- Una ECU, la cual está formada por tres objetos: capa física, capa DLL_LIN y capa de supervisor.
    - PHY_LIN: Realiza las funciones de la capa física.
    - DLL_LIN: Se encarga del nivel de enlace de datos y se compone de otros dos objetos que representan las dos subcapas que forman el protocolo de comunicaciones LIN:
        - LLC. Define las tareas independientes del método de acceso al medio.
        - MAC. Realiza el control del acceso al medio mediante la gestión del testigo, la codificación, envío, recepción e identificación de tramas, el control de errores, etc.

- Supervisor_LIN: Realiza las funciones de la capa de supervisor y se compone de un objeto que representa la subcapa que gestiona los fallos del protocolo de comunicaciones LIN:
  - Fault Confinement. Gestiona los fallos del protocolo de comunicaciones LIN.

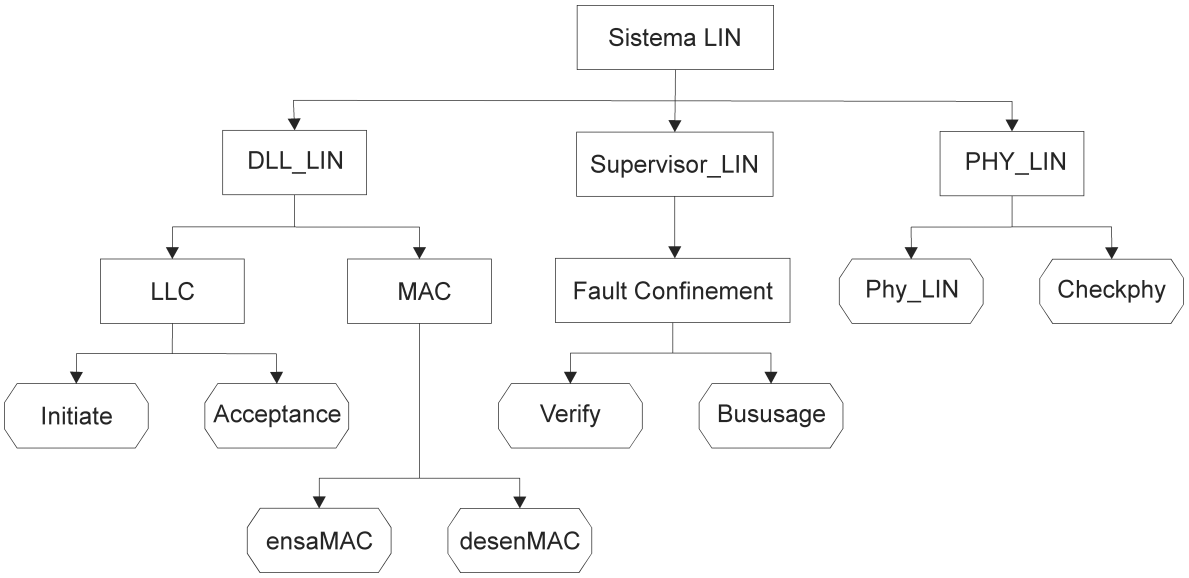

Figura 5.1. Jerarquía de los objetos de la especificación LIN.

5.1.3. Especificación de datos del protocolo LIN

Los tipos de datos usados para la especificación del protocolo LIN se declaran en la librería (*package*) CANLIN_Datos empleando la norma ANS.1 (ITU-T, 2000), esta librería está declarada a nivel de entorno del sistema, lo cual permite que sean accesibles en los demás niveles de abstracción.

La Figura 5.2 muestra los tipos de datos que permiten la creación y manipulación de las tramas y de las estructuras que las generan. Los siguientes tipos de datos son la base del protocolo LIN ya que con éstos se permite crear la mayoría de los datos empleados:

- Byte: Este tipo de datos genera una cadena de 8 bits para la creación de los datos del protocolo.

- Bit_t: Este tipo de datos crea un arreglo de t bits para identificar el campo de control de cada capa del protocolo.
- Bit_4: Este tipo de datos crea un arreglo de 4 bits para los códigos de función del protocolo.

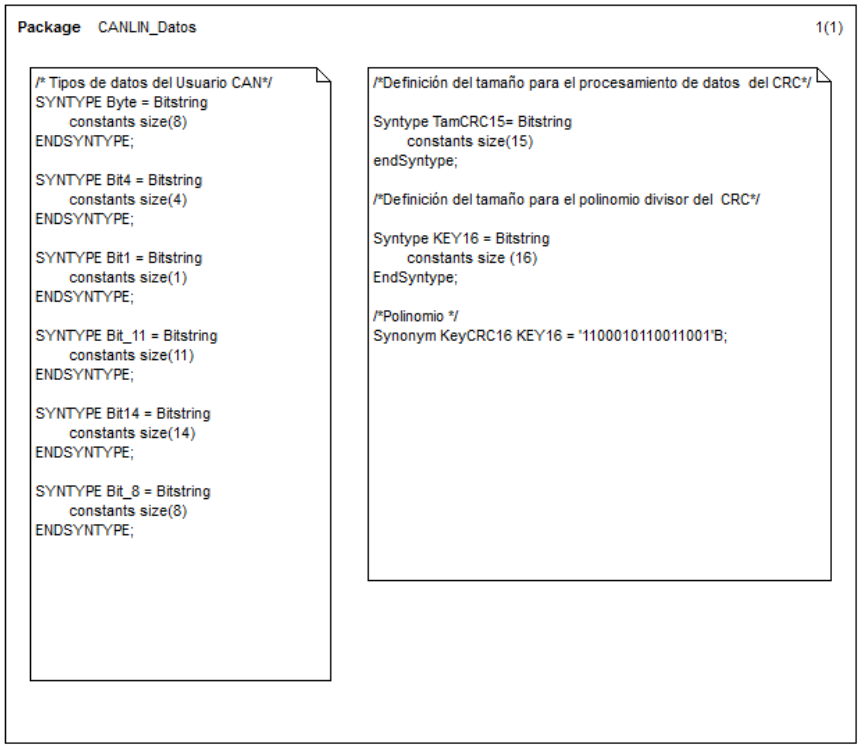

Figura 5.2. Tipo de datos para el procesamiento de tramas CAN y LIN.

5.1.4. Requerimientos del protocolo de comunicaciones CAN

Con base a las características del protocolo de comunicaciones CAN (Capítulo 4) se definen las siguientes restricciones respecto a su especificación:

- Este trabajo se enfoca en la especificación de la capa DLL y de supervisor, de acuerdo al modelo de referencia OSI (véase Figura 4.1).
- Debido a que el medio físico no está definido en la especificación del protocolo de comunicaciones CAN, se realiza una especificación mínima para ejecutar la especificación.
- No se considera la capa de aplicación, ya que ésta no es parte de la especificación del protocolo.

- Se asumirá que los datos que recibe la capa DLL, provenientes de la capa de aplicación, son correctos.
- El sistema CAN está formado por un sólo nodo activo en el bus, ya que es necesario para establecer un enlace de comunicación CAN.

5.1.5. Objetos de la especificación del protocolo de comunicaciones CAN

Una vez estudiados y analizados los estándares ISO 11898 (ISO/IEC 11898, 1993) e ISO 11519 (ISO/IEC 11519, 1994) y la especificación CAN (Robert Bosch GmbH, 1991), el siguiente paso fue la realización de la especificación formal, la definición exacta de la configuración del sistema que se va a formalizar y la identificación de las diferentes partes u objetos que lo constituyen. Con esta información se construye la especificación estática del sistema utilizando esquemas en lenguaje SDL.

Posteriormente se describieron los objetos creados para el desarrollo de la especificación del sistema, éstos consisten de bloques, sub-bloques y procesos que constituyen la especificación estática del protocolo de comunicaciones CAN. Los nombres asociados a cada objeto describen la funcion que éstos desempeñan y se presentan en orden descendente (arquitectura del sistema, arquitectura de bloque y nivel de procesos), definiendo el diseño arquitectural de la metodología de diseño. La Figura 5.3 muestra la representación de niveles jerárquicos en un diagrama de árbol de los elementos que constituyen al sistema, representados como objetos SDL, los cuales se describen a continuación:

- La unidad que realiza la función del medio físico, representando una topología de bus.
- Una ECU, la cual está formada por tres objetos, la capa física, la capa DLL_CAN y la capa de supervisor.
    - PHY_CAN: Realiza las funciones de la capa física.

- DLL_CAN: Se encarga del nivel de enlace de datos y se compone de dos objetos que representan las dos subcapas que forman el protocolo de comunicaciones CAN.
  - LLC_CAN. Define las tareas independientes del método de acceso al medio.
  - MAC_CAN. Realiza el control del acceso al medio mediante la gestión del testigo, la codificación, envío, recepción e identificación de tramas, el control de errores, etc.
- Supervisor_CAN: Realiza las funciones de la capa de supervisor, y se compone de un objeto que representa la subcapa que gestiona los fallos del protocolo de comunicaciones LIN.
  - Fault_Confinement. Gestión de fallos del protocolo de comunicaciones CAN.

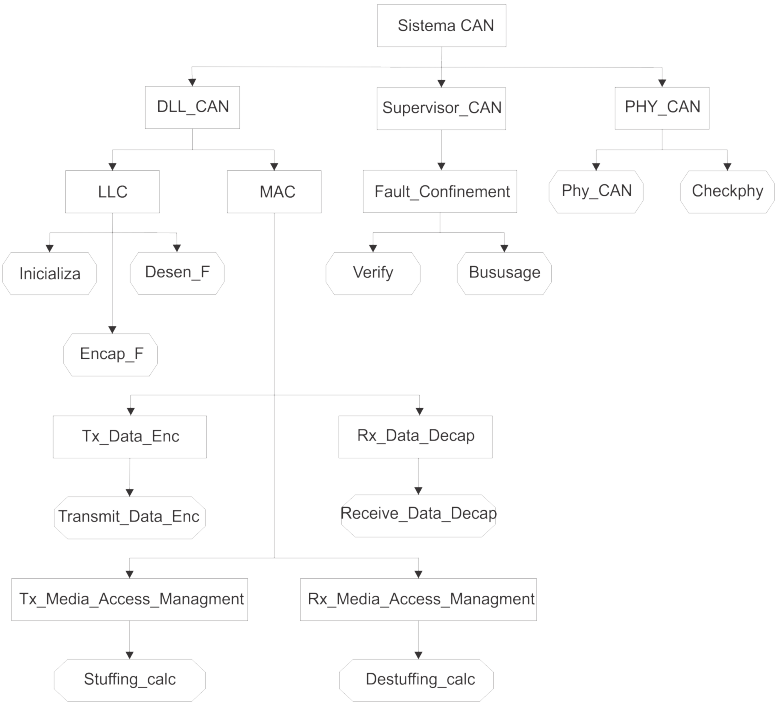

Figura 5.3. Jerarquía de los objetos de la especificación CAN.

5.1.6. Especificación de datos

Los tipos de datos usados para la especificación del protocolo CAN se declaran dentro de la librería (*package*) CANLIN_Datos empleando la norma ANS.1 (ITU-T, 2000), esta librería está declarada a nivel de entorno del sistema, lo cual permite que sean accesibles en los demás niveles de abstracción.

La Figura 5.2 muestra los tipos de datos que permiten la creación y manipulación de las tramas y de las estructuras que las generan. Los siguientes tipos de datos son la base del protocolo CAN ya que con éstos se permite crear la mayoría de los datos empleados:

- Byte: Este tipo de datos genera una cadena de 8 bits para la creación de los datos del protocolo.
- Bit_t: Este tipo de datos crea un arreglo de t bits para identificar el campo de control de cada capa del protocolo.
- Bit_4: Este tipo de datos crea un arreglo de 4 bits para los códigos de función del protocolo.
- TamCRC15: Tiene un tamaño de 15 bits y se utiliza para almacenar el CRC_15 de cada trama.
- Key16: Crea un arreglo de 16 bits para el polinomio generador del CRC_15.

5.1.7. Objetos de la especificación de la pasarela CAN/LIN

A continuación, se describen los procesos creados para el desarrollo de la especificación CAN/LIN, éstos consistieron en modificar los procesos Initiate y Acceptance pertenecientes al bloque LLC_LIN, y los procesos Desen_F e Inicializa pertenecientes al bloque LLC_CAN de cada una de las especificaciones de los protocolos de comunicaciones LIN y CAN respectivamente, para poder realizar el intercambio de datos de un protocolo a otro y viceversa. Los nombres asociados a cada proceso describen la funcion que éstos desempeñan.

5.1.8. Especificación de datos de la pasarela CAN/LIN

Los tipos de datos usados para la especificación de la pasarela CAN/LIN se declaran en la librería (*package*) CANLIN_Datos empleando la norma ANS.1 (ITU-T, 2000), hace referencia a las mismas librerías de los protocolos de comunicaciones CAN/LIN y está declarada a nivel de entorno del sistema, lo cual permite que sean accesibles en los demás niveles de abstracción.

5.2. Especificación formal de la Pasarela CAN/LIN

Mediante la utilización de un conjunto de elementos SDL, la descripción estática descompone un sistema en varias partes atendiendo a sus diferentes funciones. Cada una de estas partes es susceptible de ser dividida en unidades más sencillas hasta alcanzar el nivel de detalle que se pretende. Las unidades del último nivel contendrán la descripción del comportamiento del sistema.

Los procesos se encuentran en el ultimo nivel de abstracción y contienen la información que describe como el sistema interacciona con su entorno, como ha de responder a las situaciones de operación para las que fue diseñado. Esta información está implícita en el comportamiento de la máquina de estados que cada proceso tiene asociado. La descripción de la parte dinámica de un sistema SDL se reduce por tanto a la especificación de un conjunto de máquinas de estados. Esto se realiza directamente en el interior del proceso o, si está compuesto por servicios, en el interior de éstos últimos.

5.2.1. Especificación Estática de la Pasarela CAN/LIN

Con los objetos identificados se construye la descripción estática del sistema SDL, que los organiza en varios niveles de abstracción (véase Figura 5.4), y define cómo debe reaccionar cada uno de ellos a los eventos externos mediante la utilización de señales.

A continuación, se describen los elementos que constituyen la estructura de la especificación formal: sistema, bloques, subestructuras del bloque, tipos de bloque, procesos, canales, rutas y señales.

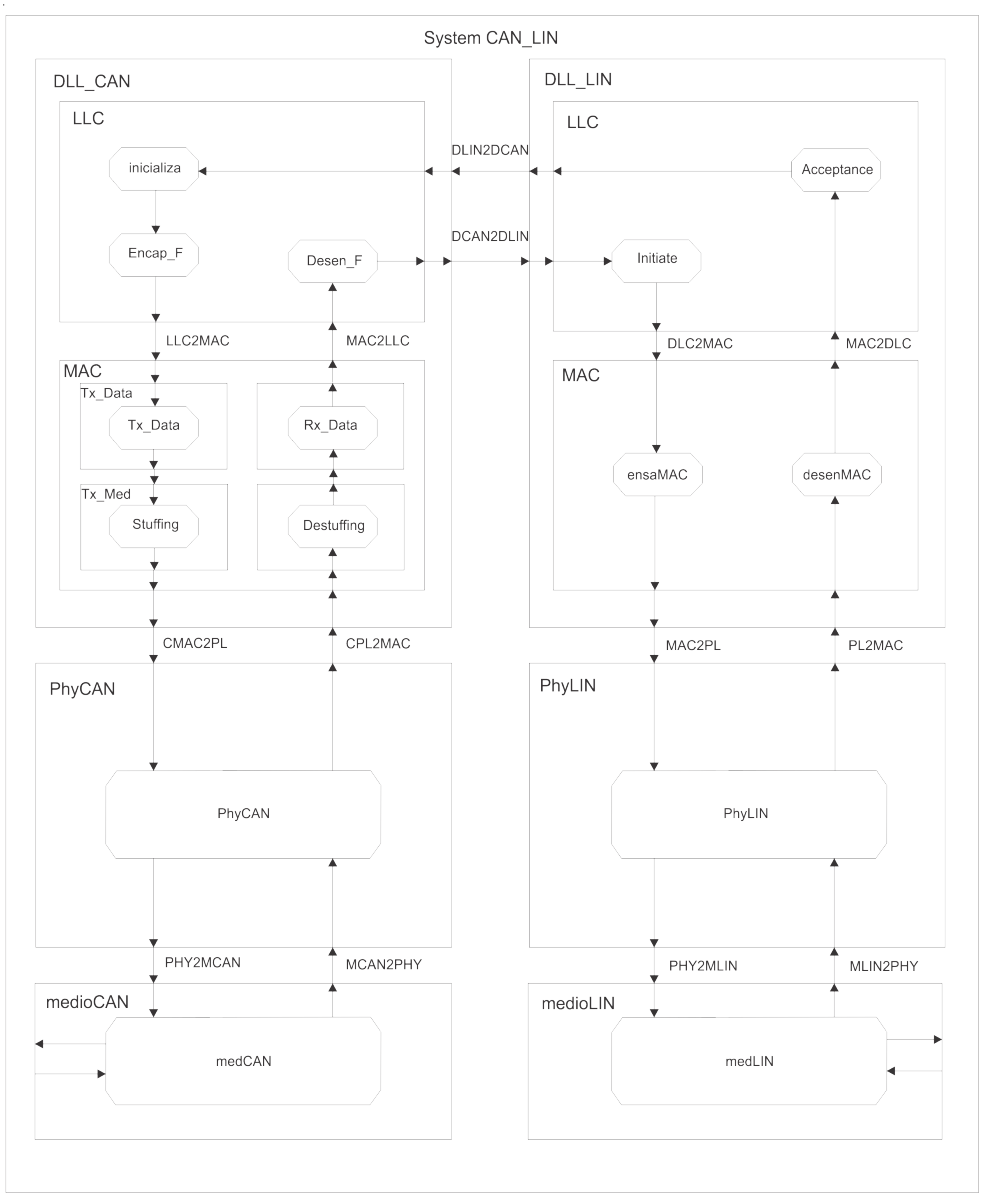

Figura 5.4. Especificación de la Pasarela CAN/LIN.

5.2.1.1. Especificación de canales, rutas de señal y compuertas

La especificación de los objetos SDL se puede realizar utilizando una representación remota basada en tipos (*type*) o representaciones remotas basadas en señales específicas. Para el desarrollo de los objetos del sistema se

utilizan especificaciones remotas basadas en señales específicas, excepto los procesos medCAN y medLIN. Debido a que los tipos (bloque o proceso) son especificaciones genéricas, no tienen conexión alguna con canales o rutas de señal específicas, es necesario un medio para el intercambio de señales; esto se lleva a cabo mediante la especificación de compuertas, las cuales pueden ser unidireccionales o bidireccionales estableciendo en cualquiera de los dos casos las señales que participarán en el intercambio.

A partir de la versión SDL-96 es posible omitir los nombres de los canales o rutas de señal, así como el identificador del alfabeto de entrada que tengan asociadas las entradas y salidas, siempre y cuando no causen ambigüedad. En el presente trabajo de tesis se hace uso de esta propiedad, para facilitar tanto la escritura como la lectura de la especificación final.

5.2.1.1.1. Especificación de señales

Las señales son la base para establecer la comunicación entre las entidades del sistema y se transportan a través de canales o rutas de señal especificadas. La Figura 5.5 muestra las señales descritas en los estándares ISO 11898 (ISO/IEC 11898, 1993) e ISO 11519 (ISO/IEC 11519, 1994), especificación LIN Rev 2.2A (LIN Consortium, 2010), y también las señales que no son parte de la especificación pero que han sido introducidas con el objetivo de construir una especificación ejecutable, éstas únicamente son válidas para una DLL_CAN y una DLL_LIN, y para evitar ambigüedades se renombraron las señales en las segundas DLL_CAN y DLL_LIN. Algunas de estas señales son parametrizadas[9], transportan información necesaria para procesamiento o control del protocolo.

[9] El término parametrización hace referencia a determinar el modo de ejecución de una acción considerando la configuración de las variables o señales.

