UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"ESPECIFICACIÓN FORMAL DEL PROTOCOLO DE COMUNICACIONES CAN FD MEDIANTE SDL"

TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN ELECTRÓNICA, OPCIÓN: SISTEMAS INTELIGENTES APLICADOS

PRESENTA

ING. EDILBERTO LÓPEZ PÉREZ

DIRECTOR DE TESIS

DR. ENRIQUE GUZMÁN RAMÍREZ

HUAJUAPAN DE LEÓN, OAXACA. ABRIL 2015

Tesis presentada el 23 de Abril de 2015
ante los sinodales:

M.S.R.C. José Antonio Moreno Espinosa

M.C. Esteban Osvaldo Guerrero Ramírez

Dr. Iván Antonio García Pacheco

Dr. Antonio Orantes Molina

Director de tesis:
Dr. Enrique Guzmán Ramírez

Dedicatoria

A mi familia quienes me han apoyado sin importar las dificultades que se presentan en la vida, y gracias a ellos soy lo que soy, porque gracias a su apoyo y consejos he llegado a realizar una de mis más grandes metas.

A mi padre:

Narciso López López

A mi madre

Eugenia Pérez López

A mi hermano

Alonso López Pérez

Edilberto.

Agradecimientos

Primeramente agradezco a Dios, por la vida misma, por estar siempre a mi lado y por poner a las personas adecuadas en mi camino.

Quiero agradecer de manera enfática al Ing. Heriberto Ildefonso Hernández Martínez por la comprensión y más que nada por su amistad, apoyo, esfuerzo, dedicación, paciencia, motivación, conocimientos, orientaciones y persistencia durante la elaboración de esta tesis.

A mi director Dr. Enrique Guzmán Ramírez por su apoyo y consejos durante el desarrollo de este trabajo de tesis.

A mis sinodales, Dr. Iván Antonio García Pacheco, Dr. Antonio Orantes Molina, M.C. José Antonio Moreno Espinosa y M.C. Esteban Osvaldo Guerrero Ramírez por el tiempo dedicado a la revisión de este trabajo, por sus observaciones y sugerencias realizadas.

A mis padres por su apoyo, consejos, comprensión, amor, ayuda en los momentos difíciles y por ayudarme con los recursos necesarios para estudiar. A ustedes que creyeron en mí y que con su ayuda me han convencido de que nada es imposible si se desea seguir adelante. Muchas gracias por ser mis padres!!!

A mi hermano, porque eres esa clase de persona que todo comprende y da lo mejor de sí mismo sin esperar nada a cambio . . . Porque sabes escuchar y brindar ayuda cuando es necesario . . . Porque te has ganado el cariño, admiración y respeto de todo el que te conoce.

A Gaby por alentarme a seguir adelante en los momentos difíciles, porque tu apoyo significó la garantía de saberse nunca solo, porque sin tu amor y cariño jamás habría llegado a la cima. Gracias por ser parte de mi vida!!!

A mi familia en general, mi abuelita Agustina, mis padrinos y en especial a Hernán por las muchas horas de pláticas.

A quienes a pesar del tiempo siguen en comunicación conmigo, por lo que los considero mis amigos: Cornelio, Enrique, David, Omar, Fito, Itzel, Donizetti, Yovany.

A todos mis compañeros de la maestría porque cada persona te enseña algo diferente: Magdiel, Misael, Segismundo, Carlos González, Ivonne, Oscar, Carlos Escobar, Cornelio y Mario.

Y a todos los que me faltaron mencionar, ya que es imposible recordar a todos cuando más ansias se tienen de terminar.

"Se paga un precio por ir tras tus sueños, pero se paga un precio mucho más alto por quedarte en el mismo lugar."

Anónimo.

Edilberto.

Índice general

Índice de figuras

Índice de tablas

Acrónimos

ACK	Acknowledgement
AGATE	Advanced General Aviation Experiments
AS-I	Actuator-Sensor Interface
AUTOSAR	Automotive Open System Architecture
BRS	Bit Rate Switch
CAIS	Common Airborne Instrumentation System
CAL	CAN Application Layer
CAN	Controller Area Network
CAN FD	CAN with Flexible Data Rate
CCITT	Comité Consultatif International Télégraphique et Teléphonique
CiA	CAN in Automation
CMS	CAN Message Specification
CRC	Cyclic Redundancy Check
CSMA/CD+AMP	Carrier Sense Multiple Access with Collision Detection and Arbitration Message Priority
DBT	Identifier Distributor
DLC	Data Length Code
DLL	Data Link Layer
E.U.A	Estados Unidos de América

ECM	Engine Control Modules
ECU	Electronic Control Units
EDL	Extended Data Length
EOF	End of Frame
ESA	European Space Agency
ESI	Error State Indicator
ETSI	European Telecommunications Standards Institute
FAA	Federal Aviation Administration
FAR	Federal Acquisition Regulation
FB	Fieldbus
FDL	Formal Description Language
FDT	Tecnicas de descripción formal Formal Description Techniques
FIP	Factory Instrumentation Protocol
FPGA	Field Programmable Gate Array
GPS	Global Positioning System
HLP	Higher Layer Protocols
HMI	Human Machine Interface
iCC	international CAN Conference
IDE	Integrated Development Environment
IDE	Identifier Extension Flag
ISO	International Organization for Standarization
ITU	International Telecommunication Union
ITU-T	Telecommunication Standardization Sector
LIN	Local Interconnect Network
LLC	Logic Link Control
LMT	Layer Management
LOTOS	Language of Temporal Ordering Specification
LSDU	Link Service Data Units
MAC	Medium Access Control

MAU	Medium Access Unit
MBS	More Bit Significant
MDI	Medium Dependent Interface
MOST	Media Oriented System Transport
NASA	National Aeronautic and Space Administration
NMT	Network Management
NRZ	Non Return to Zero
OD	Object Dictionary
OEM	Original Equipment Manufacturer
OSEK	Open Systems and their Interfaces for the Electronics in Motor Vehicles
OSI	Open Systems Interconnection
PHASE_SEG	Phase Buffer Segment
PLC	Programmable Logic Controller
PLS	Physical Signalling
PMA	Physical Medium Attachment
PROFIBUS-PA	PROFIBUS Profile for Process Automation
PROP_SEG	Propagation Delay Segment
REC	Receive Error Counter
RJW	Resynchronization Jump Width
RTR	Remote Transmission Request
SAE	Society of Automotive Engineers
SATS	Small Aircraft Transportation System
SDL	Specification and Description Language
SDS	Smart Distributed System
SeeCAN	Sistema educativo para la enseñanza del protocolo de comunicaciones CAN
SMART-1	Small Missions for Advanced Research in Technology
SOF	Star of Frame
SRR	Substitute Remote Request

STELLE	Extended Finite State Machine Language
SYNC_SEG	Synchronizacion Segment
TEC	Transmit Error Counter
TTCAN	Time Triggered CAN
TTCN	Tree and Tabular Combined Notation
TTP/A	Time Triggered Protocol
UML	Unified Modeling Language
UTM	Universidad Tecnológica de la Mixteca
VAN	Vehicle Area Network

Capítulo 1

Introducción

Las técnicas de descripción formal (FDT, *Formal Description Techniques*) permiten modelar y simular paso a paso la evolución de sistemas y protocolos de comunicaciones y realizar diferentes pruebas de comportamiento con la finalidad de evitar errores en la etapa de producción, reduciendo costos y tiempo durante la realización total [1]; recientemente se han adoptado como formas de documentación en el proceso de estandarización y para estudiar su funcionamiento.

Un lenguaje de descripción formal (FDL, *Formal Description Language*) permite evaluar las alternativas y soluciones de los sistemas mediante su análisis y simulación. En la actualidad existen FDL tales como STELLE (*Extended Finite State Machine Language*) [2], LOTOS (*Language of Temporal Ordering Specification*) [3] y SDL (*Specification and Description Language*) [4]. La Organización Internacional de Estandarización (ISO, *International Organization for Standarization*) normalizó STELLE y LOTOS, mientras SDL fue desarrollado por el CCITT (*Comité Consultatif International Télégraphique et Teléphonique*), ahora ITU-T (*Telecommunication Standardization Sector*) uno de los tres sectores de la ITU (*International Telecommunication Union*), y está detallado en la recomendación Z.100 [5].

SDL es la principal FDT utilizada en el área de las telecomunicaciones. SDL permite especificar y describir sistemas mediante una representación gráfica y/o representación textual. Los entornos de desarrollo integrado (IDEs, *Integrated Development Environment*) más conocidos para el uso de SDL son: IBM Rational Suite SDL [6] antes Telelogic Tau, Pragma-

Dev RTDS [7], SAFIRE World [8] antes Safire-SDL, Dafocus [9] antes ObjSystem, Sandrila [10] y Cinderella SDL [11]. En el Instituto de Electrónica y Mecatrónica de la Universidad Tecnológica de la Mixteca (UTM) se cuenta con la licencia de Cinderella SDL 1.3.

En la UTM se han realizado trabajos de investigación que se consideran antecedentes al presente trabajo, por ejemplo:

- Desarrollo de un sistema educativo para la enseñanza del protocolo de comunicaciones CAN. El Sistema educativo para la enseñanza del protocolo de comunicaciones CAN (SeeCAN) es una herramienta que tiene como principal objetivo el apoyo didáctico en la enseñanza de dicho protocolo. El SeeCAN se diseñó mediante una metodología de desarrollo de sistemas embebidos, y como resultado se obtuvo un innovador 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 [12].

- 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. Cabe señalar que la documentación en FlexRay ya está escrita en SDL y se verificó mediante la herramienta Cinderella SDL [13].

- 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 DNP3 [14].

1.1. Marco Teórico

1.1.1. Técnicas de Descripción Formal

Una FDT es un método de especificación basado en un lenguaje de descripción que emplea reglas rigurosas e inequívocas con respecto al desarrollo de expresiones en ese lenguaje (sintaxis formal) y a la interpretación del significado de dichas expresiones (semántica formal). Las FDT fueron desarrolladas para asegurar un lenguaje sin ambigüedades, especificaciones claras, concisas y completas que sirven como medio para el desarrollo, especificación, realización y verificación de recomendaciones y estándares [15].

Una descripción en lenguaje natural es un ejemplo de técnica de descripción no formal que emplea uno de los lenguajes utilizados para publicar recomendaciones. Puede complementarse con notaciones matemáticas, notaciones aceptadas de otros tipos, figuras, etc.

1.1.1.1. Objetivos de una FDT

El objetivo de una FDT es permitir especificaciones precisas e inequívocas. Además, se pretende que satisfagan otros objetivos, como son [16, 17]:

- Constituir una base para analizar especificaciones sobre corrección, eficiencia, etc.
- Constituir una base para determinar si las especificaciones están completas o no.
- Constituir una base para verificar las especificaciones con respecto a los requisitos de la recomendación.
- Constituir una base para determinar la conformidad de las realizaciones de las recomendaciones.
- Constituir una base para determinar la coherencia entre las especificaciones de distintas recomendaciones.
- Constituir una base de apoyo para las realizaciones prácticas.

En la actualidad, en algunas áreas podría ser necesaria más de una FDT para cumplir todos los objetivos, por ejemplo una combinación de SDL y UML (*Unified Modeling Language*). Los diagramas UML especifican todos los detalles acerca del comportamiento de las clases necesarias para generar código ejecutable mientras que con SDL se especifica el comportamiento del sistema. Usar UML y SDL en una especificación provee distintas observaciones en diferentes niveles de abstracción en el mismo sistema y complementa el proceso de documentación [18].

1.1.1.2. Ventajas de una FDT

La aplicación de una FDT puede aportar ventajas como [19]:

- Mayor calidad de las recomendaciones y estándares, que redundaría en una reducción de los costos de mantenimiento para el CCITT y para los usuarios de las recomendaciones y estándares.
- Menor dependencia del lenguaje natural para comunicar conceptos técnicos en un entorno multilingüe.
- Reducción del tiempo de elaboración de las realizaciones prácticas, gracias al empleo de medios basados en las propiedades de las FDT.
- Mayor facilidad de realización, que se traduciría en mejores productos.

1.1.2. Lenguaje de Especificación y Descripción Formal

El desarrollo de SDL comenzó en 1972, después de que algunas investigaciones indicaban que era necesario contar con herramientas para tratar las especificaciones de sistemas complejos, fue desarrollado por la ITU y sus especificaciones se encuentran detalladas en la recomendación Z.100. La primer versión de SDL fue lanzada en 1976, seguida de nuevas versiones en 1980, 1984, 1988, 1992, 1996 y 2000. Las primeras versiones de 1976 y 1980 eran muy básicas y cubrían únicamente el lenguaje gráfico, y aspectos del comportamiento de un sistema de una manera informal. La versión de 1984 añadió la posibilidad de utilizar datos y

de estructurar las especificaciones. En 1988 SDL adquiere su estado de madurez como FDT. En 1992 mejoró al incluir la orientación de objetos dentro del lenguaje y la posibilidad de usar distintos tipos de librerias llamadas *packages*. Las versiones de 1996 y 2000 incluyen mejoras y corrigen los errores detectados en las años anteriores, muchas de estas mejoras son resultado de los reportes de los usuarios [20, 21, 22].

La especificación completa de un sistema incluye diferentes tipos de información, cada tipo de información debe presentarse en el lenguaje apropiado. Dependiendo de la situación puede ser fomal, semi-formal o natural. El lenguaje formal es un lenguaje cuyos símbolos primitivos y reglas para unir esos símbolos están formalmente especificados y al conjunto de los símbolos primitivos se llama alfabeto (o vocabulario) del lenguaje, y al conjunto de las reglas se conoce como gramática formal (o sintaxis). El lenguaje natural es usado para describir metas y requerimientos pero es ambiguo e incompleto, es decir, tiene más de una interpretación que es difícil de analizar [23].

Un lenguaje de especificación permite evaluar las alternativas y soluciones de un sistema por medio del análisis y simulación. El propósito de SDL es proveer un lenguaje para la especificación y descripción formal del comportamiento de los sistemas de telecomunicaciones [24]. Las especificaciones y descripciones usando SDL son intencionadas a ser formales en el sentido de que es posible analizar e interpretar el comportamiento de un sistema sin ambigüedad [22].

La denominación de SDL como lenguaje de especificación y descripción requiere de una diferenciación entre estos dos términos:

- Una especificación de un sistema describe y define cuales son los requerimientos y exigencias de comportamiento globales de un sistema, por lo tanto una especificación ve al sistema como una caja negra

- Una descripción de un sistema indica su actual comportamiento y estructura interna; esto es la implementación.

Sin embargo sólo se trata de una indicación, puesto que el lenguaje SDL como tal no distingue entre especificaciones y descripciones [25].

En general es posible especificar varios aspectos de un sistema, como el diseño de hardware, dimensiones físicas, consumo de energía, entre otros. SDL especifica el comportamiento, estructura y el establecimiento de parámetros generales del sistema (datos). SDL intenta especificar el comportamiento de aspectos de un sistema; los parámetros generales describen las propiedades como la capacidad y el peso que tienen que ser descritos por diferentes técnicas.

1.1.2.1. Áreas de aplicación de SDL

El Instituto Europeo de Estándares y Telecomunicaciones (ETSI, *European Telecommunications Standards Institute*) ha impulsado que los estándares sean detallados mediante recomendaciones formales, siendo SDL la técnica formal preferida por el ETSI [21].

SDL fue concebido para el uso en sistemas de telecomunicaciones, incluyendo sistemas de comunicaciones de datos, pero puede ser usado en todos los sistemas de comunicaciones en tiempo real. SDL puede ser usado para describir la estructura interna de un sistema en diferentes niveles de abstracción, comenzando desde la perspectiva y llevándolo a detalles más específicos. Las aplicaciones en el área de las telecomunicaciones incluyen mantenimiento y tratamiento de fallos por ejemplo: alarmas, despeje automático de fallos, rutina de pruebas, sistemas de control por ejemplo: control de cargas, modificación y extensión de procedimientos, operación y mantenimiento de funciones, administración de redes, servicios de telecomunicaciones, protocolos de comunicaciones de datos, en general sistemas de telecomunicaciones [26].

Las principales aportaciones de usar SDL son: facilidad de requerimientos, especificación de sistemas, recomendaciones de estándares (internacionales, regionales o nacionales), especificaciones de un sistema, especificaciones detalladas, descipciones de un sistema en un alto nivel y suficientes para directamente producir implementaciones, y descripción de sistemas de pruebas [27].

1.1.2.2. Arquitectura de un sistema con SDL

La arquitectura del diseño de un sistema SDL, como muestra la Figura 1.1, se construye a partir de un nivel superior (*system*) compuesto por bloques (*blocks*); cada bloque puede

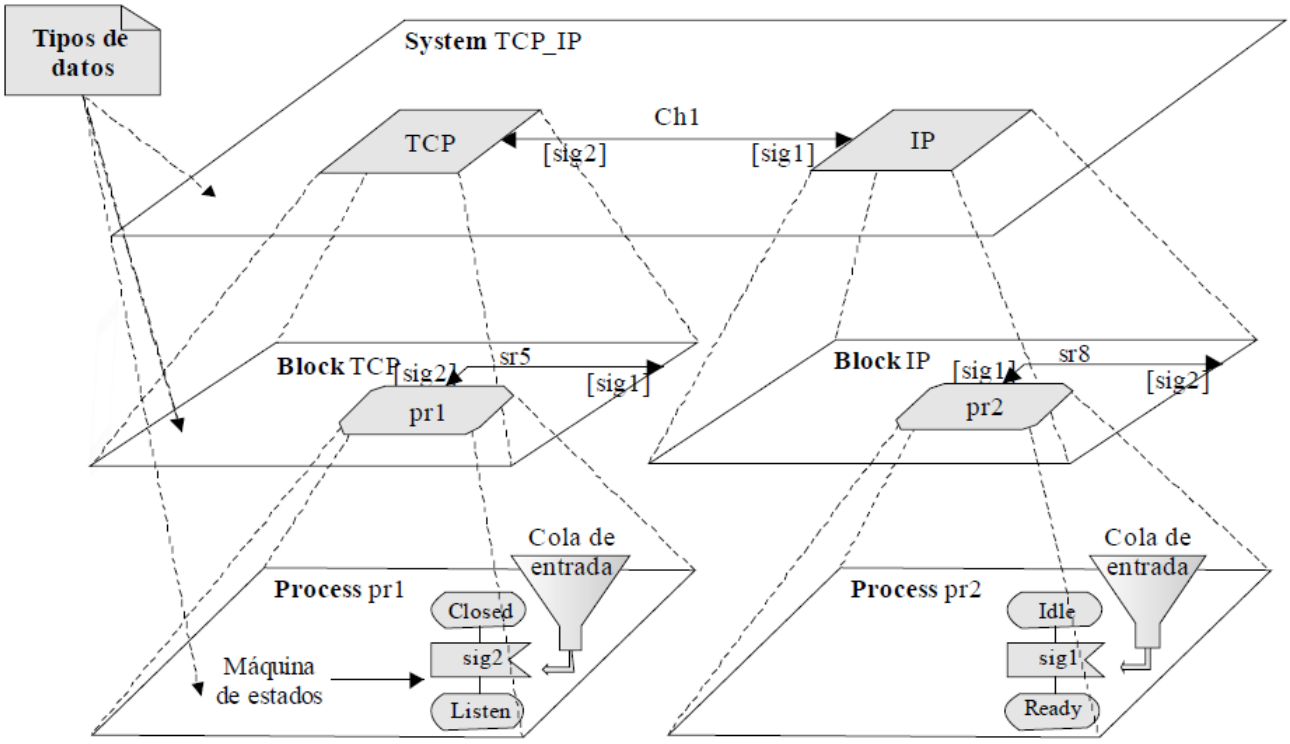

Figura 1.1: Arquitectura de sistema SDL [13].

contener bloques o procesos pero no ambos; los procesos (*process*) están compuestos por máquinas de estado que se comunican entre sí mediante señales a través de canales (*channels*) o rutas de señal (*signal route*) [22, 21].

1.1.3. Buses de campo en automoción

Desde la década de 1970, se observó un incremento exponencial en el número de sistemas electrónicos que gradualmente reemplazaron a los sistemas puramente mecánicos e hidráulicos. Esta tendencia condujo a un rápido incremento en el número de canales individuales de comunicaciones, por lo que las comunicaciones llegaron a ser un factor importante en los sistemas modernos. El incremento en la demanda de las comunicaciones condujo a la creación de especificaciones y la existencia de nuevos protocolos de comunicaciones. De acuerdo al

área de aplicación algunos protocolos han sido establecidos como estándares internacionales o como estándares *quasi de facto* [28].

Los protocolos de comunicaciones para aplicaciones industriales fueron llamadas buses de campo (FB, *Fieldbus*) que son sistemas de transferencia de información destinados a aplicaciones de tiempo real distribuidas, sistemas de automatización, sistemas de supervisión y control en el ámbito de celdas de producción [28]. Los FBs interconectan dispositivos electrónicos, tales como sensores, actuadores y unidades de control electrónico (ECU, *Electronic Control Units*), que operan desde complicados procesos industriales hasta simples procesos en el hogar.

El surgimiento de los FB fue motivado por una serie de necesidades, entre ellas: reducir el cableado en las instalaciones, dotar de inteligencia a los sensores (*smart sensors*), dotar de inteligencia a los actuadores (*smart actuators*), desarrollar sistemas de control distribuido y obtener mayor seguridad en la transferencia de información [12, 13, 14].

Los vehículos electrónicos y las comunicaciones representa un crecimiento en el área del sector automotriz, asociado al incremento de arreglos electrónicos como: sensores, actuadores, micro-procesadores, paneles de instrumentación, controladores y ECUs que interactuan en tiempo real, varios productores de automóviles y la industria automotriz comenzaron a desarrollar protocolos de comunicaciones dedicados a la automoción para cubrir las necesidades de comunicación, peso y cableado [29].

1.1.3.1. Clasificación SAE

La Sociedad de Ingenieros de Automoción (SAE, *Society of Automotive Engineers*) ha definido una clasificación formal para los protocolos automotrices basado en la velocidad de transmisión de datos y las funciones que son distribuidas en la red, las clases son [30, 31]:

- Clase A: Redes que requieren velocidades de datos menor a 10 kbps, utilizadas para transmitir señales de control e interruptores, faros, luces de paro, posición de espejos, sensor de lluvia, control de puertas y ventanas. Algunos ejemplos de protocolos de esta clase son LIN (*Local Interconnect Network*) y TTP/A (*Time Triggered Protocol*).

- Clase B: Las velocidades de transmisión de datos van de los 10 kbps a los 125 kbps, dedicadas al soporte e intercambio de datos entre las ECUs para reducir el número de sensores que comparten información; se utilizan en aplicaciones como el aire acondicionado, tablero de instrumentos, entre otros. El protocolo J1850 y CAN (*Controller Area Network*) de baja velocidad son los principales representantes de esta clase.

- Clase C: Redes que necesitan altas velocidades en tiempo real, 125 kbps a 1 Mbps; sus principales aplicaciones son en el tren motriz del automóvil (motor-transmisión-árbol de transmisión). El protocolo CAN de alta velocidad es usado para estas aplicaciones.

- Clase D: Estas redes necesitan de altas velocidades en la transmisión de datos, mayores a 1 Mbps, sus aplicaciones son orientadas a multimedia como los teléfonos, navegación basada en GPS (*Global Positioning System*). Los protocolos más representativos son FlexRay, TTP/C y MOST (*Media Oriented System Transport*).

Por otro lado, la ISO realiza una clasificación más simple:

- Baja velocidad: Velocidades de transmisión inferiores a 125 kbps.

- Alta velocidad: Velocidades de transmisión superiores a 125 kbps.

1.1.4. Protocolo de comunicaciones CAN

A inicios de 1983 la corporación Robert Bosch GmbH comenzó el desarrollo del protocolo de comunicaciones CAN, pero fue presentado oficialmente ante la SAE en 1986 en Detroit. CAN se encuentra definido en *CAN Specification 2.0* [32] y está especificado en los estándares ISO-11898 [33] e ISO-11519 [34]. CAN es un protocolo de comunicaciones serie que soporta control distribuido en tiempo real con un alto nivel de seguridad y multiplexación. El establecimiento de una red CAN para interconectar los dispositivos electrónicos internos de un vehículo tiene la finalidad de sustituir o disminuir el cableado; por ejemplo, en el modelo 307 de la firma Peugeot el número de cables se ha reducido en un 40 % con la implementación de dos buses CAN respecto al modelo 306 [30]. Las ECUs, sensores, sistemas

antideslizantes y demás se conectan mediante una red CAN a velocidades de transferencia de datos de hasta 1 Mbps.

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; por ejemplo, en el año 2010 uno de cada tres automoviles implementaban una red de comunicación CAN [35]. El protocolo CAN proporciona los siguientes beneficios [32]:

- Priorización de mensajes.
- Garantía en los tiempos de retardo.
- Flexibilidad en la configuración.
- Recepción múltiple con tiempos de sincronización.
- Robustez en los sistemas de datos.
- Detección de error y señalización.
- Retransmisión automática de mensajes corruptos tan pronto el bus se encuentre libre.
- Detección entre errores temporales y permanentes en los nodos y desconexión automática de nodos defectuosos.

1.1.5. Protocolo de comunicaciones CAN FD

La aceptación y la introducción de las comunicaciones seriales ha incrementado las aplicaciones y esto ha llevado al incremento de la demanda de ancho de banda en las comunicaciones basadas en CAN, esto ha causado que los desarrolladores buscaran una alternativa de comunicaciones para las aplicaciones. Estas aplicaciones pueden realizarse cómodamente con el nuevo protocolo de comunicaciones CAN FD (*Controller Area Network with Flexible Data Rate*) que permite velocidades superiores a 1 Mbps [36].

CAN FD comparte la capa física con el protocolo CAN (véase Figura 1.2). Mientras que el formato de transmisión es diferente; hay dos nuevos bits de control en CAN FD, el primero

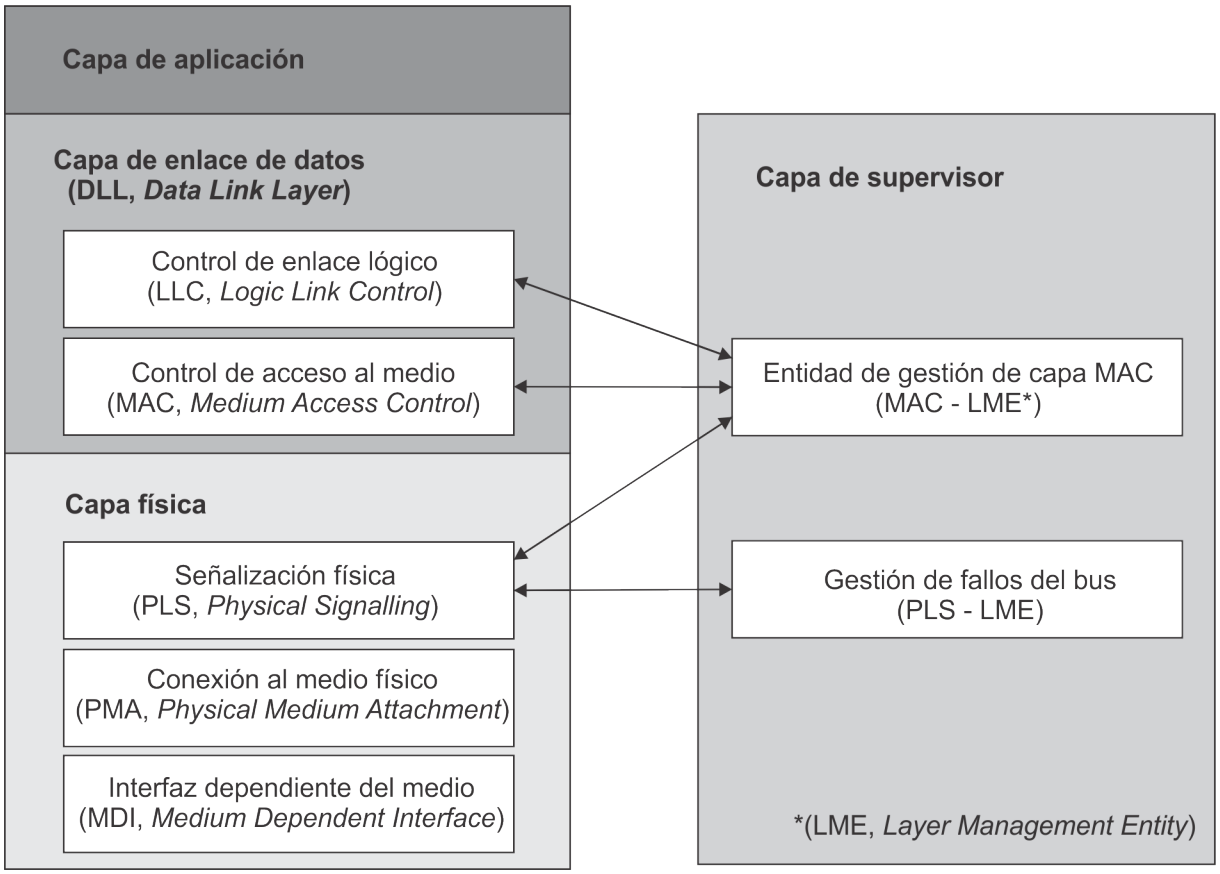

Figura 1.2: Arquitectura del protocolo de comunicaciones CAN [12].

permite el nuevo formato con diferente longitud de dato y el segundo opcionalmente cambia a una velocidad más rápida. Además se introdujo un nuevo control CRC a la seguridad de CAN FD con la misma distancia de Hamming (h=6) que el protocolo CAN, esto significa que 2 diferentes bloques de bits difieren en al menos en 6 posiciones de bits.

El nuevo formato de CAN FD ha sido definido de tal manera que los mensajes con el formato CAN puedan coexistir en el mismo bus. La *CAN Specification 2.0* no se ha modificado.

1.2. Planteamiento del problema

A mediados de la década de los ochenta se dieron a conocer diversos protocolos de FBs que atendían a diferentes soluciones en el área industrial, por ejemplo: CAN, Profibus, Interbus-S, P-Net, LIN, FIP (*Factory Instrumentation Protocol*) entre otros [28, 30, 37].

En la actualidad los sistemas electrónicos de los automóviles contienen ECUs, sensores y actuadores que se encuentran en distintas partes de los vehículos, el uso de estos sistemas se ha ido incrementando debido a que se han ido reemplazando los sistemas hidráulicos y mecánicos por sistemas eléctricos (*x-by-wire*[1]). El gran número de componentes se ha incrementado exponencialmente y dificulta realizar una conexión punto-a-punto debido a la gran cantidad de cable que se necesita para conectar todos los componentes [39]. Debido a esto se han desarrollado diversos protocolos de comunicaciones que permiten interconectar todos los componentes en un mismo bus para que los dispositivos puedan comunicarse entre ellos y así dotarlos de inteligencia. En modelos recientes, los automóviles implementan varios protocolos de comunicaciones como LIN, CAN, FlexRay, Byteflight, TTP/C, y MOST [40].

En los últimos años el protocolo de comunicaciones CAN ha ampliado su gama de aplicaciones a autos [41, 42], sistemas de transporte, técnicas de medición, equipo médico, construcciones automatizadas y a nuevas tecnologías encaminadas al monitoreo remoto de sensores [28].

En el presente trabajo de tesis se propuso estudiar el protocolo de comunicaciones CAN FD mediante SDL para documentar su funcionamiento con fines académicos y de investigación, ya que no existe una especificación formal sobre CAN FD para entender su funcionamiento y poder identificar errores antes de la estandarización.

Con base a lo anterior se formularon las siguientes preguntas de investigación: ¿Es posible que mediante la especificación formal en SDL del protocolo de comunicaciones CAN FD se puedan identificar errores y ambigüedades antes de su estandarización?, ¿Cuál es la compatibilidad del protocolo de comunicaciones CAN FD y CAN?, ¿Cuál es el futuro del protocolo de comunicaciones CAN FD?

[1] Se conoce como *drive-by-wire* o *x-by-wire* a la tecnología en la industria del automóvil que reemplaza los sistemas de control mecánicos tradicionales por sistemas de control electrónico [38].

1.3. Justificación

El protocolo CAN ha sido la referencia para otros protocolos de comunicaciones como DeviceNet, TTCAN (*Time-Triggered CAN*), CANopen, CAN FD y de las implementaciones de la capa de aplicación como CAL, SAE J1939, SDS y CANaerospace; por tal motivo, estudiar el protocolo de comunicaciones CAN FD sirvió para comprender su funcionamiento en diferentes niveles de abstracción con fines académicos y de investigación.

En el campo acádemico con el presente trabajo de tesis se describe el funcionamiento de la arquitectura de protocolos CAN FD, con el fin de que en un corto plazo se pueda enseñar en los cursos de telecomunicaciones que se imparten en las carreras de Ingeniería en Electrónica, Ingeniería en Computación e Ingeniería en Mecatrónica.

En el campo de la investigación, la descripción formal del protocolo de comunicaciones CAN FD permite obtener un modelo sin ambigüedades que sirve para comprender y documentar su funcionamiento, y con ello continuar la línea de investigación en redes industriales en el Instituto de Electrónica y Mecatrónica de esta Universidad.

1.3.1. Pertinencia

Realizar una descripción formal de un protocolo de comunicaciones es pertinente ya que en la UTM se cuenta con licencias del IDE Cinderella SDL. Además, en la UTM se han realizado las especificaciones formales de los protocolos de comunicaciones DNP3 [14] y FlexRay [13] utilizando un lenguaje de descripción formal. Cabe señalar que en el presente trabajo se realiza una especificación completa del protocolo de comunicaciones CAN FD.

1.3.2. Relevancia

El estudio del protocolo de comunicaciones CAN es importante ya que ha sido la base de la creación de nuevas tecnologias aplicadas a los FB, por tal motivo es importante estudiar y especificar formalmente el protocolo de comunicaciones CAN FD.

1.3.3. Motivación

En la Maestría en Electrónica Opción: Sistemas Inteligentes Aplicados se aborda el tema de protocolos de comunicaciones en la materia de programación de interfaces, la cual dio pauta a esta investigación. Los protocolos de comunicaciónes son ampliamente utilizados en sistemas de control inteligente, donde las conexiones punto a punto no son suficientes, tal es el caso de los automóviles, los cuales tienden a ser sistemas inteligentes. Por otro lado el protocolo de comunicaciones CAN FD es reciente y presenta nuevas caracteristicas con relación a su precursor CAN, lo cual hace interesante su estudio.

1.4. Hipótesis

Mediante la especificación formal en SDL del protocolo de comunicaciones CAN FD se podrán identificar errores y ambigüedades antes de su estandarización.

1.5. Objetivos

El objetivo principal de este trabajo de tesis es especificar formalmente el protocolo de comunicaciones CAN FD mediante SDL para la identificación de errores y ambigüedades.

1.5.1. Objetivos Secundarios

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

- Elaborar el estado del arte del protocolo de comunicaciones CAN.
- Estudiar el protocolo de comunicaciones CAN y CAN FD.
- Aprender el uso de la herramienta Cinderella SDL.
- Realizar la descripción estática del protocolo de comunicaciones CAN FD.
- Realizar la descripción dinámica del protocolo de comunicaciones CAN FD.
- Documentar el trabajo de investigación.