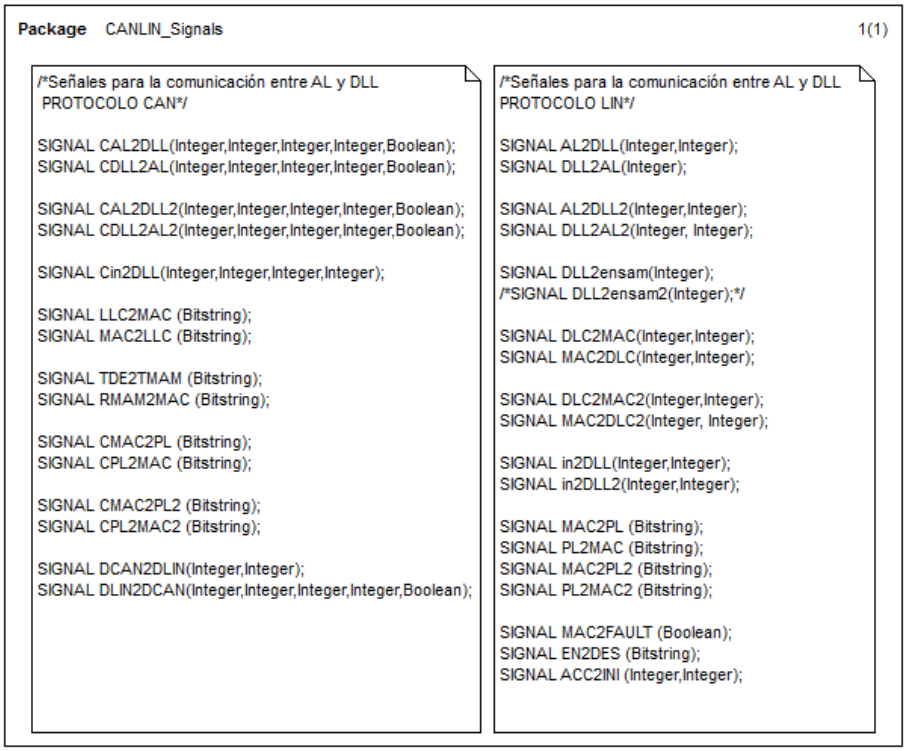

Figura 5.5. Declaración de las señales del sistema.

5.2.2. Sistema CAN_LIN

Los tipos de datos, constantes de tiempo, señales y procedimientos se declaran en librerías (véase Figura 5.6). El sistema consiste en una instancia del tipo CAN_LIN, y representa el menor nivel de abstracción. La declaración del sistema CAN_LIN está conformado por dos bloques DLL_CAN, dos bloques DLL_LIN, por un bloque PHY_CAN, un bloque PHY_LIN, un bloque Supervisor_CAN, un bloque Supervisor_LIN, por un bloque medioCAN y un bloque medioLIN. La Figura 5.7 muestra dichos bloques a nivel de sistema.

5.2.2.1. Bloque medio

El bloque medio representa el medio físico para la transmisión de datos empleando una topología Maestro/Esclavo para el nodo LIN y una topología Multimaestro para el nodo CAN, mediante una conexión punto a punto, con lo que se establecen dos medios físicos para la comunicación entre DDL_CAN y DLL_LIN.

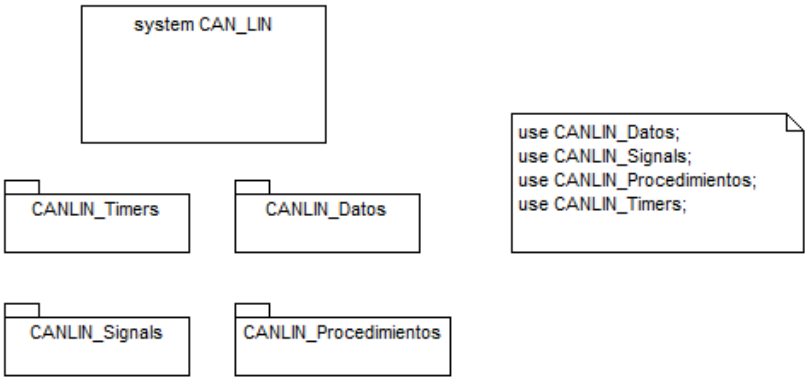

Figura 5.6. Librerías de la Especificación de la pasarela CAN/LIN.

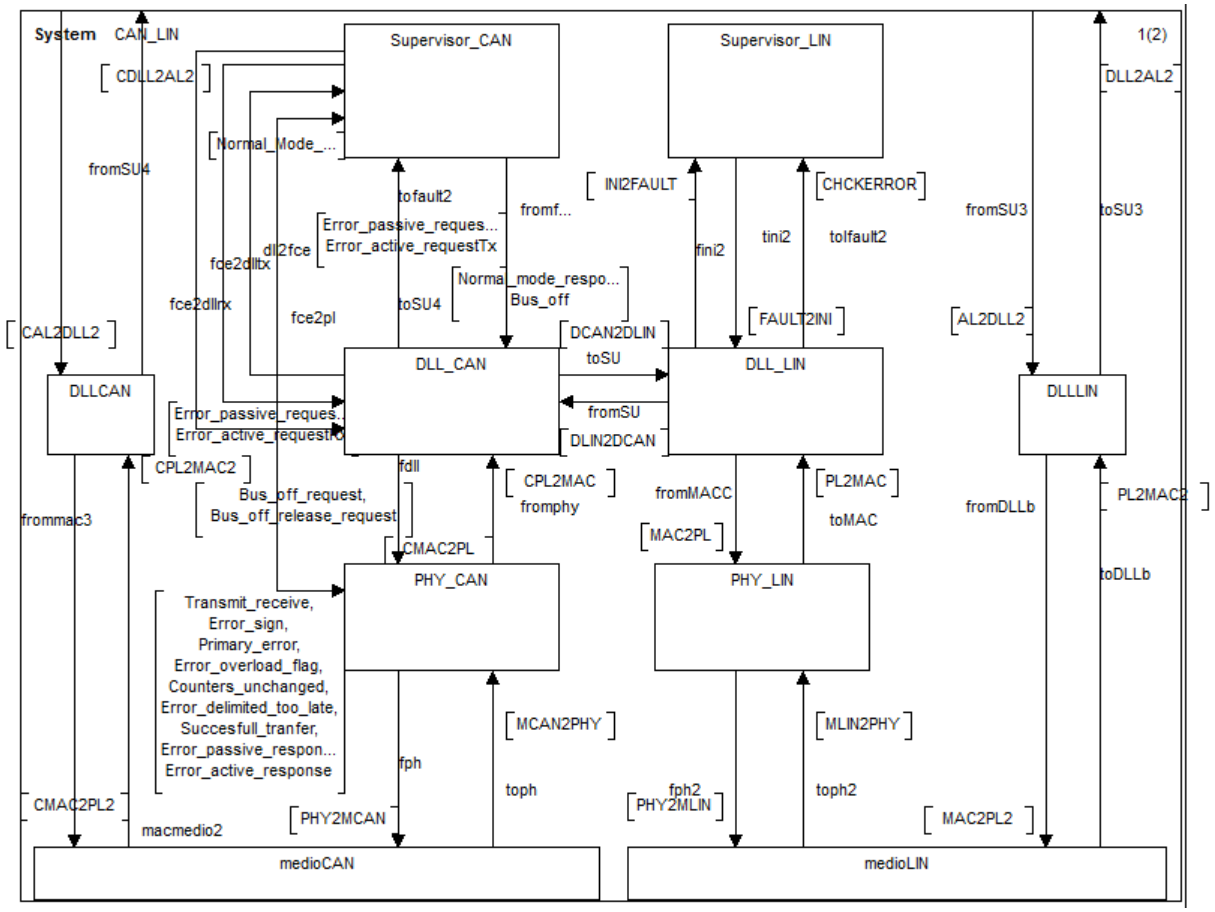

Figura 5.7. Especificación del sistema CAN/LIN.

Los bloques medioLIN y medioCAN se componen de un único proceso llamados medLIN y medCAN, respectivamente, estos procesos sólo realizan el paso de las señales mediante los canales CMAC2PL2, CPL2MAC, PHY2MCAN y MCAN2PHY para el medio CAN y los canales MAC2PL, PL2MAC, PHY2MLIN y MLIN2PHY para el medio

LIN, los cuales se encargan de enviar la información de manera correcta a los dispositivos correspondientes. Con el modelado del medio se asegura que puedan conectarse varias capas DLL al medio (con fines de un desarrollo futuro).

5.2.3. Bloque DLL_LIN

La Figura 5.8 describe la arquitectura SDL del bloque DLL_LIN, con las restricciones descritas en el apartado 5.1.1, éste está compuesto por los bloques LLC y MAC, los cuales representan la subcapa LLC y la subcapa MAC, respectivamente.

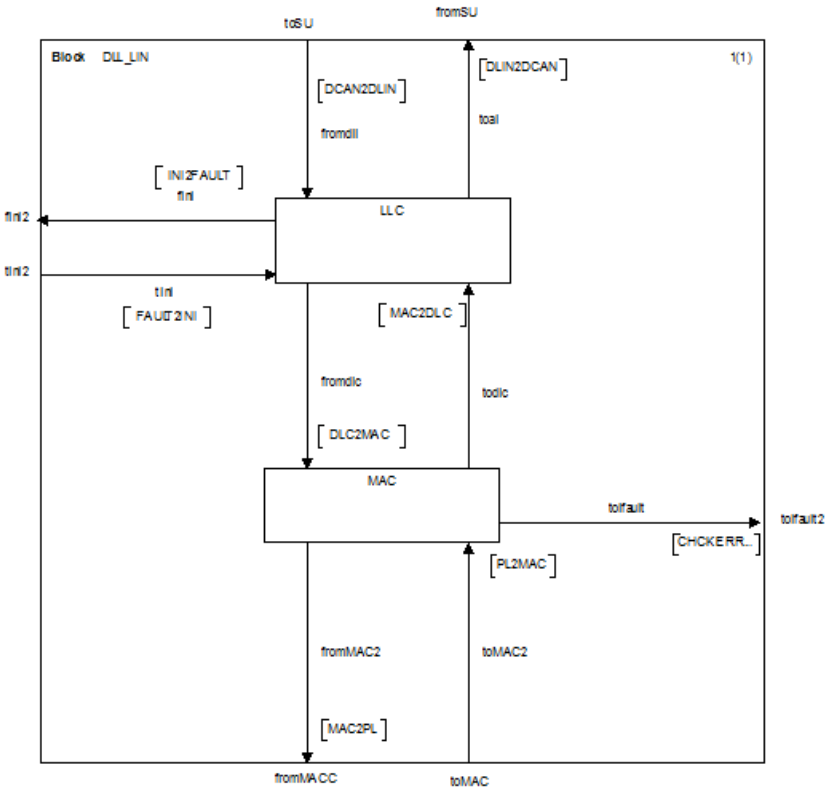

Figura 5.8. Diagrama SDL del bloque DLL_LIN.

Dado que LIN es un protocolo basado en la topología Maestro/Esclavo, el bloque DLL_CAN envía la función a realizar por medio de la señal DCAN2DLIN al bloque DLL_LIN. Esta señal es equivalente a la señal que envía la capa de aplicación, pero con la información proveniente de un nodo CAN, este bloque también tiene comunicación con la capa física a través de las señales MAC2PL y PL2MAC.

5.2.3.1. Bloque LLC

La Figura 5.9 describe la arquitectura SDL del bloque LLC, éste está compuesto por los procesos Initiate y Acceptance, los cuales representan las máquinas de estado de inicialización del bus LIN y filtro de datos provenientes del bloque MAC.

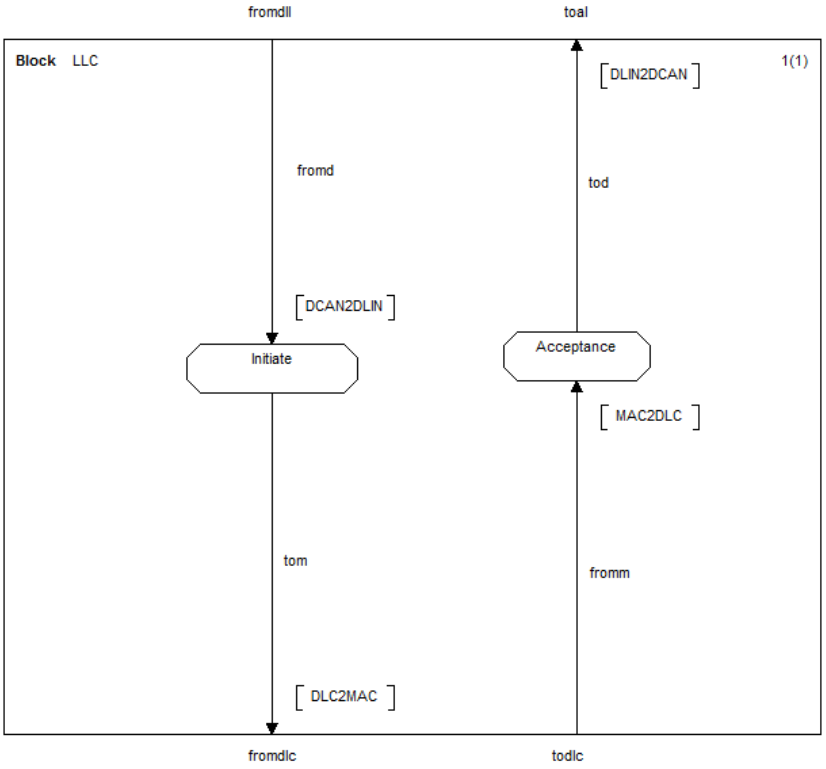

Figura 5.9. Diagrama SDL del bloque LLC.

5.2.3.2. Bloque MAC

La Figura 5.10 describe la arquitectura SDL del bloque MAC, este bloque está compuesto por los procesos ensaMAC y desenMAC, que se encargan de enviar y recibir las tramas de datos, respectivamente, además este bloque interactúa directamente con la capa física por medio de las señales MAC2PL y PL2MAC. Se comunica por medio de las señales DLC2MAC y MAC2DLC con el bloque LLC.

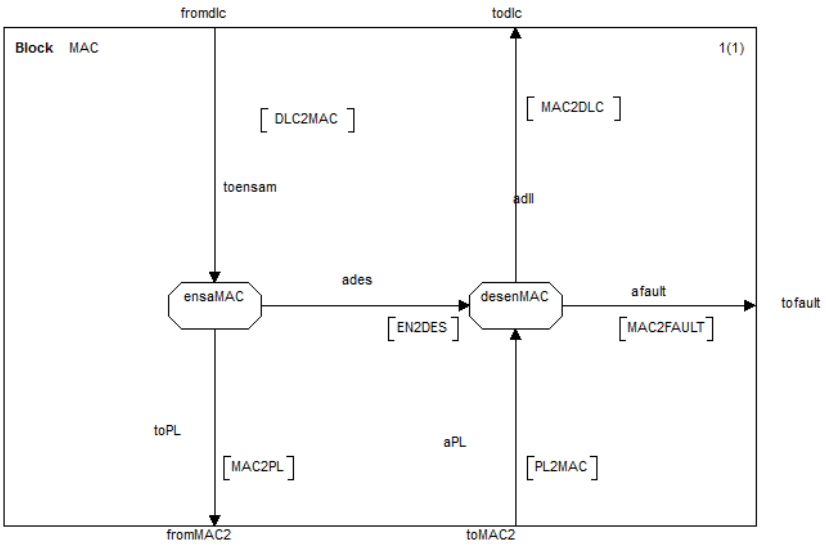

Figura 5.10. Diagrama SDL del bloque MAC.

5.2.4. Bloque DLL_CAN

La Figura 5.11 describe la arquitectura SDL del bloque DLL_CAN, con las restricciones descritas en el apartado 5.1.4, éste está compuesto por los bloques LLC_CAN y MAC_CAN, los cuales representan a la subcapa LLC y la subcapa MAC del ISO-11898, respectivamente.

Dado que CAN es un protocolo basado en una topología Multimaestro, el bloque DLL_LIN envía la función a realizar por medio de la señal DLIN2DCAN al bloque DLL_CAN con las características de la señal que envía la capa de aplicación. Este bloque tiene comunicación con la capa física a través de las señales CMAC2PL y CPL2MAC.

5.2.4.1. Bloque LLC_CAN

La Figura 5.12 describe la arquitectura SDL del bloque LLC, éste está compuesto por los procesos Inicializa, Encap_F y Desen_F, los cuales representan las máquinas de estado de inicialización del bus, empaquetamiento y desempaquetamiento de datos, respectivamente.

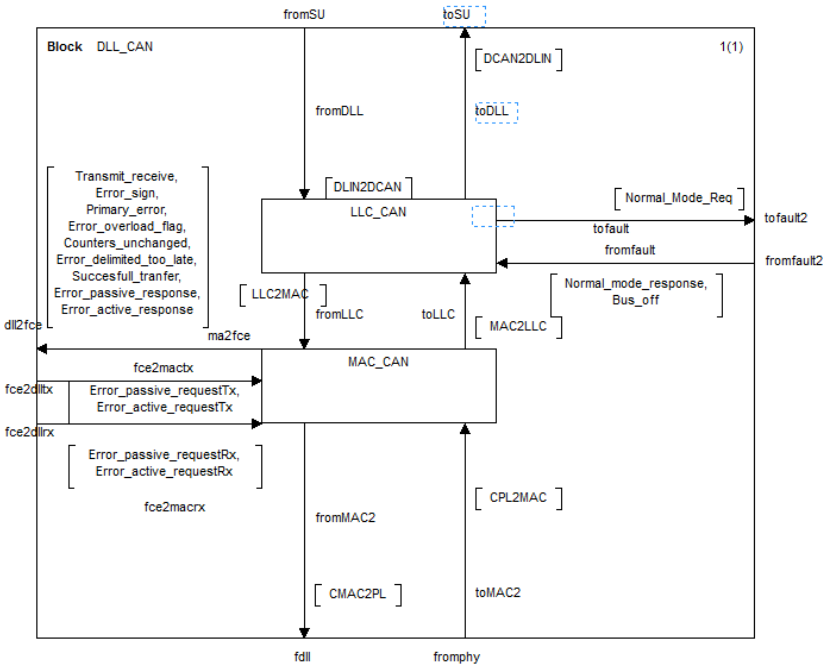

Figura 5.11. Diagrama SDL del bloque DLL_CAN.

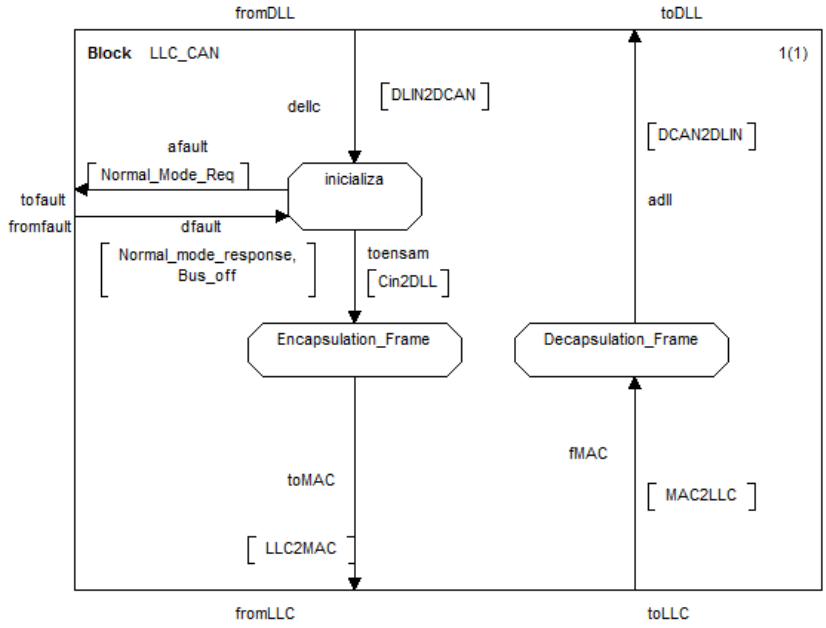

Figura 5.12. Diagrama SDL del bloque LLC.

5.2.4.2. Bloque MAC_CAN

La Figura 5.13 muestra la arquitectura SDL del bloque MAC, éste bloque está compuesto por los bloques Tx_Data_Enc, Tx_Media_Access_Management, Receive_Data_Decapsulation, Receiver_Media_Access_Management, que se encargan de

enviar y recibir las tramas de datos. Este bloque interactúa directamente con la capa física por medio de las señales CMAC2PL y CPL2MAC, y con las señales LLC2MAC y MAC2LLC con el bloque LLC.

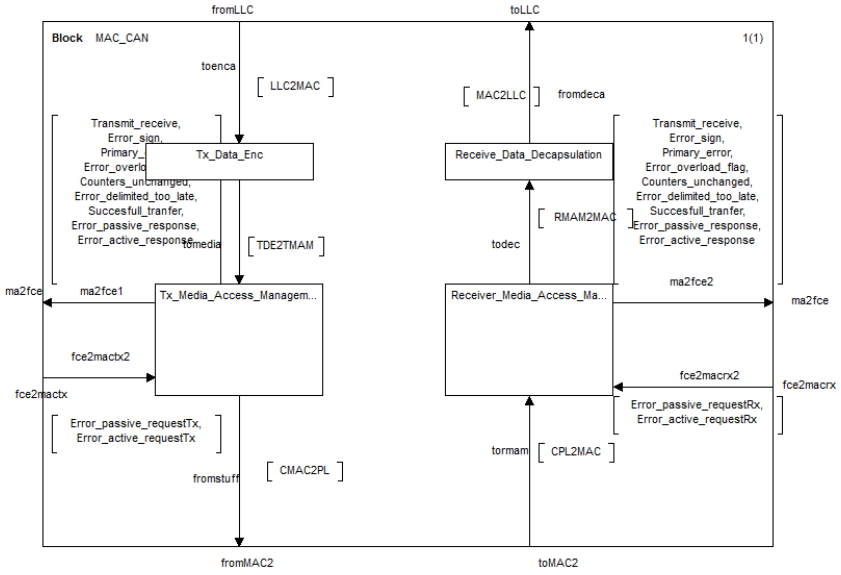

Figura 5.13. Diagrama SDL del bloque MAC.

5.2.4.2.1. Bloque Tx_Data_Enc

La Figura 5.14 describe la arquitectura SDL del bloque Tx_Data_Enc, éste está compuesto por el proceso Transmit_Data_Encapsulation, que representa la máquina de estados que genera el CRC y encapsula la trama de datos CAN.

5.2.4.2.2. Bloque Tx_Media_Access_Management

La Figura 5.15 describe la arquitectura SDL del bloque Tx_Media_Access_Management, éste está compuesto por el proceso Stuffing_calc, el cual representa la máquina de estados que prepara la trama CAN para poder transmitirse por el medio de forma serial y realiza una inserción de bit como se indica en el inciso 4.1.1.1. Este bloque se comunica con la capa física por medio de la señal CMAC2PL.

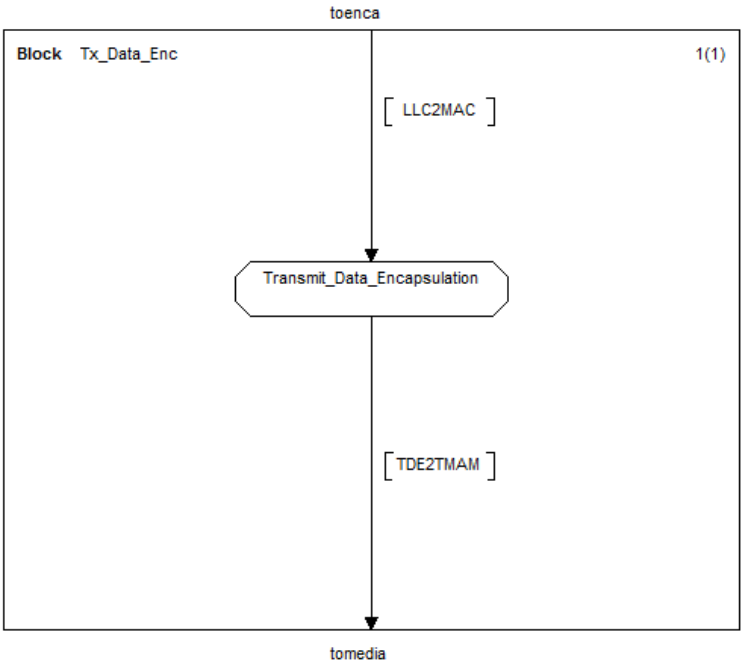

Figura 5.14. Diagrama SDL del bloque Tx_Data_Enc.

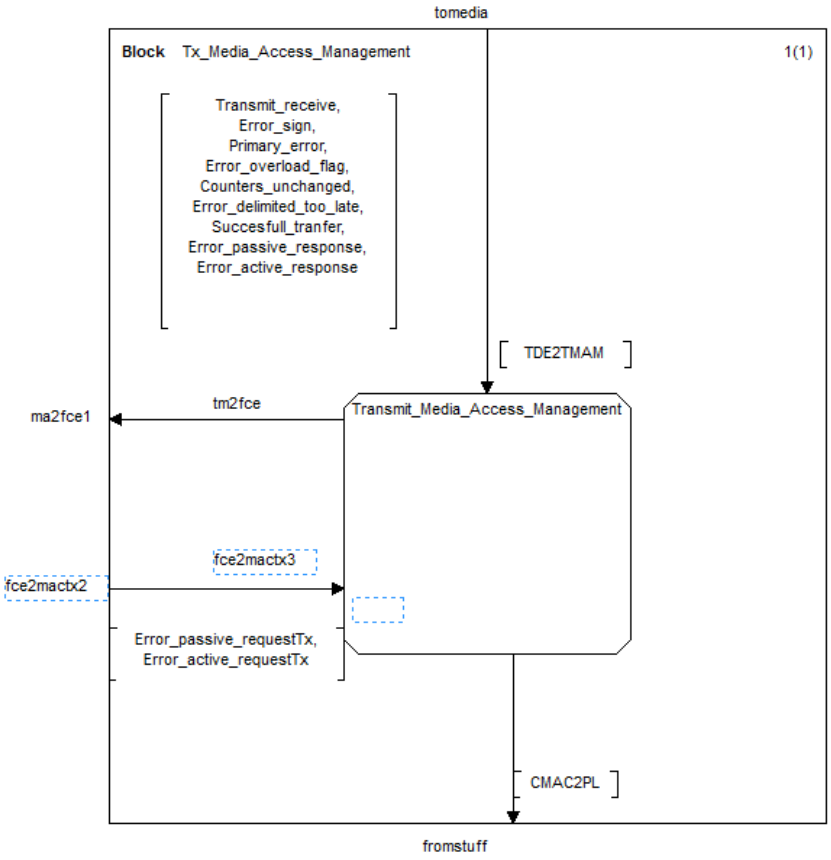

Figura 5.15. Diagrama SDL del bloque Tx_Media_Access_Management.

5.2.4.2.3. Bloque Receiver_Media_Access_Management

La Figura 5.16 describe la arquitectura SDL del bloque Receiver_Media_Access_Management, éste está compuesto por el proceso Destuffing_calc, el cual representa la máquina de estados que recibe la trama CAN de forma serial proveniente de la capa física, elimina el bit de inserción y verifica posibles errores como se indica en el inciso 4.3.7.1, además prepara los datos al bloque superior.

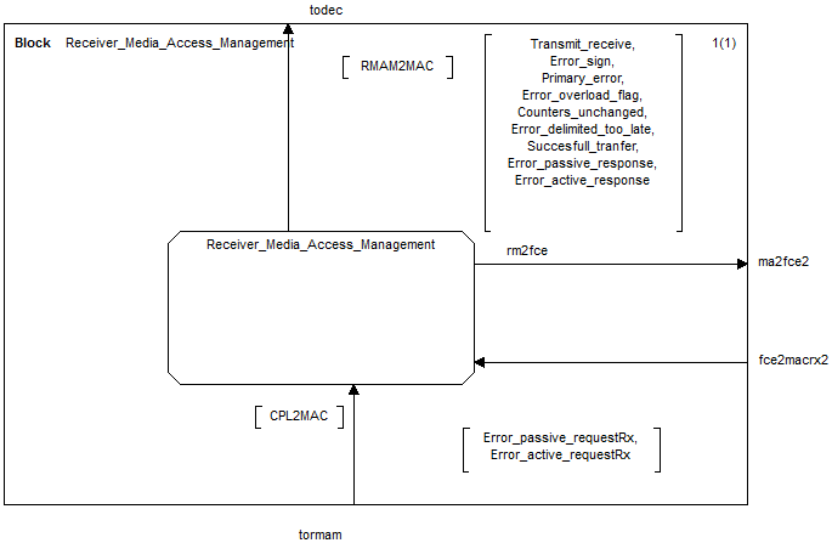

Figura 5.16. Diagrama SDL del bloque Receiver_Media_Access_Management.

5.2.4.2.4. Bloque Receive_Data_Decapsulation

La Figura 5.17 describe la arquitectura SDL del bloque Rx_Data_Decap, éste está compuesto por el proceso Receive_Data_Decapsulation, el cual representa la máquina de estados que genera el CRC con los datos provenientes del bloque inferior y lo compara con el CRC enviado, como se indica en el inciso 4.3.7.1; finalmente desencapsula la trama CAN.

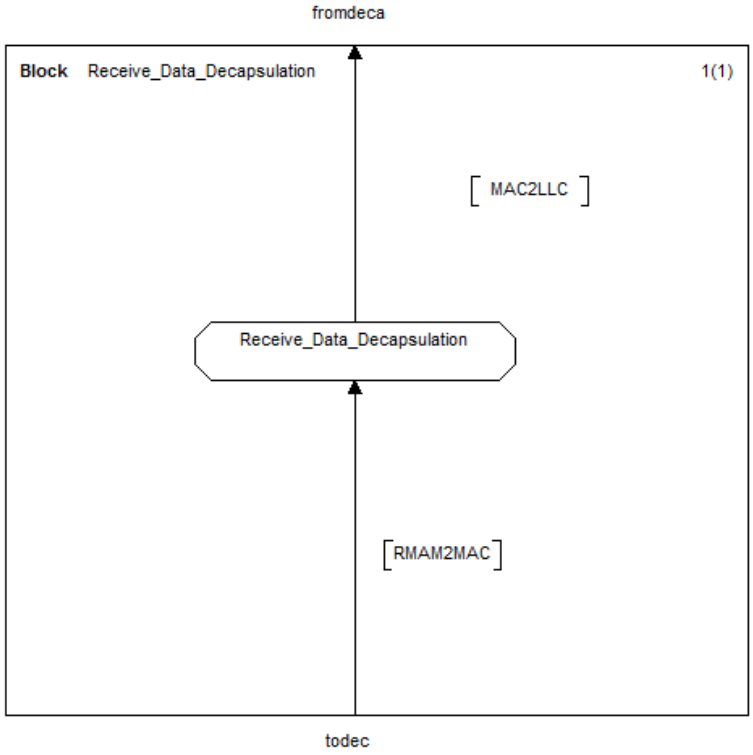

Figura 5.17. Diagrama SDL del bloque Receive_Data_Decapsulation.

6. Especificación Dinámica de la Pasarela CAN/LIN

La especificación dinámica describe el comportamiento del sistema y se encuentra dentro de los procesos que conforman la especificación. Dado que la funcionalidad de los procesos ya se ha definido en la especificación estática, a continuación, se detalla el funcionamiento interno basado en máquinas de estado.

Debido a que la especificación de los procesos es muy extensa, sobre todo en los procesos Tx_Data_Encapsulation, Receive_Data_Decapsulation, ensaMAC y desenMAC, la descripción de cada proceso se realiza de la siguiente forma:

- Se usará la primera página de la especificación formal de los procesos: Esto es debido a que la primera página de cada proceso incluye las definiciones de las señales y la declaración de variables.
- Los procesos se describen con diagramas de visión de estados: Son diagramas de comportamiento en SDL en los que se eliminan todos los símbolos, excepto los que corresponden a los estados y a las señales de entrada.

Para una revisión exhaustiva de la especificación formal de la pasarela CAN/LIN, se entregan los siguientes documentos conjuntamente con este trabajo de tesis:

- Especificación formal en formato .cbf: Formato de la especificación generada por la herramienta Cinderella SDL.

- Especificación formal en formato .pdf: Para que los lectores que no disponen de la herramienta Cinderella SDL, puedan visualizar la especificación formal.
- Diagrama MSC en formato .pdf: Los diagramas MSCs (*Message Sequence Chart*) representan el resultado de la simulación como una secuencia de mensajes ordenada en el tiempo, y que se abordan en el Capítulo 7 para documentar la evaluación de resultados.

6.1. Proceso inicializa

Una vez que el proceso inicializa (véase Figura 6.1) ejecuta el símbolo de inicio, se recibe la información proveniente del bloque DLL_LIN por medio de la señal DLIN2DCAN, cuando esto sucede se procede a enviar la solicitud de configuración inicial hacia el bloque Supervisor mediante la señal Normal_Mode_Request, la respuesta a la solicitud de configuración es recibida por medio de la señal Bus_off; en caso de que el bus se encuentre disponible se envía al proceso Encapsulation_Frame la información almacenada por medio de la señal Cin2DLL (con fines de funcionalidad). En caso contrario espera hasta que el bus se encuentre libre para que la ECU pueda transmitir.

6.2. Proceso Encapsulation_Frame

El proceso Encapsulation_Frame forma las unidades de datos de acuerdo a la estructura y formato de la trama LLC (véase Figura 6.2); al ejecutar el símbolo de inicio, se lee la señal Cin2DLL la cual tiene los parámetros IDE_AL, Iden_AL, DLC_AL, Data_AL en formato entero (*integer*), y se convierten a cadena de bits (*bitstring*); a partir de este proceso los datos se envían en formato binario, a excepción de que se indique lo contrario.

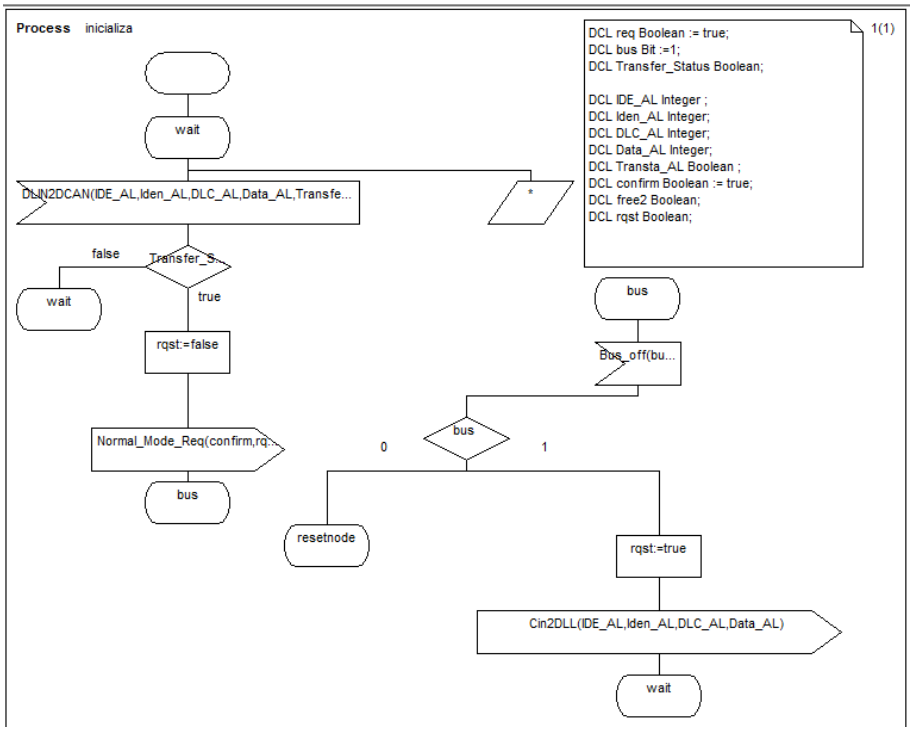

Figura 6.1. Descripción del proceso inicializa.

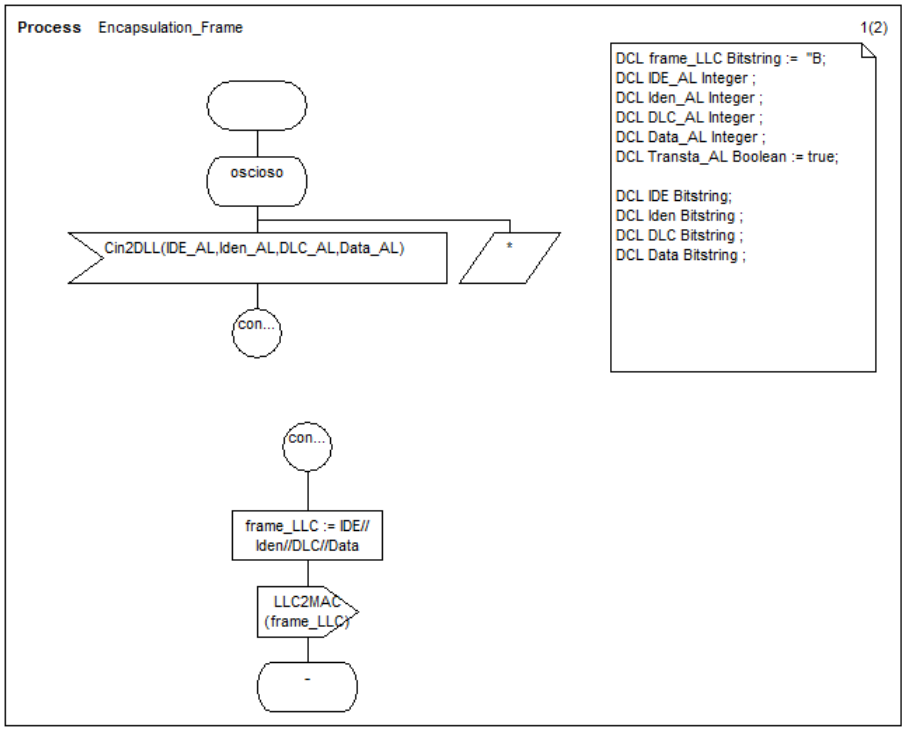

Figura 6.2. Descripción del proceso Encapsulation_Frame.

6.3. Proceso Transmit_Data_Encapsulation

El proceso Transmit_Data_Encapsulation realiza las siguientes funciones (véase Figura 6.3):

- Aceptación de las tramas LLC y la información de la interfaz de control.
- Cálculo de la secuencia CRC. El cálculo del CRC se realiza de manera paralela.
- Construcción de la trama MAC agregando el bit RTR, el bit r0, el bit ACK, el bit delimitador ACK y el CRC que previamente fue calculado por el procedimiento CRC15.

La trama de datos CAN se transmite al siguiente proceso por medio de la señal TDE2TMAM.