1.6. Metas

A continuación se listan las metas del proceso de investigación.

- Investigación documental del protocolo de comunicaciones CAN.
- Registro de la evolución del protocolo de comunicaciones CAN.
- Construcción del estado del arte del protocolo de comunicaciones CAN.
- Estudio de las normas del protocolo de comunicaciones CAN y CAN FD publicadas por la firma Bosch.
- Estudio de los estándares del protocolo de comunicaciones CAN publicadas por ISO-11898 e ISO-11519-2.
- Comprensión y análisis del protocolo de comunicaciones CAN FD.
- Estudio del estándar Z.100 y la bibliografía sobre SDL.
- Conocimiento del funcionamiento del IDE Cinderella SDL.
- Obtención de los requerimientos del sistema CAN FD.
- Especificación de datos para el sistema CAN FD.
- Descripción estática del protocolo de comunicaciones CAN FD.
- Descripción dinámica del protocolo de comunicaciones CAN FD.
- Simulación y validación del sistema CAN FD.
- Análisis y síntesis de los resultados obtenidos.
- Documentación del trabajo de investigación.

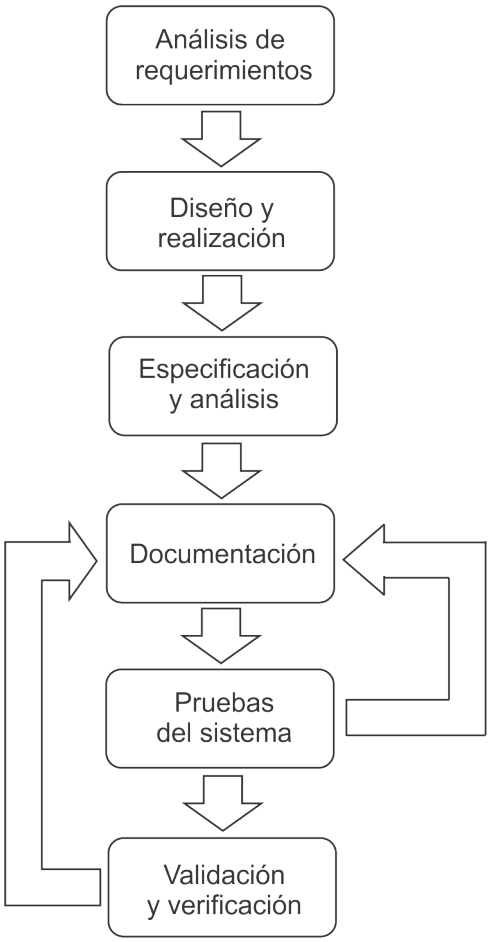

Figura 1.3: Metodología de desarrollo para la descripción formal de un sistema.

1.7. Metodología

La metodología utilizada en la aplicación de los métodos formales a un sistema implica la realización de una serie de actividades tales como: análisis de los requerimientos del sistema, especificación y análisis, diseño y realización, documentación, pruebas y verificación, la cual es descrita por Juan Benigno Nogueira Nine en su tesis doctoral [25].

A continuación se describe la metodología utilizada en realización de la descripción formal (Figura 1.3).

1. Análisis de requerimientos: En esta fase se combinan las actividades de captura de requerimentos, estudio de viabilidad y definición de los requerimientos funcionales.

2. Diseño y realización: El sistema se divide en partes más sencillas para que puedan ser analizadas y probadas de forma independiente.

3. Especificación y análisis: Se produce una especificación detallada de los requerimientos funcionales y no funcionales, a partir de los cuales se debe diseñar y realizar el sistema para que pueda ser verificado.

4. Documentación: La documentación en la descripción formal es de gran valor, ya que de esta forma, tras una lectura de la especificación se puede conocer de forma detallada qué es lo que hace el sistema, sin necesidad de ver líneas de código, esquemas eléctricos, etc.

5. Pruebas de sistema: La especificación obtenida en la fase anterior se utiliza como una caja negra, a la que se realiza un conjunto de pruebas diseñadas para verificar el correcto funcionamiento del sistema. Dichas pruebas deben comprobar todas las situaciones de funcionamiento normal del sistema, y la mayoría de las situaciones anormales, hasta cubrir el grado de confianza deseado.

6. Validación y Verificación: En esta fase se debe comprobar que la especificación del sistema satisface los requerimientos sobre su comportamiento, lo que implica realizar razonamientos lógicos basados en los textos de la especificación no formal. Durante la validación se comprueban propiedades específicas establecidas en los requerimientos, y se investigan las propiedades generales del sistema: consistencia, completitud, etc.

1.8. Estructura del documento

El documento de tesis tiene la estructura siguiente:

El capítulo 1 presenta una introducción al trabajo de tesis.

El capítulo 2 presenta el estado del arte del protocolo de comunicaciones CAN, en el que se incluye una reseña histórica del protocolo, su estandarización, su principal objetivo y cuales son sus principales aplicaciones en la industria.

El capítulo 3 describe la arquitectura del protocolo de comunicaciones CAN FD de acuerdo al modelo de referencia OSI.

El capítulo 4 presenta la especificación estática y dinámica del protocolo de comunicaciones CAN FD.

El capítulo 5 presenta los resultados obtenidos durante el desarrollo de la especificación formal del protocolo de comunicaciones CAN FD utilizando la herramienta Cinderella SDL.

El capítulo 6 plantea las conclusiones obtenidas y los trabajos futuros de investigación

Por último, se presentan las referencias bibliográficas utilizadas en el desarrollo de esta tesis.

Capítulo 2

Estado del arte del protocolo de comunicaciones CAN

La tecnología ha experimentado un rápido desarrollo y mejora en los últimos 100 años, y una de las áreas en donde ha sido más visible es la industria automotriz. Los automóviles diseñados por Henry Ford hoy lucen como artefactos ancestrales comparados con los nuevos modelos de automóviles. Mientras el cambio en las apariencias externas es sorprendente, el contenido electrónico en los automóviles se ha incrementado a un ritmo constante desde 1970, con el comienzo de esta década se introdujo el encendido electrónico y se usaron los primeros procesadores, la tendencia continúo incrementándose a través de las decadas siguientes. La mayor parte de la innovación de un automóvil se ha dado en el sector electrónico; el mercado de componentes electrónicos para el sector automotriz ha tenido un crecimiento aproximado del 7 %, en el año 2011 el costo de la electrónica de los automóviles representa un 40-50 % del costo total de un automóvil [43], mientras que en el año 2002 el costo de la electrónica involucrada en los automóviles era del 23 % de su costo total [35].

La evolución de los microcontroladores dio inicio a la introducción de sistemas con controles distribuidos en los automóviles y esto motivó la necesidad de usar redes de comunicaciones para la comunicación de los mismos. A principio de la década de 1980, la compañía alemana Robert Bosch GmbH empezó a realizar estudios para la aplicación de FB en los automóviles con la finalidad de satisfacer la creciente demanda que se tenía en las

comunicaciones en los automóviles. Como resultado se obtuvo la especificación del protocolo de comunicaciones CAN, del cual a continuación se presenta una reseña histórica y sus principales aplicaciones.

2.1. Reseña histórica del protocolo de comunicaciones CAN

A inicios de 1983 la corporación Robert Bosch GmbH comenzó el desarrollo de un nuevo protocolo de red que aportara una mayor funcionalidad, seguridad, fiabilidad y una mayor eficiencia en el consumo de combustible, basándose en el hecho de que con la reducción del peso del cableado se favorecería este hecho. Los Dr. Uwe Kiencke, Siegfried Dais, Martin Litsch dieron inicio al desarrollo del nuevo sistema; como asistente académico el Dr. Horst Wettstein y como consultor intervino el Dr. Wolfhard Lawrenz, de la Universidad de Ciencias Aplicadas de Brunswick-Wolfenbüttel, Alemania, fue este último quien dio al nuevo protocolo de comunicaciones el nombre de *"Controller Area Network"* (CAN). En la investigación de los mecanismos de detección de errores intervinieron Wolfgang Borst, Wolfgang Botzenhard, Otto Kart, Helmut Schelling y Jan Unruh [12, 44].

Los ingenieros de la empresa Mercedes-Benz pronto se involucraron en las primeras fases de creación del nuevo sistema, y también lo hizo la compañia Intel como potencial vendedor de semiconductores. Desde entonces varias empresas productoras de componentes tales como: Siemens, Infineon, Motorola, Freescale, National Instruments, Texas Instruments, Termic y Atmel comenzaron a interesarse en el bus CAN [44].

En febrero de 1986, Robert Bosch GmbH presentó oficialmente, en Detroit, E.U.A (Estados Unidos de Ámerica), ante miembros exclusivamente de la SAE, el protocolo de comunicaciones CAN. Este año se considera el nacimiento de CAN.

A mediados de 1987, Intel presentó el primer chip de controlador CAN, el 82526. Poco tiempo después, la compañia Phillips dio a conocer el controlador 82C200. Estos dos controladores CAN eran completamente distintos en cuanto a filtros de aceptación y control de mensajes. Intel adoptó el concepto de FullCAN, debido a que requería menos carga de la

CPU del microcontrolador que la implementación BasicCAN elegida por Philips. Pero por otra parte, el dispositivo de FullCAN era limitado con respecto al número de los mensajes que podían ser recibidos. Además, el controlador de BasicCAN requería menos silicio, lo que abarataba aun más su costo. Actualmente, los términos "BasicCAN" y "FullCAN"han quedado obsoletos [28].

En 1991, Robert Bosch GmbH publicó la especificación CAN en su versión 2.0 (*CAN Specification 2.0*), que posteriormente fue sometida a su estandarización internacional, proceso que se vio afectado por disputas políticas con el protocolo francés VAN (*Vehicle Area Network*). En el mismo año, la firma Kvaser dio a conocer el protocolo de capa de aplicación CANKingdom [45].

En marzo de 1992, de acuerdo a las necesidades de comunicación de esa época, empezaron a surgir varios protocolos de comunicaciones que cubrían la capa de aplicación. La mayoría de los pioneros del bus CAN utilizaban un enfoque monolítico, es decir, que las funciones de comunicación, administración de la red y código de aplicación, se realizaban en un mismo módulo de software y no consideraron las desventajas que surgían al desarrollar una solución propia. Lo anterior originó la necesidad de establecer organizaciones encargadas de reglamentar el desarrollo de aplicaciones del bus CAN; la organización CiA (*CAN in Automation*), por ejemplo, está formada por un grupo de fabricantes y usuarios internacionales que proporcionan información técnica, de productos y comercialización para fomentar la utilización del bus CAN [46]. A pocas semanas de su fundación, CiA publicó un artículo técnico en donde recomendaba la utilización de transceptores (*transceivers*) en la capa física del protocolo CAN y posteriormente publicó el protocolo de capa de aplicación CAN (CAL, *CAN Application Layer*). Otra de las tareas de CiA fue organizar el intercambio de información entre los expertos y usuarios de CAN. Por otro lado, Mercedes-Benz fue el primer fabricante que implementó el protocolo CAN en la producción del vehículo Clase S. El sistema estaba compuesto por dos redes CAN, una red de alta velocidad, en la que se comunicaban las ECUs del motor, de la caja de cambios y el tablero de instrumentos; y una red de baja velocidad, para el control del aire acondicionado y de los dispositivos electrónicos internos. El número de redes de ECUs en Mercedes, BMW, Audi, y VW fue de 5 o menos a inicios de 1990 y llegó alrededor de 40 a finales de esa misma década. Los costos de instala-

ción y mantenimiento debido a la complejidad del cableado punto a punto se incrementaron y como resultado CAN fue rápidamente adoptado por la industria automotriz para proveer la solución a estos problemas. La implementación realizada por Mercedes-Benz propició que otros fabricantes de automóviles comenzaran a utilizar redes CAN en sus modelos de lujo, por ejemplo Volvo, Saab, BMW y VW, más tarde se agregaron a la lista Fiat, Renault y PSA.

En noviembre de 1993, el protocolo CAN logró su estandarización internacional bajo la norma ISO 11898 [33], en la que se define una capa física para velocidades de transferencia de datos de hasta 1 Mbps, y en este mismo año se fundó el grupo áleman OSEK (en áleman: *Offene Systeme and deren Schnittstellen für die Elektronik in Kraftfahrzeugen*; en inglés: *Open Systems and their Interfaces for the Electronics in Motor Vehicles*) [47]. En 1994, CiA organizó la primera conferencia internacional de CAN (iCC, *international CAN Conference*); en el mismo año, la firma Allen-Bradley publicó un protocolo de capa de aplicación CAN conocido como DeviceNet, que actualmente se utiliza en los E.U.A.

En 1995, CiA lanzó al mercado CANopen, que al igual que DeviceNet es un protocolo que cubre la capa de aplicación para CAN pero con mayor aceptación y demanda por parte de los fabricantes europeos. En el mismo año se publicó una mejora al estándar ISO 11898 incluyendo las recomendaciones de la especificación CAN 2.O B de Robert Bosch GmbH. En marzo de 1995, la SAE emprendió una investigación para valorar las necesidades comunes que representa la implementación de CAN en los automóviles y para 1996 CAN se implementó en la mayoría de los sistemas de control del motor en vehículos europeos.

En 1999 comenzó el desarrollo de un nuevo protocolo accionado por tiempo basado en CAN, el cual se dio a conocer en el año 2000 con el nombre de TTCAN bajo el estándar ISO 11898-4 [48], en este año se dio una explosión en el uso del protocolo CAN para aplicaciones industriales y en la mayoría de los motores de los automóviles,

Hoy la mayoría de los automóviles fabricados en Europa son equipados con al menos un bus CAN. En los E.U.A. la Agencia de Protección Ambiental *(Enviromental Protection Agency)* autorizó el uso de CAN en las tarjetas de diagnóstico en todos los automóviles y camiones ligeros vendidos en E.U.A desde modelos del año 2008, la conferencia anual sobre producción estima que aproximadamente 65 a 67 millones de automóviles tienen entre diez

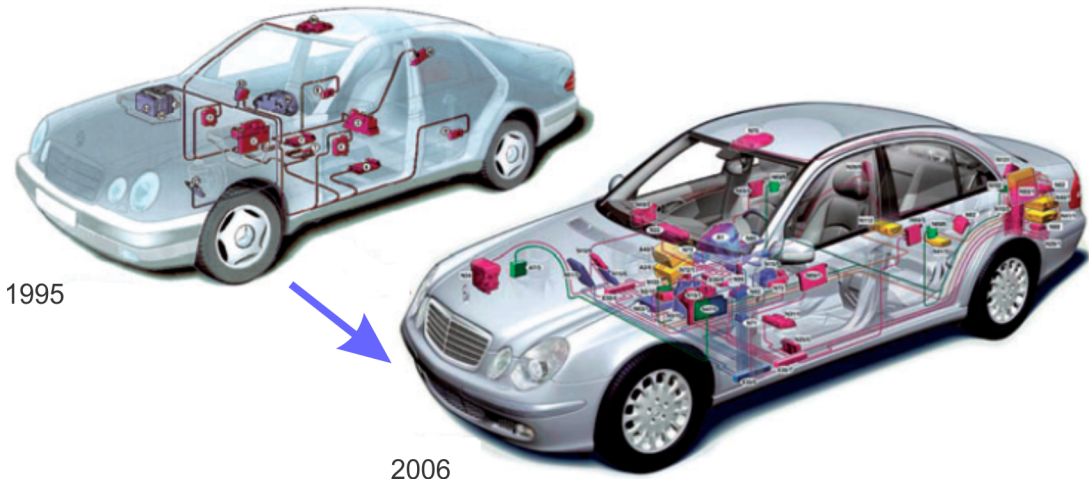

Figura 2.1: ECUs CAN en el automóvil modelo W210 (1995) y W211 (2006) de Mercedes Benz [50].

y quince nodos CAN por vehículo, de tres ECUs que se encontraban en los automóviles en 1989 han llegado a más de setenta en el año 2006 (véase Figura 2.1) y reportes de CiA indican que en el año 2014 un autómovil tienen más de cien ECUs [49, 50, 51].

En los últimos 20 años, el protocolo de comunicaciones CAN ha dominado los sistemas de comunicaciones de los automóviles y ha sido la base del desarrollo de las tecnologías que hoy existen en esta área, asociado con la aceptación e introducción de las comunicaciones seriales, las cuales han incrementado la demanda de ancho de banda en el protocolo de comunicaciones CAN, esto ocasionó que los desarrolladores buscaran una alternativa y el 17 de Abril de 2012 Robert Bosch GmbH publicó la actualización al protocolo CAN FD [36], las aplicaciones sobre CAN FD permiten velocidades de transmisión superiores a 1 Mbps. En la Figura 2.2 se muestra la evolución del protocolo de comununicaciones CAN.

2.2. Estándarización Internacional del protocolo de comunicaciones CAN

Robert Bosch GmbH en 1983 comenzó a desarrollar ECUs junto con un protocolo de comunicaciónes dedicado al sector automotriz. El resultado fue el protocolo de comunicaciones

Figura 2.2: Evolución del protocolo de comunicaciones CAN.

CAN, este protocolo de comunicaciones fue estandarizado en Noviembre de 1993 como *ISO 11899 Road vehicles - Interchange of digital information - Controller area network (CAN) for high-speed communication* [33].

2.2.1. La familia ISO 11898

El estándar describe la arquitectura del protocolo de comunicaciones CAN en términos de las capas propuestas por los modelos ISO/OSI (*Organization for Standardization/Open Systems Interconnection*), las cuales son la capa física: encargada de definir los aspectos del medio físico para la transmisión de datos entre los nodos de una red, la capa de enlace de datos, que se divide en en dos subcapas: control de enlace lógico (LLC, *Logic Link Control*) y control de acceso al medio (MAC, *Medium Access Control*) y la capa de aplicación, que

depende del protocolo de capa superior como se analizó en el apartado anterior. Existen 6 estándares derivados de la primeras especificaciones. Las partes 1 y 2 corresponden a la primera versión del estándar y fueron publicados al mismo tiempo.

- ISO 11898-1 *Road vehicles-Controller area network (CAN)-Part 1: Data link layer and physical signalling*. Especifica la capa de enlace de datos y la señalización física de CAN, y detalla la especificación de la subcapa LLC y MAC [52].

- ISO 11898-2 *Road vehicles-Controller area network (CAN)-Part 2: High-speed medium access unit*. Especifica las unidades de acceso al medio (MAU, *Medium Access Unit*) para velocidades de transmisión de hasta 1 Mbps y algunas características de interfaces dependientes al medio (MDI, *Medium Dependent Interface*), las cuales componen la capa física de CAN [53].

Posteriormente se desarrollaron las siguientes partes:

- ISO 11898-3 *Road vehicles-Controller area network (CAN)-Part 3: Low-speed, fault-tolerant, medium-dependent interface*. Especifica las características de configuración para el intercambio de información digital entre ECUs y vehículos equipados con el bus CAN en velocidades de transmisión arriba de 40 kbps hasta 125 kbps [54].

- ISO 11898-4 *Road vehicles-Controller area network (CAN)-Part 4: Time-triggered communication*. Especifica la sincronización de las tramas coordinadas por las capas LLC y MAC de acuerdo con el ISO 11898-1, para proveer la comunicación programada basada en tiempos [48].

- ISO 11898-5 *Road vehicles-Controller area network (CAN)-Part 5: High-speed medium access unit with low-power mode*. Especifica la capa física del protocolo de comunicaciones CAN para velocidades de transmisión de hasta 1 Mbps para el uso de vehículos equipados con el bus CAN. También describe las funciones de MAU y algunas características de MDI de acuerdo al ISO 8802-2 [55].

- ISO 11898-6 *Road vehicles-Controller area network (CAN)- Part 6: High-speed medium access unit with selective wake-up functionality*. Especifica la capa física del bus CAN

para velocidades de transmisión de hasta 1 Mbps y describe las funciones de MAU. Representa una extensión del ISO 11898 2 e ISO 11898 5 especificando el mecanismo selectivo para despertar una ECU usando una trama CAN configurable [56].

Además de la familia ISO 11898, hay otros estándares del protocolo de comunicaciones CAN en otros sectores de estandarización. Ejemplos de éstos son los creados por la SAE en el estándar J2284-1 bis-3, que especifica la capa física del bus CAN para diferentes velocidades de transmisión de 125 kbps (J2284-1), 250 kbps (J2284-2) y 500 kbps (J2284-3).

2.3. Aplicaciones del protocolo de comunicaciones CAN

El protocolo de comunicaiones CAN se utiliza en diversas áreas de aplicación industrial debido principalmente a su bajo costo, alto rendimiento, proliferación rápida guiada por la industria automotriz y respaldo de un gran número de fabricantes de dispositivos CAN.

CAN fue creado en un principio para uso automotriz, por lo que su aplicación más común es para comunicación de dispositivos electrónicos relacionados con esta industria. Sin embargo, conforme otras industrias han observado las ventajas de CAN, han adoptado al bus para una amplia variedad de aplicaciones.

A continuación se mencionan las principales aplicaciones CAN respaldadas por CiA dentro de los sectores automotriz, automatización industrial, control de maquinaria, marítimos y ferroviarios.

2.3.1. Sector automotriz

En los vehículos modernos la electrónica desempeña un papel importante, como sistemas de control, ayuda de conducción y asocia múltiples accesorios del vehículo. La sofisticación de los vehículos exige la incorporación de sistemas cada vez más inteligentes, que monitoricen constantemente y diagnostiquen, prevengan, detecten, almacenen y controlen los posibles fallos.

CAN es una red duradera y económica que permite a varios dispositivos comunicarse entre sí. Un beneficio es que permite a las ECUs tener una sola interfaz CAN, en lugar de

diferentes entradas analógicas y digitales para cada dispositivo en el sistema. Esto reduce el costo y peso en los automóviles.

CAN fue desarrollado, inicialmente, para aplicaciones en los automóviles y por lo tanto la plataforma del protocolo es resultado de las necesidades existentes en el área automotriz [28].

A nivel global, los fabricantes europeos implementan al menos una red CAN en sus automóviles para el control de motor; los fabricantes americanos utilizan CAN en aplicaciones de control de energía del motor; y los fabricantes del lejano oriente han iniciado el desarrollo de redes basadas en CAN para el diagnóstico de fallas en sus automóviles [45].

2.3.2. Automatización de maquinaria

La maquinaria móvil debe satisfacer los requisitos de seguridad con la finalidad de cumplir con sus directivas de seguridad. Actualmente existe una nueva solución, basada en CANopen para cumplir con las normas de seguridad.

MOBA fundada hace más de 40 años fue una de las primeras compañías en darse cuenta del potencial que ofrecen los sistemas CAN en el mercado de la automatización de maquinaria y ha desarrollado sus diseños mediante esta arquitectura descentralizada de sistema . De este modo, desarrolla y fabrica dispositivos de control electrónico en estrecha colaboración con sus clientes. Esto incluye sensores de nivelación y posicionamiento, regulación de distancias, sistemas de inclinación, interfaces hombre-máquina (HMI, *Human Machine Interface*) y el software para estos dispositivos. Por supuesto, los dispositivos proporcionan conectividad CAN y CANopen. Los sistemas modulares de MOBA se integran en los equipos dotados de esta tecnología, y a su vez, les dota de la posibilidad de ampliación gracias a su capacidad de interconectividad [57].

Por otro lado, la calidad es una prioridad en la construcción de carreteras que incluye no sólo la composición óptima del material, sino también un proceso óptimo durante los trabajos de construcción. Un aspecto que juega un papel importante en toda la cadena del proceso desde la planta de asfalto a la compactación es la temperatura del asfalto. Aplicar el asfalto a una temperatura no adecuada se traduce en costos adicionales de hasta un 46 %

debido a una vida más corta de la carretera. La maquinaria MOBA PAVE-IR basado en CANopen proporciona un control de temperatura durante el extendido de asfalto y documenta el proyecto. Como resultado, los contratistas pueden optimizar sus procesos y verificar la calidad del proceso de pavimentación. El sistema utiliza un escáner de temperatura basado en un microcontrolador a través de toda la anchura de pavimentación de hasta 8 m para medir la temperatura del asfalto. El ancho de la medición se puede ajustar individualmente con un máximo de 31 puntos de medición, el escáner alcanza una precisión de ±2 °C en las temperaturas típicas de asfalto. El perfil de temperatura se visualiza en tiempo real en la pantalla donde el operador puede reaccionar en cualquier momento si se producen irregularidades. Además, el historial se almacena con los datos de posición GPS y se transfiere a una memoria USB. Con el software *Pave Project Manager* el usuario puede evaluar y documentar los datos, como alternativa los datos pueden ser enviados a un servidor a través de la red GSM, donde se puede comunicar en cualquier momento con una aplicación web [58].

2.3.3. Aviación

Muchos aviones implementan redes de comunicaciones basadas en CAN. El protocolo de capa superior CANaerospace se utiliza en varias aeronaves, la mayoría de aplicaciones son: sensores del estado del vuelo, sistemas de navegación, y computadoras de investigación en la cabina de control [59].

La Administración Nacional de la Aeronáutica y del Espacio (NASA, *National Aeronautic and Space Administration*) y el Centro de Investigación Langley instalaron el protocolo CANaerospace/AGATA (*Advanced General Aviation Experiments*) en dos aviones de investigación utilizado por la SATS (*Small Aircraft Transportation System*). La SATS es una asociación entre varias organizaciones de E.U.A que incluyen a la NASA, la Administración Federal de Aviación (FAA, *Federal Aviation Administration*), universidades, funcionarios estatales y locales de aviación. El enfoque inicial de este proyecto fue demostrar que las nuevas capacidades operativas permiten el acceso seguro a prácticamente cualquier pista en los E.U.A. en la mayoría de las condiciones meteorológicas. Estas nuevas capacidades

de operación dependen de la informática, controles de vuelo avanzados y las tecnologías de separación y secuenciación automatizada del tráfico aéreo.

La red CAN de 1 Mbit fue utilizada como red troncal para sensores de estado de vuelo, sistemas de navegación y varias computadoras de investigación de alta resolución instalados en la cabina. Las computadoras de investigación, dotadas de la interfaz CANaerospace sirven como puertas de enlace de datos (*gateways*) entre una conexión a Internet en el aire y el bus.

Para fines de investigación, los dos aviones Langley estaban equipado con CAIS (*Common Airborne Instrumentation System*), un sistema de adquisición de datos de vuelo que registra todos los datos del estado de vuelo pertinentes, incluida la información CANaerospace. Junto con una instalación de grabación de voz/video, la gran cantidad de datos generados por los vuelos de prueba permitió un análisis de la asistencia proporcionada por la tripulación de las interfaces de la cabina. La interfaz CANaerospace para el sistema CAIS fue desarrollado por *Teletronics Technology Corp* [60] de Bristol, Inglaterra y *Stock Flight Systems* [61] de Farchach, Alemania. CAIS utiliza diversos programas de varios centros de investigación de la NASA y la Fuerza Aérea de los E.U.A.

2.3.4. Marítimas

Uno de los medios de transporte más antiguos de la humanidad para cruzar ríos y océanos son los buques y embarcaciones. La demanda para el transporte de bienes y personas por vía marítima continuamente va en aumento y su tecnología ha evolucionado hasta el grado de reinventarse a sí mismos. La industria marina ha buscado aprovechar el desarrollo técnico derivado de otros sectores industriales, por lo tanto todas las tecnologías que desencadenaron revoluciones industriales también se encuentran a bordo de los buques modernos [62].

En este sentido, existen distintas aplicaciones marítimas del protocolo de comunicaciones CAN. A continuación se presenta un ejemplo relacionado con objetivos marítimos no tripulados por la Marina de los E.U.A.

La Marina de los E.U.A. ha desarrollado una arquitectura electrónica distribuida denominada SeaCAN. La Figura 2.3 muestra una arquitectura SeaCAN para un 7m (módelo

del barco) controlado remotamente. El sistema implementa, un piloto automático basado en un control de realimentación a través de la red, evaluación de nodos Rudder, receptor GPS, inclinación/balanceo/rubro, comando/control y dos nodos de aceleración del motor. El sistema SeaCAN utiliza una serie de placas de CPU con microcontroladores Infineon C167 conectados juntos a través de un bus CAN funcionando a 125 kbps. El sistema SeaCAN utiliza un sistema operativo construido con el bus CAN y basado en el protocolo CANKingdom de capa de aplicación. El sistema operativo contiene un programador para tareas, que se sincronizan con una capa de aplicación mediante un reloj global. Por lo tanto, es posible la coordinación del comportamiento entre dos nodos sin ninguna comunicación de red adicional. Esta funcionalidad se utiliza para la generación del muestreo periódico utilizado en el control del servo del timón. En ese caso, el nodo sensor del timón toma muestras al ángulo del timón a una velocidad de 10 Hz. La recepción de los mensajes de datos desde el nodo de detección al nodo controlador del accionador del timón provoca la rutina del ciclo de control. Debido a que estos mensajes tienen alta prioridad en el bus y que están sincronizados a una velocidad conocida, la variabilidad del bucle de control y el retraso de datos son muy bajos, lo que permite un ciclo de control de buen rendimiento [45].

2.3.5. Ferroviarias

El constante crecimiento de la demanda de los servicios de transporte y el aumento de los requisitos de velocidad y comodidad para viajar implica la necesidad de mejorar la calidad de las vías del tren. Para satisfacer estas demandas es necesario diseñar maquinaria que sea capaz de realizar sus tareas con la máxima eficiencia. En aplicaciones ferroviarias se utiliza CAN en vagones, tranvías, subterráneos, trenes ligeros, y trenes de distancias largas para comunicar las unidades de las puertas o controladores de freno, unidades de conteo de pasajeros, y más [63].

Un ejemplo de una máquina que utiliza una red CAN es la *breakstone cleaner OT84* (véase Figura 2.4) diseñada y construida por ZPS (*Stargard Szczecinski*) de Polonia, en colaboración con *Bosch Rexroth* (Polonia). El *breakstone cleaner* es un vehículo (máquina

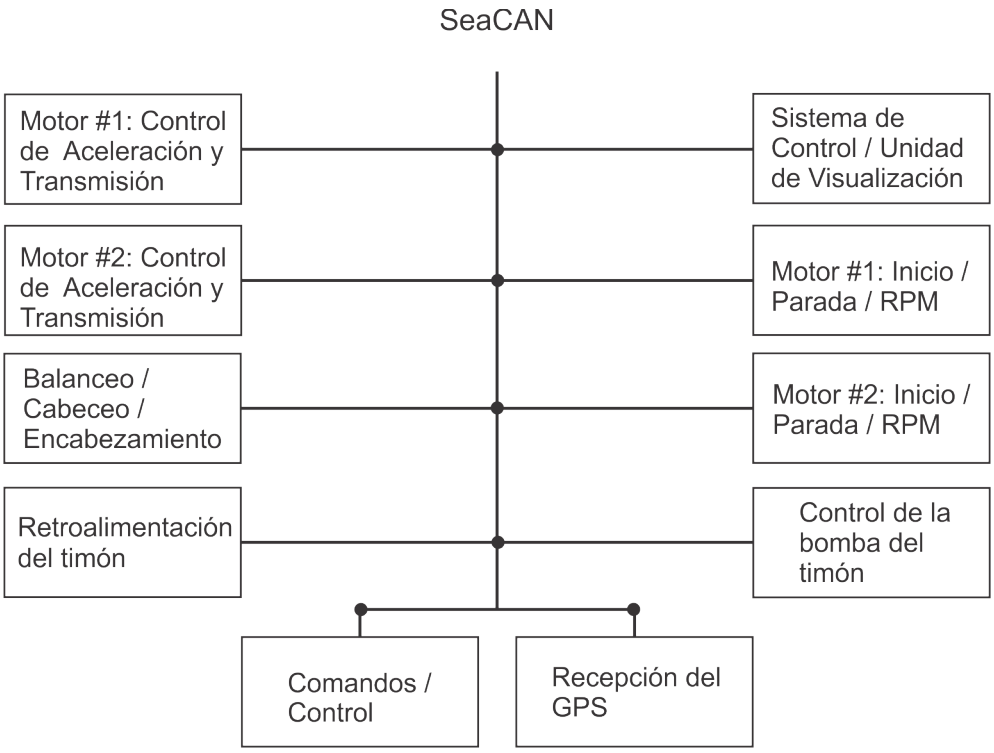

Figura 2.3: Arquitectura del control distribuido de un bote naval [45].

de ferrocarril) que con accionamientos hidráulicos realiza operaciones de limpieza en vías ferroviarias [64].

El sistema de control se realiza mediante herramientas electrónicas inteligentes de *Bosch Rexroth* , que mediante la utilización de software permite la ejecución suave y fiable de las operaciones que la máquina realiza. Cada uno de los subconjuntos tiene su controlador separado que se encarga de los componentes asociados del sistema de control (sensores, actuadores, señales de salida y válvulas hidráulicas). La máquina comprende cuatro controladores del tipo RC8-8/22 (para las unidades de tracción, transportadores, cadena y pantalla), así como el controlador RC36-20/30 que actúa como la unidad principal de la red para controlar y supervisar el funcionamiento de los otros controladores esclavos, también controla las funciones hidráulicas auxiliares tales como: aire acondicionado, iluminación y otros.

Un componente crucial en el sistema de control es la red CAN que permite la transmisión de información digital entre los controladores individuales, así como entre los controladores y los dispositivos periféricos. El *breakstone cleaner OT84* incorpora 69 dispositivos con in-

Figura 2.4: Breakstone Cleaner OT84 [64].

terfaz CAN, incluyendo controladores, 58 módulos de entrada/salida y cinco pantallas. La aplicación del sistema CAN fue implementada por la necesidad de cumplir con los requisitos de los fabricantes con el fin de reducir las interconexiones de los cables entre gran número de dispositivos de medición y actuadores. El sistema de control basado en la arquitectura CAN resultó ser más compacto en tamaño y transparente en el diseño, además se eliminaron las interferencias en líneas eléctricas. El uso de la red CAN hace al sistema escalable, ya que se puede mejorar con componentes adicionales, tales como: sensores, actuadores, registradores, módulos de comunicación y unidades de GSM/GPS. Con el fin de manejar todos los dispositivos del sistema, siete rutas de comunicación separadas fueron configuradas de tal manera que si hay un fallo al menos una seguirá conectada [64].

2.3.6. Aeroespaciales

El protocolo de comunicaciones CAN se encuentra en muchas aplicaciones aeroespaciales, que van desde el análisis de datos en vuelos a los sistemas de control del motor de la nave, sistemas de combustible, bombas y actuadores lineales. A continuación se muestra un ejemplo del uso del protocolo de comunicaciones CAN en la sonda SMART-1 (*Small Mis-*

sions for Advanced Research in Technology), la primera del programa SMART de la Agencia Espacial Europea (ESA, *European Space Agency*).