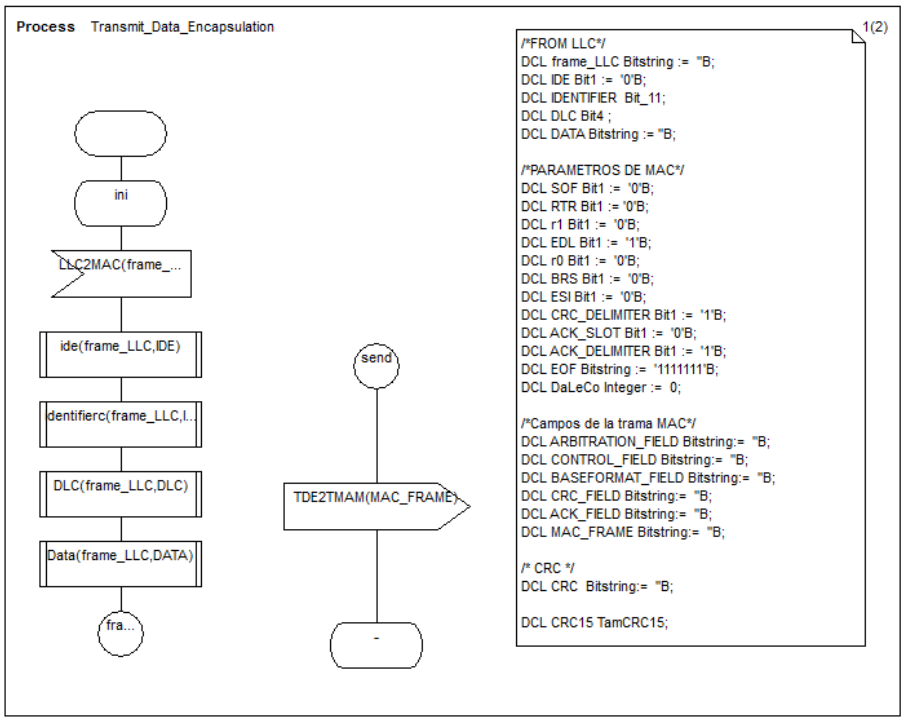

Figura 6.3. Descripción del proceso Transmit_Data_Encapsulation.

6.4. Proceso Transmit_Media_Access_Management

El proceso Tansmit_Media_Access_Management realiza las siguientes funciones (véase Figura 6.4):

- Inicialización de la transmisión después de reconocer que el bus está libre.
- Serialización de la trama MAC.
- Inserción de bits de relleno.
- Arbitraje, y en caso de que se pierda el arbitraje, pasar a modo receptor.
- Detección de errores de formato, monitoreo de bits, verificación del procedimiento de inserción de bit.
- Revisión del bit de reconocimiento, para determinar que la trama fue enviada con éxito.
- Reconocimiento de una condición de sobrecarga.
    1. Construcción de una trama de sobrecarga e inicialización de la transmisión.

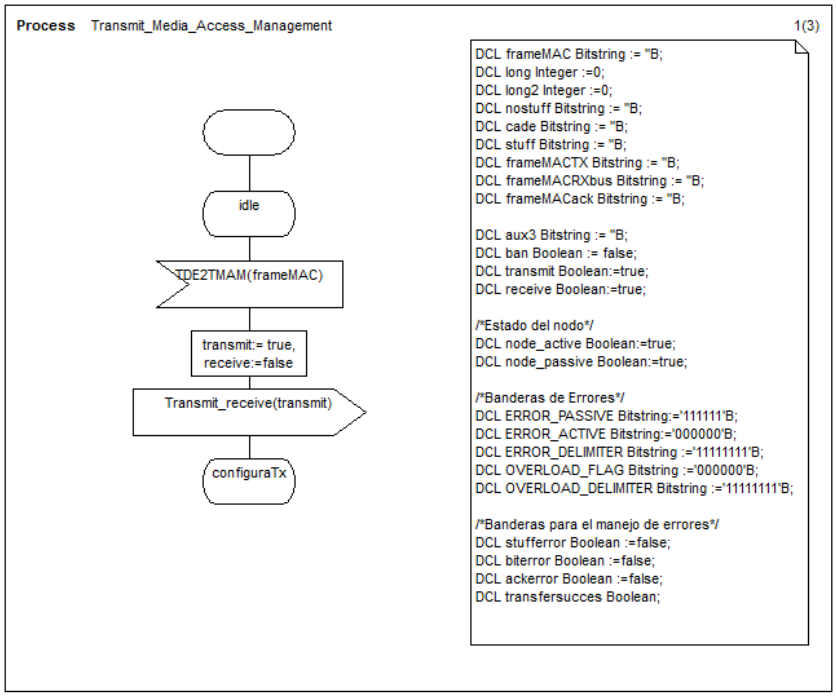

Figura 6.4. Descripción del proceso Transmit_Media_Access_Management.

6.5. Proceso medCAN

Una vez que el proceso medCAN (véase Figura 6.5) ejecuta el símbolo de inicio se pasa al estado Recibe, el cual está a la espera de recibir la trama CAN por parte

del proceso superior mediante las señales CMAC2PL2, cuando esto ocurre se habilita un temporizador y se realiza la transición al estado correspondiente a la señal recibida espera1; al estar en cualquier estado se espera el envío de la señal por parte del temporizador habilitado (TBus) y se envían los datos recibidos para regresar al estado recibe. La trama de datos CAN tiene la forma que se describe en el subcapítulo 4.3 y se transmitirá por el medio físico.

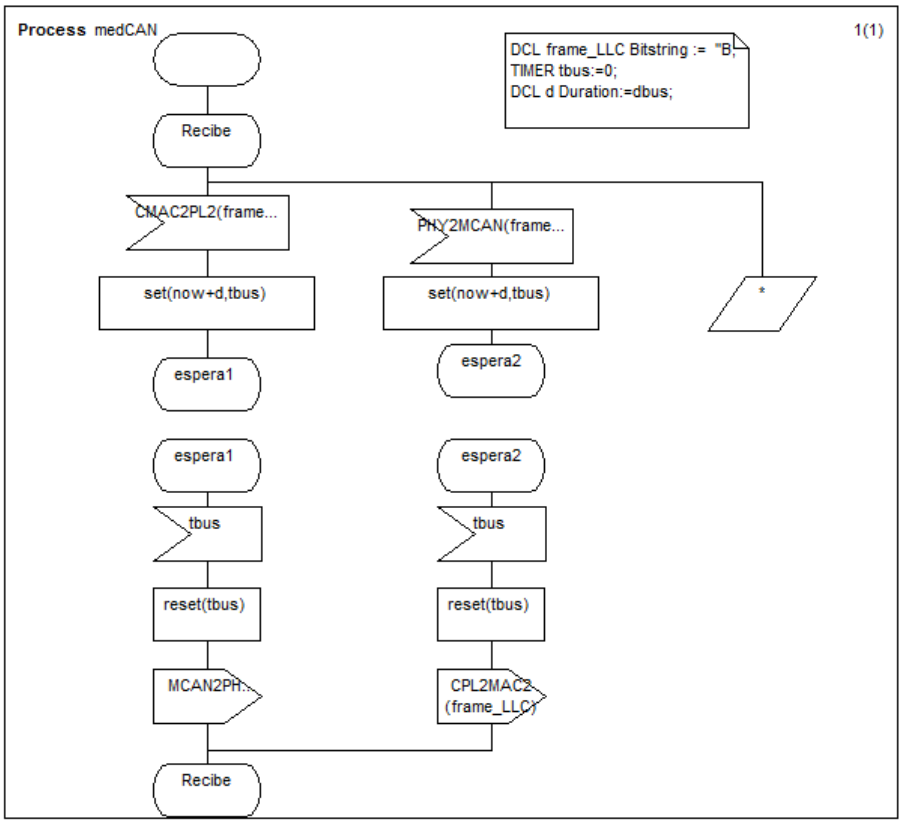

Figura 6.5. Descripción del proceso medCAN.

6.6. Proceso Receiver_Media_Access_Management

El proceso Receiver_Media_Access_Management realiza las siguientes funciones (véase Figura 6.6):

- Recepción del flujo de bits provenientes de la capa física.
- Deserialización y ensamblado de la estructura de la trama.
- Eliminación de los bits de relleno.

- Detección de errores de CRC, formato y violaciones a la regla de inerción de bits.
- Transmisión del bit de aceptación.
- Construcción de la trama de error e inicialización de la transmisión.
- Reconocimiento de una condición de sobrecarga.
- Reactivación y construcción de una trama de sobrecarga e inicialización de la transmisión.
- Distinción de tramas entre el formato estándar y el formato extendido.

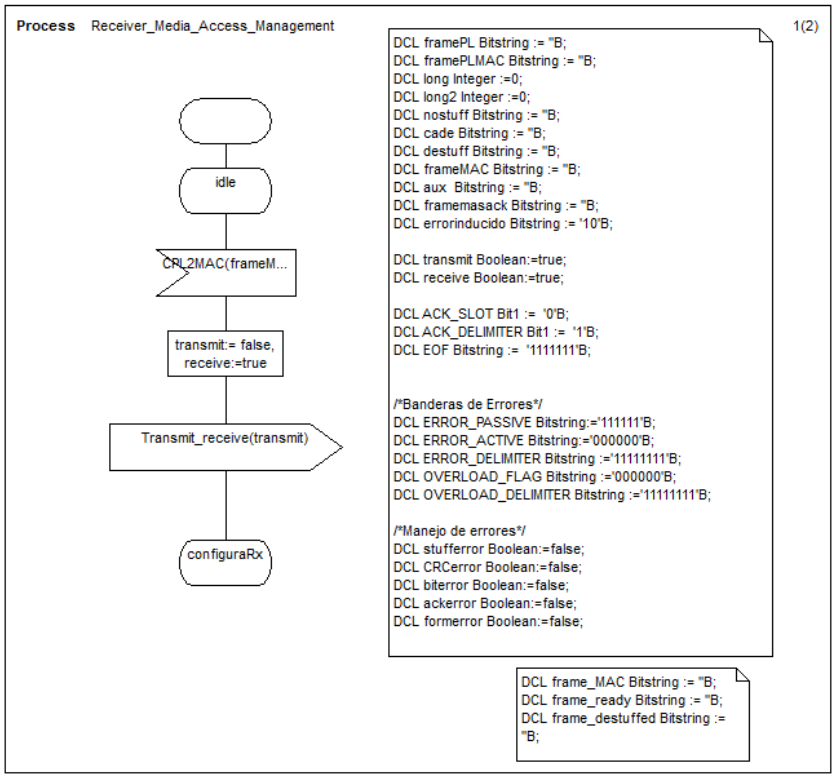

Figura 6.6. Descripción del proceso Receiver_Media_Access_Management.

6.7. Proceso Receive_Data_Decapsulation

El proceso Receive_Data_Decapsulation realiza las siguientes funciones (véase Figura 6.7):

- Eliminación de la información específica de la trama MAC.
- Presentación de la trama LLC y de información de la interfaz de control a la subcapa LLC.

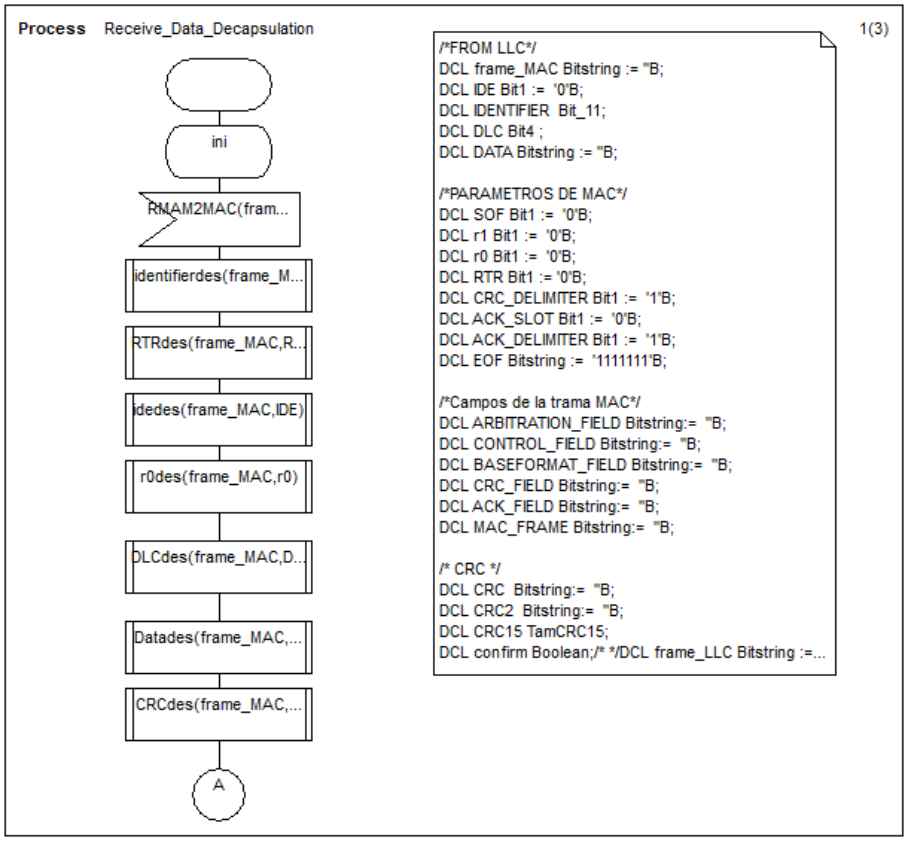

Figura 6.7. Descripción del proceso Receive_Data_Decapsulation.

6.8. Proceso Decapsulation_Frame

La Figura 6.8 muestra la descripción del proceso Decapsulation_Frame; una vez ejecutado el símbolo de inicio se pasa al estado ocioso, el cual está a la espera de recibir la información por parte del bloque MAC, una vez recibido el dato se procede a desensamblar la trama; ya separados los datos se realiza la conversión de tipos de datos, de binarios a enteros y se presentan a su correspondiente capa DLL_LIN por medio de la señal parametrizada DCAN2DLIN realizando el intercambio de datos del protocolo CAN al protocolo LIN.

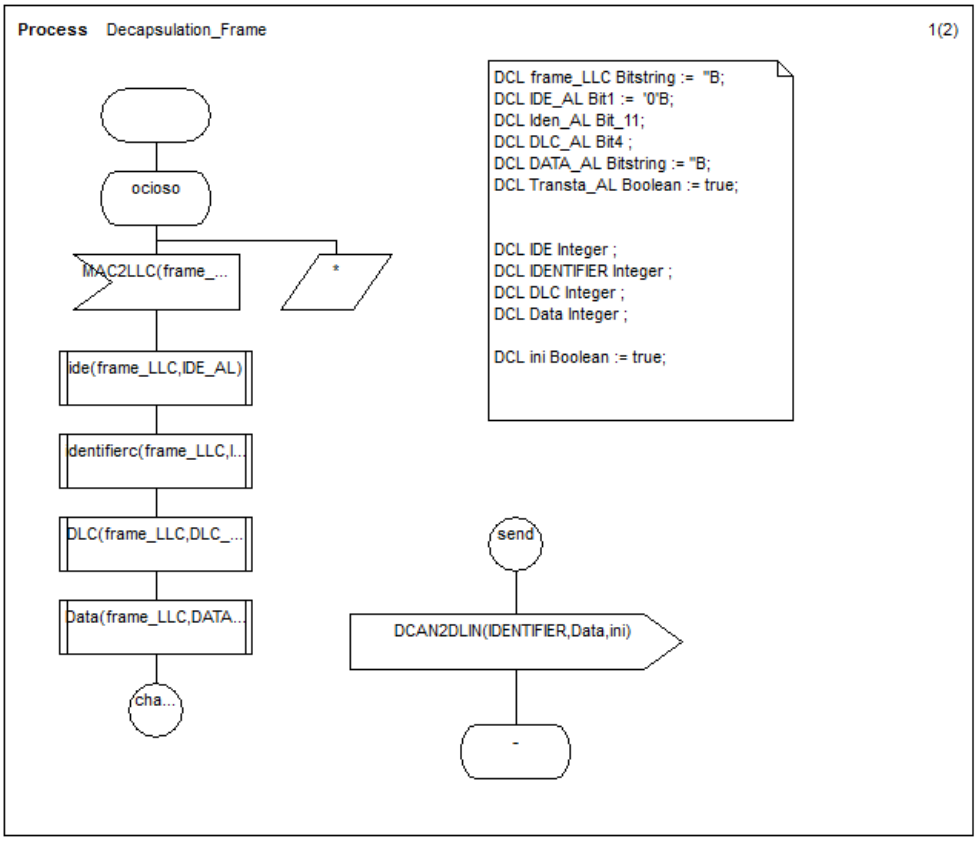

Figura 6.8. Descripción del proceso Desen_F.

6.9. Proceso initiate

Una vez que el proceso initiate (véase Figura 6.9) ejecuta el símbolo de inicio, se recibe la información proveniente del bloque DLL_CAN por medio de la señal DCAN2DLIN, cuando esto sucede se procede a enviar la solicitud de configuración inicial hacia el bloque Supervisor mediante la señal INI2FAULT, se recibe la respuesta a la solicitud de configuración por medio de la señal FAULT2INI; en caso de que el bus LIN se encuentre disponible se envía la información almacenada al proceso ensaMAC por medio de la señal DLC2MAC, en caso contrario espera hasta que el bus se encuentre libre para que la ECU pueda transmitir.

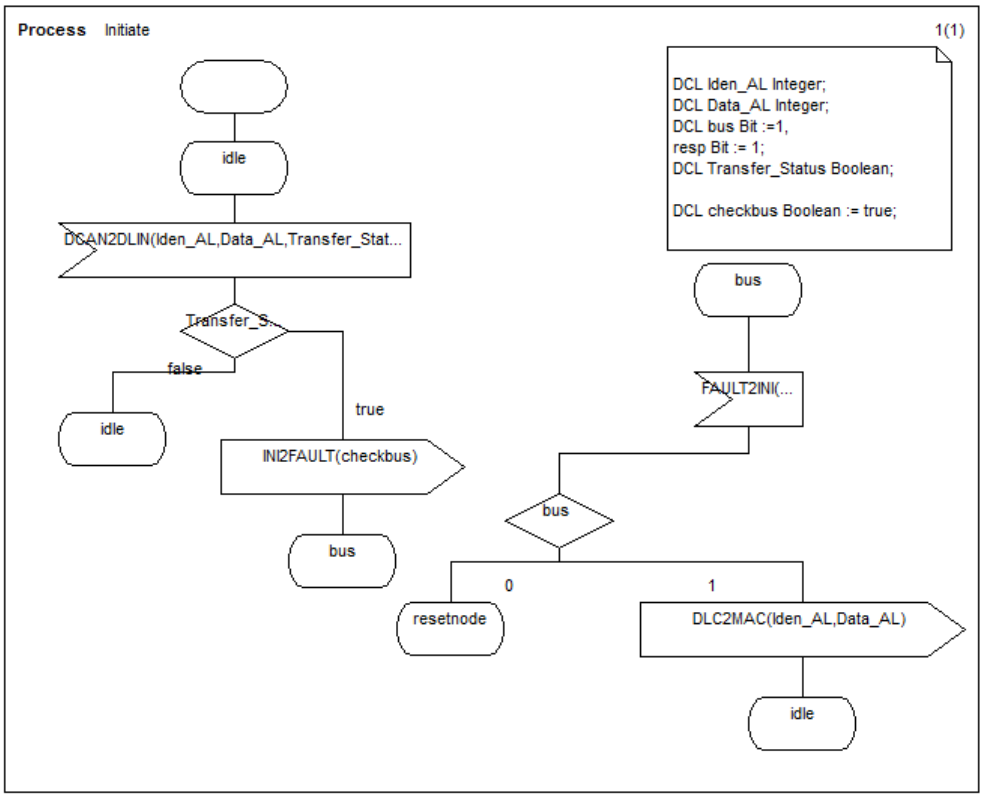

Figura 6.9. Descripción del proceso Initiate.

6.10. Proceso ensaMAC

El proceso ensaMAC realiza las siguientes funciones (véase Figura 6.10):

- Aceptacion de la trama LLC.
- Cálculo del *checksum*, éste se realiza en forma paralela.
- Construcción de la trama MAC agregando el *breakfield*, el byte *sync* y el *checksum* si se requiere.
- Inicialización de la transmisión después de reconocer que el bus está libre.
- Serialización de la trama MAC.