El satélite SMART tiene 7 cargas útiles (*payload* [2]) de diferentes instituciones, universidades y empresas privadas, cada una con su propia interfaz mecánica, térmica, eléctrica y manejo de datos. Las siete cargas útiles a bordo de la nave espacial SMART- 1 están diseñadas para observaciones científicas. Las cargas EPDP y SPEDE, monitorean el desempeño y los posibles efectos secundarios de la propulsión eléctrica, como la erosión de la superficie y la contaminación por polvo. El transpondedor de banda KaTE X/Ka proporciona mediciones doppler muy precisas, estos datos se utilizan para determinar el rendimiento del empuje del motor midiendo el cambio en la velocidad de la nave espacial y también para llevar a cabo investigaciones sobre la libración lunar. Las cargas útiles SIR y D-CIXS son espectrómetros de rayos X en el infrarrojo cercano y participan en un mapeo mineralógico de la luna. La carga útil AMIE CCD es una cámara que proporciona imágenes a lo largo de la misión y se utiliza en conjunción con la SIR y D-CIXS para el mapeo de superficie de la luna. La carga útil XSM monitorea la actividad solar y ayuda en la calibración de los datos de D-CIXS [65].

Las cargas útiles interconectan sistemas de apoyo de la nave espacial en las áreas mecánica, eléctrica, térmica y de manipulación de datos. Las interfaces eléctricas, en particular, consideran la interfaz de bus de datos como un punto particularmente crítico. La nave espacial SMART- 1 utiliza un bus de datos de tipo CAN para todas las comunicaciones con el fin de asegurar la funcionalidad de las cargas útiles. La Corporación Sueca del Espacio (*Swedish Space Corporation*) proporciona el diagrama de detallado de la interfaz CAN para cada equipo, así como todos los componentes, incluyendo un chip FPGA (*Field Programmable Gate Array*) programado que contiene el protocolo de transferencia de datos.

Antes del traslado de los datos a la memoria de la nave espacial, el controlador del protocolo subdivide cada paquete en una serie de mensajes de 8 bytes que se envían con un intervalo de tiempo fijo. El extremo receptor devuelve el mensaje antes de guardarlo en la memoria de la nave espacial. Dado que las cargas útiles producen diferentes cantidades de datos por sesión de observación, el tiempo de intervalo de mensaje se ajusta individualmente

[2]Es la razón de la misión del satélite, por ejemplo; satélite de observación, radar de rayos X, para las telecomunicaciones es el transpondedor.

para hacer frente a la velocidad de datos requerida. La Tabla 2.1 muestra las velocidades de transmisión de las cargas útiles [66].

Carga útil	**Velocidad de transmisión [kbps]**
AMIE	70
SIR	45
D-CIXS	75
KATE	4
EPDP	1
SPEDE	0.24

Tabla 2.1: Velocidades de transmisión que requiere cada carga útil.

Otro ejemplo del uso del protocolo de comunicaciones CAN son los satélites experimentales Giove que forman parte del programa de navegación por satélite Galileo de la Unión Europea y la ESA. La plataforma del sistema de control electrónico se basa en un hardware dual redundante que incluye una red CAN. Los dispositivos típicos como sistema de potencia y el controlador de propulsión son conectados por una red CAN. Las redes CAN vinculan cada microcontrolador OBC386 al sensor de datos (por ejemplo, el sol, giroscopio y la tierra). La computadora controla, vía CAN, los actuadores (por ejemplo *magnetorquer*, rueda de propulsión). La ESA ha publicado internamente una práctica recomendada para CANopen con el fin de lograr un mayor grado de funcionalidad *plug-and-play* [67].

2.3.7. Equipo médico

Dado que los dispositivos médicos se vuelven más sofisticados, los fabricantes ya no desarrollan y fabrican todos los elementos de un sistema, para reducir costos de desarrollo y cumplir con los requisitos, compran productos prediseñados de aplicación específica para integrar en sus dispositivos médicos. Como consecuencia, las normas de interfaz abiertas son importantes para los fabricantes de dispositivos médicos, no sólo porque simplifican la integración de productos de terceros, sino también porque permiten la sustitución de un producto con una alternativa de otro fabricante. El protocolo de capa superior CANopen

es un facilitador para la normalización de las interfaces de los componentes y se utilizan en muchos campos de la medicina y el cuidado de la salud, por ejemplo en equipo de análisis de laboratorio para automatizar sus procedimientos de prueba y mantener su competitividad, reducir los costos, mientras que las compañías farmacéuticas y de biotecnología lo usan para acelerar sus ciclos de desarrollo [68].

Durante más de 15 años, *Phillips Medical Systems* ha utilizado CAN y CANopen como la tecnología de comunicación de datos en las interfaces de usuario, detección de imágenes, generación de rayos X, la colimación[3] de rayos X y el control de los elementos mecánicos móviles del sistema de rayos X. La organización CiA tiene el perfil 412 destinado a aplicaciones médicas, el perfil CiA 412-2 es el perfil de dispositivo para colimadores de rayos X automáticos dirigidos al mercado de fabricantes de equipos originales (OEM, *Original Equipment Manufacturer*) desarrollado conjuntamente por *GE Healthcare, Phillips Medical Systems y Siemens Medical Solutions*. La tarea principal de un colimador de rayos X es limitar el haz de rayos X a un formato definido. El perfil de dispositivo soporta varias funciones para enfocar el haz de rayos X, tales como la colimación rectangular o circular [69].

Actualemte se están desarrollando los perfiles CiA 412-3 para generadores de rayos X, CiA 412-4 para mesas de rayos X, CiA 412-5 para *stands* de rayos X.

CiA 412-6 es el perfil del dispositivo para los sistemas de medición de dosis, un sistema de medición de dosis se utiliza para medir la dosis de rayos X en una máquina de rayos X. La especificación describe los dispositivos que reciben la distancia entre el punto focal y la tabla de referencia de pacientes de la red CANopen. También describe los dispositivos que calculan la distancia entre el generador del rayo X y la mesa de referencia del paciente. El perfil proporciona objetos de datos para describir la medición del tiempo de irradiación y la dosis de medición que influyen en los valores, tales como temperatura de la cámara y la presión de aire. El perfil de aplicación define cómo los valores de los campos (valores medidos reales) pueden ser convertidos a los valores deseados del proceso (valor medido deseado). Además, el perfil de dispositivo ofrece la posibilidad de volver a ajustar el valor de proceso a través de la red CANopen.

[3] De acuerdo a la Real Academia de Lengua Española, un colimador es un sistema que a partir de un haz (de luz, de electrones, etc.) divergente se obtiene un haz paralelo.

Para estandarizar un perfil de aplicación CANopen en la automatización del laboratorio, la CIA ha establecido un grupo de interés especial (SIG). El SIG se inició con el desarrollo del perfil dilutor CiA 434-2.

Capítulo 3

Protocolo de comunicaciones CAN FD

El protocolo de comunicaciones CAN ha existido desde la década de 1980 y se ha convertido en el más utilizado para la comunicación entre sensores, actuadores y ECUs en los vehículos. En la actualidad el número de sistemas y dispositivos embebidos que requieren transmitir datos está en constante aumento debido a la creciente demanda de los consumidores y a las exigencias legislativas respecto a la seguridad, lo cual ha generado la necesidad de transmitir más datos a mayores velocidades, para cumplir con los requerimientos funcionales que demandan los vehículos.

Para satisfacer las necesidades de comunicaciones que exigen los vehículos modernos, en marzo de 2012, Robert Bosch GmbH presentó el protocolo de comunicaciones CAN FD, el cual soporta control distribuido en tiempo real con alto nivel de seguridad y complementa las aplicaciones CAN que requieren mayores velocidades de comunicación como son: ECUs de motor, sensores, sistemas anti-derrape, ventanas eléctricas, faros, entre otros. Los sistemas y dispositivos que implementan el protocolo de comunicaciones CAN FD pueden coexistir en una red de comunicaciones CAN y combinan las características del bus[4] CAN permitiendo velocidades de transferencia de datos de hasta 15 Mbps, lo cual lo ha posicionado entre una red CAN de alta velocidad (1 Mbps) y una red FlexRay (10 Mbps). En aplicaciones automotrices, las tarjetas CAN FD tienen un promedio de velocidad de transmisión de datos de 2.5 Mbps con los transmisores actuales [70].

[4]Medio físico para la transmisión de datos

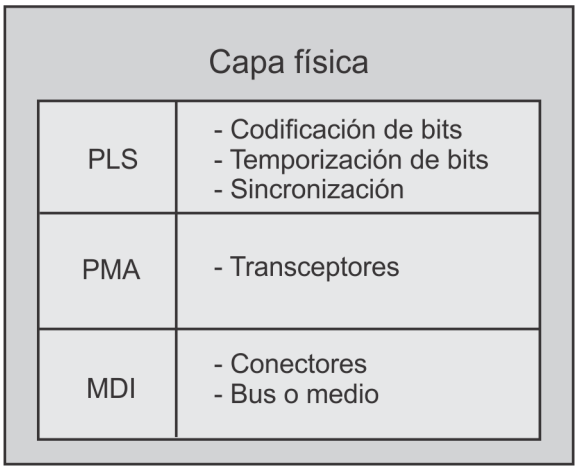

Figura 3.1: Arquitectura de la capa física del protocolo de comunicaciones CAN [44].

La arquitectura de protocolos CAN FD es la misma que la del protocolo del comunicaciones CAN de acuerdo al modelo de referencia OSI (*Open Systems Interconnection*), la cual 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 la Figura 1.2).

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 [28]. 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.

Las especificaciones de los protocolos de comunicaciones CAN [32] y CAN FD [36] no definen la capa física, ya que el medio físico (eléctrico u óptico) y los niveles de señal se pueden configurar para cualquier aplicación en específico. Los estándares ISO 11898 [33] e ISO 11519 [34] establecen las características que deben cumplir las aplicaciones para la transferencia en alta y baja velocidad. Las características definidas para la capa física se deben implementar en todos los nodos que se encuentren conectados a la red CAN. La capa física se divide en tres subcapas (véase Figura 3.1): señalización física (PLS, *Physical Signalling*), medio de

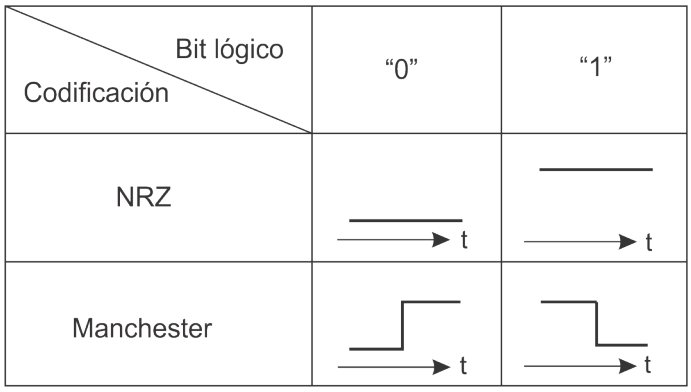

Figura 3.2: Representación de bit en código NRZ y Manchester [71].

acoplamiento físico (PMA, *Physical Medium Attachment*) e interfaz dependiente del medio (MDI, *Medium Dependent Interface*).

3.1.1. Subcapa de señalización física

La subcapa PLS define las funciones relacionadas con: la codificación/decodificación, tiempo y sincronización de los bits; y está implementada en los controladores (*drivers*) de los protocolos CAN y CAN FD [71].

3.1.1.1. Codificación y decodificación de bits

Las tramas de los protocolos de comunicaciones CAN y CAN FD se codifican de acuerdo al método NRZ (*Non Return to Zero*), el cual establece que durante todo el tiempo de generación de bit se genera un nivel de señal constante que puede ser dominante (d) o recesivo (r). Se define como nivel dominante al “0” lógico y como nivel recesivo al “1” lógico. Una ventaja de este 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 (véase Figura 3.2).

Otra de las características de la codificación NRZ es que al mantener un nivel constante de bit no contiene información acerca del reloj de bit, sin embargo esto puede ocasionar problemas con la sincronización para grandes longitudes de datos, además de acarrear bits erróneos debido a que la codificación NRZ no proporciona flancos que puedan utilizarse en

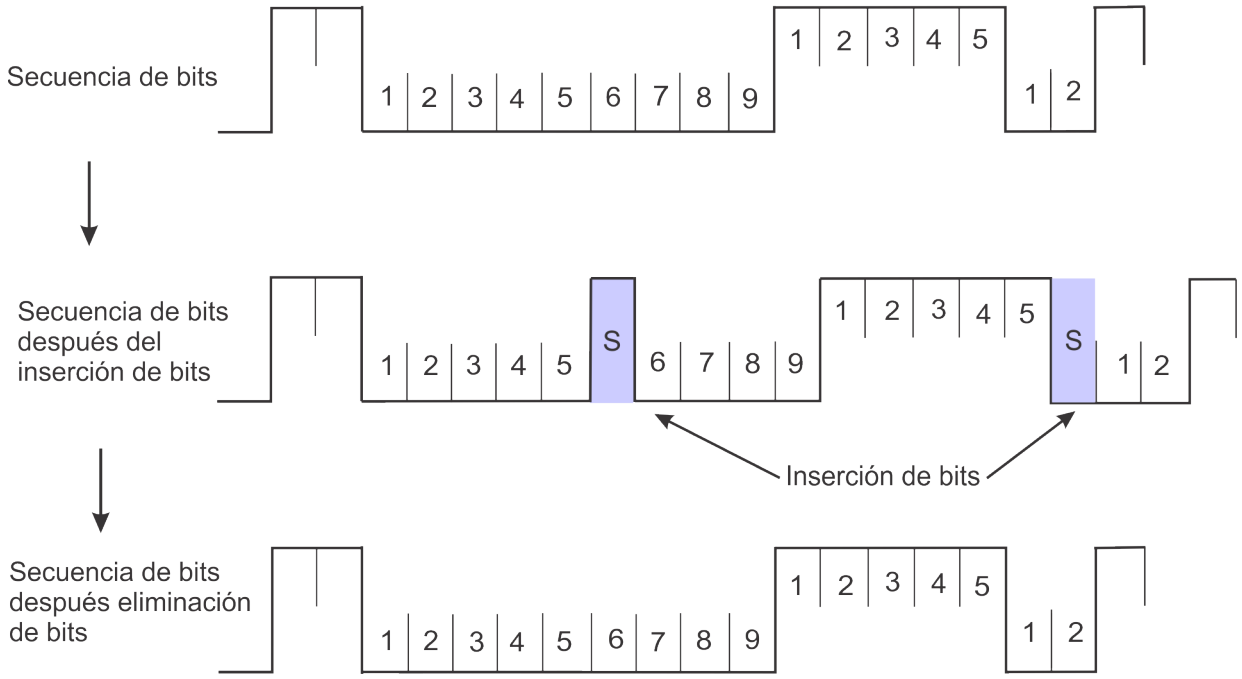

Figura 3.3: Ejemplo del procedimiento de inserción de bit [44].

la sincronización. Para evitar los problemas con la sincronización 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 3.3.

El procedimiento de inserción de bit se aplica a los campos: inicio de trama, campo de arbitraje, campo de control, datos y código de redundancia cíclica; y los campos restantes, que son: delimitador de verificación por redundancia cíclica (CRC, *Cyclic Redundancy Check*), campo de aceptación, fin de trama, tramas de error o de sobre flujo tienen un formato fijo y se transmiten sin emplear el procedimiento de inserción de bit.

3.1.1.2. Temporización de bits

El protocolo de comunicaciones CAN soporta velocidades de transferencia de datos que van desde 1 kbps hasta un 1 Mbps. Cada nodo en una red CAN opera con un reloj individual, generado por un oscilador de cuarzo. Una característica importante del protocolo CAN es su

flexibilidad para determinar los parámetros de velocidad de transferencia, punto de muestreo de bit y número de muestras realizadas en un periodo de bit.

El diseño de una red CAN debe considerar los siguientes conceptos [32, 34, 36, 52]:

- Tiempo de bit (t_B): se define como el tiempo de duración de un bit.
- Velocidad de transferencia nominal (f_B): es el número de bits por segundo transmitidos sin resincronización por un transmisor ideal.
- Tiempo de bit nominal: está compuesto de un número específico de pulsos de ciclo del sistema, basado en que cada nodo tiene un reloj interno. Se calcula mediante la ecuación 3.1.

$$Tiempo\ de\ bit\ nominal = \frac{1}{f_B} \tag{3.1}$$

- *time quantum* (t_q): El parámetro t_q es la unidad básica de tiempo más pequeña utilizada por un nodo CAN. La longitud de los segmentos de tiempo en un intervalo de bit está definida por múltiplos enteros de t_q derivada del periodo del oscilador t_{CLK}. También existe un preescalador programable, con valores enteros y su rango es de 1 a 32 t_q (véase Figura 3.4).

El protocolo de comunicaciones CAN FD define dos tiempos de bit, el primero para la fase de arbitraje (*arbitration phase*) con un tiempo de bit más grande y el segundo para la fase de datos (*data phase*) con un tiempo de bit menor. La definición del primer tiempo de bit es la misma que en el protocolo de comunicaciones CAN, mientras que la definición del segundo tiempo de bit requiere de una configuración diferente. El protocolo de comunicaciones CAN FD define dos registros de configuración para definir los segmentos de tiempo de las dos velocidades de transferencia de bit. Los tiempos de bit para CAN y CAN FD se dividen en cuatro segmentos de tiempo no traslapados; de acuerdo a la Figura 3.5 estos segmentos son:

- Segmento de sincronización (SYNC_SEG, *Synchronizacion Segment*): se utiliza para sincronizar varias ECUs dentro del bus mediante un flanco dentro del mismo segmento.

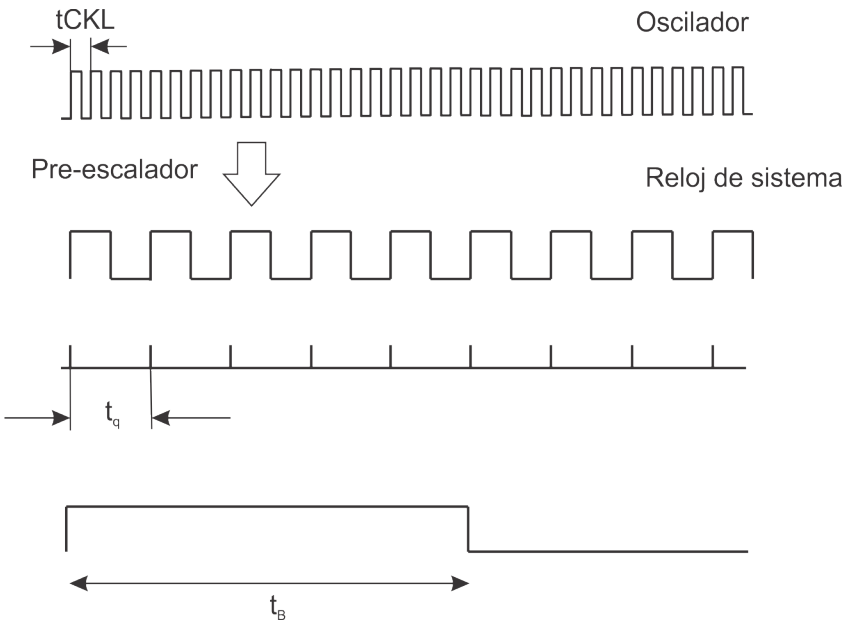

Figura 3.4: Principio de derivación del periodo de bit [12].

- Segmento de tiempo de propagación (PROP_SEG, *Propagation Delay Segment*): esta parte del tiempo de bit se utiliza para compensar los tiempos de retardos físicos dentro de la red. Estos retardos consisten en la propagación de la señal en el bus y por los retardos internos de las ECUs.

- Segmento de memoria temporal de fase 1 (PHASE_SEG1, *Phase Buffer Segment 1*): este segmento se utiliza para compensar variaciones de tiempo entre ECUs. Durante la resincronización se puede ajustar e incrementar la longitud de este segmento.

- Segmento de memoria temporal de fase 2 (PHASE_SEG2, *Phase Buffer Segment 2*): este segmento se utiliza para compensar variaciones de tiempo entre ECUs. Durante la resincronización se puede ajustar e incrementar la longitud de este segmento.

Se deben considerar los siguientes conceptos en los requerimientos de tiempo de bit:

- Punto de muestreo: es el instante de tiempo en el que se lee el nivel del bus y se interpreta su valor del bit respectivo. Se encuentra localizado al final del segmento PHASE_SEG1.

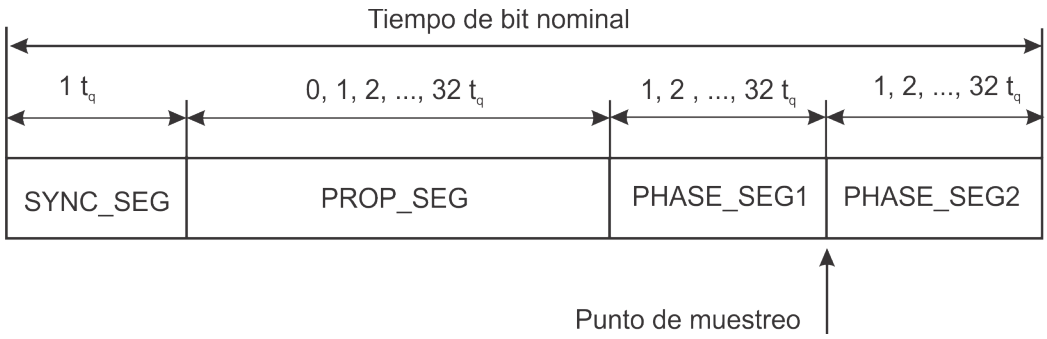

Figura 3.5: Segmentos del tiempo de bit [36].

- Tiempo de procesamiento de la información: es el periodo de tiempo que comienza con el punto de muestreo y se utiliza para calcular el nivel de bit subsecuente. Esta longitud está determinada en la implementación del controlador CAN.

El valor mínimo t_q puede tener una longitud de acuerdo a la ecuación 3.2:

$$t_q(n) = m(n) * valor\ minimo\ t_q \tag{3.2}$$

donde:
$m(n)$ es el valor del preescalador.

En el protocolo de comunicaciones CAN FD define dos valores para el preescalador, $m(N)$ para el tiempo de bit nominal y $m(D)$ para el tiempo de bit de datos, uno para cada velocidad de transferencia de bit que producen dos diferentes longitudes de t_q, los cuales deben ser programados en el rango de 8 a 25.

La longitud del segmento de tiempo para las velocidades de transferencia de bit es:

- SYNC_SEG(N), es de una longitud de 1 t_q.
- PROP_SEG(N), es programable para tener una longitud entre 1, 2, ..., 32 o más t_q.
- PHASE_SEG1(N), es programable para tener una longitud entre 1, 2, ..., 32 o más t_q.
- PHASE_SEG2(N), es la longitud máxima de la PHASE_SEG1(N) y del tiempo de procesamiento de la información.

- El tiempo de procesamiento de la información tiene una longitud menor o igual a 1 $t_q(N)$.

La primera parte de una trama CAN FD, hasta el bit de cambio de velocidad (BRS, *Bit Rate Switch*), se transmite a la velocidad de bit nominal. Se modifica la velocidad de bit si el bit BRS es recesivo, hasta el delimitador de CRC o hasta que el controlador CAN FD detecte una condición de error que resulte en el inicio de una trama de error (*error frame*). Las tramas de error y sobrecarga en CAN FD, así como ACK, EOF y todas las tramas en el formato CAN son transmitidas a la velocidad de transferencia nominal f_B.

La longitud del segmento de tiempo para las velocidades de transferencia de datos es:

- SYNC_SEG(D), es de una longitud de 1 t_q.
- PROP_SEG(D), es programable para tener una longitud entre 0, 1, 2, ..., 8 o más t_q.
- PHASE_SEG1(D), es programable para tener una longitud entre 1, 2, ..., 8 o más t_q.
- PHASE_SEG2(D), es la longitud máxima de la PHASE_SEG1(D) y del tiempo de procesamiento de la información.
- El tiempo de procesamiento de la información tiene una longitud menor o igual a 1 $t_q(D)$.

La posición del punto de muestreo puede diferir en las dos configuraciones de bit de sincronización, y se puede reducir la longitud de PROP_SEG en la velocidad de transferencia de datos. Cuando se modifica la velocidad de bit en el bit BRS o en el bit del delimitador CRC, ésta debe ser cambiada inmediatamente después del punto de muestreo. La suma de las longitudes de estos dos bits debe ser la suma de un bit del tiempo de transferencia nominal y un bit de la velocidad de transferencia de datos. Cuando se cambia el bit de velocidad, es porque se detecta una condición de error y puede ser movido el cambio de tiempo después del punto de muestreo por la longitud del tiempo de procesamiento de la información.

3.1.1.3. Sincronización de bits

El protocolo de comunicaciones CAN usa una transmisión de datos síncrona, a diferencia de una transmisión asíncrona como lo hacen los FBs comunes, lo cual mejora la capacidad de transmisión, pero requiere de métodos de transmisión más sofisticados de sincronización 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 es necesario que el receptor se resincronice continuamente [71].

En una red CAN, cada ECU tiene un oscilador interno que opera continuamente, por lo que todas las ECUs conectadas no necesariamente tienen el mismo valor. Debido a esto pueden ocurrir desplazamientos de fase en los diferentes nodos. Otra característica importante es que los protocolos de comunicaciones CAN y CAN FD utilizan una codificación NRZ que dificulta la recuperación de un bit y el siguiente bit cuando la estructura está ausente, por lo tanto, mientras reciben una trama CAN los controladores CAN proporcionan un mecanismo de sincronización y recuperación para compensar los desplazamientos y cuando está ausente la estructura.

Antes de empezar a analizar las formas de sincronización descritas por el protocolo CAN FD, es necesario definir los siguientes conceptos:

- Error de fase del flanco (*phase error of an edge*): el error de fase del flanco (e) está dado por la posición del flanco respecto al segmento de sincronización y se mide en unidades t_q.

  El signo del error de fase se define de la siguiente manera:

  - $e = 0$, si el flanco se detecta dentro del segmento SYNC_SEG.
  - e > 0, si el flanco se detecta entre SYNC_SEG y el punto de muestreo.
  - e < 0, si el flanco se detecta entre el punto de muestro y los bits siguientes de SYNC_SEG.

- RJW (*Resynchronization Jump Width*): es el valor programado de unidades t_q que se suma a la longitud de PHASE_SEG1 o se reduce de la longitud de PHASE_SEG2.

La cantidad sumada o reducida del segmento de memoria temporal tiene un límite superior dado por RJW. RJW(N) debe ser programada entre 1 y el mínimo de 16 y PHASE_SEG1(N) (min(16, PHASE_SEG1(N))), mientras que RJW(D) debe ser programada entre 1 y el mínimo de 4 y PHASE_SEG1(D) (min(4, PHASE_SEG1(N))).

El protocolo de comunicaciones CAN FD define dos tipos de sincronización [36]:

- 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 (*resynchronization*): cuando la magnitud de un error de fase del flanco que generó la resincronización es menor o igual al valor programado de RJW, el efecto de la resincronización es el mismo que el de una sincronización; y cuando la magnitud del error de fase es mayor que el valor de RJW:
  - si el error de fase es positivo: el segmento de fase 1 se prolonga por una cantidad igual al valor de RJW;
  - si el error de fase es negativo: el segmento de fase 2 se reduce por una cantidad igual al valor de RJW.

La sincronización al comienzo de trama y la resincronización son dos formas de sincronización y siguen las siguientes reglas [36]:

1. Se permite solamente una sincronización entre dos puntos de muestreo.
2. Se debe de usar un flanco para la sincronización sólo si el valor es detectado antes del punto de muestreo (antes de leer el valor del bus) y difiere del valor del bus inmediatamente después del flanco.
3. Se realiza una sincronización siempre que haya un cambio de flanco de recesivo a dominante cuando el bus esté desocupado (*bus idle*), o la transmisión sea suspendida,

y cuando el cambio de flanco se realice en el segundo o tercer bit del intervalo. La sincronización también actúa cuando existe un flanco de recesivo a dominante cuando el bit EDL (*Extended Data Length*) cambia a r0 en la trama CAN FD.

4. Todos los flancos recesivos a dominantes que satisfagan las reglas 1 y 2 se utilizan para una resincronización, a excepción de que un nodo que se encuentre transmitiendo un bit dominante no puede realizar resincronización como resultado de una flanco recesivo a dominante con un error de fase positivo.

5. Un transmisor no puede realizar la resincronización mientras está transmitiendo la fase de datos (*data-phase*).

3.1.2. Subcapa de unión al medio físico

La subcapa PMA define la conexión entre el controlador CAN y el medio físico, también describe las características funcionales y eléctricas que debe cumplir el transceptor (*transceiver*) para la transmisión/recepción de datos entre una ECU y el bus en sus implementaciones en circuitos integrados, con la finalidad de detectar fallos en el bus [12].

Básicamente la interfaz eléctrica con el bus consiste de un transmisor y un receptor. La Figura 3.6 ilustra el diseño básico en diagrama a bloques de un transceptor en una ECU CAN; el controlador CAN provee la funcionalidad básica del transceptor, que consiste de un comparador en la recepción y un amplificador en la salida.

El transceptor como transmisor se encarga de adaptar las señales entre el controlador CAN y el bus, además de las siguientes funciones: suministrar la capacidad de rendimiento al controlador CAN, proteger al controlador CAN contra sobrecargas, y reducir la radiación electromagnética.

El transceptor como receptor generalmente se encuentra en modo activo y recesivo y sus funciones son las siguientes: suministrar un nivel de señal recesivo, proteger el comparador de entrada del controlador CAN contra sobrecargas de voltaje de las líneas del bus y detectar errores en el bus tales como ruptura de la línea, corto-circuito y conmutación a operación asimétrica de una sola línea.

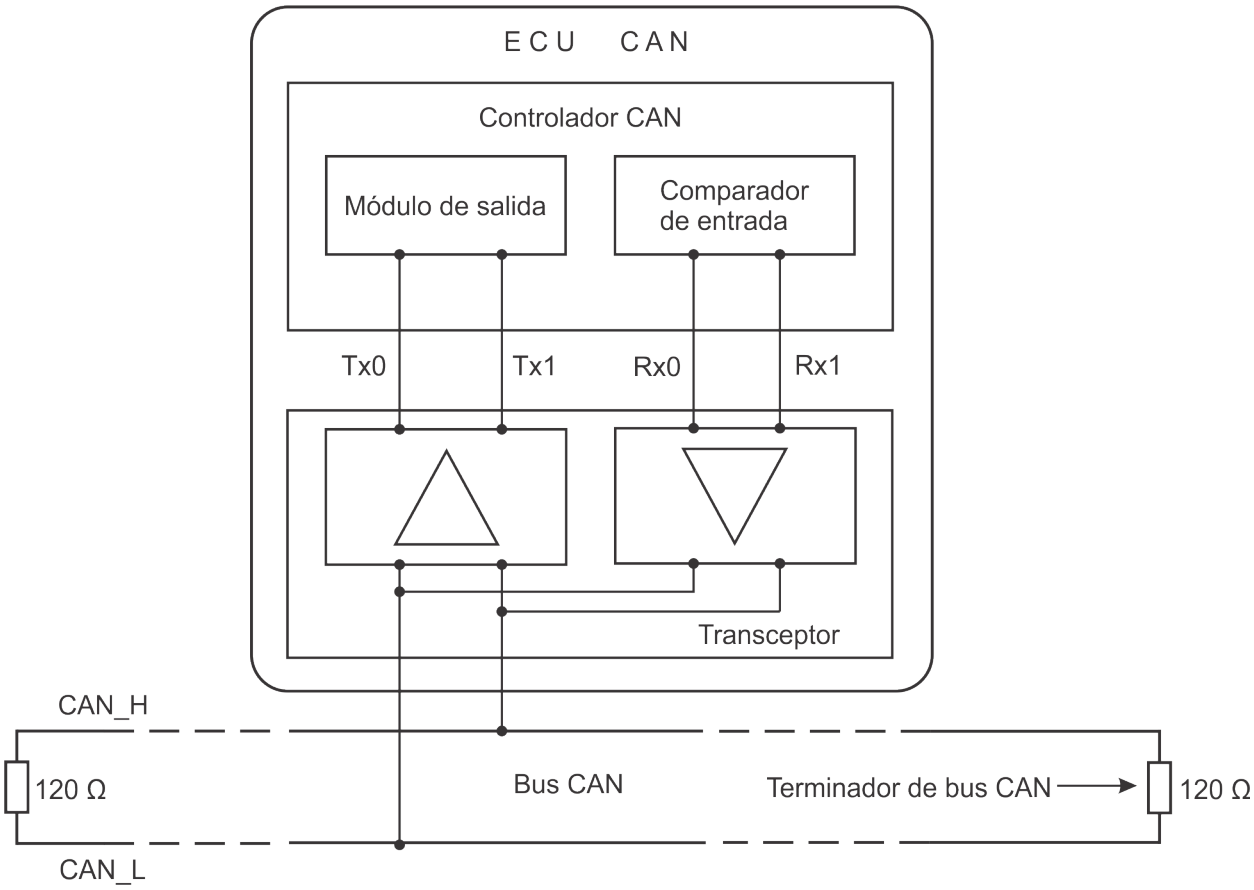

Figura 3.6: Arquitectura de una ECU conectada al bus CAN.

Además, el transceptor cuenta con la función opcional de proporcionar aislamiento galvánico al nodo CAN de las líneas de bus, generalmente mediante el uso de opto acopladores.

3.1.3. Subcapa de interfaz dependiente del medio

La subcapa MDI define los aspectos eléctricos para la conexión entre el medio físico y el nodo. Las especificaciones CAN [32] y CAN FD [36], así como en los estándares ISO 11898 [33] e ISO 11519 [34], no definen las especificaciones mecánicas ni los parámetros de los materiales de la capa física, sin embargo sí definen las características de la interfaz respecto al medio de transmisión y al tipo de conector.

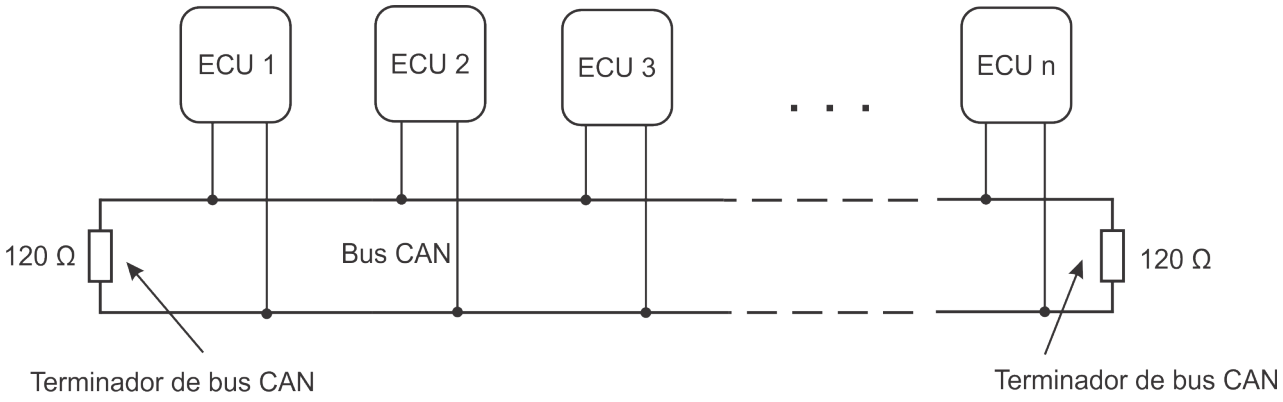

Figura 3.7: Bus CAN con medio de transmisión eléctrico de dos hilos [71].

3.1.3.1. Medio físico

Para realizar el método de arbitraje en el bus, es indispensable representar los niveles de señal recesivo y dominante. Dicha representación se realiza con medios eléctricos u ópticos, los cuales se describen a continuación.

3.1.3.1.1. Medio de transmisión eléctrico

Generalmente en la implementación de las redes CAN (véase Figura 3.7) se utiliza como medio físico el bus de dos hilos (*two wire bus*) que permite una transmisión diferencial y es resistente a los errores de modo común. Las líneas del bus deben contar con un terminador de bus en cada extremo, el cual consiste en resistores de 120 Ω para evitar la reflexión de la señal.

Para garantizar la transmisión de señales en ambientes con interferencia eléctrica se recomienda el uso de cables de tipo par trenzado, además si se implementa un manejo adecuado de errores en el bus, es posible que la comunicación continúe en condiciones de inmunidad de ruido reducida aún si una línea se rompe o se encuentra en corto circuito.

Otra implementación, es el bus de un hilo (*single wire bus*), el cual da por hecho que está disponible una tierra común para todos los nodos (véase Figura 3.8) y sólo se utiliza para aplicaciones aisladas, por ejemplo en aplicaciones automotrices para la interconexión de

dispositivos electrónicos internos. El bus de un hilo está expuesto a la interferencia inducida eléctricamente cuando no está blindado, por lo tanto es necesario realizar un desplazamiento del nivel de la señal para mejorar la relación señal/ruido, lo cual afecta el grado de emisiones electromagnéticas y por lo tanto reduce las velocidades de transferencia de datos.

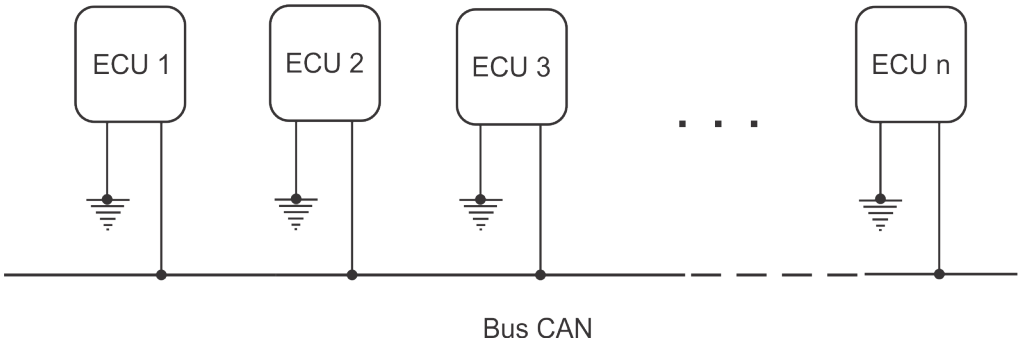

Figura 3.8: Bus CAN con medio de transmisión eléctrico de un hilo [71].

Se han desarrollado soluciones en las que se transmiten en el mismo bus las señales CAN junto con la fuente de alimentación como lo hacen los FBs AS-I (*Actuator-Sensor Interface*) y PROFIBUS-PA (*PROFIBUS Profile for Process Automation*). El suministro de voltaje sobre el bus es conveniente en sistemas CAN, sin embargo no se recomienda la transmisión simultánea de alimentación y de datos debido al arbitraje no destructivo bit a bit que utiliza dicho protocolo. La transmisión simultánea de datos y alimentación implica la utilización de circuitos de alto costo, pérdida de inmunidad a la interferencia y reducción en la longitud del bus, por lo que esta solución no es competente frente a las ya existentes en el mercado [71].

3.1.3.1.2. Medio de transmisión óptico

La transmisión por medios ópticos tiene una gran importancia en el desarrollo de redes CAN, especialmente en el área de la fibra óptica de plástico. El comportamiento eléctricamente neutro de un medio óptico es ideal para aplicaciones en ambientes potencialmente explosivos y entornos electromagnéticos perturbados.

Las líneas de transmisión y recepción deben proporcionarse de forma separada, debido al acoplamiento directo en el medio óptico, además cada línea de recepción debe acoplarse

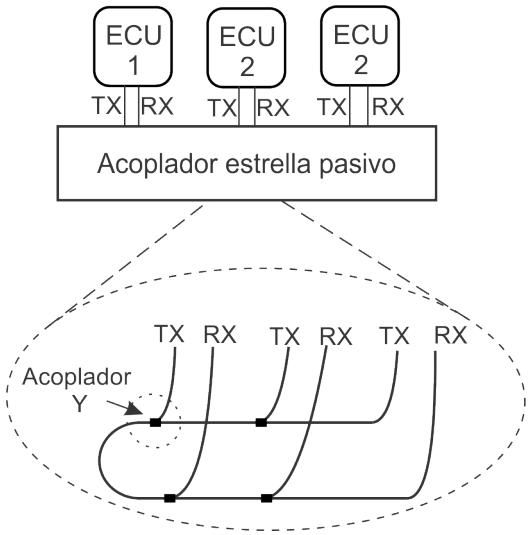

Figura 3.9: Estructura básica del bus CAN con medio de transmisión óptico y acoplador tipo estrella [71].

externamente con cada línea de transmisión con el propósito de asegurar el monitoreo de bits que requiere el protocolo CAN, esta función puede implementarse por un acoplador de tipo estrella (véase Figura 3.9). La principal desventaja de este tipo de medio físico es que el uso de acopladores pasivos tipo estrella sólo soporta una cantidad pequeña de ECUs, lo cual limita la expansión de este tipo de redes, ya que el número máximo de ECU conectados a un medio óptico depende de la potencia del haz de luz, la potencia de atenuación a través de la línea de transmisión y de los acopladores.

3.1.3.2. Topología de una red CAN

En muchas aplicaciones es inevitable utilizar topologías extendidas, por ejemplo para conectar herramientas de diagnóstico o servicio. Para superar las limitaciones de la topología de bus CAN se emplean repetidores, puentes y pasarelas con la finalidad de adaptar la topología de red de acuerdo con las necesidades geográficas de cada aplicación específica. La topología de bus es la que permite mayor velocidad de transferencia de datos y mayor longitud de línea; los automóviles modernos utilizan diversas topologías, mientras que en las aplicaciones industriales se prefiere la topología de bus lineal. A continuación se describen las diferentes topologías en las redes CAN.

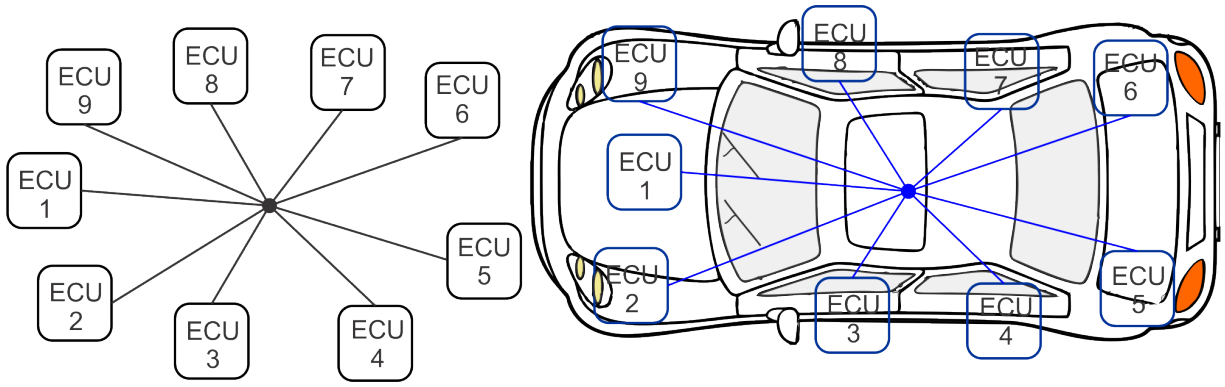

Figura 3.10: Esquemático y aplicación de una topología estrella simple [71].

3.1.3.2.1. Estrella simple

En la topología de estrella simple (*single star*) todas las ECUs se conectan en el centro de una estrella (véase Figura 3.10), y utilizan como terminador de bus una resistencia de terminación de 60 Ω. La ventaja de esta topología es la flexibilidad y la posibilidad de conectar un alto número de ECUs en la red. La longitud máxima del cable que conecta a la ECU con el bus es de 9 m y se alcanzan velocidades de transferencia de datos de hasta 500 kbps.

3.1.3.2.2. Estrella doble

La topología estrella doble (*twin star*) consiste de la interconexión de dos estrellas simples (véase Figura 3.11), con lo que se optimiza la adaptación del cableado. En esta topología las resistencias de terminación se colocan en el centro de cada estrella y la máxima velocidad de transferencia de datos es de 500 kbps.

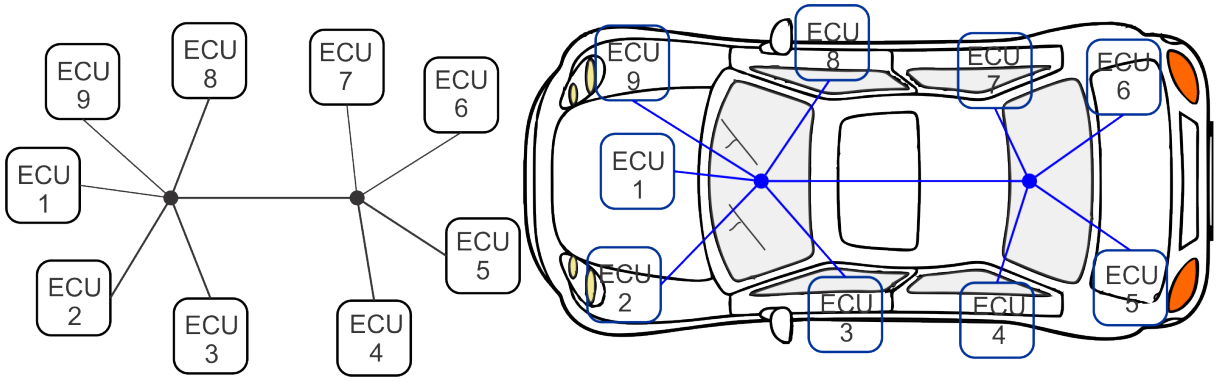

Figura 3.11: Esquemático y aplicación de una topología estrella doble [71].

3.1.3.2.3. Bus lineal

La mayoría de aplicaciones industriales y automovilísticas emplean la topología de bus lineal (*linear bus*) (véase Figura 3.12), cabe señalar que en los automóviles permite velocidades superiores a 1 Mbps. El estándar ISO 11898-2 recomienda una longitud máxima de 30 cm entre la ECU y el bus, y la longitud total del bus debe ser menor a 40 m sin repetidores.

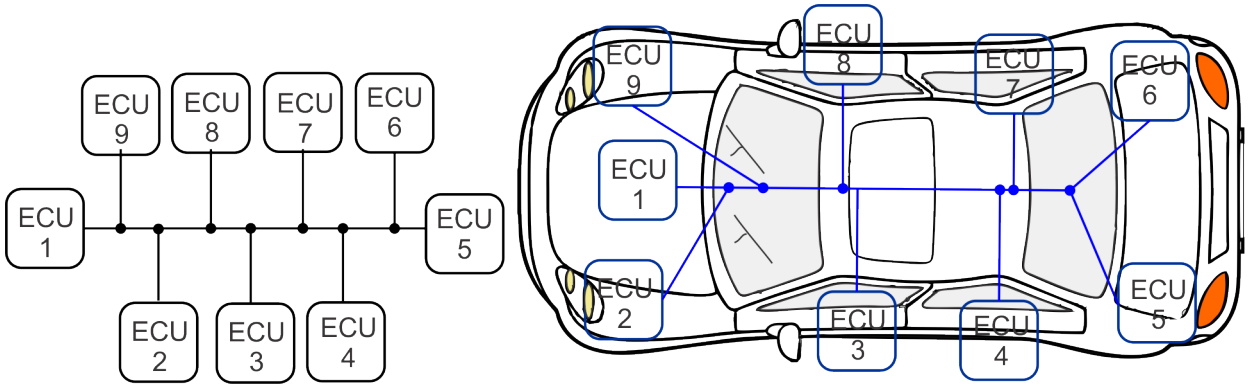

Figura 3.12: Esquemático y aplicación de una topología bus lineal [71].

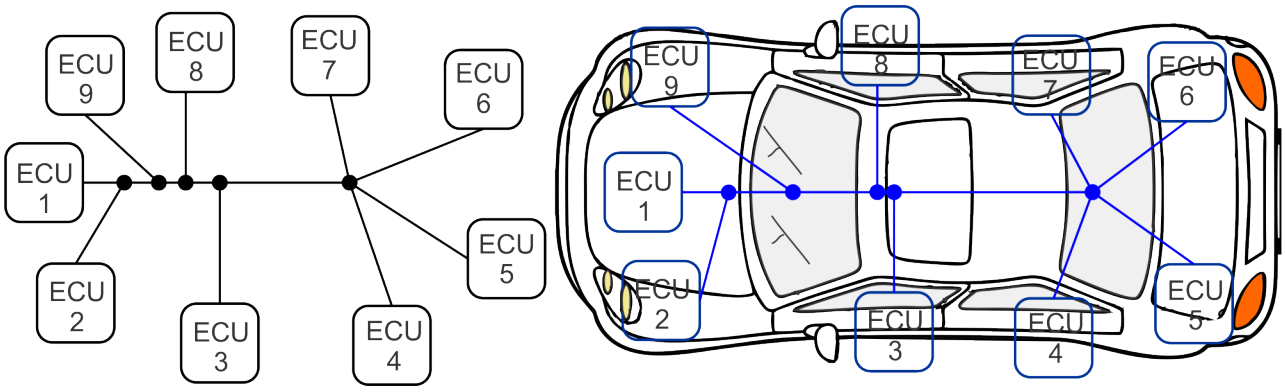

Figura 3.13: Esquemático y aplicación de una topología hibrida [71].

3.1.3.2.4. Topología híbrida

La topología híbrida es una combinación de una topología de bus lineal y una topología de estrella simple (véase Figura 3.13). Esta tecnología combina las ventajas y desventajas de ambas topologías. La máxima velocidad de transferencia de datos es de 1 Mbps y soporta menores longitudes de cableado que otras soluciones. Además, el costo de esta topología es más elevado que las anteriores.

3.2. Capa enlace de datos

De acuerdo a los estándares ISO 7498 [72] e ISO 8802-2 [73] la capa de enlace de datos (DLL, *Data Link Layer*) se divide en dos subcapas (véase Figura 3.14): control de enlace lógico (LLC, *Logic Link Control*) y control de acceso al medio (MAC, *Medium Access Control*).

3.2.1. 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, asimismo proporciona dos tipos de servicios de transmisión sin conexión al usuario LLC (*LLC user*):

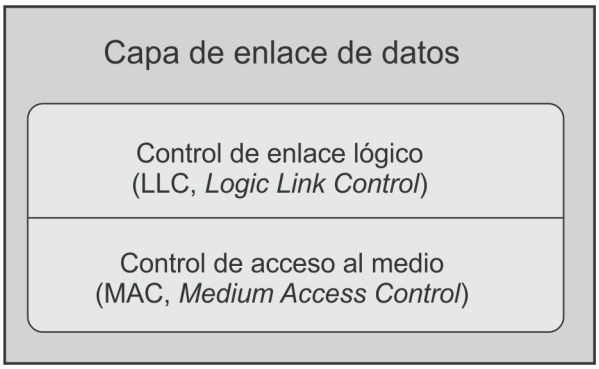

Figura 3.14: Arquitectura de la capa de enlace de datos [52].

- 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 LDSU sin establecer una conexión de enlace de datos.

De acuerdo con los tipos de servicio, el protocolo de comunicaciones CAN definen dos formatos de tramas, de datos LLC y remota LLC. Ambos formatos definen identificadores de 11 bits (estándar) y de 29 bits (extendida); en el caso del protocolo de comunicaciones CAN FD no existen las tramas remotas LLC, pero cuenta con soporte para las tramas remotas de CAN.

3.2.1.1. Funciones de la subcapa LLC

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. Como máximo sólo se pueden generar dos tramas de sobrecarga.

- 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.

3.2.2. Control de acceso al medio

La subcapa MAC describe la parte baja de la capa DLL y es el núcleo de los protocolos de comunicaciones CAN y CAN FD. Su función es presentar los mensajes recibidos de la subcapa LLC y realizar la transmisión al bus. La transmisión se realiza en tiempo real, a altas velocidades de transferencias de datos y requiere de retrasos mínimos. En redes multimaestro la técnica de acceso al medio es muy importante, ya que toda ECU activa cuenta con los mismos derechos para controlar la red y hacer uso de los recursos del bus.

Para acceder al medio, las ECUs CAN utilizan un mecanismo de arbitraje, el cual consiste en que cuando una ECU necesita enviar información a través de una red CAN, la capa de aplicación realiza una petición de forma asíncrona para transmitir una trama. Es posible que varias ECUs inicien la transmisión al mismo tiempo, por lo que el protocolo CAN resuelve la colisión generada al asignar prioridades mediante el uso de un identificador de cada mensaje; 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 en las redes CAN y CAN FD 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, las ECUs en la red que necesitan transmitir información deben esperar a que el bus esté libre (detección de portadora) y al

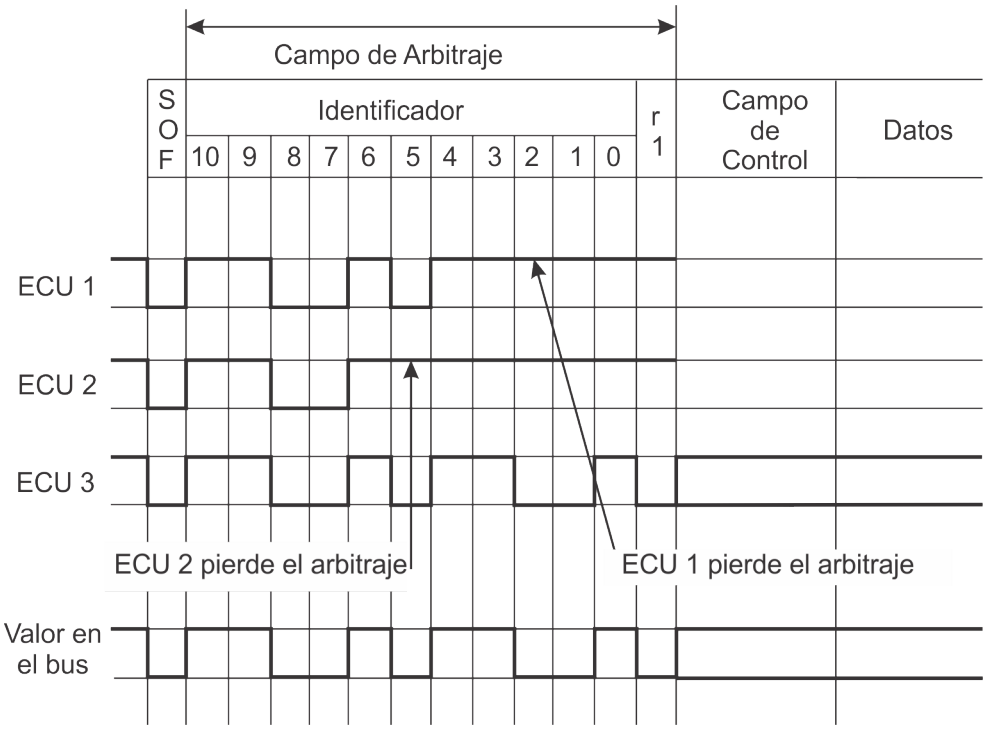

Figura 3.15: Procedimiento de arbitraje [44].

cumplirse esta condición, las ECUs transmiten un bit de inicio (acceso múltiple). Cada nodo lee el bus bit a bit durante la transmisión de la trama y comparan 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.

La Figura 3.15 muestra el caso en la que tres ECU comienzan a transmitir simultáneamente, en la posición del quinto bit del identificador las ECUs 1 y 3 transmiten un bit dominante y el nodo 2 transmite un bit recesivo, todas las ECUs comparan los valores transmitidos con los que reciben y el nodo 2 detecta una diferencia (detección de colisión), lo que indica que ha perdido el arbitraje e inmediatamente deja de transmitir para conmutar a modo de recepción. Las ECUs 1 y 3 continúan transmitiendo hasta que se presenta la misma situación y el nodo 3 es quien gana el arbitraje y transmite su trama. Este tipo de transmisión se conoce como arbitraje no destructivo bit a bit ya que a pesar de que varias ECUs inician la transmisión de sus tramas al mismo tiempo, el ganador del arbitraje (mensaje con

la mayor prioridad), continua 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 espacio del ACK, se activa el proceso de control de errores por parte del administrador de la ECU, quien lo considera un error de bit.

En el caso del protocolo CAN, si se inicia simultáneamente la transmisión de una trama de datos y una trama remota 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 arbitraje en las redes CAN limita la extensión máxima de la red a una velocidad de transferencia de datos específica.

3.2.2.1. Transmisión de mensajes

El protocolo de comunicaciones CAN FD establece cuatro diferentes formatos de tramas (véase Tabla 3.1) que difieren en la longitud del campo de arbitraje (*arbitration field*) y del campo de control (*control field*).

Formatos de trama	**Longitud del campo de arbitraje**
CAN BASE	11 bits y velocidad de bit constante
CAN EXTENDED	29 bits y velocidad de bit constante
CAN FD BASE	11 bits y velocidad de bit compartido
CAN FD EXTENDED	29 bits y velocidad de bit compartido

Tabla 3.1: Tipos de formatos de trama en CAN FD.

Para la transmisión y control de mensajes CAN y CAN FD, se definen cuatro tipos de tramas:

- Trama de datos (*data frame*): se encarga de llevar los datos del transmisor al receptor, existen cuatro subtipos de trama de datos en el protocolo de comunicaciones CAN FD y son:
  - Trama de datos CAN en formato estándar (*data frame in CAN base format*).

- Trama de datos CAN en formato extendido (*data frame in CAN extended format*).
- Trama de datos CAN FD en formato estándar (*data frame in CAN FD base format*).
- Trama de datos CAN FD en formato extendido (*frame in CAN FD extended format*).

- Trama remota (*remote frame*): es transmitida por un módulo del bus para solicitar la transmisión de una trama de datos con el mismo identificador. El protocolo de comunicaciones CAN FD no define las tramas remotas.
- Trama de error (*error frame*): es transmitida por cualquier ECU que detecta un error en el bus.
- Trama de sobrecarga (*overload frame*): es usada para sincronizar un evento ocioso y proveer de un retardo extra entre la trama precedente y la que se encuentra sucediendo (trama de datos o remota)

Para separar las tramas remotas y de datos de tramas precedentes se utiliza un espacio intertrama (*inter-frame space*)

3.2.2.1.1. Trama de datos

La trama de datos se compone de siete campos (véase Figura 3.16):

- Inicio de trama (SOF, *Star Of Frame*): marca el inicio de una trama de datos o una trama remota, y consiste en un bit dominante. Una ECU sólo puede empezar a transmitir cuando el bus está libre. Todas las ECUs activas en la red se tienen que sincronizar con el flanco causado por el SOF. El SOF es el mismo para las tramas estándar y extendida de CAN y CAN FD.
- Campo de arbitraje: la estructura del campo de arbitraje es diferente para la trama estándar y extendida, pero difiere según el formato de trama del protocolo CAN y CAN FD (véase Figura 3.17).

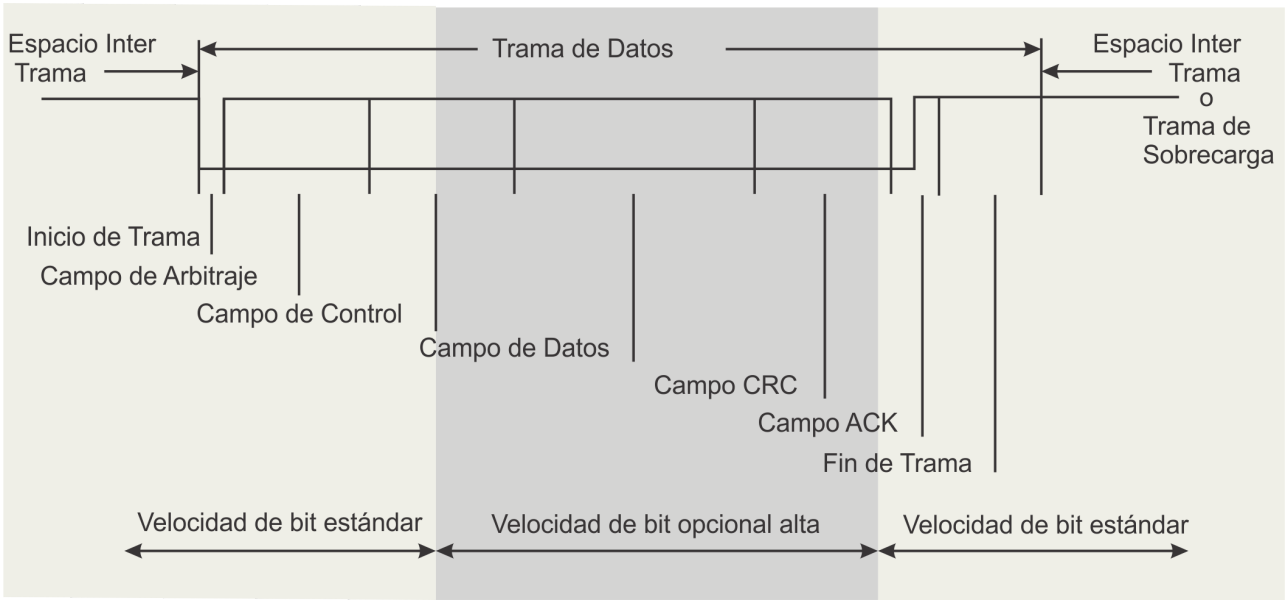

Figura 3.16: Formato de trama de datos CAN FD [36].

- El formato estándar consiste de un identificador base (*base identifier*) de 11 bits. En el caso del protocolo de comunicaciones CAN después del identificador base sigue un bit de petición de transmisión remota (RTR, *Remote Transmission Request*) y para el protocolo de comunicaciones CAN FD le sigue el bit r1. De acuerdo al estándar ISO 11898 [33] el bit menos significativo (ID-0) del identificador se transmite al último y los 7 bits más significativos (ID-10...ID-4) no pueden ser todos recesivos, la especificación CAN FD [36] no tiene esta última restricción.
- El formato extendido consiste del identificador extendido (*extended identifier*) de 29 bits, el bit de petición remota substituta (SRR, *Substitute Remote Request*), el bit de la bandera de extensión del identificador (IDE, *Identifier Extension Flag*) y el bit RTR (formato CAN) o r1 (formato CAN FD). El identificador se divide en dos secciones, la primera de 11 bits (denotado ID-28...ID-18) denominada identificador base (*base identifier*) que corresponde al identificador del formato estándar, y la segunda sección de 18 bits (denotado ID-17...ID-0) conocida como extensión del identificador (*identifier extension*).

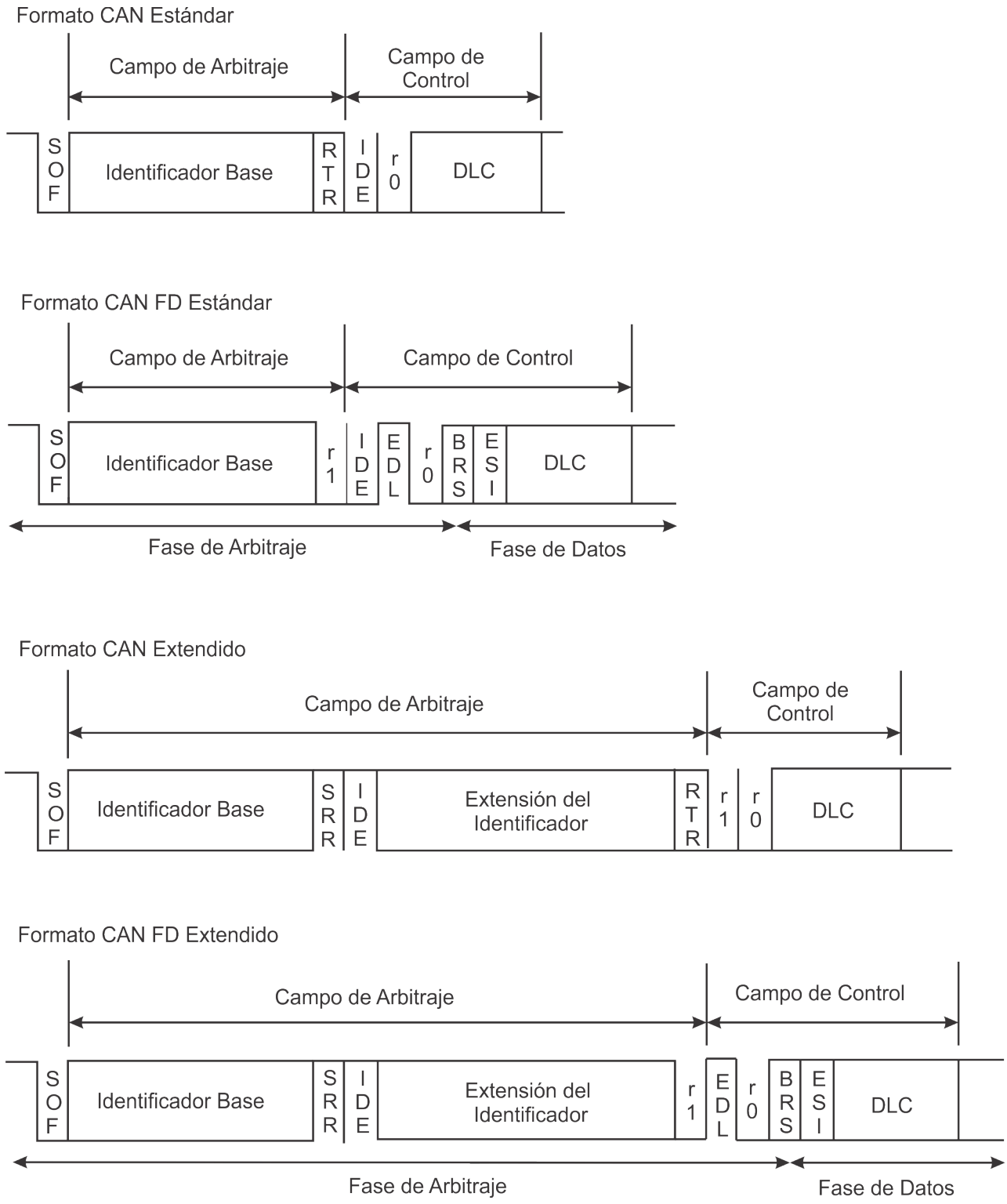

Figura 3.17: Formatos de trama de datos estándar y extendida de CAN y CAN FD [36].

El bit IDE forma parte del campo de arbitraje cuando se usa un formato de trama extendida y como parte del campo de control para el formato de trama estándar. Para distinguir entre estos dos formatos el bit IDE se transmite con un valor dominante en el formato estándar y con un valor recesivo en el formato extendido.

En el formato extendido el bit SRR se transmite en la posición del bit RTR del formato estándar con un valor recesivo. Además, las posibles colisiones entre ambos tipos de formatos de trama que tengan el mismo valor en el campo identificador base se resuelven de manera que el formato estándar predomina sobre el formato de trama extendida.

El bit RTR solamente existe en las tramas CAN, debe ser dominante para ambos formatos de trama de datos y recesivo con tramas remotas. En el formato de tramas CAN FD este bit es reemplazado por el bit reservado r1 con un valor dominante. Cabe recordar que CAN FD no define las tramas remotas.

- Campo de control (véase Figura 3.17): la estructura del campo de control es diferente para los formatos estándar y extendido de los protocolos de comunicaciones CAN y CAN FD. A continuación se muestran los bits que componen cada campo.

  - En el formato CAN estándar el campo de control consiste de los bits IDE, r0 y 4 bits que forman el código de longitud de datos (DLC, *Data Length Code*).
  - En el formato CAN FD estándar el campo de control consiste de los bits IDE, EDL, r0, BRS, ESI y 4 bits del DLC.
  - En el formato CAN extendido el campo de control consiste de los bits r0, r1 y 4 bits del DLC.
  - En el formato CAN FD extendido el campo de control consiste de los bits EDL, r0, BRS, ESI y 4 bits del DLC.

El bit de longitud de datos extendido (EDL, *Extended Data Length*) es un bit recesivo y solamente existe en las tramas CAN FD, este bit distingue entre una trama CAN y CAN FD. En el formato de trama CAN se transmite el bit r0 con valor dominante en lugar del bit EDL. En tramas con identificador de 11 bits, el bit EDL se transmite después del bit IDE, mientras que en tramas con identificador de 29 bits el bit EDL se transmite después del bit

r1. El bit EDL está siempre precedido por el bit dominante r0, el cual está reservado para futuras expansiones del protocolo.

El bit de cambio de velocidad (BRS, *Bit Rate Switch*) decide si la velocidad de bit se cambia dentro de la trama en el formato CAN FD. Si el bit se transmite con un valor recesivo, indica que la velocidad de bit se cambió desde la velocidad de bit estándar desde la fase de arbitraje (*arbitration phase*) para configurar velocidades de bit alternativas de la fase de datos (*data phase*). Si se transmite el bit BRS con un valor dominante es porque la velocidad de bit no ha sido cambiada. Es importante mencionar que el bit BRS no existe en el formato de tramas CAN.

El bit de indicador del estado de error (ESI, *Error State Indicator*) es una bandera que se transmite con un valor dominante por las ECUs que están en modo de error activo (*error active*) y con un valor recesivo si las ECUs se encuentran en modo de error pasivo (*error passive*). El bit ESI no existe en el formato de tramas CAN.

Se recomienda que los bits reservados r0 y r1 se transmitan con un valor dominante, aunque los receptores aceptan bits dominantes y recesivos en todas sus combinaciones y también aceptan bits SRR dominantes.

El campo DLC indica el número de bytes que contiene el campo de datos. Este código tiene una longitud de 4 bits y es diferente en los protocolos de comunicaciones CAN y CAN FD. En la Tabla 3.2 se muestra la codificación de los bits DLC, en donde los primeros nueve son los mismos, pero los siguientes cambian ya que en el formato CAN el campo de datos tiene una longitud máxima de 8 bytes, mientras que el formato CAN FD permite enviar un máximo de 64 bytes.

	Bytes de datos	**DLC$_3$**	**DLC$_2$**	**DLC$_1$**	**DLC$_0$**
Código en formato CAN y CAN FD	0	0	0	0	0
	1	0	0	0	1
	2	0	0	1	0
	3	0	0	1	1
	4	0	1	0	0
	5	0	1	0	1
	6	0	1	1	0
	7	0	1	1	1
	8	1	0	0	0
Código en formato CAN FD	12	1	0	0	1
	16	1	0	1	0
	20	1	0	1	1
	24	1	1	0	0
	32	1	1	0	1
	48	1	1	1	0
	64	1	1	1	1

Tabla 3.2: Codificación del número de bytes de datos mediante los bits del DLC.