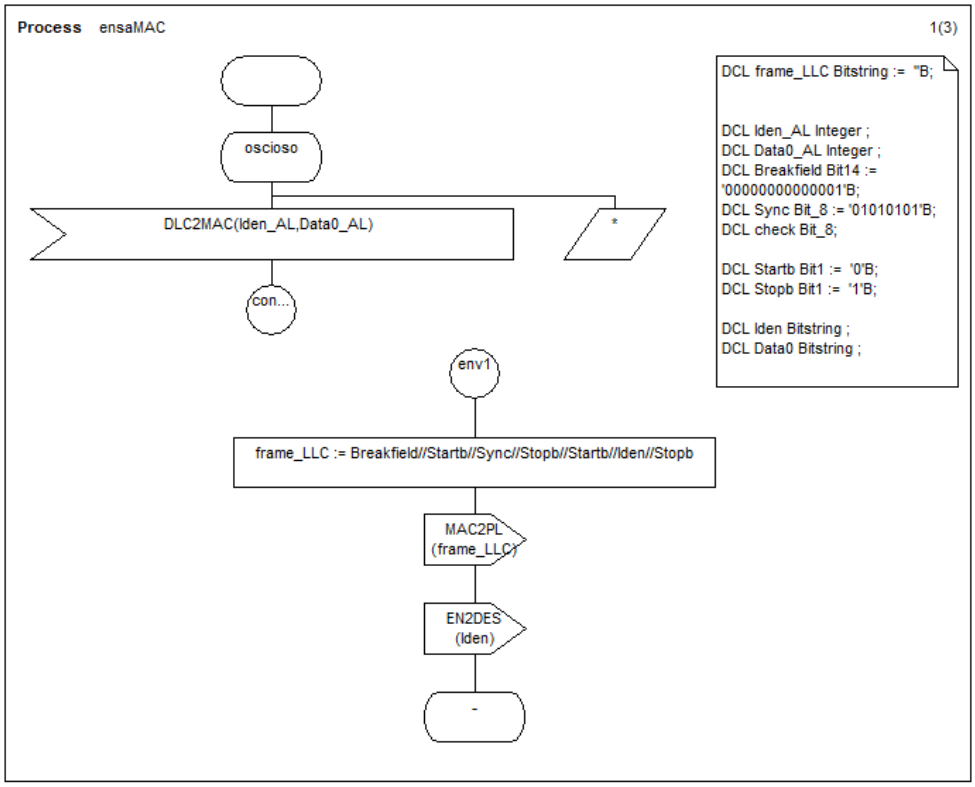

Figura 6.10. Descripción del proceso ensaMAC.

6.11. Proceso medLIN

Una vez que el proceso med_LIN (véase Figura 6.11) ejecuta el símbolo de inicio se pasa al estado Recibe, el cual está a la espera de recibir la trama LIN por parte del proceso superior mediante de la señal PHY2MLIN, cuando esto ocurre se habilita un temporizador y se realiza la transición al estado correspondiente a la señal recibida espera1; al estar en este estado se espera el envío de la señal por parte del temporizador habilitado (TBus) y se envían los datos recibidos para regresar al estado recibe. La trama de datos ha sido transformada del protocolo CAN al protocolo LIN y tiene la forma descrita en el capítulo 3. Con fines ilustrativos se añadió otra capa DLL_LIN para poder procesar la trama LIN y entregarla al usuario final en otra ECU.

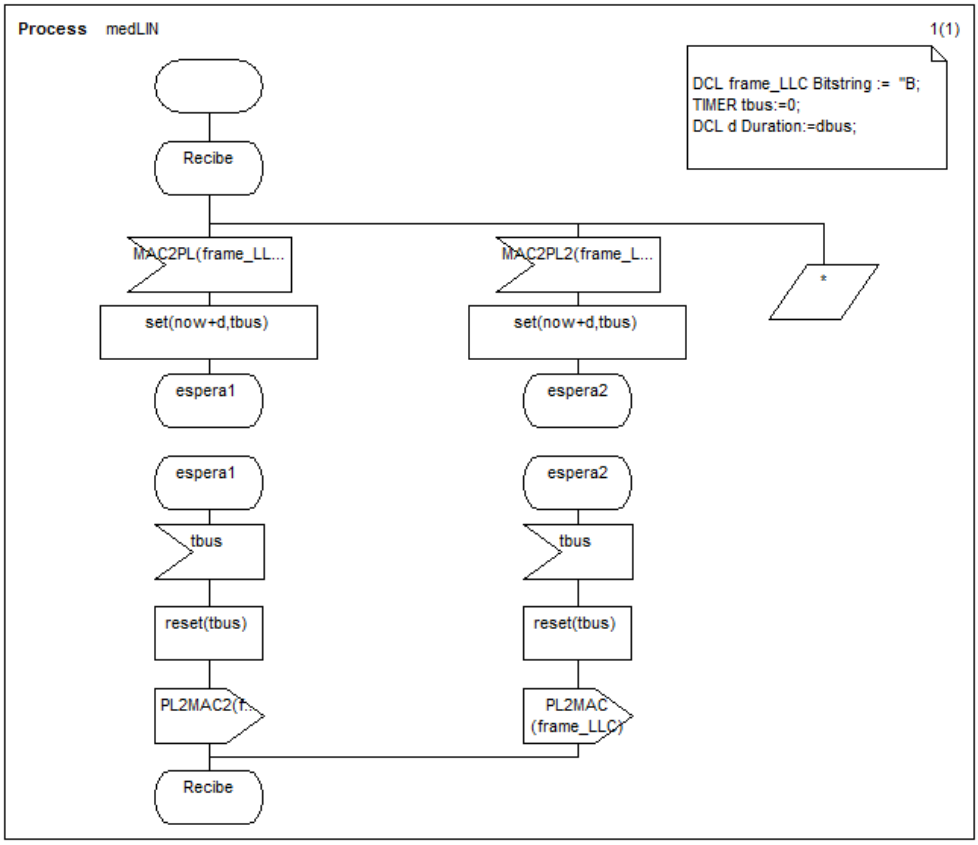

Figura 6.11. Descripción del proceso MedLIN.

6.12. Proceso desenMAC

El proceso desenMAC realiza las siguientes funciones (véase Figura 6.12):

- Recepción del flujo de bits provenientes de la capa física.
- Deserialización y ensamblado de la estructura de la trama.
- Detección de error de Checksum.
- Eliminación de información específica de la trama MAC.
- Presentación de la trama LLC a la subcapa LLC.

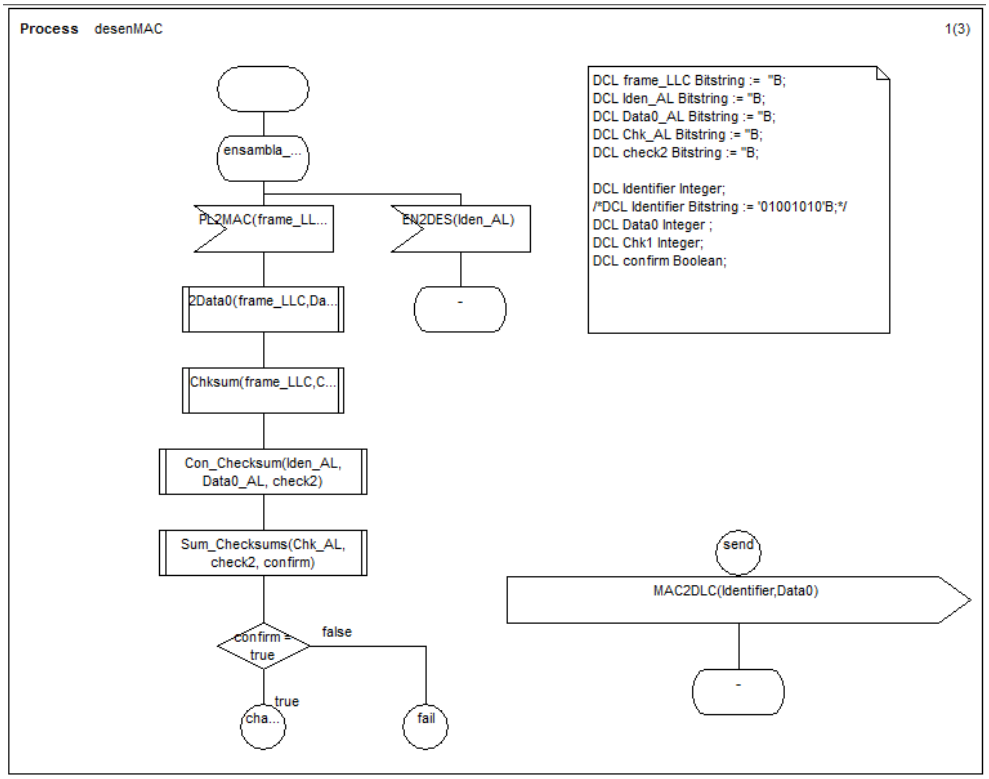

Figura 6.12. Descripción del proceso desenMAC.

6.13. Proceso Acceptance

Una vez que el proceso acceptance (véase Figura 6.13) ejecuta el símbolo inicio se pasa al estado resetnode, el cual está a la espera de recibir algún dato por parte del bloque MAC, una vez recibido el dato se procede a desensamblar la trama; ya separados los datos se realiza un filtro de mensajes con la ayuda del identificador, para que puedan ser entregados al usuario; el identificador debe coincidir con el del nodo, si son iguales se envían los datos al bloque DLL_CAN por medio de la señal DLIN2DCAN. En este punto de la especificación los datos provenientes del medio LIN se han extraido y están listos para llevar a cabo la conversión de un protocolo de comunicaciones a otro.

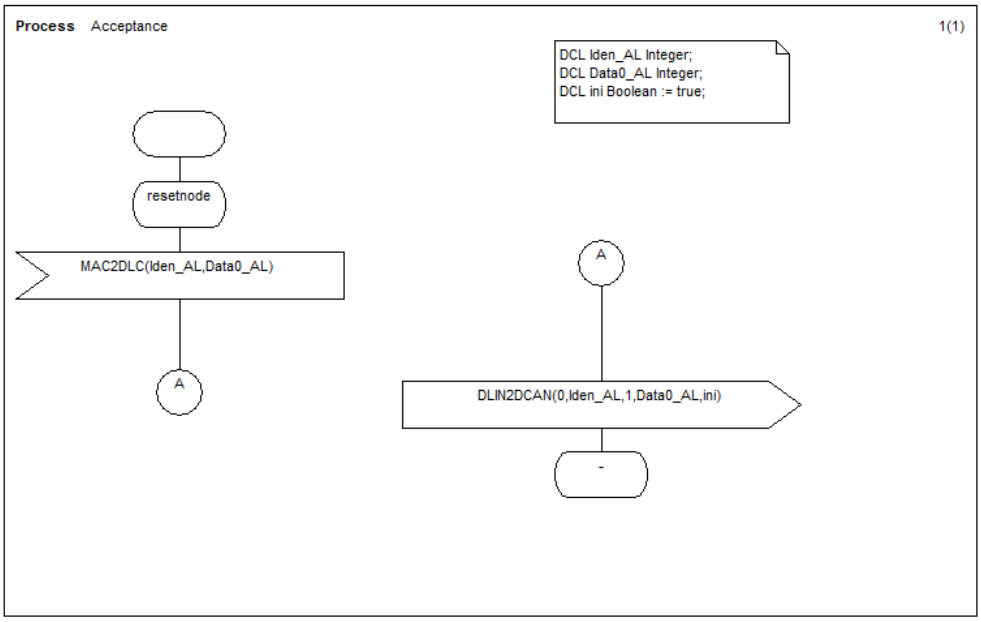

Figura 6.13. Descripción del proceso Acceptance.

6.14. Proceso FCE

La Figura 6.14 muestra la descripción del proceso FCE; una vez ejecutado el símbolo de inicio pasa al estado idle, el cual está a la espera de recibir alguna solicitud proveniente del bloque LLC_CAN mediante la señal Normal_mode_request que establece el proceso FCE a un estado inicial. Para atender la solicitud del bloque LLC_CAN, el proceso FCE da respuesta mediante la señal Normal_mode_response y por medio de la señal Bus_off indica el estado del bus.

Los mensajes intercambiados entre el proceso FCE y el bloque MAC_CAN se realizan mediante las siguientes señales:

- Transmit_receive: Indica a la ECU el modo actual de transferencia (transmisor o receptor).
- Error_sign: Esta señal indica que el bloque MAC_CAN detectó algún error durante la transmisión o la recepción.
- Primary error: Por medio de esta señal se indica que el bloque MAC_CAN detectó un bit dominante después de haber mandado una bandera de error.

- Error_overload_flag: Determina que la ECU está transmitiendo banderas de error o de sobrecarga.
- Counters_unchanged: Esta señal permite determinar que los contadores TEC y REC permanecen sin cambios.
- Error_delimited_too_late: Indica que el bloque MAC_CAN está esperando demasiado tiempo para un delimitador de error.
- Succesfull_transfer: Por medio de esta señal se indica que la transmisión o la recepcion se completaron éxitosamente.
- Error_passive_response: Esta señal indica que la ECU fue establecida en el estado de error pasivo.
- Error_active_response: Esta señal indica que la ECU fue establecida en el estado de error activo.
- Error_passive_request: Esta señal solicita que la ECU se establezca el estado de error pasivo.
- Error active request: Esta señal solicita que la ECU se establezca el estado de error activo.

Para conocer el estado del bus y poder dar respuesta a las solicitudes de los bloques LLC_CAN y MAC_CAN, el proceso FCE utiliza las siguientes señales:

- Bus_off_request: Por medio de esta señal se solicita el estado del bus a la capa física.
- Bus_off_release_request: Mediante esta señal se establece que la ECU tendrá un comportamiento de transmisor o de receptor.
- Bus_off_response: Por medio de esta señal la capa física responde al bloque FCE el estado del bus.
- Bus_off_release_response: Mediante esta señal la capa física confirma que la solicitud de la ECU para transmitir o recibir está establecida.

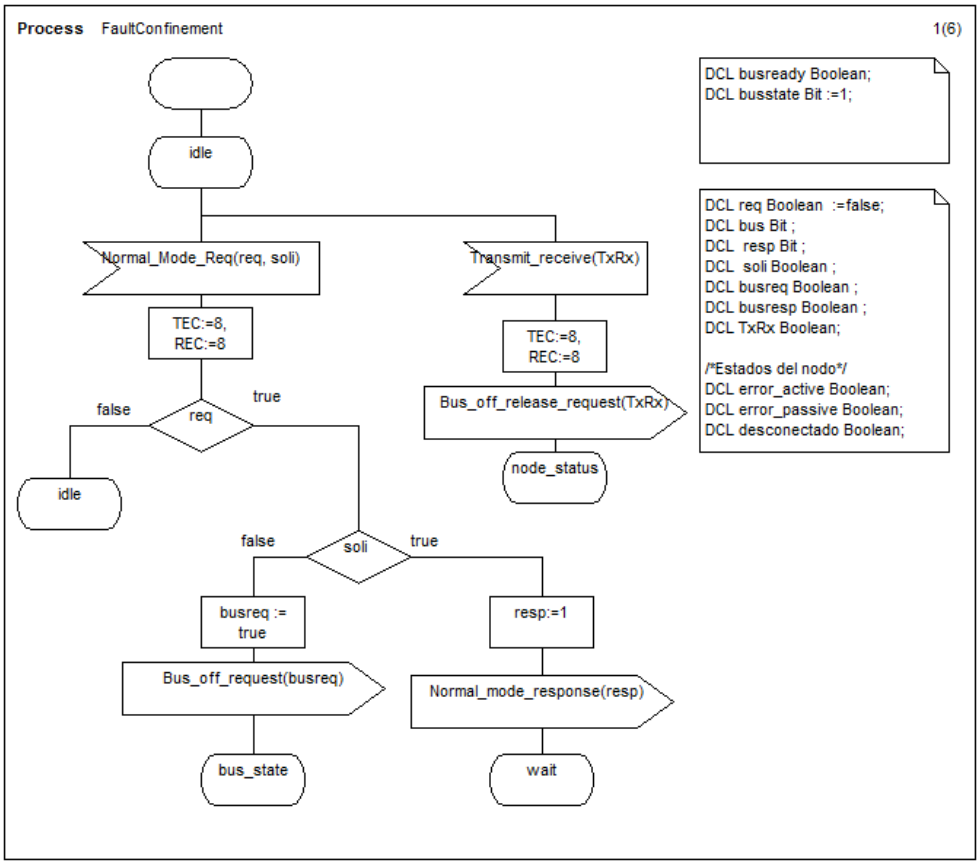

Figura 6.14. Descrición del proceso FCE.

6.15. Procedimientos

En este apartado se describen los procedimientos más importantes que los protocolos de la pasarela CAN/LIN realizan, entre los que destacan aquellos que son responsables de generar el CRC, generar el *checksum*, inserción del bit de relleno y eliminación del bit de relleno de la trama, así como verificar que no existan errores de bit, CRC, inserción de bit, de formato y verificación de aceptación.

6.15.1. Procedimiento Mod_2_15

El procedimiento Mod_2_15 realiza la función de desplazamiento, mediante el empleo de la operación XOR a nivel de bits (véase Figura 6.15). Este procedimiento recibe y devuelve una cadena de bits. Una vez que se ejecuta el inicio del procedimiento se calcula la longitud de la cadena recibida,

posteriormente se localizan el primer "1" dentro de la cadena recibida mediante una búsqueda incremental; cuando se localiza un "1" se realiza la operación XOR con la cadena actual y la variable KEY15 que contiene el valor del polinomio generador del CRC (apartado 4.3.1). Al finalizar este procedimiento se regresa la cadena modulo.

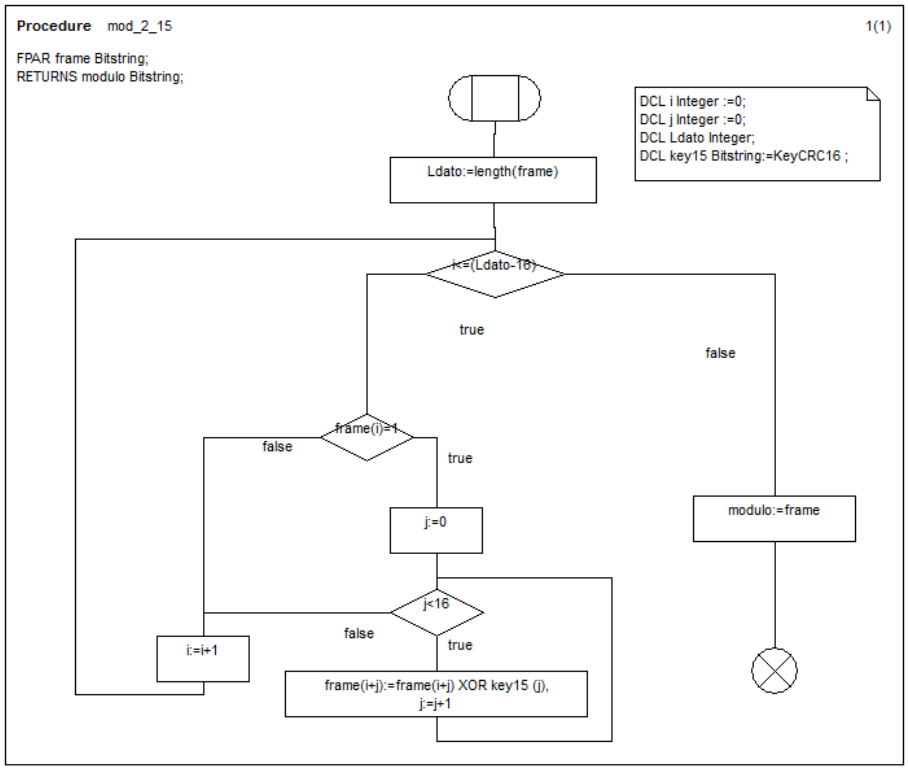

Figura 6.15. Descripción del procedimiento Mod_2_15.

6.15.2. Procedimiento CRC_15

Una vez descrito el procedimiento Mod_2_15 empleado en la realización del procedimiento CRC_15, se describe el procedimiento para generar la secuencia de CRC (véase Figura 6.16).