- Campo de datos (*data field*): consta de la carga útil que son transmitidos en la trama de datos. Los datos a transmitir pueden ser de 0 a 8 bytes en el formato CAN y de 0 a 64 bytes en el formato CAN FD. Los bytes contienen 8 bits que son transmitidos por el bit más significativo (MBS, *More Bit Significant*). En las tramas remotas y en en las tramas de sincronización el campo DLC es cero.

- Campo CRC (*CRC field*): contiene la secuencia CRC (*CRC sequence*) seguida por el limitador de CRC (*CRC delimiter*), este último se transmite con un valor recesivo (véase Figura 3.18).

La secuencia CRC en las tramas CAN y CAN FD están derivadas del código de redundancia cíclica (*BCH code*). Las ECUs en el protocolo de comunicaciones CAN FD usan

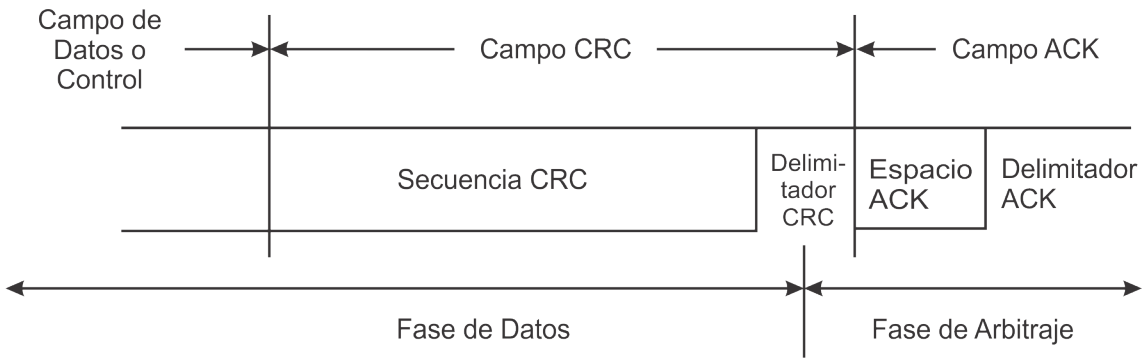

Figura 3.18: Formato del campo CRC [36].

diferentes polinomios generadores de CRC. El primer polinomio CRC_15 es usado para todas las tramas en formatos CAN y CAN FD. El segundo polinomio CRC_17 es usado en tramas en formato CAN FD cuando el campo de datos tiene una longitud de hasta 16 bytes. El tercer polinomio CRC_21 es usado en tramas en formato CAN FD cuando el campo de datos tiene una longitud mayor de 16 bytes. Cada polinomio tiene una distancia de Hamming H=6.

CRC_15	0xC599	$(x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+1)$
=		$(x+1)\cdot(x^7+x^3+1)\cdot(x^7+x^3+x^2+x+1)$
CRC_17	0x3685B	$(x^{17}+x^{16}+x^{14}+x^{13}+x^{11}+x^6+x^4+x^3+x^1+1)$
=		$(x+1)\cdot(x^{16}+x^{13}+x^{10}+x^9+x^8+x^7+x^6+x^3+1)$
CRC_21	0x302899	$(x^{21}+x^{20}+x^{13}+x^{11}+x^7+x^4+x^3+1)$
=		$(x+1)\cdot(x^{10}+x^3+1)\cdot(x^{10}+x^3+x^2+x^1+1)$

La longitud de la secuencia del CRC (n_{CRC}, orden del polinomio generador) está establecida a 15, 17 y 21 bits para el CRC_15, CRC_17 y CRC_21 respectivamente. Al inicio de la trama se calcula simultáneamente las tres secuencias de CRC en todas las ECUs incluyendo al transmisor. La ECU que gana el arbitraje selecciona la secuencia de CRC dependiendo de los valores de los bits EDL y DLC. Todas las ECUs receptoras deben considerar únicamente el polinomio CRC seleccionado para verificar un error de CRC (*CRC error*).

El flujo de bits relevantes para el cálculo del CRC consiste de los bits SOF, campo de arbitraje, campo de control, y campo de datos si está presente, complementados con n_{CRC}

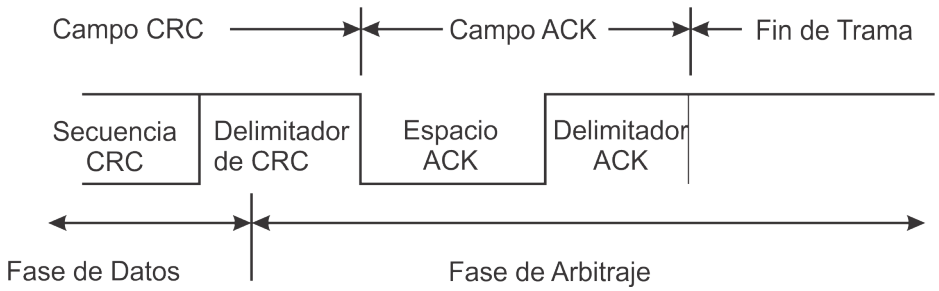

Figura 3.19: Formato del campo ACK [36].

bits de 0's. Para el cálculo del CRC, el polinomio definido por los coeficientes del flujo de bits relevantes es dividido en módulo-2 por el polinomio generador. El resultado de la división es la secuencia del CRC y es transmitida al bus.

La secuencia del CRC es seguida por el delimitador de CRC. En el formato de tramas CAN el delimitador de CRC es un bit recesivo, mientras que en el formato de tramas CAN FD el delimitador de CRC puede consistir de uno o dos bits. Un transmisor deberá mandar solamente un bit recesivo como delimitador de CRC, pero también deberá aceptar dos bits recesivos antes del flanco que comienza con el espacio de aceptación (*ACK slot*). Un receptor deberá mandar un bit de aceptación después del primer bit de delimitador CRC. Los controladores del protocolo de comunicaciones CAN FD cambian de la fase de datos a la fase de arbitraje cuando se alcanza el punto de muestreo del primer bit del delimitador CRC.

- Campo de aceptación (*ACK field*): contiene el espacio de aceptación y el delimitador de aceptación (*ACK delimiter*) (véase Figura 3.19). En el campo de aceptación la ECU que se encuentra transmitiendo manda bits recesivos y las ECUs que han recibido una trama válida, lo reportan a la ECU transmisora mandando un bit dominante al inicio del espacio de aceptación.

Todas las estaciones que han recibido una secuencia CRC igual a la calculada reportan la correcta transmisión mediante el bit del espacio de aceptación, reescribiendo el bit recesivo del transmisor por un bit dominante. En el formato CAN FD, todas las ECUs deberán aceptar dos bits de fase dominantes traslapados como una aceptación válida. El formato CAN considera un error cuando existe un bit dominante seguido de un bit solo de aceptación.

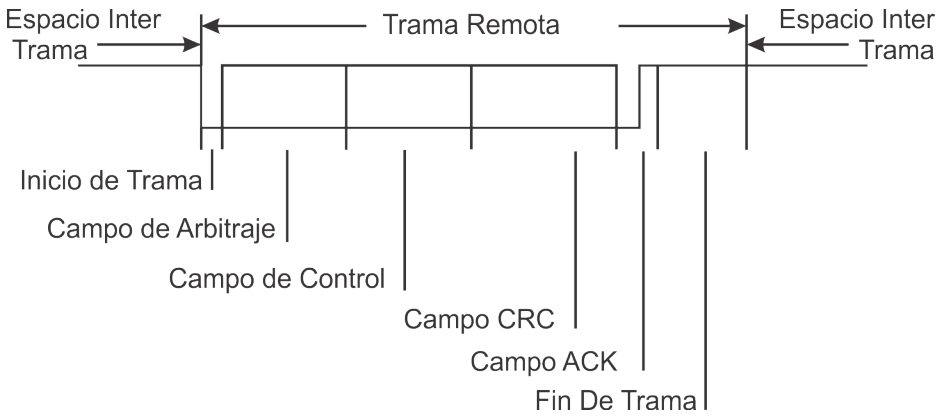

Figura 3.20: Formato de una trama remota [36].

El delimitador de aceptación es un bit recesivo y es el último bit del campo de aceptación, como consecuencia el espacio de aceptación se encuentra rodeado de dos bits recesivos que corresponden al delimitador CRC y de aceptación.

- Fin de trama (EOF, *End Of Frame*): cada trama de datos o trama remota es delimitada por una secuencia de siete bits recesivos. Cuando el EOF está activo se realiza una violación al procedimiento de inserción de bit, por ello dicho procedimiento no se aplica a este campo.

3.2.2.1.2. Trama remota

Una ECU 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 como en formato extendido (véase Figura 3.20). 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. La trama remota está definida únicamente en el formato CAN, ya que tanto las tramas remotas y el bit RTR no existen en el protocolo de comunicaciones CAN FD.

3.2.2.1.3. 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 y CAN FD permiten la generación de dos tramas de sobrecarga como máximo para retrasar la transmisión de la siguiente trama. Existen tres condiciones de sobrecarga que llevan a la transmisión de banderas de sobrecarga:

1. Condiciones internas del receptor que requieren un retardo en la transmisión de la siguiente trama de datos o trama remota.
2. Detección de un bit dominante en el primer o segundo bit de intermisión.
3. Si una ECU CAN FD muestra un bit dominante en el octavo bit (último) de un delimitador de error o sobrecarga, o si un receptor CAN FD muestra un bit dominante en el último bit del EOF, empezará a transmitir una trama de sobrecarga. Los contadores de errores no se incrementarán.

El inicio de una trama de sobrecarga debido a la condición de sobrecarga 1 es solamente permitido en el primer bit de intermisión, mientras que con las condiciones 2 y 3 empiezan después de que se detecta un bit dominante. Una trama de sobrecarga se considera una forma especial de trama de error y consta de los siguientes campos (véase Figura 3.21):

- Bandera de sobrecarga (*overload flag*): está formada por seis bits dominantes. La forma completa corresponde a la bandera de error activo.
- Delimitador de sobrecarga (*overload delimiter*): está formado por ocho bits recesivos.

3.2.2.1.4. Trama de error

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

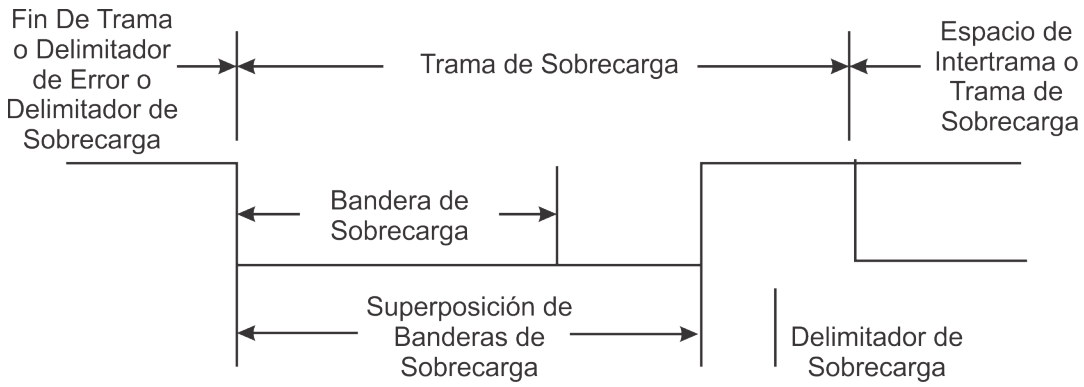

Figura 3.21: Formato de una trama de sobrecarga [36].

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 3.22):

- 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.
- 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,

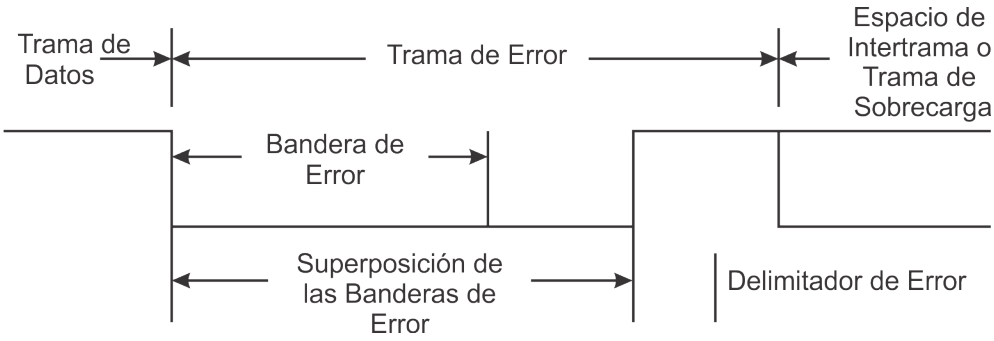

Figura 3.22: Formato de una trama de error [36].

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.

3.2.2.1.5. Espacio de intertrama

Las tramas de datos y remotas están separadas de tramas precedentes (datos, remotas, errores y sobrecarga) por un espacio entre tramas llamado espacio de intertrama (*interframe space*), 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 intertrama 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 ninguna otra ECU inicie la transmisión de una trama de datos o remota. En una ECU CAN FD con una transmisión pendiente y que además está en estado de error activo o que debió recibir una trama y si el tercer bit del intermisión es dominante, empezará a transmitir este mensaje con el primer bit del identificador base en el próximo bit sin transmitir el bit EOF y continuará como receptor.

- Bus libre (*bus idle*): este periodo es de longitud arbitraria y tiene un nivel recesivo hasta que alguna ECU inicie la transmisión de una nueva trama. La detección de un bit dominante en el bus se interpreta como un bit SOF.

- Suspender transmisión (*suspend transmission*): adicionalmente, el espacio inter tramas contiene un tiempo de inhibición de transmisión de ocho bits para las ECUs que se encuentren en estado de error pasivo.

El espacio intertrama tiene dos formas de representación, para las ECU que no tienen un error pasivo o que han recibido un mensaje previo (véase Figura 3.23). La otra forma de representación es cuando las ECUs tiene un error pasivo y ha transmitido un mensaje previo (véase Figura 3.24).

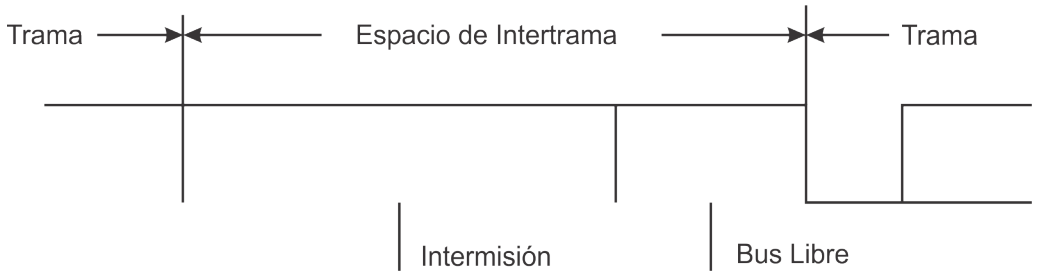

Figura 3.23: Formato del espacio intertrama para ECU sin error pasivo [36].

3.2.2.2. Validación de tramas

El instante de tiempo en el que se valida una trama difiere según el transmisor y el receptor del mensaje:

- Transmisor: la trama es válida para el transmisor si no existen errores hasta el final del campo EOF. Si existe un error en la trama se activa el proceso de retransmisión de acuerdo a la priorización de la trama. La retransmisión de una trama puede ser limitada (por configuración) a un valor en específico. Por defecto el número de retransmisiones es ilimitado.
- Receptor: la trama es válida para el receptor si no existen errores hasta el antepenúltimo bit de EOF.

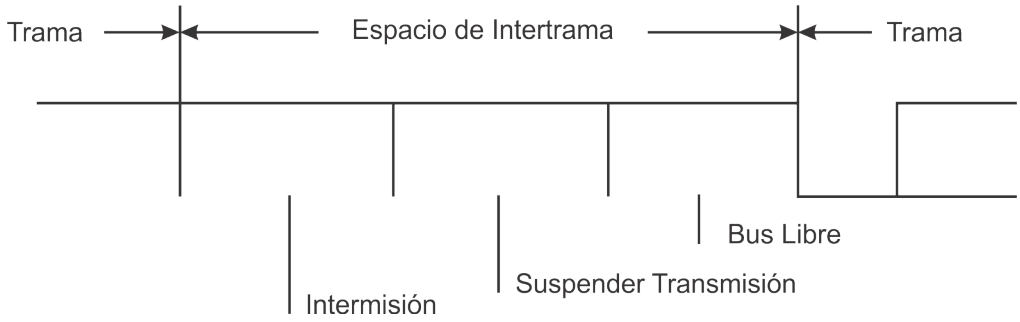

Figura 3.24: Formato del espacio intertrama para ECU con error pasivo [36].

3.2.2.3. Manejo de errores

Un controlador CAN cuenta con la capacidad de detectar y manejar los errores que surjan en una red. 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*), y con ello no afecten a los demás nodos [28].

3.2.2.3.1. Detección de errores

Para cumplir con las demandas relativas a la seguridad en la transmisión de datos, el protocolo CAN FD define cinco mecanismos para detección de errores, los cuales no son mutuamente exclusivos y se describen a continuación:

- 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. Una excepción ocurre cuando se transmite un bit recesivo en el flujo de inserción de bit en el campo de arbitraje o durante el espacio ACK, entonces no ocurre un bit de error cuando es monitoreado un bit dominante.

- Verificación del procedimiento de inserción de bit: hace referencia al hecho de detectar un error de inserción de bit (*stuff error*) cuando ocurren seis niveles consecutivos de bits con el mismo valor en un campo de trama codificado por el procedimiento de inserción de bit.

- Verificación de redundancia cíclica: se presenta un error de CRC (*CRC error*) cuando la secuencia CRC recibida no se corresponde con la secuencia CRC calculada. Los receptores calculan de la misma forma el CRC que el transmisor.

- Verificación de trama: detecta un error cuando un campo de bit de formato fijo contiene uno o más bits no válidos (*form error*). Las excepciones a este tipo de errores ocurren

cuando se detecta un bit dominante al final del EOF en el receptor, o cuando se detecta un bit dominante durante el último bit del delimitador de error o delimitador de sobrecarga por cualquier nodo.

- Verificación de aceptación: un transmisor detecta un error de aceptación (*ACK error*) cuando el espacio ACK (*ACK slot*) no cambia a estado dominante.

3.2.2.3.2. Señalización 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. Para una ECU que se encuentra en estado de error activo (*active error*) las banderas de señalización son de tipo error activo (*active error flag*), mientras que para una ECU en estado error pasivo (*passive error*) las banderas de señalización son de tipo pasivo (*passive error flag*).

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.

En el protocolo de comunicaciones CAN FD las ECUs que se encuentran operando en la fase de datos deberán cambiar a la fase de arbitraje cuando empieza una bandera de error.

3.3. Capa de supervisor

Las comunicaciones de bus serie presentan el problema de que una ECU defectuosa puede bloquear el funcionamiento del sistema completo. Cuando una ECU está activa y transmite una bandera de señalización de error, al detectar algún tipo de error, puede ocasionar que una ECU defectuosa acapare el medio físico, para eliminar este riesgo el protocolo CAN y CAN FD definen un mecanismo autónomo para detectar y desconectar una ECU defectuosa del bus, dicho mecanismo se conoce como gestor de fallos (*fault confinement*).

De acuerdo al gestor de fallos una ECU puede estar en tres estados (véase Figura 3.25), los cuales se detallan a continuación:

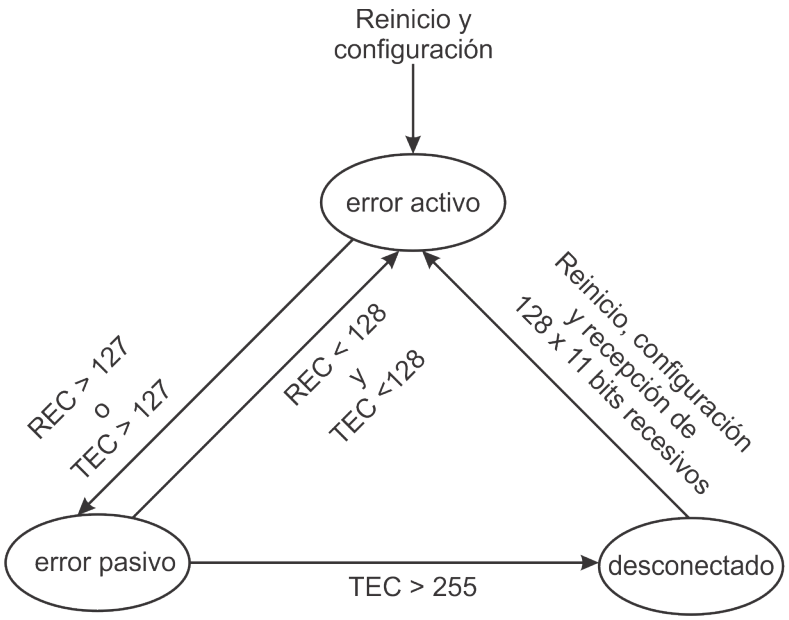

Figura 3.25: Diagrama de estados de una ECU CAN y CAN FD [36].

- Error activo (*error active*): una ECU toma parte en la comunicación de bus y cuando detecta un error envía una bandera de error activo, es decir, 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 (*error passive*): una ECU no puede enviar una bandera de error activo pero aún 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, una ECU no tiene ninguna influencia en el bus y por ello no puede transmitir tramas de datos, de aceptación, de error o de sobrecarga.

Para el gestor de fallos se implementan dos contadores: Contador de errores de transmisión (TEC, *Transmit Error Counter*), contador de errores de recepción (REC, *Receive Error Counter*)

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

de las ECUs se modifica en relación a los errores, dependiendo de los valores del contador correspondiente. La especificación CAN FD [36] establece las siguientes reglas (una a o más reglas pueden ser aplicadas durante la transferencia de información) para el incremento o decremento de los contadores:

1. Cuando un receptor detecta un error, el valor de REC debe incrementarse en 1, excepto cuando el error detectado es un error de bit durante la transmisión de una bandera de error activo o una bandera de sobrecarga.

2. Cuando un receptor detecta un bit dominante como primer bit después de una bandera de error, el valor de REC se debe incrementar en 8.

3. Cuando el transmisor transmite una bandera de error, el valor de TEC debe incrementarse en 8, sin embargo existen dos excepciones en las que el valor de TEC no se incrementa:

   - Excepción 1: Si el transmisor está en error pasivo y detecta un error de aceptación debido a que no detectó un bit ACK dominante y no detecta un bit dominante cuando está transmitiendo una bandera de error pasivo.
   - Excepción 2: Si el transmisor señaliza una bandera de error debido a un error de inserción de bit durante el arbitraje y éste debe de ser recesivo pero se monitorea como dominante.

4. Si un transmisor detecta un error de bit mientras está transmitiendo una bandera de error activo o una bandera de sobrecarga, el valor de TEC se incrementa en 8.

5. Si un receptor detecta un error de bit mientras está transmitiendo una bandera de error activo o una bandera de sobrecarga, el valor de REC se incrementa en 8.

6. Cualquier nodo soporta hasta siete bits dominantes consecutivos después de las banderas de error pasivo, error activo y sobrecarga. Después de detectar el catorceavo bit dominante (en caso de una bandera de error de activo o bandera de sobrecarga) o después de detectar el octavo bit dominante seguido de una bandera de error pasivo y

después de cada secuencia adicional de ocho bits dominantes, cada transmisor incrementar el valor del TEC en 8 y cada receptor debe incrementar el valor de REC en 8.

7. Después de una transmisión exitosa del mensaje (obteniendo un ACK correcto y sin errores hasta el EOF), el valor de TEC se decrementa en 1 y si éste tiene un valor de 0 no se decrementará.

8. Después de una recepción exitosa del mensaje (sin errores hasta el espacio de ACK y envió correcto del bit ACK), el valor REC se decrementa en 1, siempre y cuando su valor se encuente 1 y 127. Si el valor de REC está en 0, permanecerá en 0 y si es mayor a 127, deberá establecerse en un valor entre 119 y 127.

9. Una ECU está en error pasivo cuando los contadores REC y TEC son iguales o exceden a 128. Si la ECU se encuentra en error pasivo causa que mande banderas de error activo.

10. Una ECU pasa a estado desconectado cuando el valor TEC es mayor o igual a 256.

11. Una ECU que se encuentra en error pasivo pasa a error activo cuando ambos contadores TEC y REC son menores o iguales a 127.

12. Una ECU que se encuentra desconectada, pasa a error activo cuando ambos contadores se encuentran entre 0 y 128 después de monitorear 11 bits recesivos consecutivos en el bus.

La especificación CAN FD [36] agrega dos notas.

- Un error de cuenta mayor a 96 indica una alteración en el bus. Esto puede ser una desventaja en el significado de una condición de prueba.

- Si durante el inicio sólo una ECU está en línea y si esta ECU transmite algún mensaje y no recibe una aceptación, detecta un error y repite el mensaje. Puede estar en error pasivo pero no como desconectado.

3.4. Capa aplicación

El protocolo de comunicaciones CAN 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 (HLP, *Higher Layer Protocols*)). Con la implementación de sistemas distribuidos basados en CAN han surgido nuevos requerimientos que no fueron considerados en el estándar ISO 11898, siendo los más importantes los siguientes [12]:

- Disponibilidad de servicios de transmisión para bloques de datos mayores a 8 octetos.
- Soporte al modelo cliente/servidor.
- Funciones dedicadas a la gestión de red.
- Métodos para asignar identificadores de mensaje y configuración de parámetros específicos de la ECU, 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 FB industriales. La ventaja de tener un estándar es considerable, ya que admite el desarrollo independiente de los componentes individuales de la red, lo cual permite a los fabricantes de automóviles utilizar componentes de diferentes proveedores. Actualmente la mayoría de los sistemas CAN que se implementan utilizan protocolos de capa superior propietarios.

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. Algunos ejemplos de capas de aplicación del protocolo de comunicaciones CAN son CAL, CANopen, DeviceNet, SAE J1939, SDS (*Smart Distributed System*), CANKingdom, CANaerospace [46].

Se han desarrollado estándares para definir y desarrollar software especifico de la capa de aplicación, entre ellos los más importantes son OSEK y AUTOSAR.

3.4.1. SAE J1939

Este protocolo define el control de comunicación en la capa de aplicación, fue desarrollado por la SAE y está dirigido a la industria automotriz en camiones y comunicacion de remolques, vehículos agrícolas y forestales así como sistemas de navegación en aplicaciones marítimas. El protocolo de comunicaciones SAE J1939 especifica cómo leer y escribir los datos, pero también cómo calibrar ciertos subsistemas. La velocidad de transmisión de datos en SAE J1939 es de 250 kbps, lo cual permite transmitir aproximadamente 1850 mensajes por segundo.

El hecho de que un identificador de mensaje CAN no sólo identifica el mensaje sino que también le da cierta prioridad en el bus, hace que el protocolo J1939 aborde una tarea difícil para un diseñador del sistema, esta tarea consiste en que el protocolo fija la prioridad del mensaje sin embargo esta prioridad puede ser inadecuada en algunos sistemas de control en tiempo real.

En los E.U.A., un camión pesado (*trailer*) puede tener el chasis de un fabricante OEM, con motor y transmisión de otros fabricantes. Esto permite que los fabricantes de camiones pesados pueden equipar sus vehículos con sistemas de diferentes fabricantes. Por ejemplo, camiones de diferentes marcas (*Volvo, Freightliner, Navistar*) pueden tener el mismo motor (*Cummings*). El protocolo J1939 es el estándar en motores diésel para el control de módulos del motor (ECM, *Engine Control Modules*).

3.4.2. CAL

Para facilitar la implementación de redes CAN en aplicaciones de control industrial la organización CiA estandarizó el protocolo de comunicaciones CAL. La funcionalidad de CAL consiste de los siguientes cuatro servicios, que se pueden configurar para diseñar sistemas con diferentes capacidades y requerimientos [28, 71]:

- Especificación de mensajes CAN (CMS, *CAN Message Specification*): proporciona los medios para la descripción e implementación de una comunicación orientada a objetos. Contiene distintos tipos de datos estructurados y diferentes objetos de comunicación

con características de transmisión. Mediante reglas de codificación, especifica un formato de datos común para todos los mensajes de la red.

- Gestión de la red (NMT, *Network Management*): soporta la configuración, manejo y monitoreo de las ECUs. Asegura un inicio ordenado de toda la red y contiene las medidas de precaución necesarias para la supervisión e intercambio/inserción de ECUs en tiempo de ejecución.

- Distribuidor de identificadores (DBT, *Identifier Distributor*): se encarga de la asignación dinámica de identificadores durante el inicio del sistema y trabaja en conjunto con el servicio NMT.

- Gestión de capa (LMT, *Layer Management*): se encarga de la configuración y especificación de parámetros de transmisión y recepción de las ECUs en la red.

3.4.3. CANopen

CANopen es un estándar de comunicaciones y dispositivos basado en CAL, es una norma de CiA, por lo tanto la especificación está disponible únicamente para miembros de la CiA. CANopen es ampliamente utilizado en Europa como estándar *de facto* para la aplicación de CAN en automatización industrial distribuida y sus campos de aplicación son: equipo médico, electrónica marítima, transporte público, domótica, entre otros. CANopen especifica los perfiles de comunicación (*communication profile*) y de dispositivos, que permiten un uso independiente de la aplicación de CAN. El perfil de comunicación define el mecanismo de comunicación subyacente. Existen perfiles de dispositivos (*device profile*) comunes en la automatización industrial, tales como módulos de entradas/salidas digitales y analógicas, codificadores y controladores. Los perfiles de dispositivos se puede configurar a través de CANopen independientemente de su fabricante ya que CANopen cuenta con un diccionario de objetos (OD, *Object Dictionary*) estandarizado en formato ASCII.

CANopen distingue en tiempo real entre el intercambio de datos y el intercambio de datos críticos. Proporciona comunicación estandarizada para los datos, los datos de configu-

ración, gestión de la red en tiempo real y ciertas funciones especiales, por ejemplo, sincronización de mensajes.

3.4.4. DeviceNet

El protocolo DeviceNet fue desarrollado originalmente por la empresa estadounidense Allen-Bradley (ahora propiedad de Rockwell Automation), y es ampliamente utilizado en automatización industrial distribuida. Se utiliza principalmente en los E.U.A. y Asia. DeviceNet adapta la tecnología de ControlNet, otro protocolo industrial desarrollado por Allen-Bradley, y la tecnología CAN, por lo que es de bajo costo, eficiente, simple y robusto en comparación con los protocolos basados en RS-485 tradicionales.

3.4.5. SDS

El protocolo de capa de aplicación SDS es un sistema de bus basado en CAN para la conexión de sensores y actuadores inteligentes en aplicaciones discretas e híbridas en la automatización industrial, aunado al crecimiento de la familia de productos que incluye dispositivos inteligentes como sensores de proximidad y fotoeléctricos, interfaces de control, control de motores y la distinta variedad de plataformas de control, como controladores lógicos programables (PLC, *Programmable Logic Controller*) y sistemas de control basados en una computadora personal; el protocolo SDS cumple con los requerimientos establecidos para la conexión de estos sistemas.

Algunas de sus principales características son la utilización de métodos de detección, corrección de errores y confiabilidad en el reconocimiento de mensajes, también proporciona un conjunto de mensajes que abarca desde mensajes de cambio de estado controlados por eventos, hasta operaciones complejas transportando valores binarios, analógicos y alfanuméricos.

3.4.6. CANKingdom

CANKingdom es un protocolo de capa de aplicación y fue publicado en 1990 se considera antecesor de protocolos de capas de aplicación altas basadas en CAN. CANKingdom es

utilizado en sistemas de control de movimiento y máquinas, también se utiliza en la industria marítima.

CAN Kingdom describe un sistema como un país (*country*), un reino (*Kingdom*), con una capital y ciudades (*capital and cities*). El rey (*king*) gobierna al reino desde la capital y cada ciudad tiene un Alcalde (*Mayor*), responsable del gobierno local. El único medio para comunicarse dentro de la ciudad es el correo (*mail*). La red CAN se describe como el sistema postal real (*The royal postal system*), cada ciudad tiene una oficina de correos (*post office*) y un director de correos (*postmaster*), el cual simboliza a un controlador CAN. Cada ciudad produce algo y puede importar o exportar información por correo. El alcalde de la ciudad organiza cualquier información de importación o exportación dentro de listas, estas listas forman parte de la documentación del módulo. El diseñador de sistemas elige los módulos específicos que se utilizarán en su máquina, para ello debe conocer completamente las listas, por ejemplo, el identificador local de cada variable.

CANKingdom permite el cambio de comportamiento de la red en cualquier momento incluso mientras el sistema está funcionando, por ejemplo CANKingdom permite la solución de problemas del sistema para apagar los nodos individuales; los identificadores de nodo CAN y las condiciones de disparo para el envío de mensajes se pueden cambiar mientras el sistema está funcionando, un ejemplo es cuando la reconfiguración de la red en tiempo real está en condiciones de fallo; otro ejemplo es la pérdida de un enlace de radio de la ECU en una aplicación marítima, el monitor de red, también conocido como el Rey, primero apaga el nodo de radio para evitar el envío de más comandos, y luego le dice a los nodos apropiados para obtener los datos del Rey, esta operación elimina el problema de un nodo que recibe dos comandos conflictivos simultáneos, también elimina el problema de dos nodos de envío con el mismo identificador CAN [74].

3.4.7. CANaerospace

El protocolo CANaeroespace originalmente fue desarrollado en Alemania por los sistemas de almacén de vuelos (*Stock Flight Systems*). La NASA y el programa de experimentos generales avanzados en aviación (AGATE, *Advanced General Aviation Experiments*) han

estandarizado el protocolo de capa de aplicación de próxima generación para su uso en la aviación en el año 2001. Sus principales ventajas son la fiabilidad, la sencillez y la auto identificación del formato de mensajes, además de apoyar la interoperabilidad de los sistemas producidos por diferentes fabricantes. CANaerospace se utiliza como una red de sistemas distribuidos en el avión monomotor turbohélice Ae270 y ha sido certificado por las autoridades aeronáuticas de la República Checa.

El protocolo CANaerospace está presente en otros programas de la NASA como Sofía, donde sirve como enlace de datos entre varios sistemas de control en tiempo real para el telescopio de la astronomía infrarroja, que se extendió por todo el fuselaje de la aeronave de investigación de Boeing 747SP. El avión Sofía es una colaboración entre la NASA y la Organización de Investigación Aeronáutica Alemana (*German Aeronautics Research Organization*). El avión Sofía será operado desde el Centro de Investigación Ames de la NASA en *Moffet Field*, California. Debido a la buena experiencia durante la integración de sistemas, el Centro de Investigación Langley de la NASA está evaluando el uso de CANaerospace en otros aviones con fines de investigación [75, 76].

3.4.8. OSEK

En la década de 1990 los fabricantes OEM introdujeron OSEK/VDX como un sistema operativo multitarea y en tiempo real, con la finalidad de establecer un estándar uniforme para el software de las ECUs en el desarrollo embebido de la industria automotriz. Sus propiedades incluyen bajo consumo de energía de los procesadores, recursos de memoria, manejo de eventos y tareas. OSEK/VDX introduce una arquitectura abierta que comprende tres áreas [77]:

- Comunicación (*OSEK COM*): proporciona servicios para el intercambio de datos entre tareas y/o rutinas de servicio de interrupción (*ISR*), comunicación interna y externa entre diferentes ECU; el acceso a los servicios de comunicación OSEK se realiza mediante interfaces de programación de la aplicación específica (API, *Application Program Interface*).

- Gestión de red (*Network Management*): define un conjunto de servicios para determinar y supervisar la configuración del nodo. Debe adaptarse a los requerimientos específicos del sistema de bus utilizado (métodos globales) o a los recursos de cada nodo (métodos locales).

- Sistema operativo (OS, *Operating System*): las aplicaciones automotrices se caracterizan por tener requerimientos de tiempo real rigurosos. El OS de OSEK proporciona la funcionalidad necesaria para dar soporte a sistemas de control manejados por eventos. Los servicios especificados por el OS constituyen una base para hacer posible la integración de módulos software realizados por distintos fabricantes.

3.4.9. AUTOSAR

El sistema de arquitectura abierta automotriz (AUTOSAR, *Automotive Open System Architecture*) es una arquitectura estandarizada de software para aplicaciones automotrices. Fue desarrollada por fabricantes de automóviles, proveedores y desarrolladores. Es una alianza entre fabricantes OEM y proveedores de trabajar juntos para establecer un estándar *de facto* en la industria automotriz. Una de las características importantes de AUTOSAR es el soporte para el reúso de software y hardware de los componentes electrónicos en el sistema del automóvil [78]. Las metas de AUTOSAR son:

- Cumplimiento de requerimientos automotrices futuros, como: disponibilidad, seguiridad, mantenimiento, actualización y mejora del software.

- Incremento de la escalabilidad y flexibilidad de integrar y transferir funciones.

- Alta penetración en el mercado de software y hardware a través de líneas de productos.

- Mejora de contención de productos y procesos de complejidad y riesgos.

- Optimización de costos de sistemas escalables.

Capítulo 4

Especificación formal del protocolo de comunicaciones CANFD

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, en 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 únicos.

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. Las especificaciones son realizadas en documentos textuales en la mayoría de los casos, 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 se necesitan comunicar las ECUs en los vehículos, el protocolo CAN FD fue creado 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 el protocolo en un nivel más profundo con el fin de ser capaz de tomar ventaja de sus capacidades. Por lo tanto, en este capítulo se presenta la especificación formal del protocolo de comunicaciones CAN FD utilizando el FDL Cinderella SDL.

4.1. Requerimientos y acotamiento del sistema

En base a los requerimientos del sistema de comunicaciones CAN FD (Capítulo 3) se definen las siguientes restricciones respecto a la especificación:

- Este trabajo se enfoca en la especificación de las capas DLL y de supervisor, de acuerdo al modelo de referencia OSI (véase Figura 1.2).
- Debido a que el medio físico no está definido en la especificación del protocolo de comunicaciones CAN y CAN FD, 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 FD está formado por dos nodos activos en el bus, ya que es el mínimo de nodos necesarios para establecer un enlace de comunicaciones CAN FD.

4.2. Objetos de la especificación

Una vez estudiados y analizados los estándares ISO 11898 [33] e ISO 11519 [34] y las especificaciones CAN [32] y CAN FD [36], el siguiente paso es la realización de la especificación

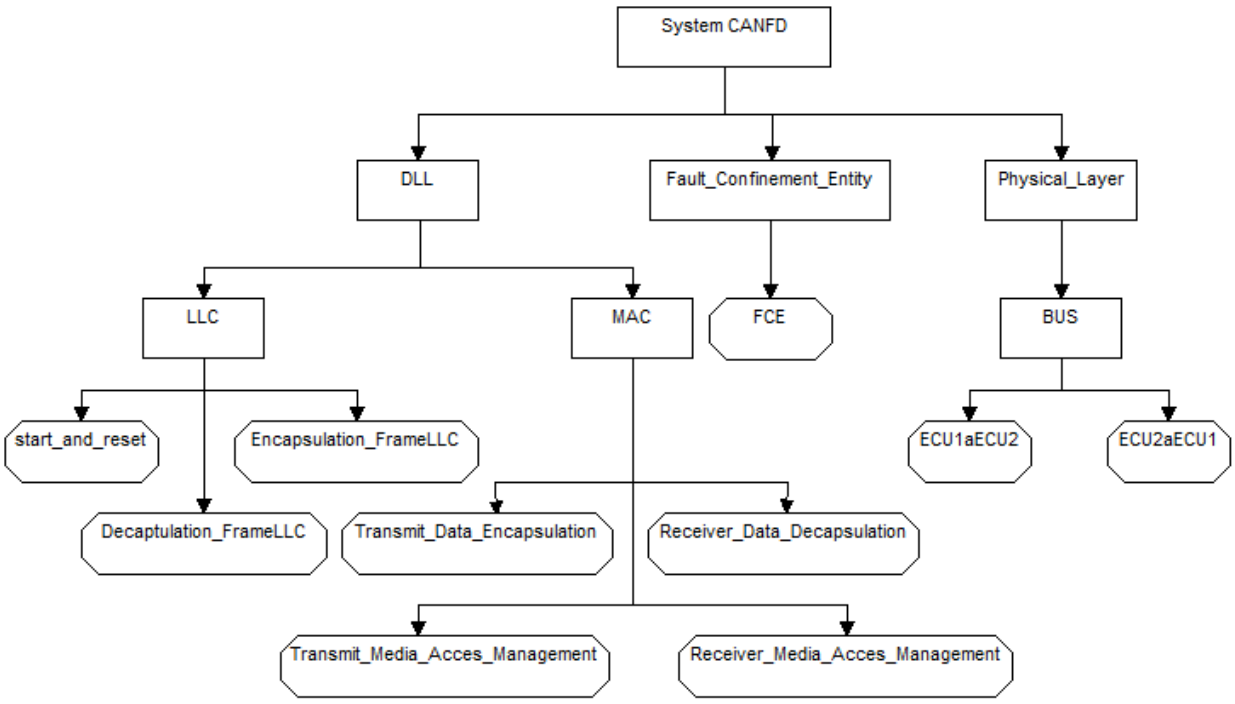

Figura 4.1: Jerarquía de los objetos de la especificación.

formal, es decir, 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 describen los objetos creados para el desarrollo de la especificación del sistema, estos consisten en bloques, sub-bloques y procesos que constituyen la especificación estática del protocolo de comunicaciones CANFD. Los nombres asociados a cada objeto describen la función 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 4.1 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:

- Una unidad que realiza la función del medio físico, representando una topología de bus.
- Un conjunto de ECUs (dos para la simulación), cada una de las cuales está formada por tres objetos que representan las capas física, DLL y de supervisor.

1. `Physical_Layer`. Realiza las funciones de la capa física.
2. `Fault_Confinement_Entity`. Realiza las funciones de la capa de supervisor.
3. `DLL`. 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 CAN FD.
   - `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.

4.3. Especificación de datos

Los tipos de datos [79] usados para la especificación del protocolo CAN FD se declaran en la librería (*package*) `CAN_Datos` empleando la norma ASN.1, esta librería está declarada y enlazada al nivel de entorno del sistema, lo cual permite que sean accesibles en los demás niveles de abstracción.

La Figura 4.2 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 FD ya que con éstos se permite crear la mayoría de los datos empleados:

- `Byte`: Crea una cadena de 8 bits para la creación de los datos del protocolo.
- `Bit1`: Crea un arreglo de un bit para identificar el formato (estándar o extendido) del protocolo.
- `Bit4`: Crea un arreglo de 4 bits para el campo DLC.
- `Bit_11`: Crea un arreglo de 11 bits para el identificador base de cada ECU CAN FD.
- `Bit_18`: Crea un arreglo de 18 bits para la extensión del identificador de cada ECU CAN FD.

```
/* Tipos de datos del Usuario CAN*/

SYNTYPE Byte = Bitstring
      constants size(8)
ENDSYNTYPE;

SYNTYPE Bit4 = Bitstring
      constants size(4)
ENDSYNTYPE;

SYNTYPE Bit1 = Bitstring
      constants size(1)
ENDSYNTYPE;

SYNTYPE Bit_11 = Bitstring
      constants size(11)
ENDSYNTYPE;

SYNTYPE Bit_18 = Bitstring
      constants size(18)
ENDSYNTYPE;
```

```
/*Definición del tamaño para el procesamiento de datos  del CRC*/

Syntype TamCRC15= Bitstring
      constants size(15)
endSyntype;

Syntype TamCRC17= Bitstring
      constants size(17)
endSyntype;

Syntype TamCRC21= Bitstring
      constants size(21)
endSyntype;

/*Definición del tamaño para el polinomio divisor del  CRC*/

Syntype KEY16 = Bitstring
      constants size (16)
EndSyntype;

Syntype KEY18 = Bitstring
      constants size (18)
EndSyntype;

Syntype KEY22 = Bitstring
      constants size (22)
EndSyntype;

/*Polinomio */
Synonym KeyCRC16 KEY16 = '1100010110011001'B;
Synonym KeyCRC18 KEY18 = '110110100001011011'B;
Synonym KeyCRC22 KEY22 = '1100000010100010011001'B;
```

Figura 4.2: Tipos de datos utilizados en el protocolo de comunicaciones CAN FD.

- `TamCRC15`: Tiene un tamaño de 15 bits y se utiliza para almacenar el CRC_15 de cada trama.
- `TamCRC17`: Tiene un tamaño de 17 bits y se utiliza para almacenar el CRC_17 de cada trama.
- `TamCRC21`: Tiene un tamaño de 21 bits y se utiliza para almacenar el CRC_21 de cada trama.
- `KEY16`: Crea un arreglo de 16 bits para el polinomio generador del CRC_15.
- `KEY18`: Crea un arreglo de 18 bits para el polinomio generador del CRC_17.
- `KEY22`: Crea un arreglo de 22 bits para el polinomio generador del CRC_22.

4.4. Especificación estática

Con los objetos identificados se construye la descripción estática del sistema en SDL que los organiza en varios niveles de abstracción (véase Figura 4.1), y define cómo debe reaccionar cada uno de ellos a los eventos externos mediante la utilización de señales.

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

4.4.1. 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. Dado 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.

4.4.1.1. Especificación de señales

Las señales son la base para establecer la comunicación entre las distintas entidades del sistema y se transportan a través de las rutas especificadas, canales o rutas de señal. La Figura 4.3 muestra las señales descritas en los estándares ISO 11898 [33] e ISO 11519 [34], 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 ECU CAN FD y para evitar ambigüedades se renombró las señales de la segunda ECU.

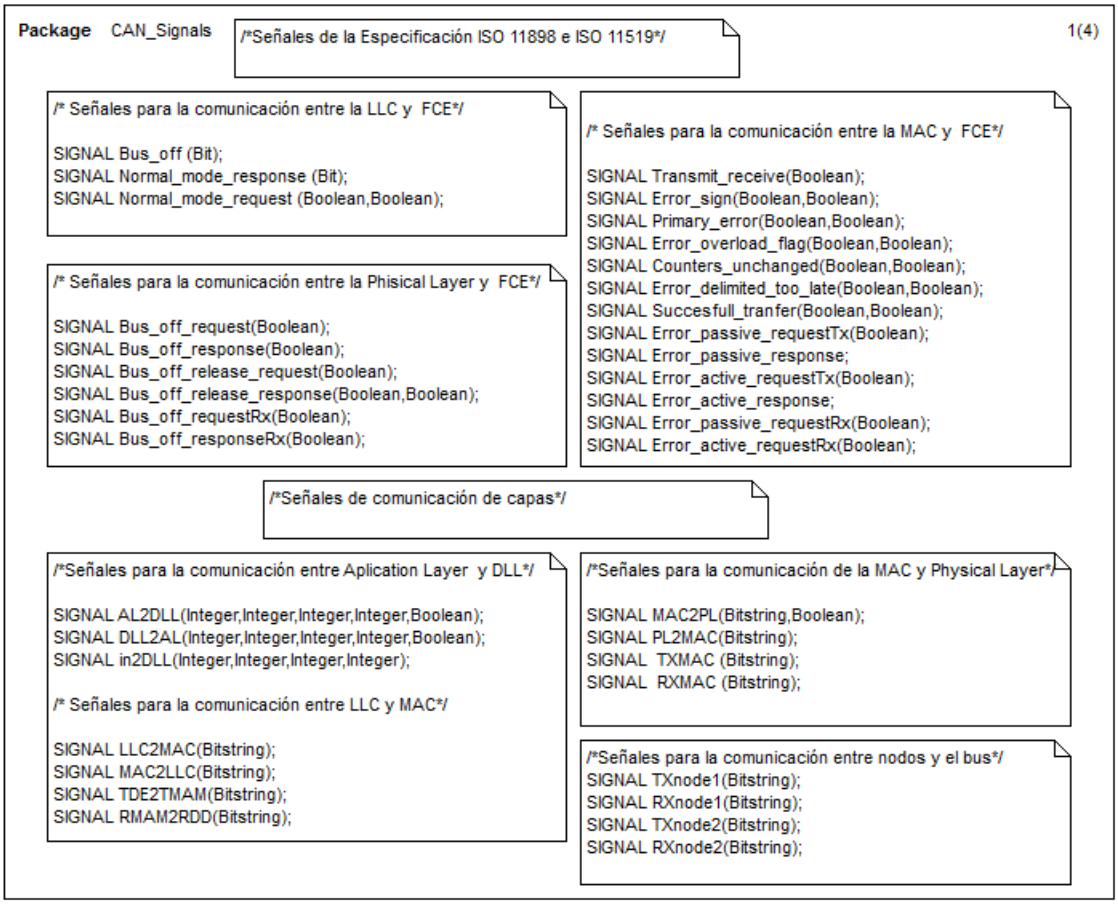

Figura 4.3: Declaración de las señales para una ECU CAN FD.

Algunas de estas señales son parametrizadas, es decir, transportan información necesaria para procesamiento o control del protocolo.

4.4.2. Sistema CAN FD

Las señales, procedimientos y tipos de datos se declaran en las librerías `CAN_Signals`, `CAN_Procedimientos` y `CAN_Datos` respectivamente (veáse Figura 4.4). La especificación del sistema CAN FD está conformada por los bloques `Physical_Layer`, `DLL`, `Fault_Confinement_Entity`, que corresponden al modelado de la ECU 1, para el modelado de la ECU 2 se usan los bloques `DLL2` y `Fault_Confinement_Entity2`. La Figura 4.5 muestra la relación y la arquitectura de los bloques a nivel sistema. Dado que los bloques `DLL` y `DLL2` junto con los bloques `Fault_Confinement_Entity` y `Fault_Confinement_Entity2` son iguales, en el

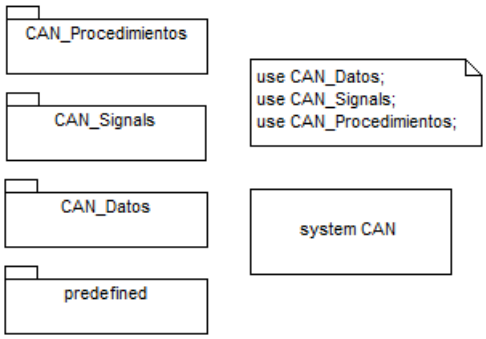

Figura 4.4: Librerías de la especificación del protocolo CAN FD.

presente documento únicamente se detallarán los bloques y procesos correspondientes a la ECU 1.

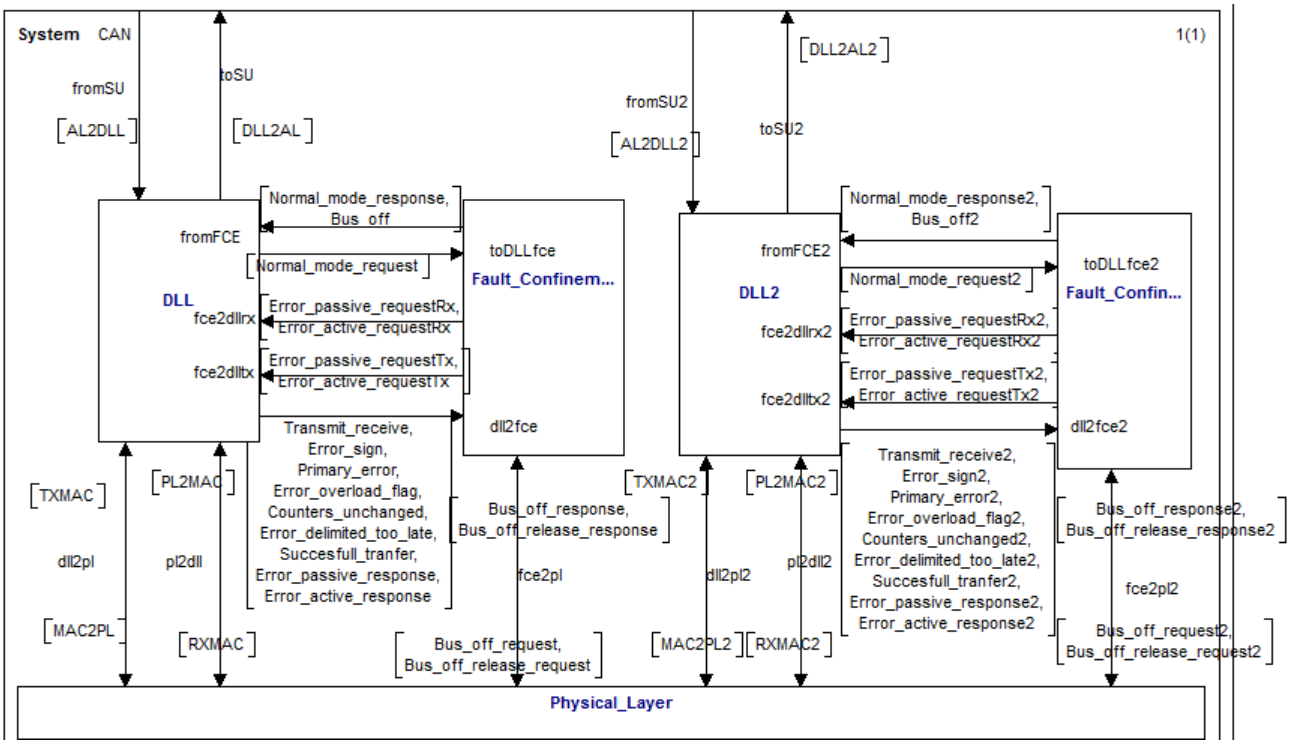

Figura 4.5: Especificación del sistema CAN FD.

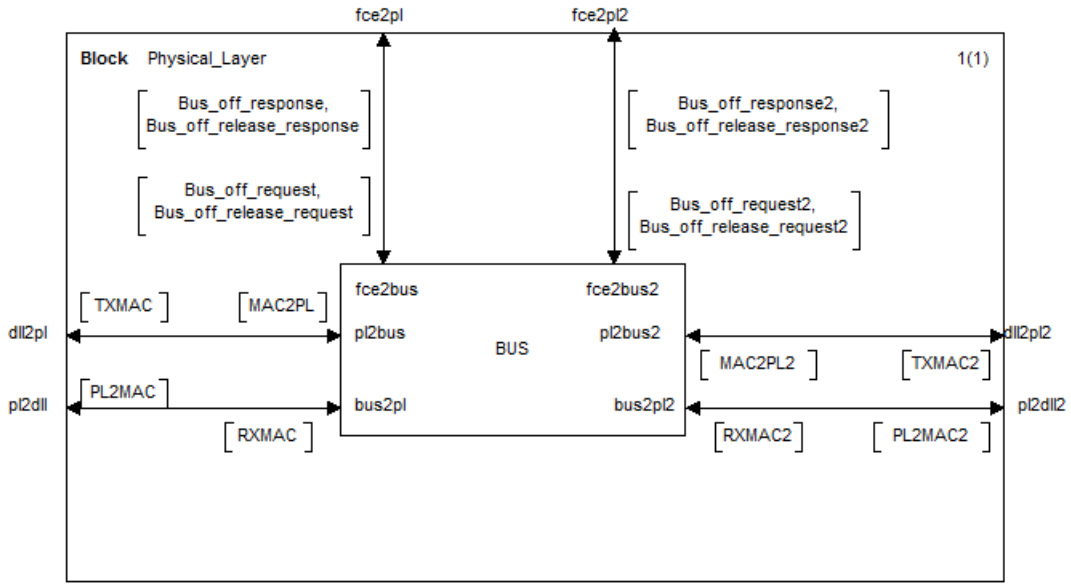

Figura 4.6: Diagrama SDL del bloque `Physical_Layer`.

4.4.2.1. Bloque Physical_Layer

El bloque `Physical_Layer` modela una topología de bus, basada en conexiones punto a punto, y representa el medio físico para la transmisión de datos mediante el cual se establece un enlace de comunicación entre los nodos conectados a la red CAN FD.

El bloque `Physical_Layer` se compone de un bloque llamado `BUS` conectado a los canales `dll2pl`, `pl2dll`, `fce2pl`, `dll2pl2`, `pl2dll2`, `fce2pl2` a través de las rutas de señal `pl2bus`, `bus2pl`, `fce2bus`, `pl2bus2`, `bus2pl2`, `fce2bus2`, respectivamente (véase Figura 4.6).

4.4.2.1.1. Bloque BUS

El bloque `BUS` realiza la función de intercambiar datos entre las ECU1 y ECU2, se compone de dos procesos nombrados `ECU1aECU2` y `ECU2aECU1` conectados a los canales `pl2bus`, `bus2pl`, `fce2bus`, `pl2bus2`, `bus2pl2`, `fce2bus2` mediante las rutas de señal `n1tx`, `n1rx`, `bus2node1`, `n2tx`, `n2rx`, `bus2node2`, respectivamente (véase Figura 4.7). Cuando se establece una solicitud de enlace de comunicación por medio de la señal `Bus_off_release_request`, se le indica al bus que la ECU comenzará a transmitir o recibir dependiendo de la solicitud de la ECU. La respuesta a esta solicitud es enviada por la señal `Bus_off_release_response`

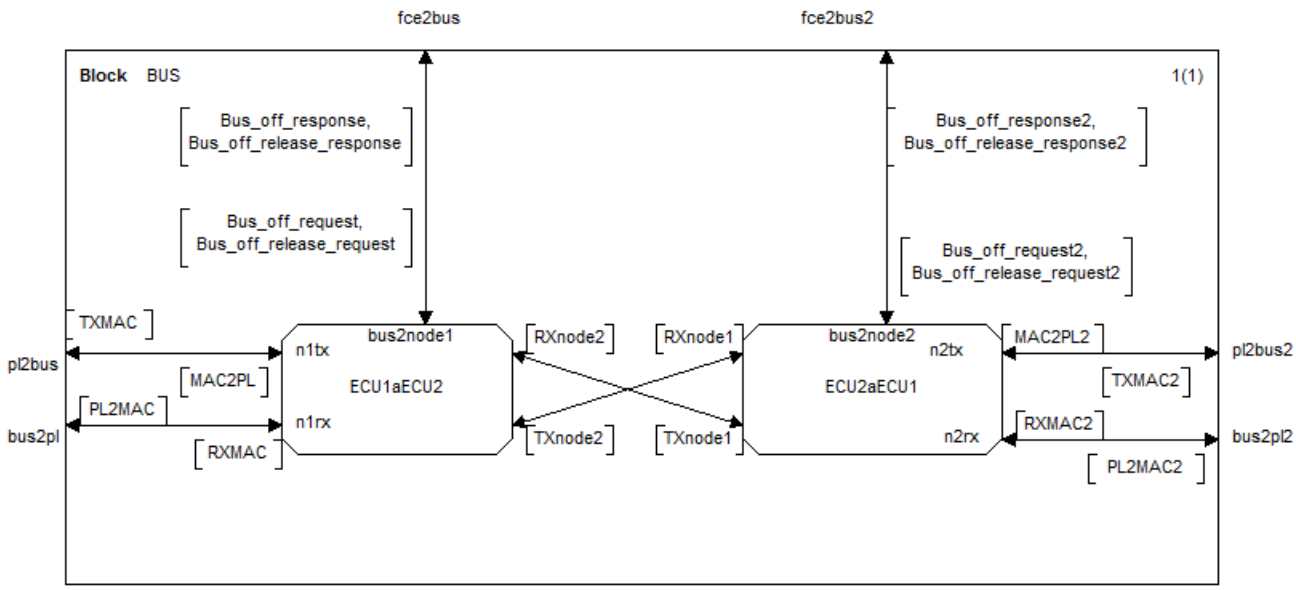

Figura 4.7: Diagrama SDL del bloque `BUS`.

a la ECU solicitante. El intercambio de datos entre las ECU1 y ECU2 se realiza mediante las señales `RXnode1`, `TXnode1`, `RXnode2` y `TXnode2`.

4.4.2.2. Bloque DLL

La Figura 4.8 muestra la arquitectura SDL de la capa DLL, en donde el bloque `DLL` se compone de los bloques `LLC` y `MAC` correspondientes a las subcapas definidas en los estándares ISO 7498 [72] e ISO 8802-3 [73]. Estas estaciones se encargan del envío y recepción de los datos provenientes de las capas superiores, capa física y capa de supervisor. Los datos de la capa de aplicación se reciben por la señal `AL2DLL` y se transmiten por la señal `DLL2AL`; la trama CAN FD se transmite al bus mediante la señal `MAC2PL` y la ECU lo recibe mediante la señal `PL2MAC`.

4.4.2.2.1. Bloque LLC

El bloque `LLC` se compone de tres procesos `start_and_reset`, `Encapsulation_FrameLLC` y `Decaptulation_FrameLLC` (véase Figura 4.9) que realizan la máquina de estados corres-

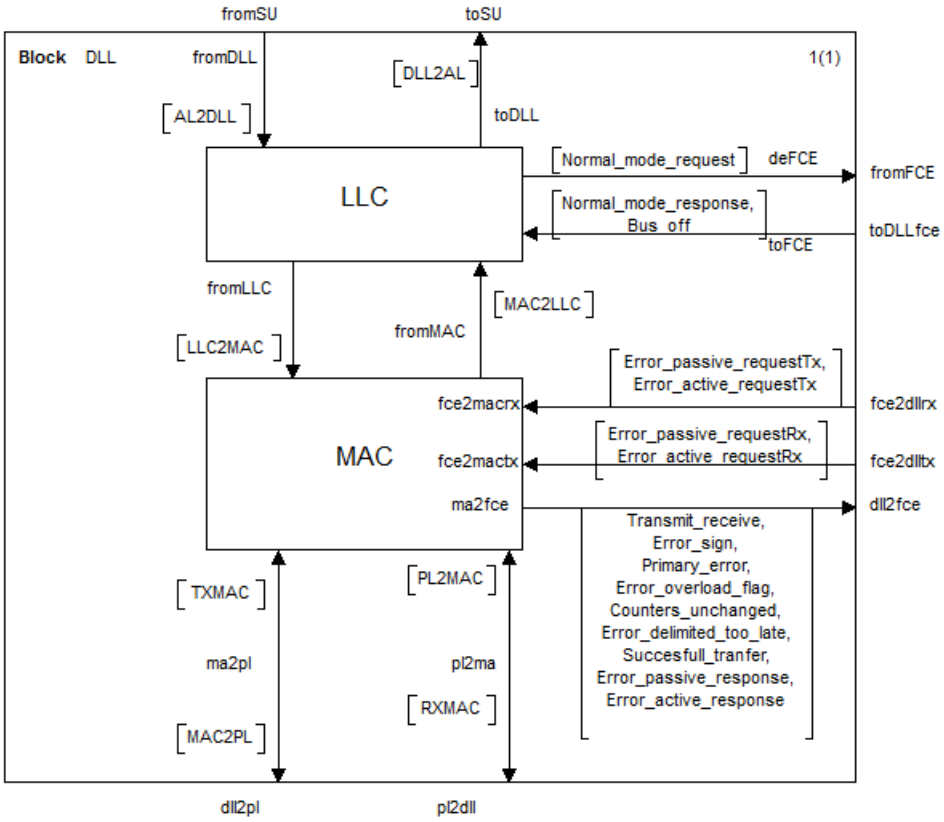

Figura 4.8: Diagrama SDL del bloque `DLL`.

pondiente al comportamiento descrito en la sección 3.2.1, además este bloque se encarga de interactuar con la capa de supervisor para establecer la comunicación con el bus y presentar los datos al bloque `MAC`.

4.4.2.2.2. Bloque MAC

El bloque `MAC` es el núcleo del protocolo de comunicaciones CAN FD. Su función es presentar los mensajes recibidos de la subcapa LLC y realizar la transmisión. Se compone de dos partes totalmente independientes, la parte de transmisión y la de recepción (véase Figura 4.10). Los procesos para la transmisión son: `Transmit_Data_Encapsulation`, `Transmit_Media_Acces_- Management`, mientras que los procesos para la recepción son: `Receiver- _Data_Decapsulation`, `Receiver_Media_Acces_Management` que realizan la máquina de estados correspondiente al comportamiento descrito en la seccción 3.2.2.

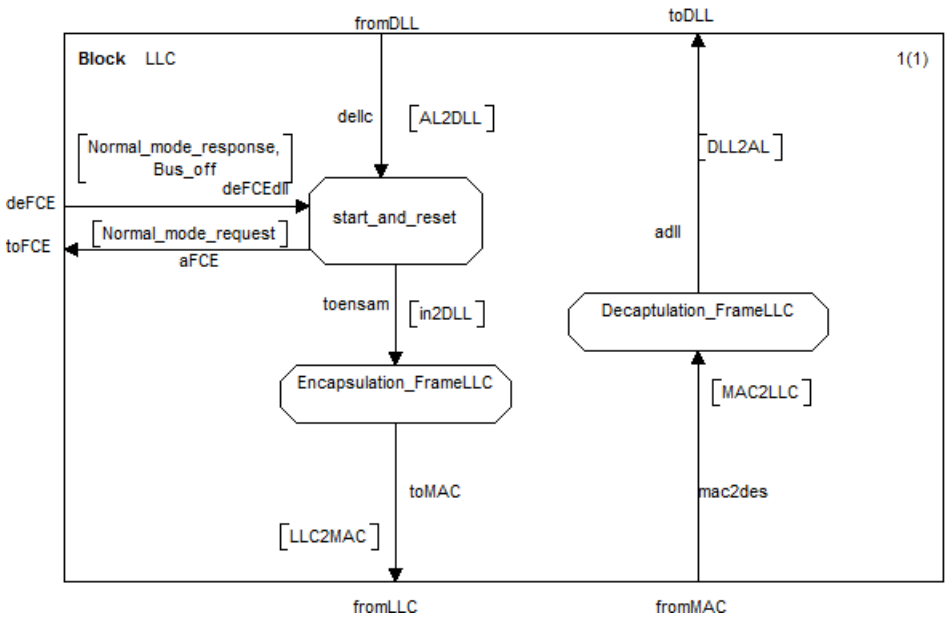

Figura 4.9: Diagrama SDL del bloque `LLC`.

4.4.2.3. Bloque Fault_Confinement_Entity

La Figura 4.11 muestra la arquitectura SDL de la capa de supervisor, en donde el bloque `Fault_Confinement_Entity` está compuesto por el proceso `FCE` con el cual se realiza la máquina de estados de dicha estación. El principal objetivo de este bloque es preservar la disponibilidad del sistema de transmisión de datos en caso de un nodo defectuoso. Las estrategias que debe proveer este bloque son: Distinción entre errores temporales y fallos permanentes y localización y apagado de nodos defectuosos.

4.5. Especificación dinámica

La especificación dinámica describe el comportamiento del sistema mediante los procesos y máquinas de estados que lo conforman, la funcionalidad de dichos procesos ya se ha definido en la especificación estática y a continuación se detalla el funcionamiento interno de las máquinas de estado.

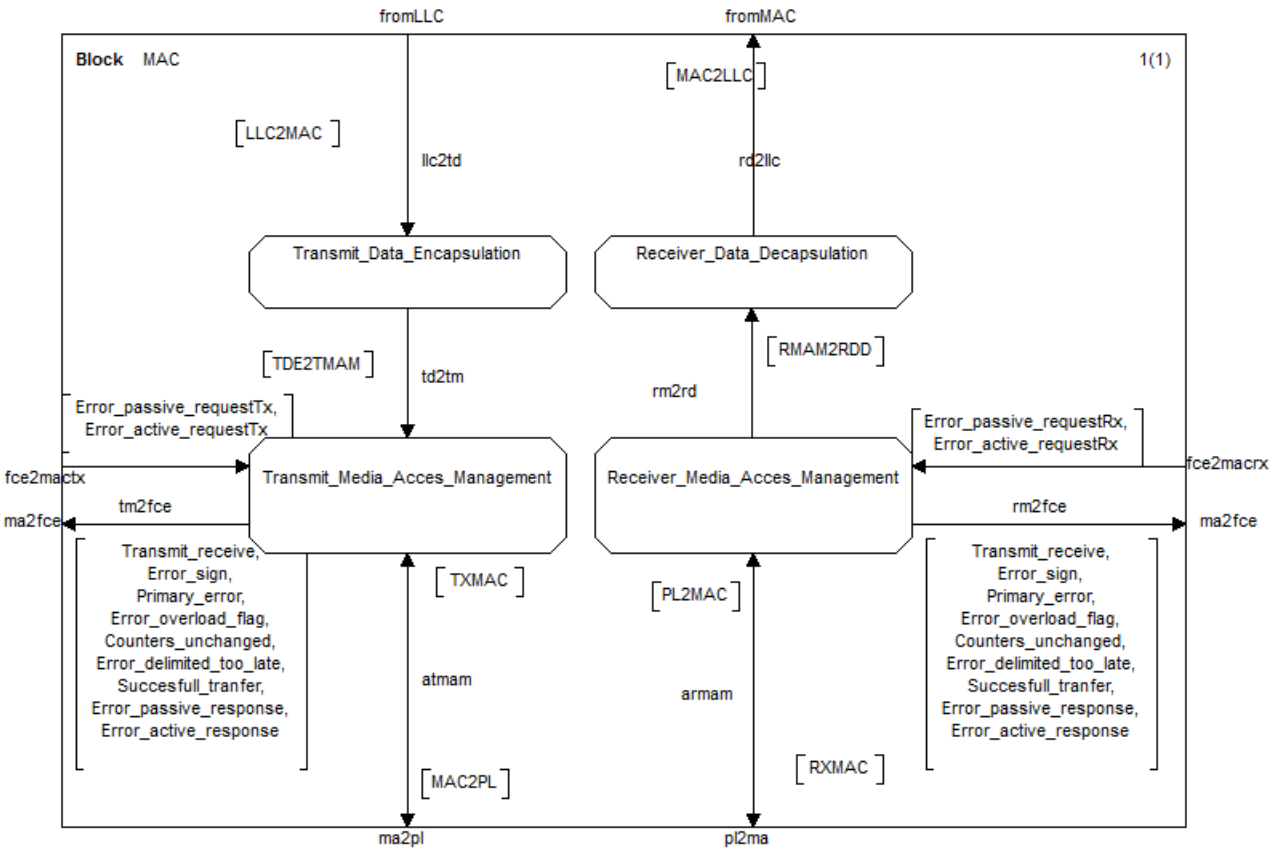

Figura 4.10: Diagrama SDL del bloque `MAC`.