El procedimiento CRC_15 recibe y regresa una cadena de datos de tipo *bitstring*. Después de ejecutar el símbolo de inicio del procedimiento, se concatena a la variable DatosI una variable de tipo TamCRC15 (CRCaux) y se determina la longitud de la cadena DatosI, posteriormente se envía la cadena

DatosI al procedimiento mod_2_15 y se extraen los últimos 15 bits de la cadena mediante la operación substring.

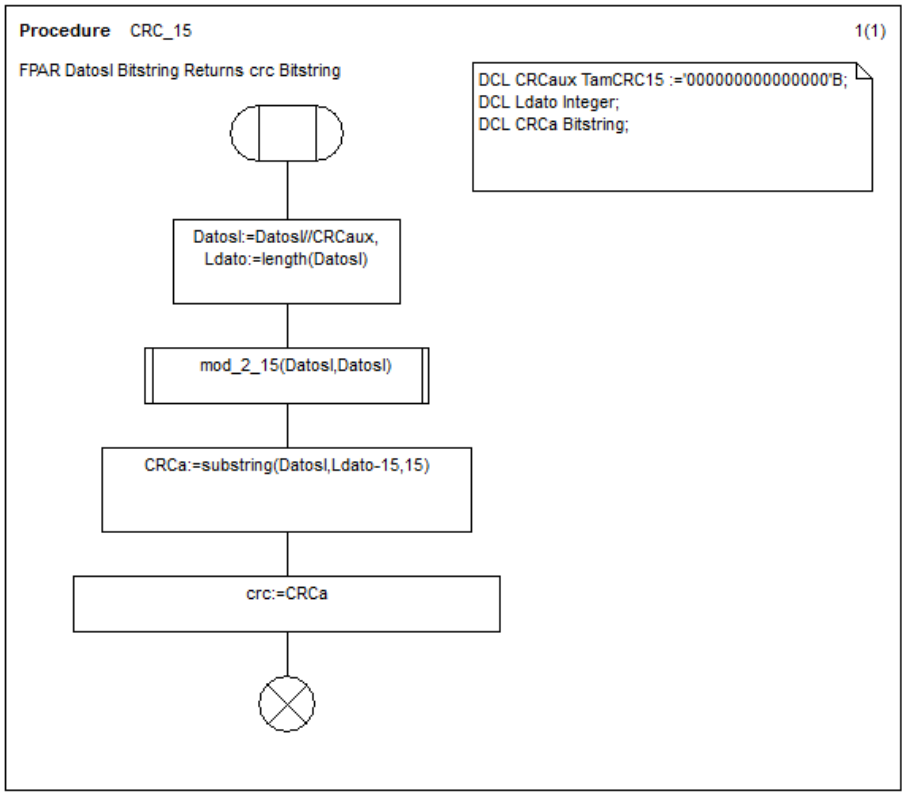

Figura 6.16. Descripción del procedimiento CRC_15.

6.15.3. Procedimiento Gen_Checksum

El procedimiento Gen_Checksum realiza la suma binaria del identificador con los datos a transmitir del protocolo LIN como se discutió en el apartado 3.3.2.1. Después de ejecutar el símbolo de inicio del procedimiento, se suma bit a bit la información en memoria, si resulta un acarreo se suma el bit de acarreo a la variable original y así sucesivamente hasta terminar con el último bit de suma. Al finalizar este procedimiento se regresa la cadena resultado.

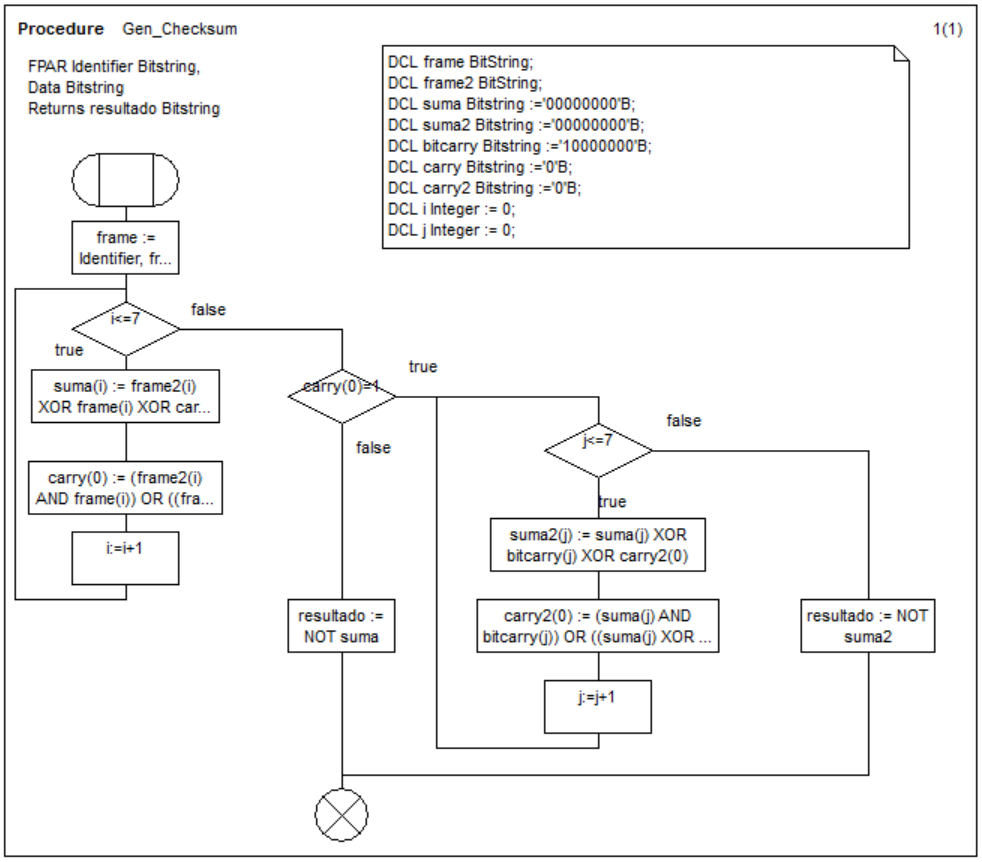

Figura 6.17. Descripción del procedimiento Gen_Checksum.

6.15.4. Procedimiento prepara_cadena

El procedimiento prepara_cadena realiza la función de crear un arreglo fijo de bits para la inserción de bits (véase Figura 6.18). Primero se analiza la longitud de la cadena de bits a la cual se le aplicará la inserción de bit. El tamaño máximo de una cadena después de aplicar el procedimiento de inserción de bit está dado por la ecuación 6.1:

$$TaMax = long + \frac{long}{5} + 1 \qquad \text{Ecuación 6.1}$$

donde:

$TaMax$ es el valor máximo de la cadena después de la inserción de bit.

$long$ es el tamaño de la cadena antes de aplicar la inserción de bit.

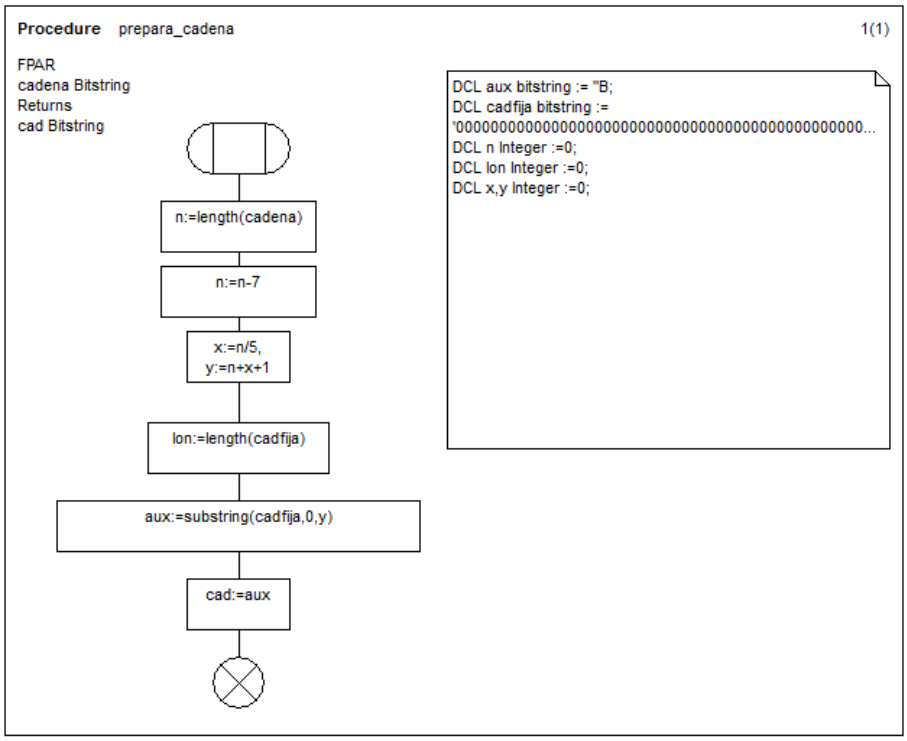

Figura 6.18. Descripcion del procedimieno prepara_cadena.

6.15.5. Procedimiento Stuffing

El procedimiento Stuffing asegura que en la transmisión de una trama sólo exista un máximo de cinco bits consecutivos con la misma polaridad (véase Figura 6.19). Este procedimiento recibe y regresa una cadena de datos de tipo *bitstring*. Después de ejecutar el símbolo de inicio del procedimiento, se hace una llamada al procedimiento prepara_cadena y se determina la longitud de la variable cadena. Posteriormente se cuentan las veces que se repite el mismo valor y cuando la cuenta llega a cinco se realiza la inserción de bit, cambiando el bit siguiente por el valor opuesto al bit actual. Una vez finalizada la inserción de bits se analiza cuántas veces se realizó la inserción de bit y se eliminan los bits que no se utilizaron.

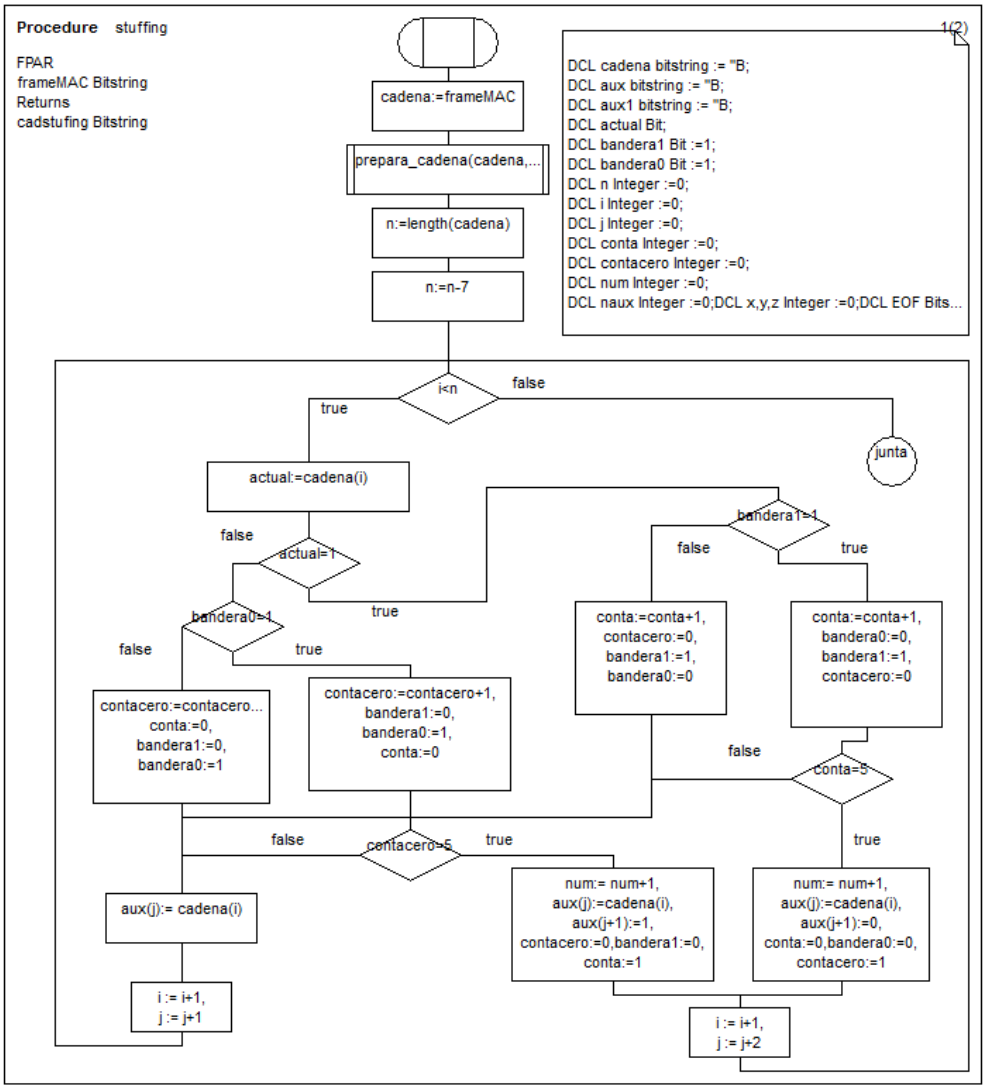

Figura 6.19. Descripción del procedimiento Stuffing.

6.15.6. Procedimiento Destuffing

El procedimiento Destuffing elimina los bits que fueron insertados por el procedimiento stuffing (véase Figura 6.20).

Este procedimiento recibe y regresa una cadena de datos de tipo *bitstring*. Después de ejecutar el símbolo de inicio del procedimiento, se hace una llamada al procedimiento prepara_cadena2 y se determina la longitud de la variable cadena. Posteriormente se cuentan las veces que se repite el mismo valor y cuando la cuenta llega a cinco se elimina el siguiente bit. Una vez finalizada la eliminación

de los bits de relleno se analiza cuántos bits fueron eliminados y mediante la operación substring se eliminan los bits que no se utilizaron.

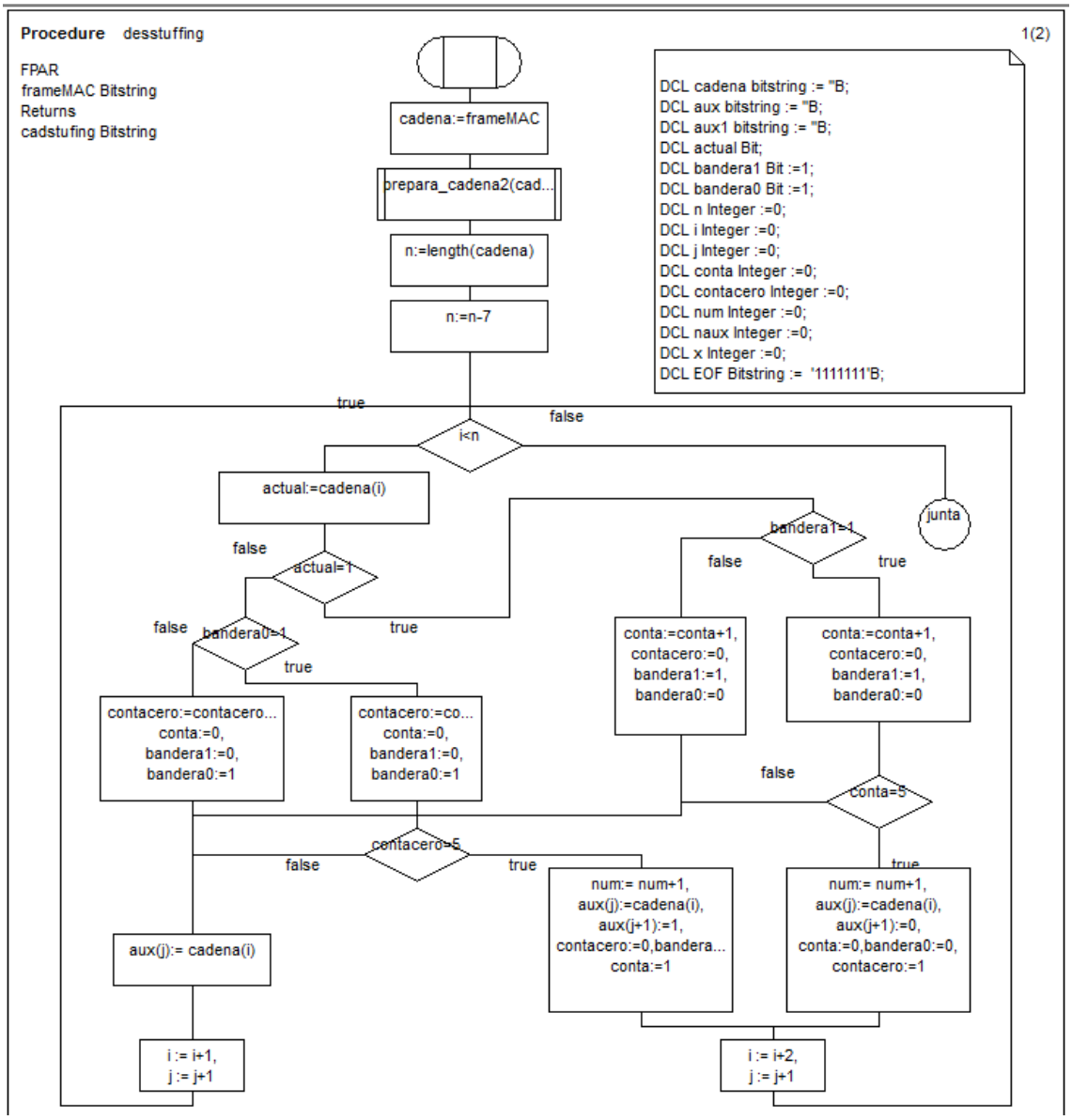

Figura 6.20. Descripción del procedimiento Desstuffing.

6.15.7. Procedimiento Check_CRC_error

El procedimiento Check_CRC_error verifica que el CRC enviado sea el mismo que el calculado por la ECU receptora (véase Figura 6.21). Este procedimiento recibe una cadena de datos de tipo *bitstring* y regresa una variable de tipo *boolean*. Después de ejecutar el símbolo de inicio del procedimiento, se analiza la

longitud de la cadena de bits recibida. Posteriormente se extrae el código de longitud de datos para determinar el tamaño de CRC que se utilizará en el cálculo del CRC de la ECU receptora. Terminado el cálculo de CRC se extrae el CRC enviado y se compara con el CRC calculado, la comparación se realiza bit a bit y en caso de que un bit sufra un cambio, se modifica la bandera aux para indicar que el CRC enviado con el calculado no coincide.