Debido a que la especificación de los procesos es muy extensa, a excepción de los procesos `start_and_reset`, `Encapsulation_FrameLLC` y `Decaptulation_FrameLLC`, la descripción de cada proceso se realiza de la siguiente forma:

- Se usa 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 puertas de enlace, listas de señales y 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 correspondientes a los estados y las señales de entrada (también puede contener los símbolos de las señales de salida). Simplifica la representación del interior de los procesos, ocul-

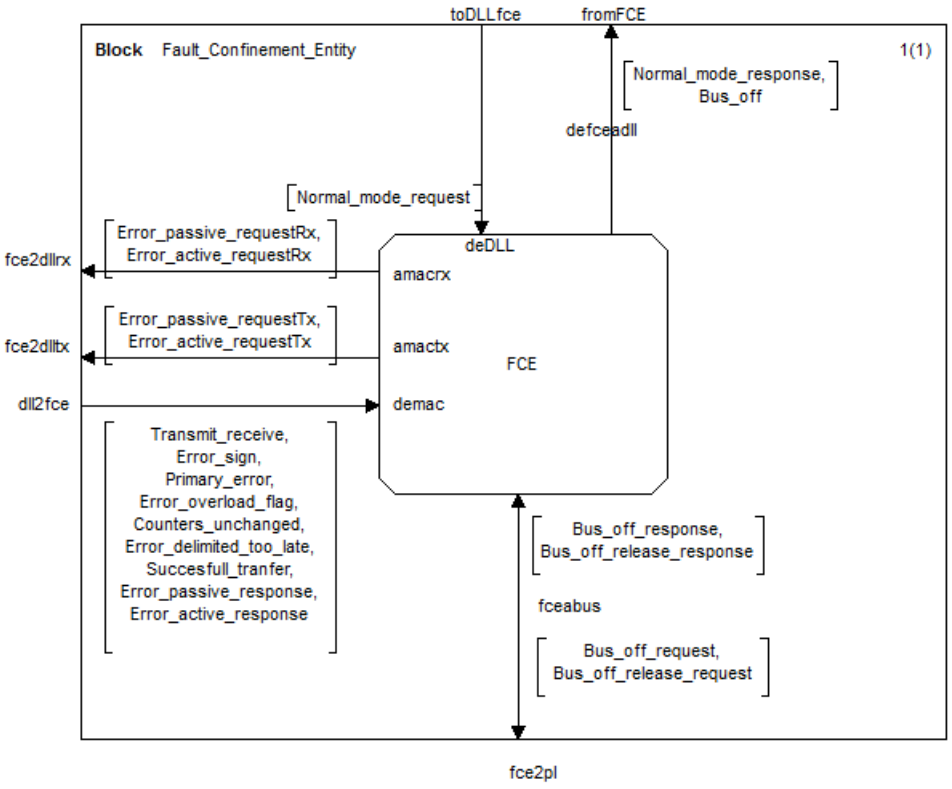

Figura 4.11: Diagrama SDL del bloque `Fault_Confinement_Entity`.

tando información sobre las tareas realizadas en las transiciones, de tal forma que se pueda visualizar la evolución de procesos complejos en un sólo diagrama.

- La descripción de los procesos sigue el orden de los bloques: `LLC`, `MAC`, `FCE` y `BUS`.

Para una revisión exhaustiva de la especificación formal del protocolo CAN FD, se entregan los siguientes documentos conjuntamente con este trabajo de tesis:

- Especificación formal en formato $*$.cbf: Formato de la especificación generado por la herramienta CinderellaSDL.
- Especificación formal en formato $*$.pdf: En caso que el lector no disponga de la herramienta CiderellaSDL, pueda acceder a la especificación formal.
- Diagrama MSC en formato $*$.pdf: Los diagramas MSC (*Message Sequence Chart*) representan el resultado de la prueba (ejecución) de la simulación como una secuencia

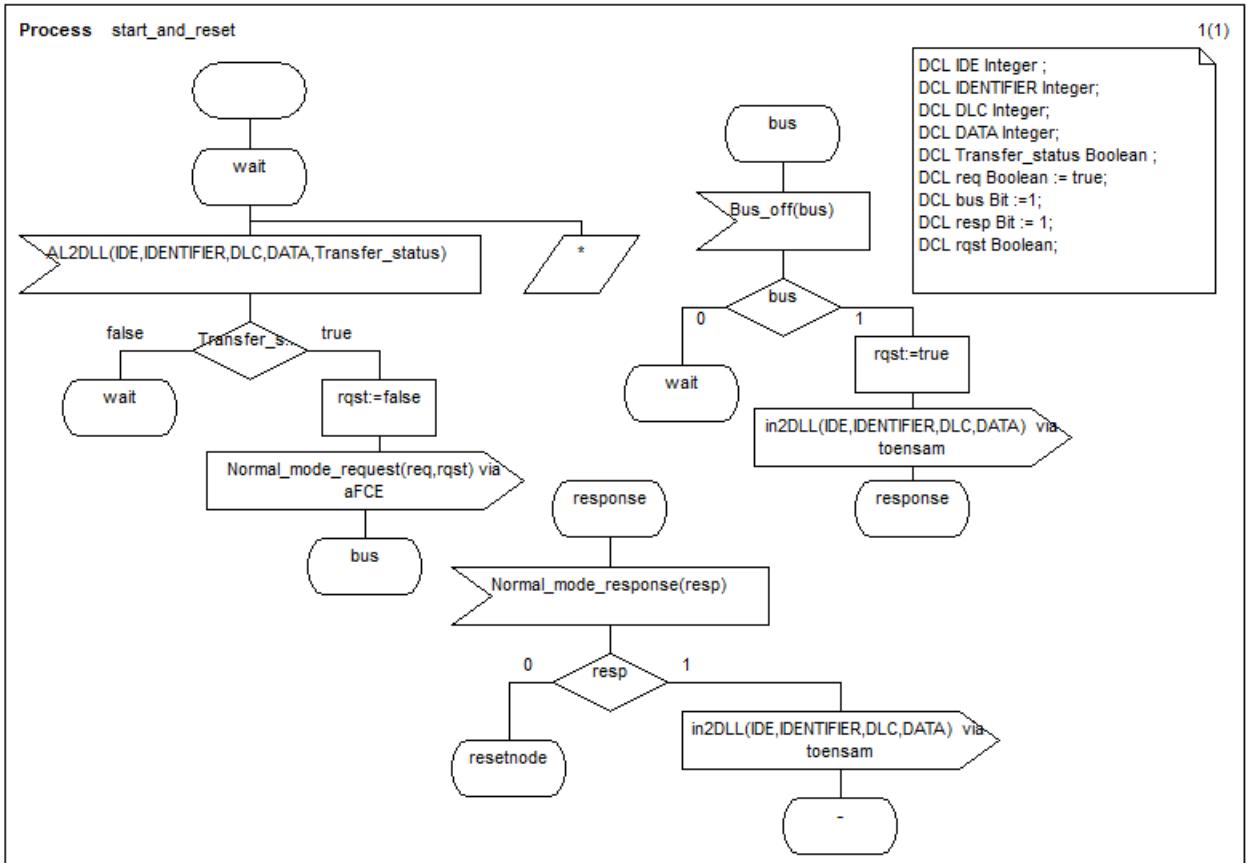

Figura 4.12: Descripción del proceso `Start_and_reset`.

de mensajes ordenada en el tiempo, los cuales se han usado durante la evolución del proyecto y se abordan en el Capítulo 5 para documentar la evaluación de resultados.

4.5.1. Proceso Start_and_reset

Una vez que el proceso `Start_and_reset` (véase Figura 4.12) ejecuta el símbolo de inicio, se procede a enviar la solicitud de configuración inicial hacia el bloque `Fault_Confinement_Entity` mediante la señal `Normal_mode_request`, la respuesta a la solicitud de configuración es recibida por medio de la señales `Normal_mode_response` y `Bus_off`, en caso de que el bus este libre se envían los datos de la capa de supervisor al siguiente bloque por medio de la señal

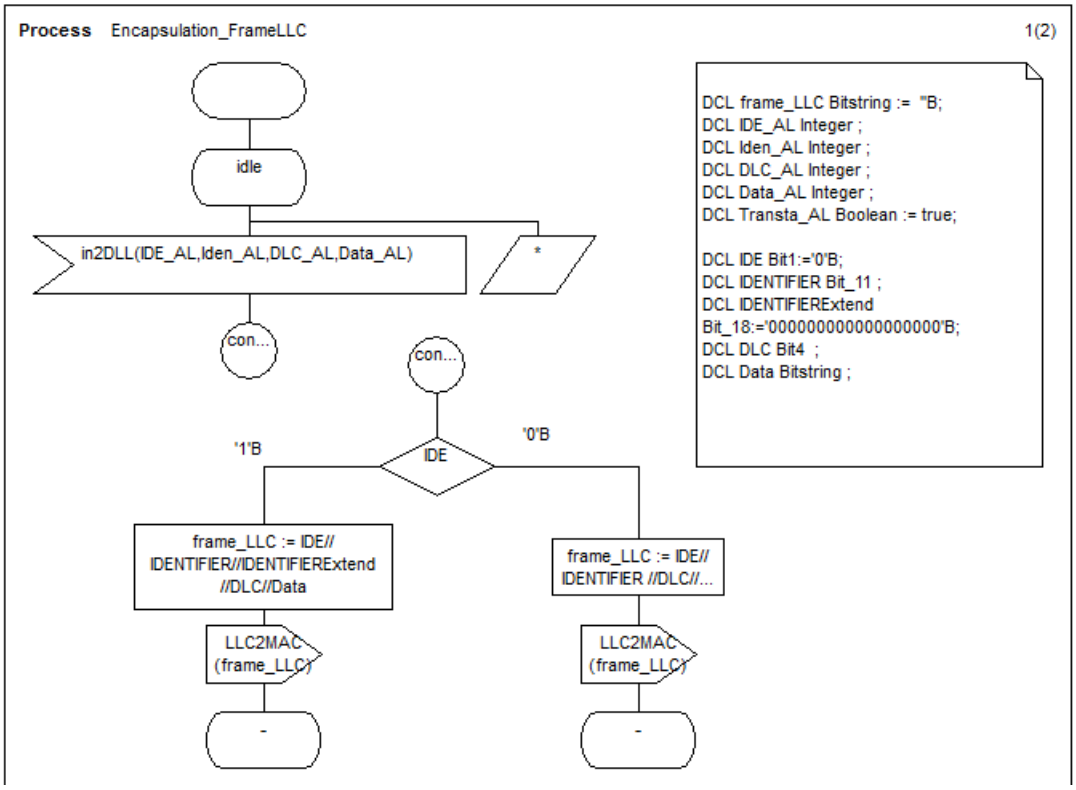

Figura 4.13: Descripción del proceso `Encapsulation_FrameLLC`.

`in2DLL`, en caso contrario espera hasta que el bus se encuentre libre para que la ECU pueda transmitir.

4.5.2. Proceso Encapsulation_FrameLLC

El proceso `Encapsulation_FrameLLC` forma la unidades de datos de acuerdo a la estructura y formato de la trama LLC (véase Figura 4.13). Al ejecutar el símbolo de inicio, se lee la señal `in2DLL` la cual tiene los parámetros de `IDE_AL`, `IDENTIFIER_AL`, `DLC_AL` y `DATA_AL` en formato entero (*integer*), estos valores enteros 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.

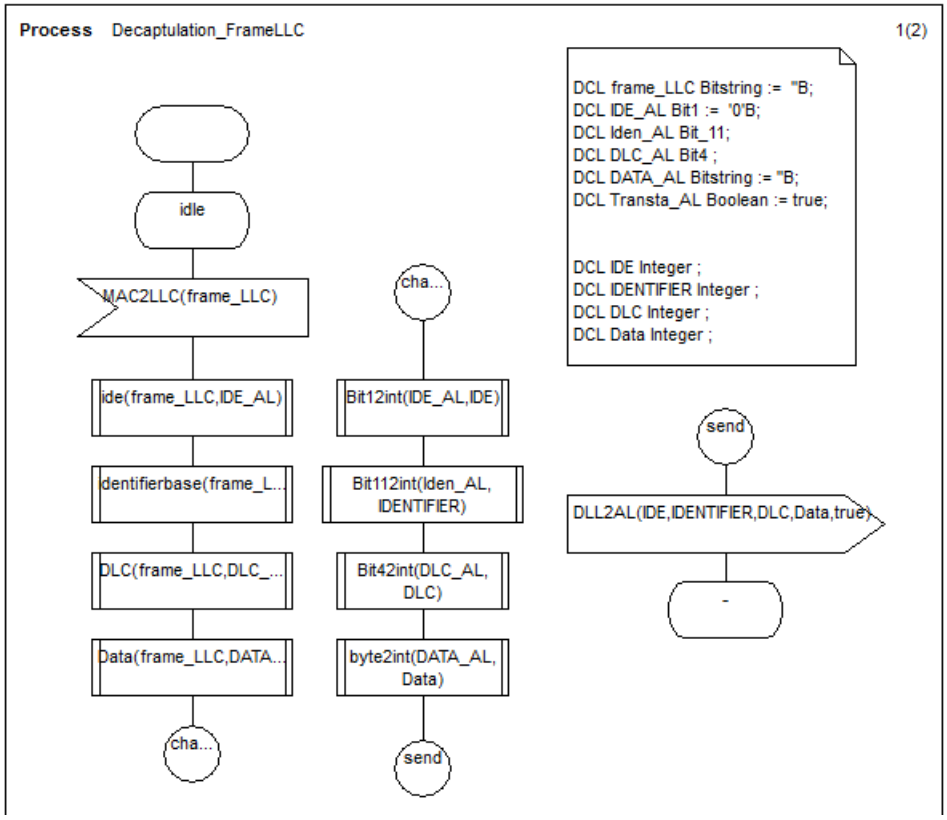

Figura 4.14: Descripción del proceso `Decaptulation_FrameLLC`.

4.5.3. Proceso Decaptulation_FrameLLC

La Figura 4.14 muestra la descripción del proceso `Decaptulation_FrameLLC`; una vez ejecutado el símbolo de inicio se pasa al estado `idle`, 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 una conversión de tipos de datos, de binarios a enteros y se pasan a la capa superior por medio de la señal parametrizada `DLL2AL`.

4.5.4. Proceso Transmit_Data_Encapsulation

El proceso `Transmit_Data_Encapsulation` realiza las siguientes funciones (véase Figura 4.15):

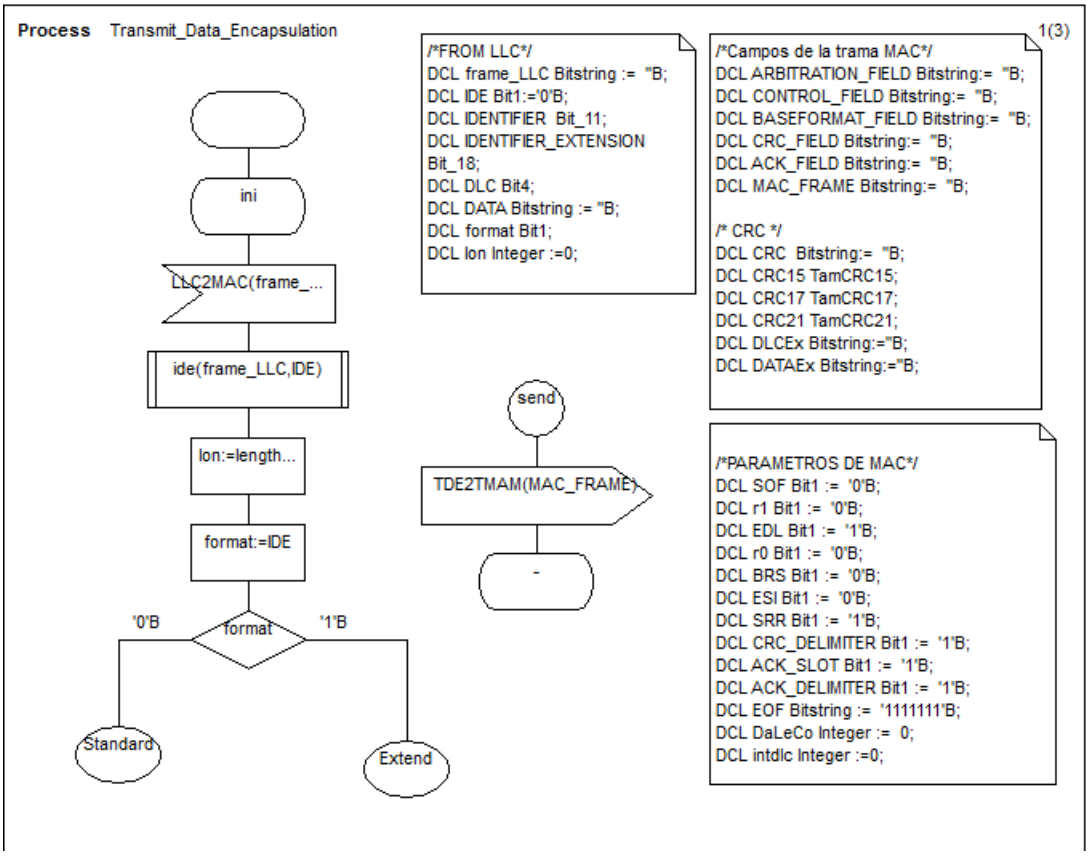

Figura 4.15: Descripción del proceso `Transmit_Data_Encapsulation`.

1. Aceptación de las tramas LLC y la información de la interfaz de control.

2. Cálculo de la secuencia CRC. El cálculo del CRC se realiza de manera paralela y se selecciona el CRC que formará la trama dependiendo de los bits DLC.

3. Construcción de la trama MAC agregando los bits SOF, r1, EDL, r0, BRS, ESI, SRR y EOF a la trama LLC dependiendo del formato estándar o extendido.

4.5.5. Proceso Transmit_Media_Acces_Management

El proceso `Transmit_Media_Acces_Management` realiza las siguientes funciones (véase Figura 4.16):

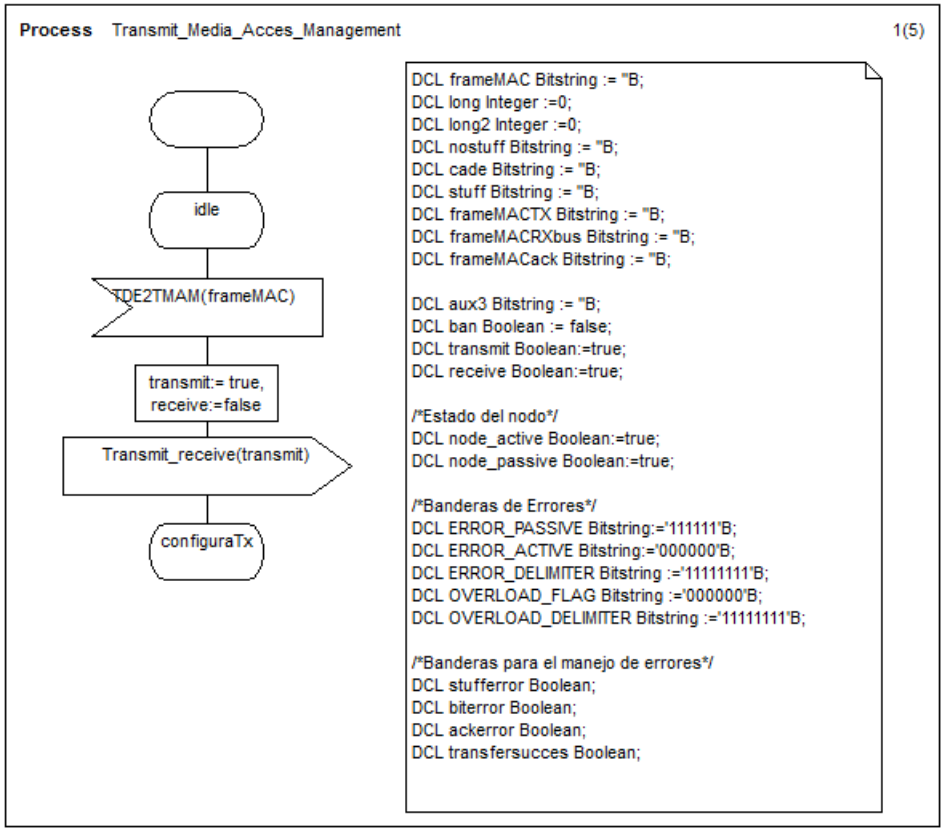

Figura 4.16: Descripción del proceso `Transmit_Media_Acces_Management`.

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

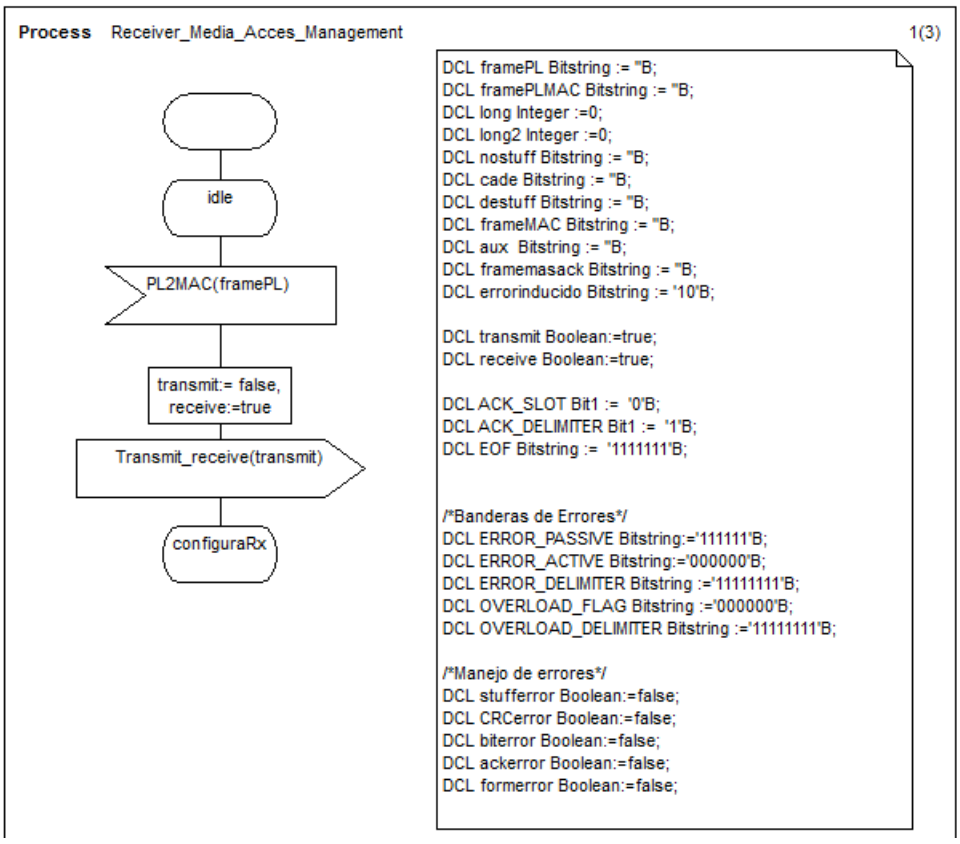

Figura 4.17: Descripción del proceso `Receiver_Media_Acces_Management`.

9. Construcción de trama de error e inicialización de la transmisión.

10. Presentación de los bits en forma serial para la transmisión a la capa física.

4.5.6. Proceso Receiver_Media_Acces_Management

El proceso `Receiver_Media_Acces_Management` realiza las siguientes funciones (véase Figura 4.17):

1. Recepción del flujo de bits provenientes de la capa física.

2. Deserialización y ensamblado de la estructura de la trama.

3. Eliminación de los bits de relleno.

4. Detección de errores de CRC, formato y violaciones a la regla de inserción de bits.

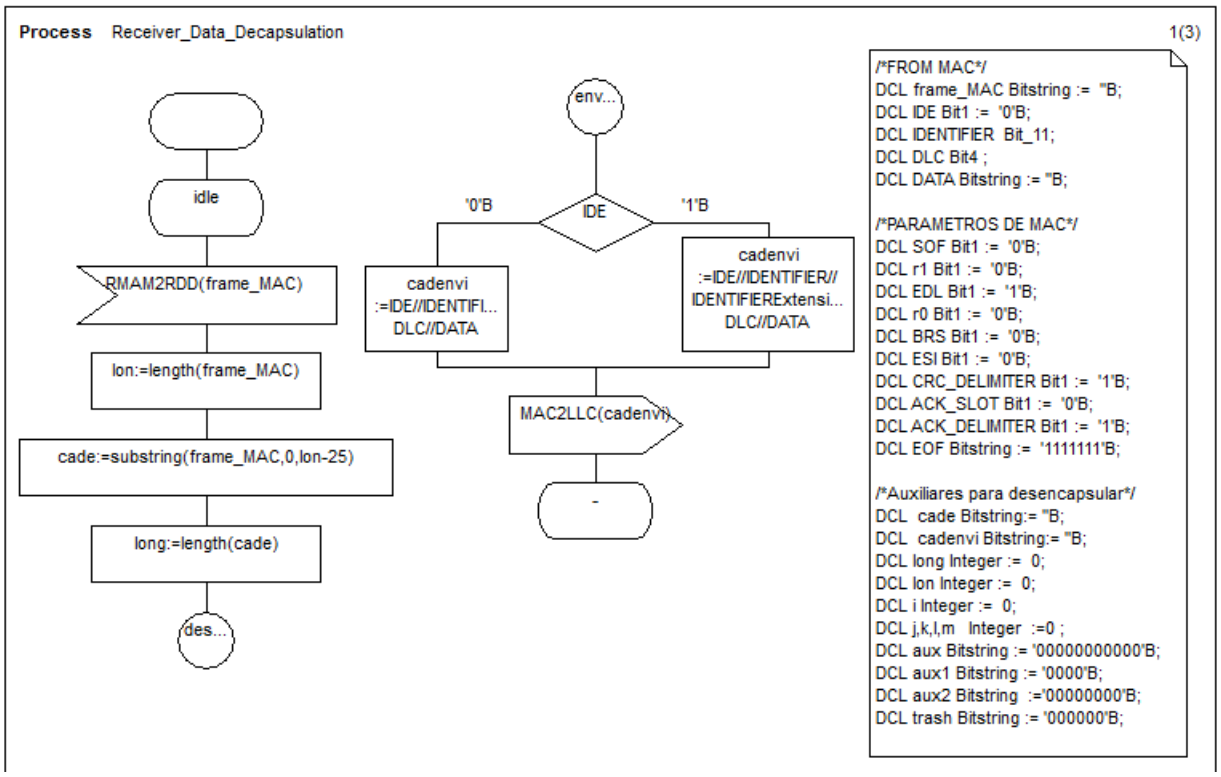

Figura 4.18: Descripción del proceso `Receiver_Data_Decapsulation`.

5. Transmisión del bit de aceptación.

6. Construcción de trama de error e inicialización de la transmisión.

7. Reconocimiento de una condición de sobrecarga.

8. Reactivación y construcción de una trama de sobrecarga inicialización de la transmisión.

9. Distinción de tramas entre el formato estándar y el formato extendido.

4.5.7. Proceso Receiver_Data_Decapsulation

El proceso `Receiver_Data_Decapsulation` realiza las siguientes funciones (véase Figura 4.18):

1. Eliminación de información específica de la trama MAC.

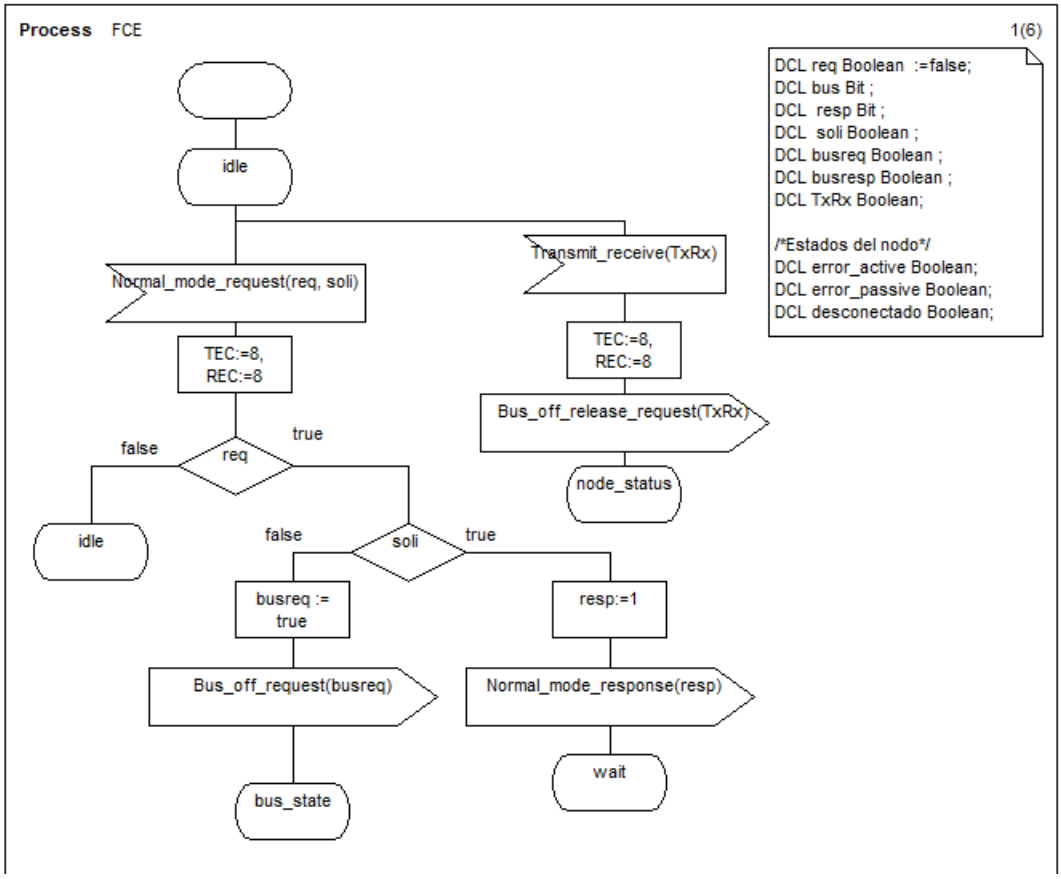

Figura 4.19: Descripción del proceso `FCE`.

2. Presentación de la trama LLC y de información de la interfaz de control a la subcapa LLC.

4.5.8. Proceso FCE

La Figura 4.19 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` mediante la señal `Normal_mode_request` que establece el proceso FCE a un estado inicial. Para atender la solicitud del bloque `LLC` el proceso `FCE` da respuesta mediante la señal `Normal_mode_response` y por medio de la señal `Bus_off` le indica el estado del bus.

Los mensajes intercambiados entre el proceso `FCE` y el bloque `MAC` 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 detectó algún error durante la transmisión o la recepción.
- `Primary_error`. Por medio de esta señal se indica que el bloque MAC 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 está esperando demasiado tiempo para un delimitador de error.
- `Succesfull_tranfer`. Por medio de esta señal se indica que la transmisión o la recepción se completaron exitosamente.
- `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 así dar respuesta a las solicitudes de los bloques `LLC` y `MAC` 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 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.

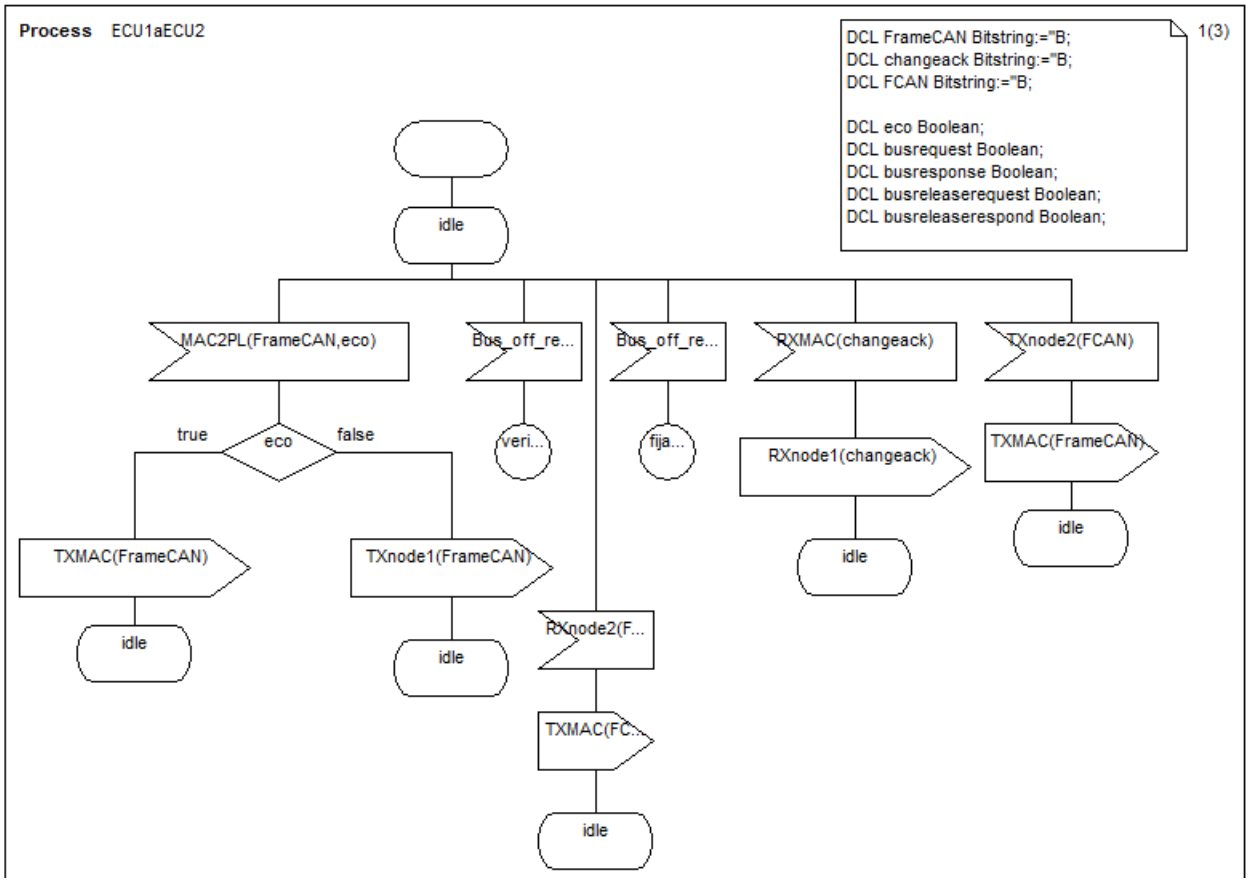

Figura 4.20: Descripción del proceso `ECU1aECU2`.