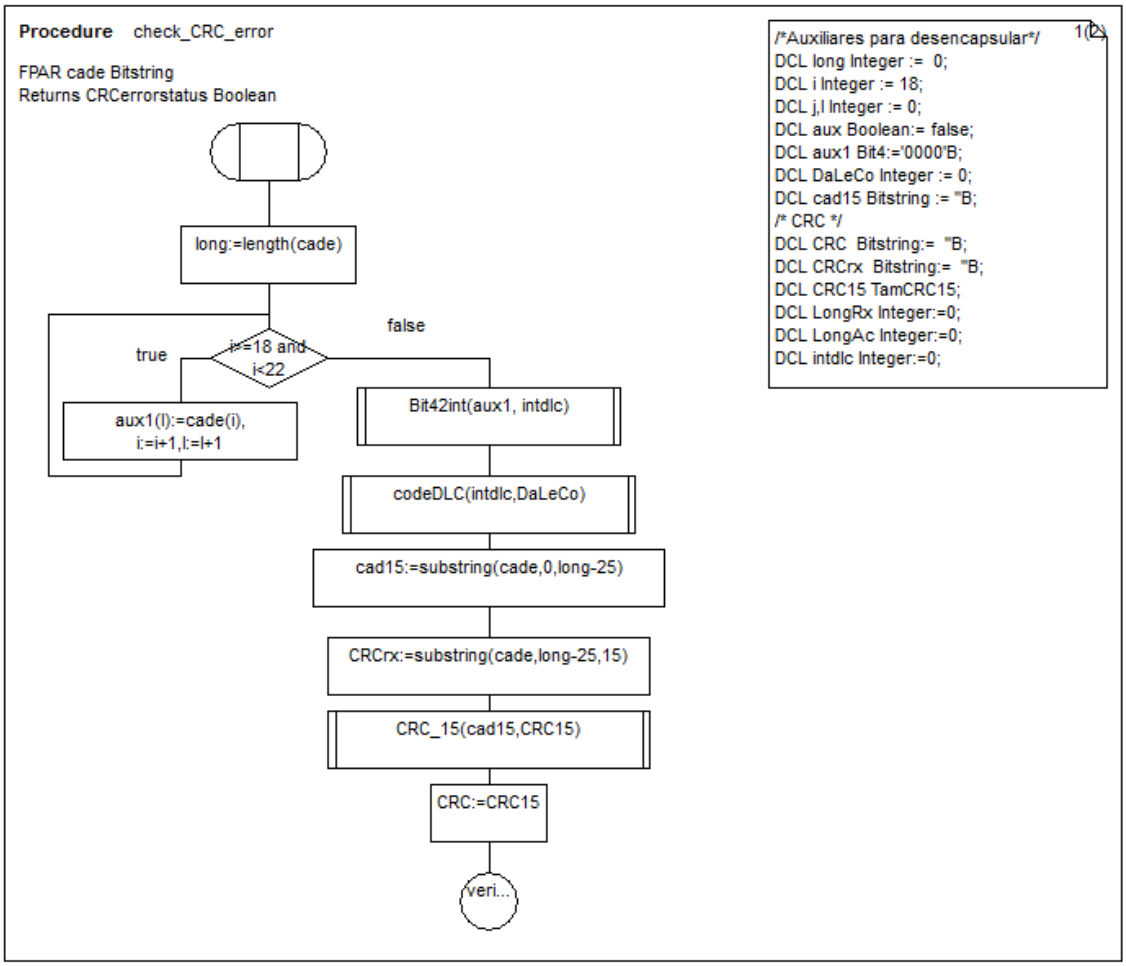

Figura 6.21. Descripción del procedimiento Check_CRC_error.

6.15.8. Procedimiento Check_stuff_error

El procedimiento Check_stuff_error verifíca que en la trama de datos no existan más de cinco bits del mismo valor (véase Figura 6.22). Este procedimiento recibe una cadena de datos de tipo *bitstring* y retorna una variable de tipo *boolean*. Si al realizar la cuenta de bits del mismo valor existen al menos seis bits del mismo

valor, se aumenta el contador num. Al final se analiza el contador num, si tiene un valor mayor o igual a uno indica que existe un error de insersión de bit.

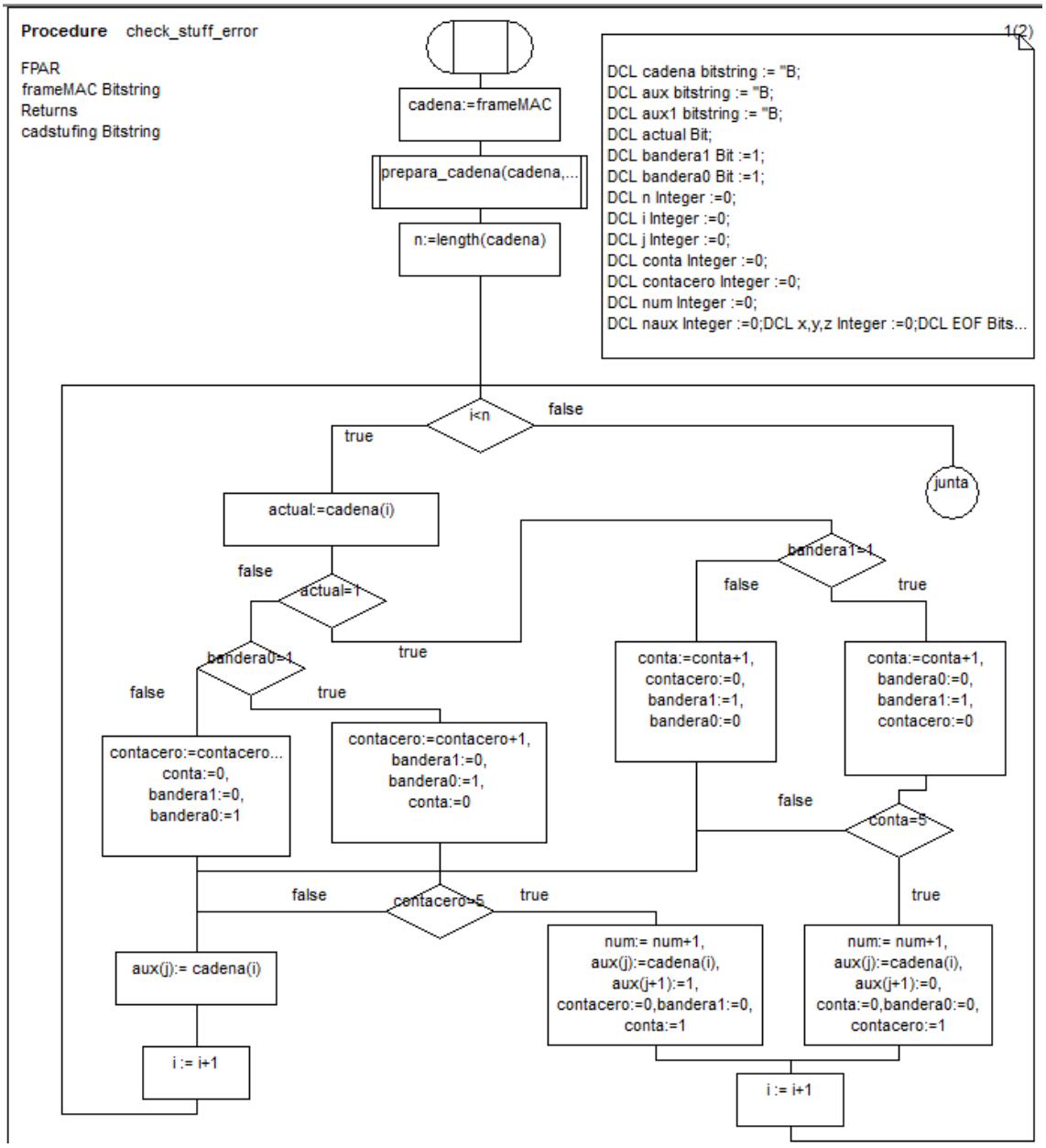

Figura 6.22. Descripción del procedimiento Check_Stuff_error.

6.15.9. Procedimiento Check_ack_error

El procedimiento Check_ack_error verifica que la trama enviada haya sido recibida correctamente mediante el análisis del bit de espacio ACK, si el valor es recesivo indica que la trama fue enviada correctamente, si el valor es dominante

indica que la trama no fue enviada correctamente (véase Figura 6.23). Este procedimiento recibe una cadena de datos de tipo *bitstring* y regresa una variable de tipo *boolean*.

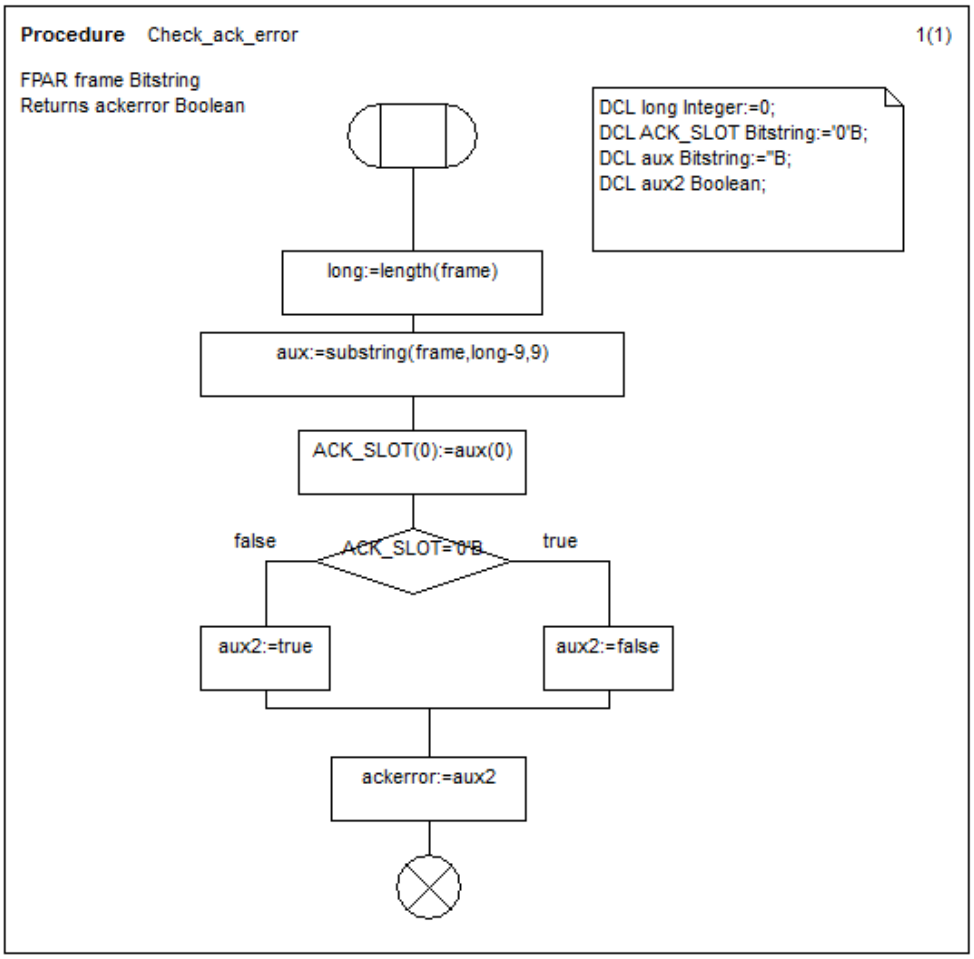

Figura 6.23. Descripción del procedimiento Check_ack_error.

6.15.10. Procedimiento Check_bit_error

El procedimiento Check_bit_error simula el arbitraje y analiza que los bits enviados al bus sean los mismos que lee; se realiza mediante una funcion de efecto local (eco) (véase Figura 6.24). El procedimiento hace una comparacion bit a bit y en caso de que algún bit no coincida indica que se perdió el arbitraje o que existe algún bit erróneo. Este procedimiento recibe una variable tipo *bitstring* y regresa una variable de tipo *boolean*.

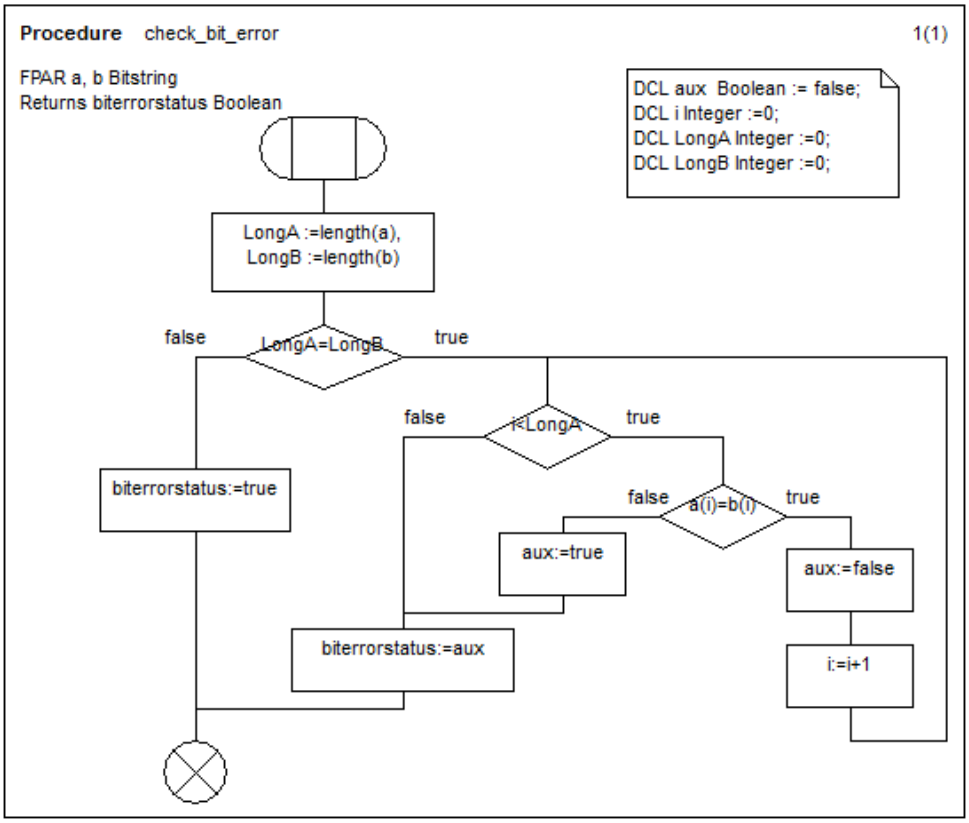

Figura 6.24. Descripción del procedimiento check_bit_error.

7. Resultados

La etapa de diseño que define la metodología de desarrollo (véase Figura 1.7) se ha detallado en el Capítulo 1, cuyo resultado principal es la especificación formal de la pasarela CAN/LIN (Capítulo 5). A continuación, se presenta la simulación, depuración y refinamiento de dicha especificación.

Como parte de la metodología de desarrollo, una vez obtenida la especificación formal ésta debe verificarse, para ello se realizan tres fases: simulación, validación y pruebas de comportamiento (test), las cuales deben realizarse conforme a la especificación y con ayuda de herramientas de software (Ellsberger, Hogrefe, y Sarma, 1997). En este caso, no se dispuso de herramientas adicionales, por lo que las fases mencionadas se realizaron en forma secuencial y durante el desarrollo de la especificación, empleando las propiedades de edición, análisis (sintáctico y semántico), simulación integral y generación de MSCs de la herramienta Cinderella SDL.

Las fases de simulación y validación se aplican a la especificación formal para corregir posibles errores de diseño y comprobar que se satisfagan los requerimientos respecto al comportamiento. La fase de pruebas de comportamiento tiene como objetivo verificar que una realización particular de la especificación reaccione ante los eventos de manera adecuada y se obtenga un grado elevado de confiabilidad del sistema. Debido a que el desarrollo de esta última fase es una tarea compleja, se requiere el dominio de lenguajes como TTCN-3 (*Testing and Test Control Notation*, tercera versión) y el uso de

herramientas de validación y de verificación. En el presente trabajo de tesis se realizó la validación y verificación de la especificación formal de la pasarela con la herramienta Cinderella SDL, como se describe a continuación.

Fases de simulación y validación

La especificación SDL de la pasarela CAN/LIN se realizó mediante la construcción de un sistema compuesto por bloques, los cuales contienen los procesos que describen el funcionamiento de ambos protocolos. Para realizar la fase de simulación, o prueba del sistema (prueba de caja negra), el usuario del sistema proporcionó los datos en formato decimal: (0,75,1,20), obteniendo los siguientes resultados:

- Correcta generación y comprobación del CRC en nodo CAN.
- Generación de las cabeceras pertenecientes a cada capa diseñada en el nodo CAN.
- Recepción correcta de las tramas que establecen la comunicacion entre DLLCAN y DLL_CAN.
- Recepción, transmision y procesamiento de los datos pertenecientes a capas superiores (datos) en el nodo CAN.
- Correcta generación y comprobación del *Checksum* en el nodo LIN.
- Generación de las cabeceras pertenecientes a cada capa diseñada en el nodo LIN.
- Recepción correcta de las tramas que establecen la comunicacion entre y DLLLIN y DLL_LIN.
- Recepción, transmision y procesamiento de los datos pertenecientes a capas superiores (datos) en el nodo LIN.

Durante la fase de simulación de la especificación de la pasarela CAN/LIN, se llevó a cabo la fase de validación mediante la supervisión constante del explorador de Cinderella SDL (véase Figura 7.1), el cual visualiza el estado en que se encuentra el proceso que se está simulando y los valores de las variables

pertenecientes a dicho proceso. Durante la fase de simulación se analizó la reacción del proceso ante la recepción de las señales que constituyen su alfabeto de entrada.

Cinderella SDL permite realizar el proceso de depuración, ya que tiene la posibilidad de establecer puntos de ruptura donde se requiere detener la ejecución, habilitar la ejecución paso por paso o seguir la ejecución de llamadas a procedimientos. De forma que la herramienta permite hacer un seguimiento, en modo gráfico, del símbolo SDL que se está procesando en todo momento; sin embargo, analizar el funcionamiento de un conjunto de procesos en el interior de un bloque o de un proceso con muchos estados, dificulta el seguimiento de la ejecución (como la mayoría de los procesos de la especificación del protocolo).

La Figura 7.2 muestra el aspecto de un conjunto de señales dentro del esquema MSC[10], con lo cual se pudo observar la evolución de la simulación como una secuencia de eventos ordenados en el tiempo. Durante este proceso se detectaron errores en el comportamiento de la especificación; una vez corregidos los errores, se dio inicio a un nuevo ciclo de simulación. En cada caso, la validación se consideró completa después de verificar un conjunto de esquemas MSC representando varios casos de uso y tras una exploración de estados en la que no se detectaron errores.

La herramienta Cinderella SDL cuenta con un proceso de verificación en tiempo real, por lo que la identificación de errores dependio del tiempo de ejecución de las simulaciones de la especificación formal de la pasarela CAN/LIN.

7.1. Análisis de los resultados

Durante la simulación se detectaron y corrigieron errores básicos de programación en la especificación dinámica del sistema, dentro de los principales errores detectados y corregidos están: variables fuera de rango, operaciones mal

[10] Para generar el MSC se interrumpió temporalmente la simulación mientras se analizaba el MSC.

calculadas, tramas ensambladas erróneamente, códigos de CRC calculados erróneamente, códigos *checksum* calculados erróneamente, transiciones implícitas, problemas de inicialización de estructuras de datos e interpretación errónea de algún aspecto de los protocolos.

La validación general de la especificación de la pasarela CAN/LIN se realizó mediante la verificación de la detección de errores, empleando el cálculo del CRC y del *checksum*, así como las funciones realizadas en cada uno de los procesos del sistema; también se analizaron los datos enviados por medio de señales a cada uno de los procesos definidos en la especificación de los protocolos.

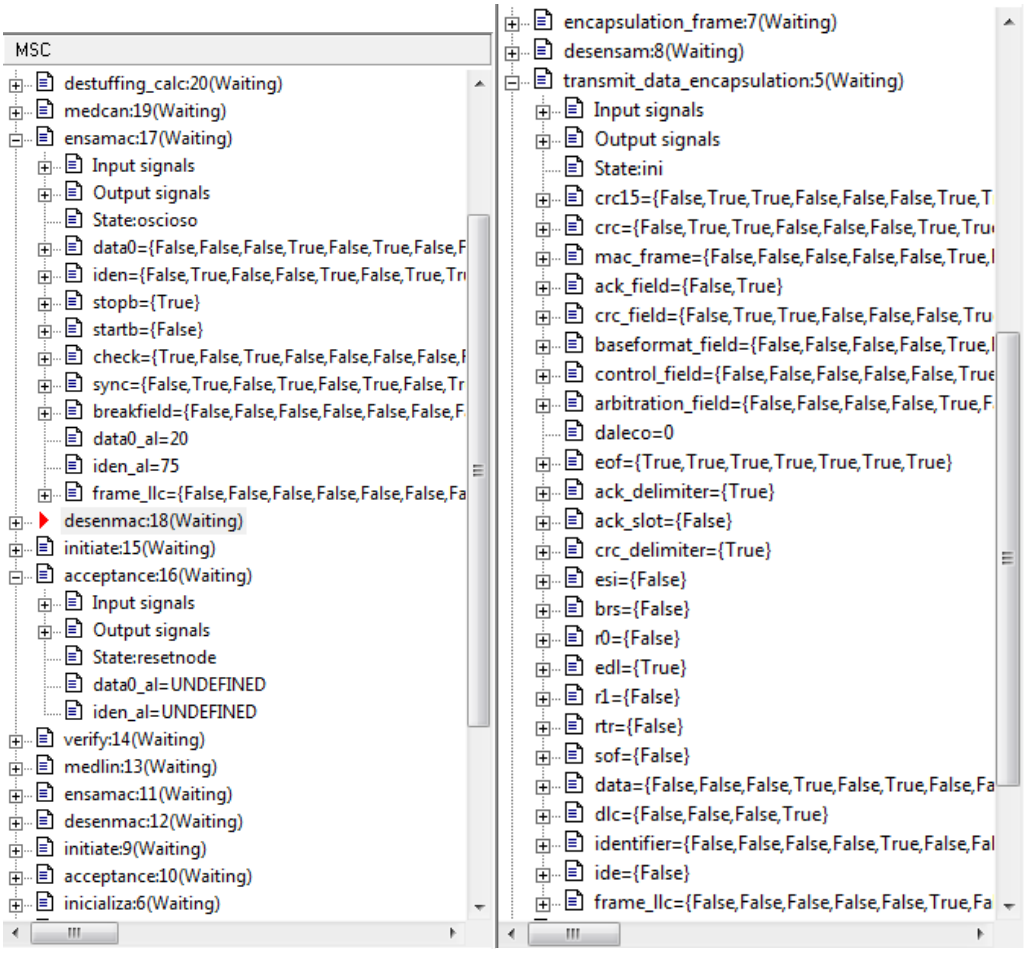

Figura 7.1. Exploración de los procesos durante la simulación.

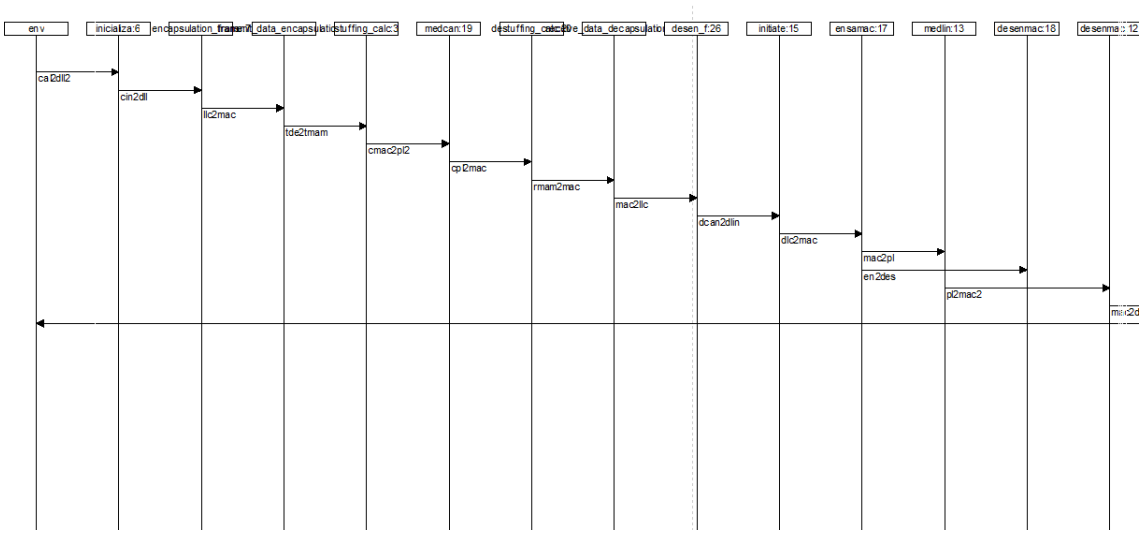

Figura 7.2. Ejemplo de los casos de uso (MSC) generados con Cinderella SDL.

8. Conclusiones y Trabajos Futuros

En el presente trabajo de tesis se realizó una investigación en dos áreas importantes en el desarrollo tecnológico e industrial, por un lado, los protocolos de comunicaciones CAN y LIN, y por otro, la FDT SDL. Resultado de dicho trabajo es la especificación ejecutable de la pasarela CAN/LIN como caso de estudio, que deriva el siguiente conjunto de aportaciones:

- Se realizó un estudio detallado sobre la historia del protocolo LIN, cuál es su objetivo y cuáles son sus principales aplicaciones en la industria.
- A partir de las especificaciones y otros textos escritos del protocolo LIN, se realizó un estudio en profundidad de las capas de supervisor, física y de enlace de datos (Capítulo 3).
- Se realizó un estudio detallado sobre la historia del protocolo CAN, su estandarización, cuál es su objetivo y cuáles son sus principales aplicaciones en la industria.
- A partir de las especificaciones, estándares y otros textos escritos del protocolo CAN, se realizó un estudio en profundidad de las capas de supervisor, física y de enlace de datos (Capítulo 4).
- Se estudió la norma SDL, detallada en la recomendación Z.100, lo cual fue fundamental para lograr el desarrollo de la pasarela CAN/LIN.
- Fue necesario estudiar el lenguaje ASN.1, detallado en la recomendación Z.105, durante las fases de pruebas y de desarrollo, ya que el procedimiento para calcular los CRC tomaba demasiado tiempo de

ejecución y con la manipulación correcta de esos tipos de datos se redujo de manera significativa el tiempo empleado en el cálculo del CRC.

- Se obtuvo una especificación estática de la pasarela CAN/LIN que permite analizar las señales, datos, bloques y procesos que conforman el sistema.
- Se obtuvo una especificación dinámica de la pasarela CAN/LIN que permiten analizar y evaluar el comportamiento del sistema.
- Se obtuvo una especificación formal SDL sin ambigüedades de la capa de enlace de datos, capa de supervisor y una reducida descripción de la capa física del protocolo de comunicaciones CAN. La especificación formal contribuye a la introducción de las FDT en la especificación de buses de campo para comunicaciones industriales.
- Se obtuvo una especificación formal SDL sin ambigüedades de la capa de enlace de datos, capa de supervisor y una reducida descripción de la capa física del protocolo de comunicaciones LIN.
- Es importante destacar que la especificación estática de la pasarela CAN/LIN no está descrita completamente en la especificación de ambos protocolos, sólo indica una idealización de la misma, por lo que fue necesario el desarrollo de bloques y procesos que no están descritos en el protocolo (medLIN, medCAN, ensaMAC, desenMAC).
- Con la especificación de la pasarela CAN/LIN se obtuvo una aplicación que permite al usuario observar e interactuar con el funcionamiento de una red CAN y una red LIN, con fines académicos y de investigación.

Durante el desarrollo de esta tesis, la experiencia adquirida en el lenguaje SDL como herramienta de especificación formal ha sido satisfactoria puesto que permite realizar un planteamiento general del problema a resolver de forma rápida, utilizando elementos visuales que sientan las bases para la construcción del sistema. Estos elementos visuales facilitan la comunicación de ideas entre varias personas en un equipo de desarrollo, por lo que cada etapa puede ser simulada y probada para verificar su correcto funcionamiento, con lo que los errores se detectan durante las fases iniciales, en las que su corrección resulta

más sencilla y más económica. El tiempo de desarrollo y el porcentaje de errores en el producto final se reduce considerablemente.

La herramienta Cinderella SDL brinda soporte al lenguaje SDL y su utilización es sencilla y atractiva. Al contar con una especificación formal, los diseñadores de sistemas únicamente se concentran en las funcionalidades, sin preocuparse en detalles concretos de la realización como son: sistema operativo y recursos físicos, entre otros. De tal manera que la especificación de un sistema puede ser utilizada en la generación de aplicaciones para diferentes plataformas adaptando únicamente las funciones de comunicación del entorno.

8.1. Trabajos futuros

Para dar continuidad a la línea de investigación propuesta por el Instituto de Electrónica y Mecatrónica de la Universidad Tecnológica de la Mixteca, sobre el estudio y utilización de lenguajes de descripción formal y su aplicación a los sistemas de redes industriales, en la que se enmarca este trabajo de tesis, se proponen los siguientes trabajos futuros:

- Extender la especificación formal de la pasarela CAN/LIN para incluir la capa de transporte en el protocolo LIN, con el propósito de realizar pruebas de diagnóstico reales.
- Realizar la simulación de la especificación con más ECUs en ambos protocolos, para poder realizar el intercambio de datos entre las ECUs y poder realizar la especificación de la capa física en su totalidad.
- Realizar un estudio detallado del comportamiento de la pasarela utilizando las demás topologías que permite que CAN y LIN se comuniquen entre sí.
- Realizar un estudio detallado de la eficacia del CRC implementado por el protocolo CAN respecto a los CRC utilizados en protocolos similares.

- Implementar cada uno de los protocolos en FPGAs, DSPics y microcontroladores para el diseño de una red que contenga al menos una DLL_LIN y una DLL_CAN.
- Realizar una valoración de la verificación usando herramientas de software con la finalidad de obtener una mayor rigurosidad.
- Utilizar SDL para especificar otros sistemas de comunicaciones industriales.

Bibliografía

Axelsson, J., Fröberg, J., Hansson, H., Norström, C., Sandström, K., & Villing, B. (2003). Correlating Business Needs and Network Architectures in Automotive Applications - A Comparative Case Study. *IFAC*, 1-10.

Belina, F., Hogrefe, D., & Sarma, A. (1991). *SDL with applications from protocol specification.* Hertfordshire, Great Britain: Prentice Hall.

BMW AG. (1999). *Byteflight Specification Rev 0.5.* Munich: BMW AG.

Burri, M., & Renard, D. (1993). *Single wire MI Bus controlling stepper motors.* Geneva: Motorola.

CAN in Automation. (Diciembre de 2014). Recuperado el 1 de Noviembre de 2014, de Página de CAN CiA: www.can-cia.org

CANkingdom International Inc. (2003). Recuperado el 1 de Noviembre de 2014, de Página Web Cankingdom: http://www.cankingdom.org

Cast Inc. (Octubre de 2013). Recuperado el 4 de Diciembre de 2014, de Página de Cast: www.cast-inc.com

Cena, G., Valenzano, A., & Vitturi, S. (2005). Advances in automotive digital communications. *Computer Standards & Interfaces, 27*(6), 665-678.

Chamú Morales, C. (2005). *Desarrollo de un sistema educativo para la enseñanza del protocolo de comunicaciones CAN.* Huajuapan de León: Universidad Tecnológica de la Mixteca.

Cinderella ApS. (12 de Octubre de 2006). *Página Web de Cinderella Apps*. Recuperado el 1 de Noviembre de 2014, de Cinderella SDL: http://www.cinderella.dk/

DSI Consortium. (2011). *DSI3 Bus Standard Rev 1.00*. DSI Consortium.

Ellsberger, J., Hogrefe, D., & Sarma, A. (1997). *SDL*. Hertfordshire, Great Britain: Prentice Hall.

Etschberger, K. (2001). *Controller Area Network*. Weingarten, Germany: IXXAT Automation GmbH.

Firlit, T., Sandoval, J., & Ellerbrock, P. (2004). *IntelliBus Protocol: Flexible and Cost-Effective Automotive Bus*. Colorado: Aeroflex Colorado Springs.

FlexRay Consortium. (15 de Diciembre de 2005). *FlexRay Communitation Systems Protocol Specification Rev 2.1A*. FlexRay Consortium.

Freescale Semiconductor. (24 de Junio de 2013). *MC9S12P Family*. Denver: Freescale Semiconductor. Recuperado el 1 de Diciembre de 2014, de www.freescale.com: http://cache.freescale.com/files/microcontrollers/doc/ref_manual/MC9S12P128RMV1.pdf?pspll=1

González Salinas, R. (2008). *Especificación del protocolo FlexRay utilizando un lenguaje de descripción formal*. Huajuapan de León: Universidad Tecnológica de la Mixteca.

Hall, A. (2005). Realising the Benefits of Formal Methods. *7th International Conference on Formal Engineering Methods* (págs. 1-4). Manchester: Springer-Verlag Berlin Heidelberg.

Infineon Technologies AG. (28 de Abril de 2009). *TLE8458Gx LIN Transceiver with integrated Low Drop Voltage Regulator*. Munich: Infineon Technologies AG. Recuperado el 1 de Noviembre de 2014, de Pagina de

Infineon: http://www.infineon.com/dgdl/Infineon-TLE8458GX-DS-v01_01-en.pdf?fileId=db3a304320d39d590121f2948c610a82&ack=t

ISO/IEC 11519. (15 de Junio de 1994). *Road Vehicles - Low Speed Serial Data Communication*. Geneva, Switzerland: International Organization for Standardization.

ISO/IEC 11898. (15 de Noviembre de 1993). *Road vehicles - Interchange of Digital Information, Controller Area Network*. Geneva, Switzerland: International Organization for Standardization.

ISO/IEC 15765-2. (2002). *Road vehicles - Diagnostics on Controller Area Network (CAN) - Part 2: Network Layer Services*. Geneva: International Organization for Standardization.

ISO/IEC 7498. (1989). *Open Systems Interconnect Basic Reference Model*. Geneva: International Organization for Standardization.

ISO/IEC 9074. (1989). *Information Processing Systems, Open System Interconnection, ESTELLE: A formal description technique based on an Extended Transition Model*. Geneva: International Organization for Standardization.

ISO/IEC 9646. (1994). *Information processing systems, Open Systems Interconnection, conformance testing methodology and framework*. Geneva: International Organization for Standardization.

ISO/IEC 9907. (1989). *Open System Interconnection, LOTOS: A formal description technique based on the Temporal Ordering of the Observational Behavior*. Geneva: International Organization for Standardization.

ITU-T. (2000). *Recommendation Z.105. Use of SDL with ASN.1*. Geneva: Telecommunications Standardization Sector of the International Telecommunication Union.

ITU-T. (2000). *Z.100 Specification and description language (SDL).* Geneva, Switzerland: Telecommunications Standardization Sector of the International Telecommunication Union.

Kumar, N., Sharma, R., & Ramya, B. (Abril de 2010). Design And Development Of Fault Tolerant CAN - LIN Gateway For In-Vehicle Communication. *SasTech, 9*, págs. 17-22.

Lawrenz, W. (1997). *CAN system engineering: From Theorical to Practical Applications.* Wolfenbuettel: Springer.

Leen, G., & Hefferman, D. (2002). Expanding Automotive Electronic Systems. *Computer, 35*, págs. 88-93.

LIN Consortium. (2010). *LIN Specification Package Rev 2.2A.* LIN Consortium.

López Pérez, E. (2015). *Especificación formal del protocolo de comunicaciones CAN FD mediante SDL.* Huajuapan de León: Universidad Tecnológica de la Mixteca.

Mariño, P., Hernández, H., Domínguez, M. A., Poza, F., & Machado, F. (2003). Computer Engineering Education in Network Protocols. *Proceedings of the 2003 International Conference on Parallel and Distributed Processing Techniques and Applications.* Las Vegas: Springer-Verlag Berlin, Heidelberg.

Mariño, P., Hernández, H., Domínguez, M. A., Poza, F., Machado, F., & Vázquez, F. (2003). Industrial Network Technologies and Related Information Systems. *Proceedings of the International Conference on Computer, Communication and Control Technologies. I*, págs. 64-69. Orlando: IIIS.

Microchip Technologies Inc. (30 de Enero de 2007). *MCP201 LIN Transceiver with Voltage Regulator.* Chandler: Microchip Technologies Inc. Recuperado el 1 de Noviembre de 2014, de Pagina de Microchip Inc.: http://ww1.microchip.com/downloads/en/DeviceDoc/21730F.pdf

Microchip Technologies Inc. (5 de Marzo de 2013). *PIC16F1829LIN Datasheet*. Chandler: Microchip Technologies Inc. Recuperado el 1 de Noviembre de 2014, de Pagina de Microchip Technologies Inc: http://ww1.microchip.com/downloads/en/DeviceDoc/41673A.pdf

MOST Cooperation. (2010). *Especificacion MOST Rev 3.0*. Karlsruhe: MOST Cooperation.

Müller, D., Sommer, D., & Stegemann, S. (2009). *Local Interconnet Network*. Berlin: Beuth University of Applied Sciences Berlin.

Navet, N., Song, Y., Simonot-Lion, F., & Wilwert, C. (2005). Trends in Automotive Communication Systems. *Proceedings of the IEEE. 93*, págs. 1204-1223. Piscataway: IEEE.

Nogueira Nine, J. (2000). *Metodologías de Especificación Formal Aplicadas a Modelos Normalizados de Protocolos de Comunicación Industriales, Tesis Doctoral*. Vigo: Universidad de Vigo.

Nolte, T., Hansson, H., & Lo Bello, L. (2005). Automotive Communications - Past, Current and Future. *IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2005). 1*, págs. 985-992. Catania: IEEE.

OSEK-VDX. (2004). Recuperado el 3 de Noviembre de 2014, de Página de Osek Vdx: http://www.osek-vdx.org

Pérez Ortiz, D. D. (2011). *Especificación del protocolo DNP3 utilizando un lenguaje de descripción formal*. Huajuapan de León: Universidad Tecnológica de la Mixteca.

Philips Semiconductor. (13 de Enero de 2004). *TJA LIN Transceiver*. Eindhoven: Philips Semiconductor. Recuperado el 1 de Noviembre de 2014, de Pagina de NXP: http://www.nxp.com/documents/data_sheet/TJA1020.pdf

R. M., S., & G. D., P. (1999). Formal description technique SDL for manufacturing systems specification and description. *International*

Conference on Advances in Production Management Systems. 24, págs. 449-456. Berlin: Springer.

Ribbens, W. B., & Mansour, N. P. (2003). *Understanding Automotive Electronics.* Burlington, USA: Newnes.

Richards, P. (2001). *AN754 Understanding Microchip's CAN Module Bit Timing.* Chandler: Microchip Tecnology Inc.

Robert Bosch GmbH. (1991). *CAN Specification Version 2.0.* Stuttgart, Baden-Württemberg: Robert Bosch GmbH.

SAE J1850. (Julio de 1995). *Class B Data Communication Network Interface.* Warrendale: SAE Vehicle Network for Multiplexing and Data Communications Standards Comitee.

SAE J2056-2. (1994). *Survey of known Protocols.* Warrendale: SAE International.

Seo, S.-H., Lee, S.-W., Hwang, S.-H., & Wook Jeon, J. (2006). Development of Network Gateway Between CAN and FlexRay Protocols For ECU Embedded Systems. *SICE-ICASE International Joint Conference*, págs. 2256-2261.

Thompson, M. (February de 1996). The Thick and Thin of Car Cabling. *IEEE Spectrum, 33*(2), págs. 42-45.

Tsugawa, S. (2005). Issues and recent trends in vehicle safety. *IATSS Research. 29*, págs. 7-15. International Association of Traffic and Safety Sciences.

Turner, K. J. (1993). *Using Formal Description Techniques: An Introduction to ESTELLE, LOTOS and SDL.* Wiley.

Vivekanaandan, B., Sabane, M. K., & Krishna, S. (2003). Configurable Vehicle Networks. *Proceedings of the First National Conference on Automotive Infotronics.* Chennai: SAE India.

Waern, M. (2003). *Real-Time Communication: Evaluation of Protocols for Automotive Systems. Master´s Thesis.* Stockholm: KTH Royal Institute of Technology.

Wiewesiek, W. (2007). Infotainment Using IDB-1394 - In-vehicle Audio and Video Data Transfer. *ATZ Elektronik Worldwide Edition*, págs. 30-32.

X-by-wire project. (1998). *X-by-wire Safety related fault tolerant systems in vehicles (final report).* Stuttgart.

Xing, W., Chen, H., & Ding, H. (1999). The application of controller area network on vehicle. *Proceedings of the IEEE International Vehicle Electronics Conference. 1*, págs. 455-458. Changchun: IEEE.