4.5.9. Proceso ECU1aECU2

La Figura 4.20 muestra la descripción del proceso `ECU1aECU2`; una vez ejecutado el símbolo de inicio pasa al estado `idle`, el cual está a la espera de recibir cualquier señal proveniente de los bloques `FCE`, `DLL` o del proceso `ECU2aECU1`. Si las señales provienen del bloque `DLL` el proceso realiza el efecto local con la finalidad de simular el arbitraje, y como transmisor/receptor a las demás ECU conectadas al bus. En caso de que sea alguna solicitud de la capa de supervisor, el proceso atiende dicha solicitud y devuelve la respuesta por las señales conectadas al canal `bus2node1`. Cada vez que el proceso recibe una señal y atiende las respuestas propias de la señal el proceso se establece en el estado `idle` en espera de cualquier otra señal.

4.5.10. Procedimientos

En este apartado se describen los procedimientos más importantes que el protocolo debe realizar, entre los que destacan aquellos que son responsables de generar el CRC, 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.

4.5.10.1. 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 4.21). 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 `KEY` que contiene el valor del polinomio generador del CRC (sección 3.2.2.1.1). Al finalizar este procedimiento se regresa la cadena `modulo`.

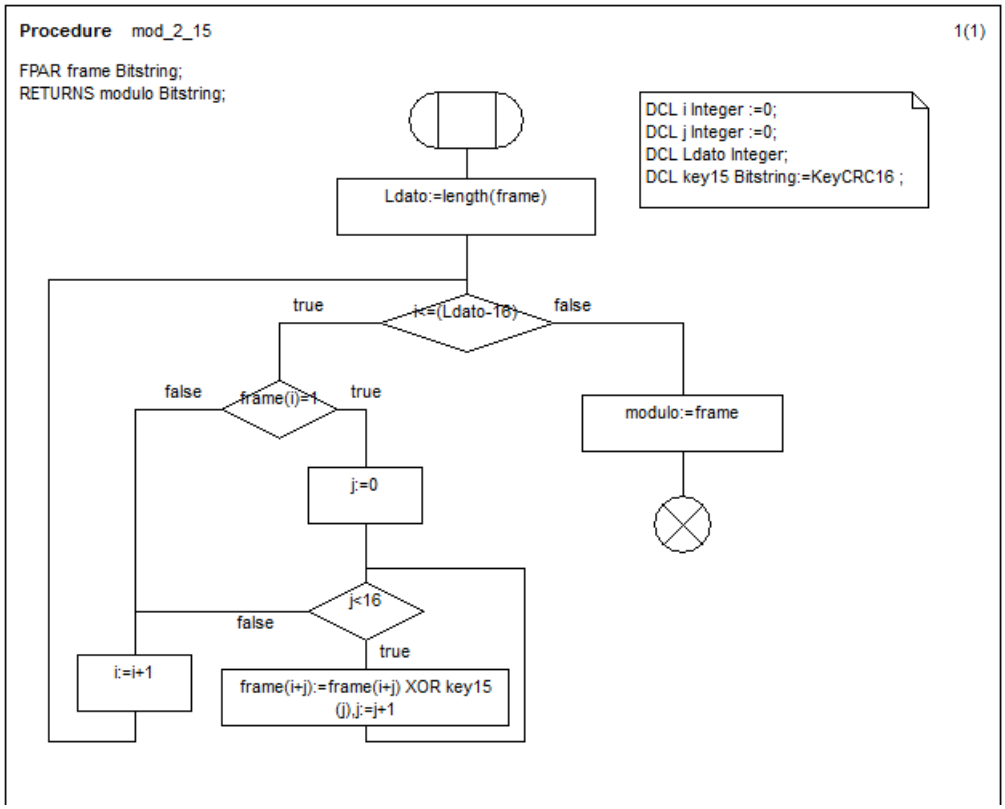

Figura 4.21: Descripción del procedimiento `Mod_2_15`.

4.5.10.2. 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 4.22).

El procedimiento `CRC_15` recibe y retorna 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`.

4.5.10.3. Procedimiento Prepara_cadena

El procedimiento `Prepara_cadena` realiza la función de crear un arreglo fijo de bits para la inserción de bits (véase Figura 4.23). Primero se analiza la longitud de la cadena de bits

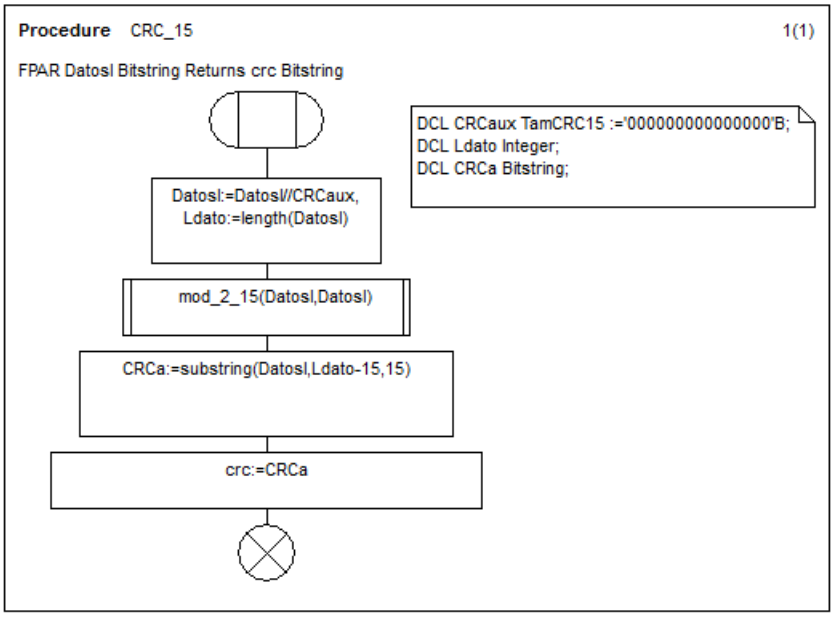

Figura 4.22: Descripción del procedimiento `CRC_15`.

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 4.1.

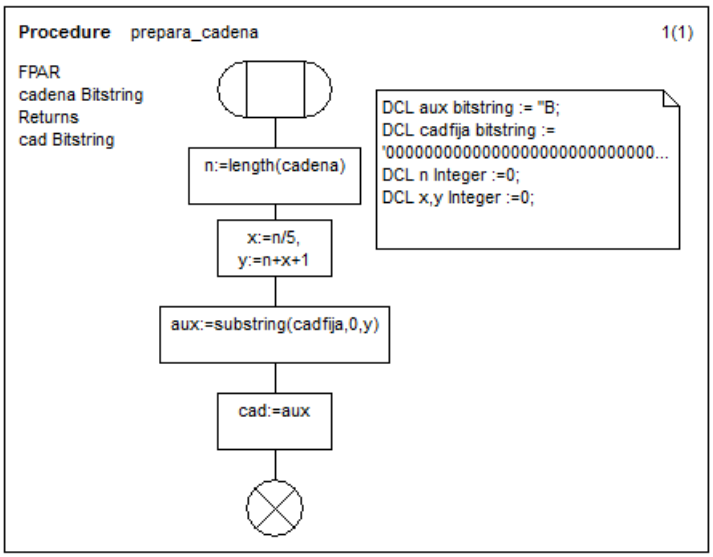

Figura 4.23: Descripción del procedimiento `prepara_cadena`.

$$TaMax = long + \frac{long}{5} + 1 \tag{4.1}$$

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.

4.5.10.4. 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 4.24). Este procedimiento recibe y retorna 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.

4.5.10.5. Procedimiento Desstuffing

El procedimiento `Desstuffing` elimina los bits que fueron insertados por el procedimiento `stuffing` (véase Figura 4.25).

Este procedimiento recibe y retorna 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.

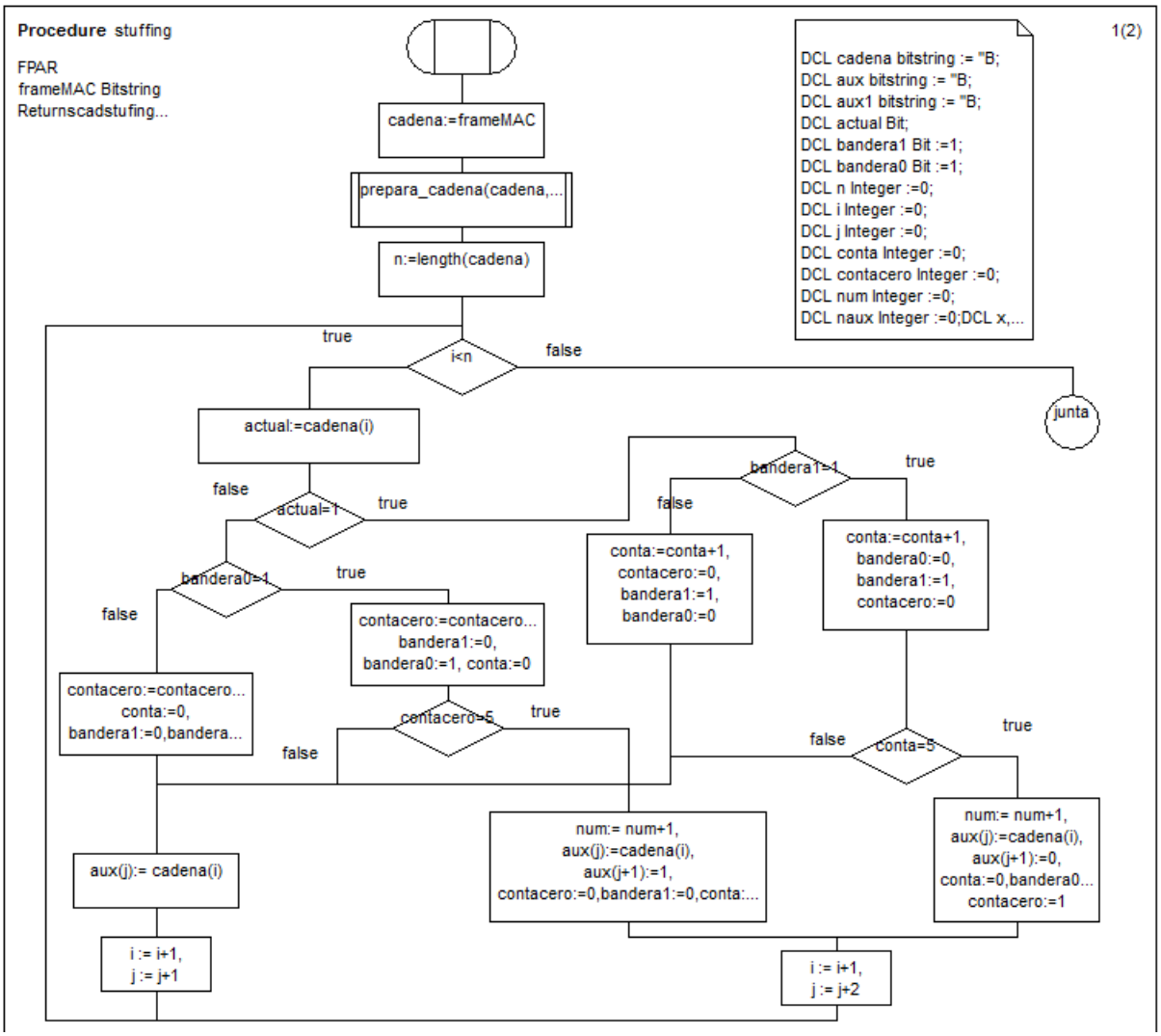

Figura 4.24: Descripción del procedimiento `Stuffing`.

4.5.10.6. 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 4.26). Este procedimiento recibe una cadena de datos de tipo `bitstring` y retorna 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

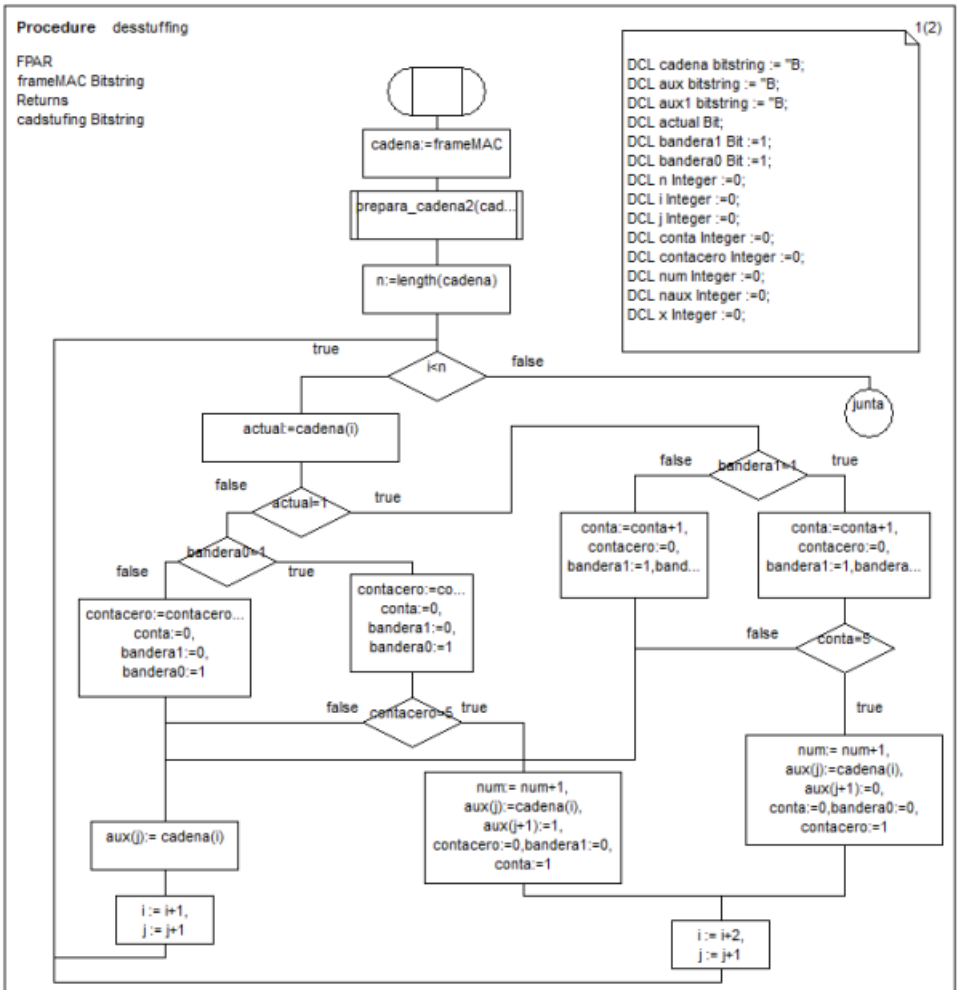

Figura 4.25: Descripción del procedimiento `Desstuffing`.

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 cambia la bandera `aux` para indicar que el CRC enviado con el calculado no coinciden.

4.5.10.7. Procedimiento Check_stuff_error

El procedimiento `Check_stuff_error` verifica que no existan más de cinco bits del mismo valor (véase Figura 4.27). 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

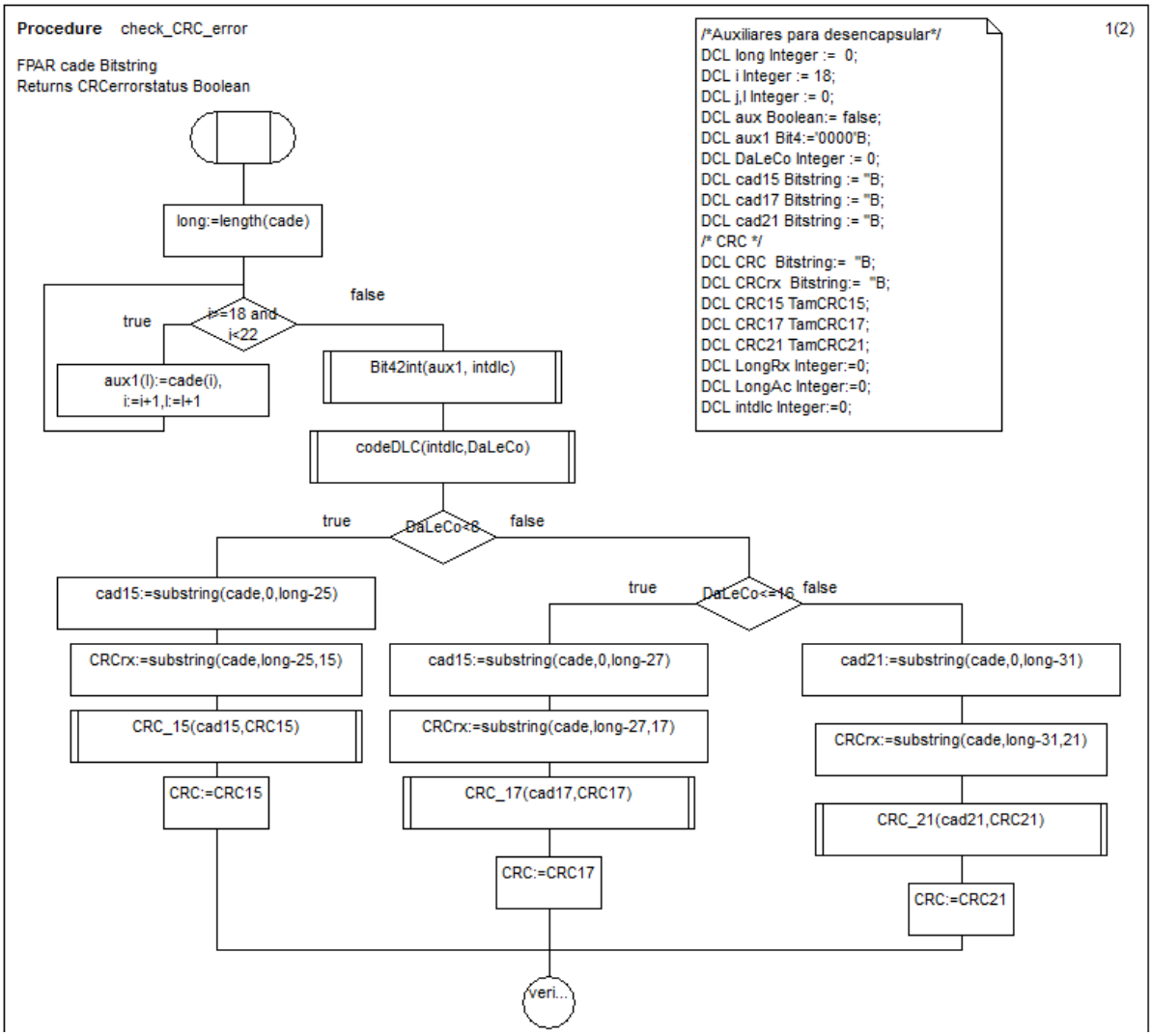

Figura 4.26: Descripción del procedimiento `Check_CRC_error`.

contador `num`, si tiene un valor mayor o igual a uno indica que existe un error de insersión de bit.

4.5.10.8. 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

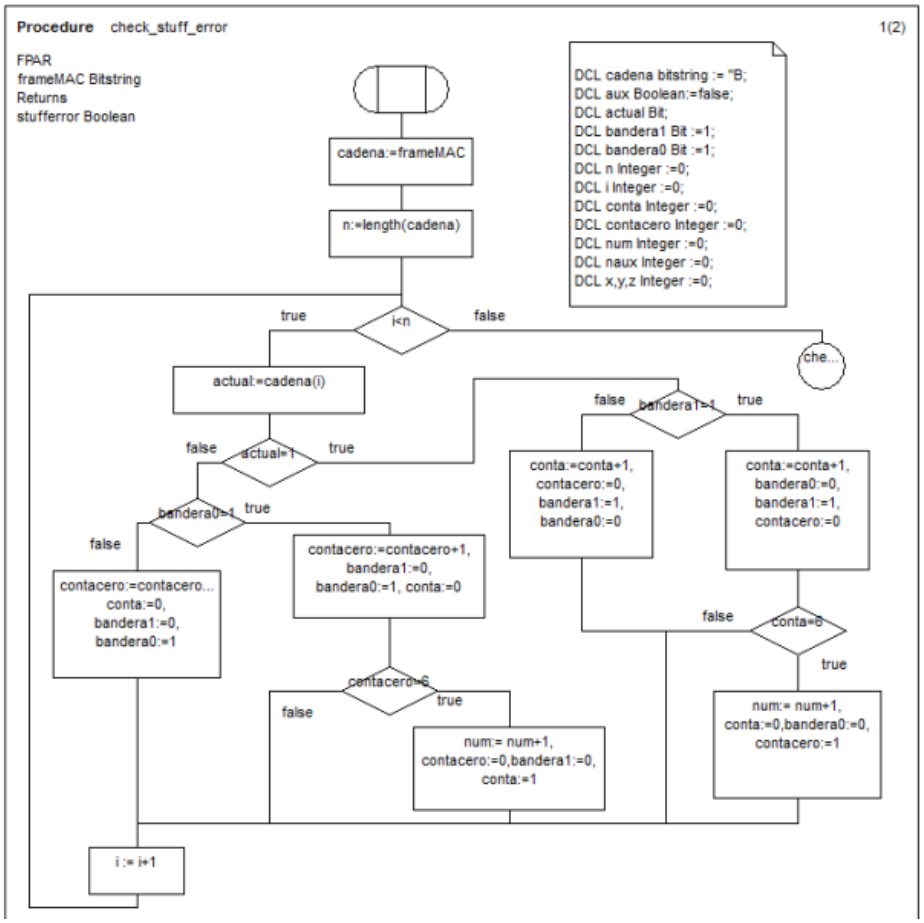

Figura 4.27: Descripción del procedimiento `Check_stuff_error`.

enviada correctamente (véase Figura 4.28). Este procedimiento recibe una cadena de datos de tipo cadena de bits y retorna una variable de tipo booleano *boolean*.

4.5.10.9. 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 función de efecto local (eco) (véase Figura 4.29). El procedimiento hace una comparación 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 cadena de bits y retorna una variable de tipo booleano.

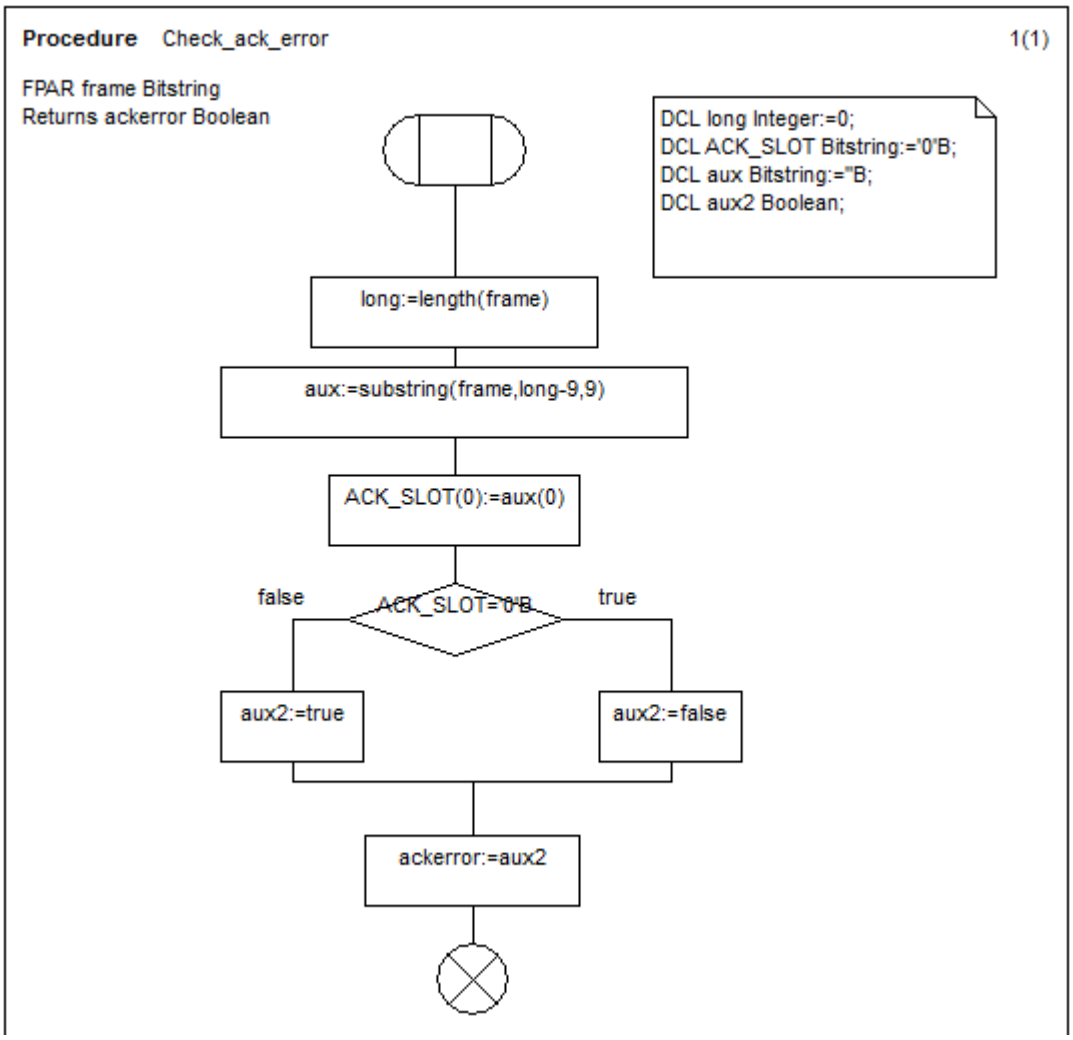

Figura 4.28: Descripción del procedimiento `Check_ack_error`.

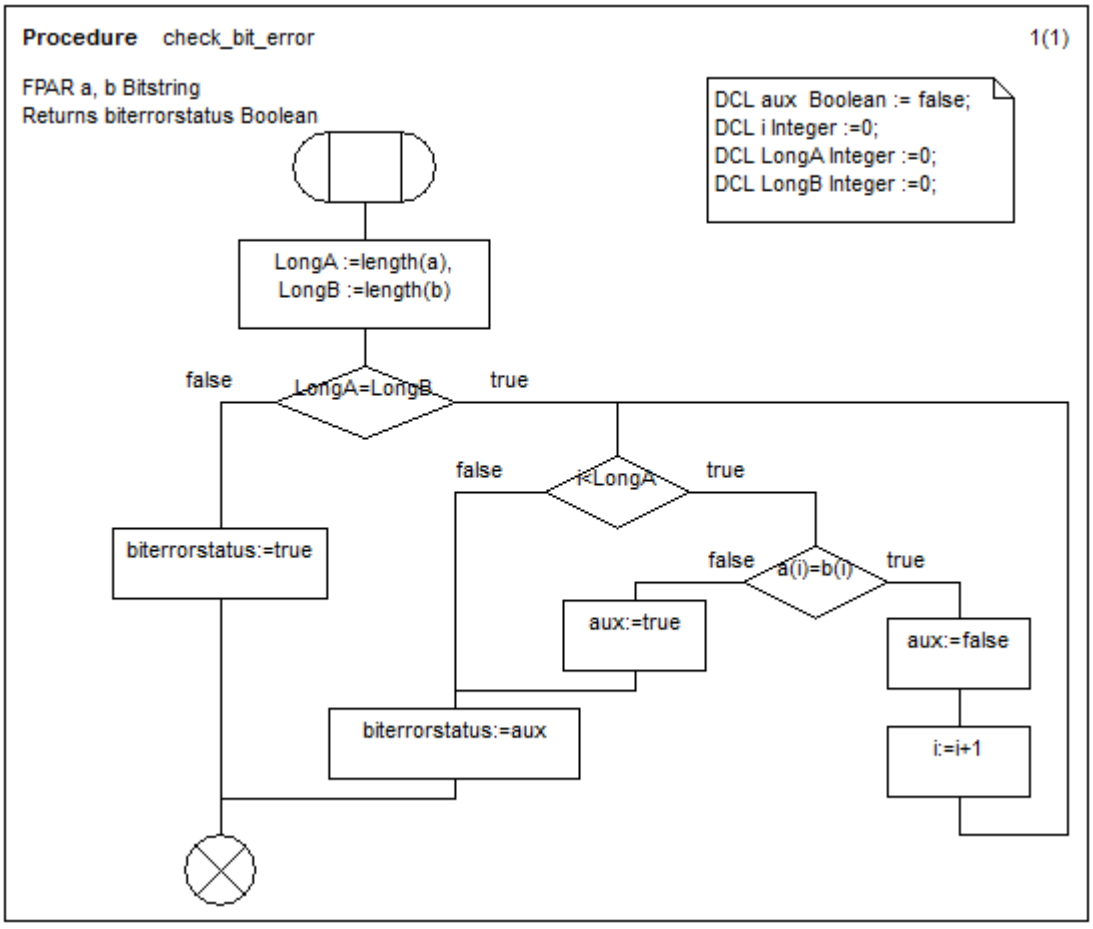

Figura 4.29: Descripción del procedimiento `Check_ack_error`.

Capítulo 5

Resultados

Una vez que se obtiene la especificación formal de un sistema, norma o protocolo, ésta debe pasar por tres etapas, simulación, validación y pruebas de comportamiento (*test*); de manera formal dichas etapas deben llevarse a cabo simultáneamente al desarrollo de la especificación con ayuda de herramientas de software específicas [80, 81]; en este caso no se dispone de herramientas adicionales, por lo que las etapas mencionadas se realizaron en forma secuencial al desarrollo de la especificación empleando las propiedades de edición, análisis (sintáctico y semántico), simulación integral y generación de MSC que proporciona la herramienta Cinderella SDL.

Las etapas de simulación y de validación se aplican a la especificación formal para corregir errores de diseño y comprobar que satisfaga los requerimientos de comportamiento y el diseñador puede verificar si el comportamiento modelado es el comportamiento esperado [82]. La etapa de pruebas de comportamiento tiene como finalidad 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 sistemas puede ser complejo, en particular los servicios distribuidos, es imposible completar todas las pruebas posibles de comportamiento. Existen lenguajes estandarizados para realizar pruebas de comportamiento como TTCN (*Tree and Tabular Combined Notation*) que generan código de pruebas a la implementación [16]. Por tal motivo, en el presente trabajo las pruebas se

realizan identificando los escenarios esenciales e importantes del sistema por no contar con las herramientas de validación y de verificación formales.

5.1. Etapas de simulación y validación

La especificación en SDL del protocolo CAN FD se realizó mediante la construcción de un sistema compuesto por un conjunto de bloques, los cuales contienen procesos que describen su funcionamiento. Para realizar la etapa de simulación, o prueba general del sistema (prueba de caja negra), se realizó la configuración inicial de cada nodo, incluyendo los valores a transmitir y como resultado se obtuvo la siguiente información:

- Los valores con que se ha configurado cada nodo.
- Correcto manejo de errores de la capa de supervisor.
- Transmisión correcta del formato CAN FD.
- Diagrama MSC para el entendimiento del protocolo de comunicaciones CAN FD.

La validación, simulación y pruebas de comportamiento de un sistema, no suceden cronológicamente a la obtención de la especificación, sino que se realizan en paralelo con ésta. La simulación y validación de la especificación CAN FD se llevó a cabo mediante la supervisión constante del explorador de Cinderella SDL (véase Figura 5.1), el cual visualiza el estado en que se encuentra la máquina de estados asociada al proceso que se esté validando y los valores de las variables declaradas en dicho proceso. Durante la etapa de simulación se analizó la reacción del proceso ante la recepción de las señales que constituyen su alfabeto de entrada.

Las ventajas que proporciona el simulador, en cuanto a capacidades de depuración, se encuentran al mismo nivel que las de cualquier depurador de lenguajes de alto nivel como C o Java los cuales permiten establecer puntos de ruptura, mediante la selección directa del símbolo SDL donde se quiere detener la ejecución, posibilidad de hacer ejecuciones de tipo "*step*" y "*transition step*" en llamadas a procedimientos, visualización y modificación de variables durante la ejecución, etc.

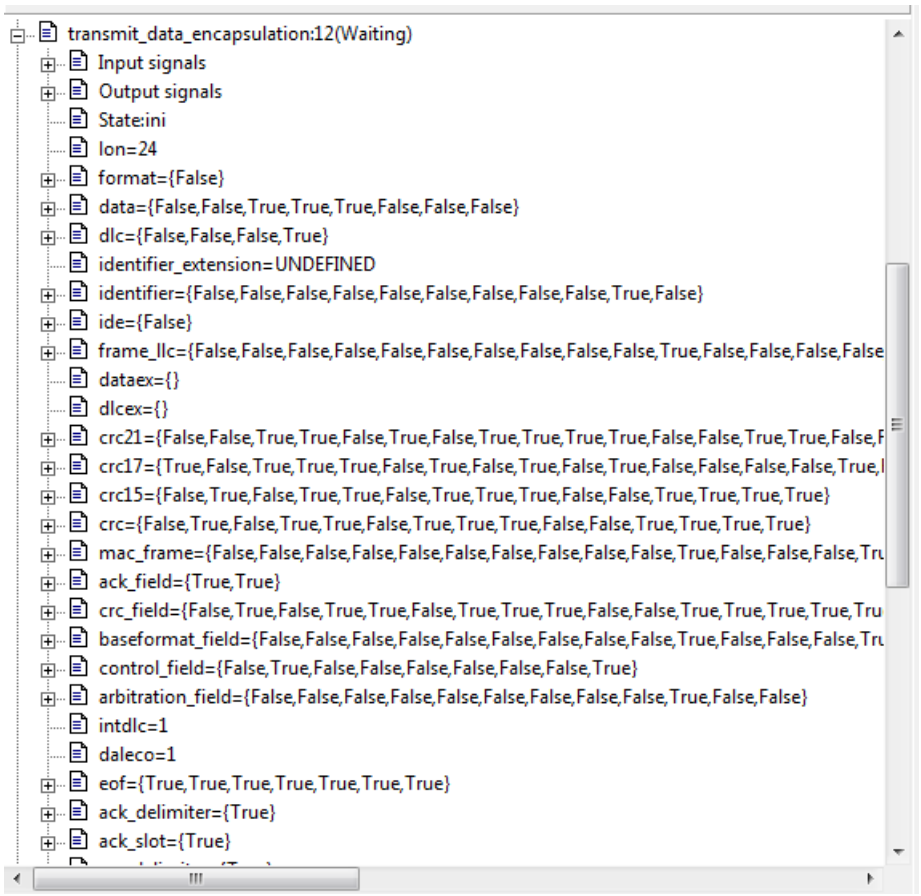

Figura 5.1: Exploración de procesos durante la etapa de simulación.

La herramienta Cinderella SDL permite hacer un seguimiento en modo gráfico del símbolo SDL que se está procesando en todo momento, sin embargo, su utilización para analizar el funcionamiento de un conjunto de procesos y procedimientos en el interior de un bloque, como ocurre con el proceso `MAC`, dificulta el seguimiento de la ejecución. Para ello se recurrió al generador de MSC y se parametrizaron algunas señales para determinar el origen de los errores. La Figura 5.2 muestra el aspecto de un conjunto de señales dentro del esquema MSC, el cual permite observar la evolución de la simulación como una secuencia de eventos ordenados en el tiempo.

En todas las tareas realizadas para verificar la especificación formal, simulación, validación y generación de pruebas de comportamiento, se ha seguido un análisis "*bottom-up*" el cual consistió en que se inició la verificación desde el nivel de abstracción más bajo (*bottom*),

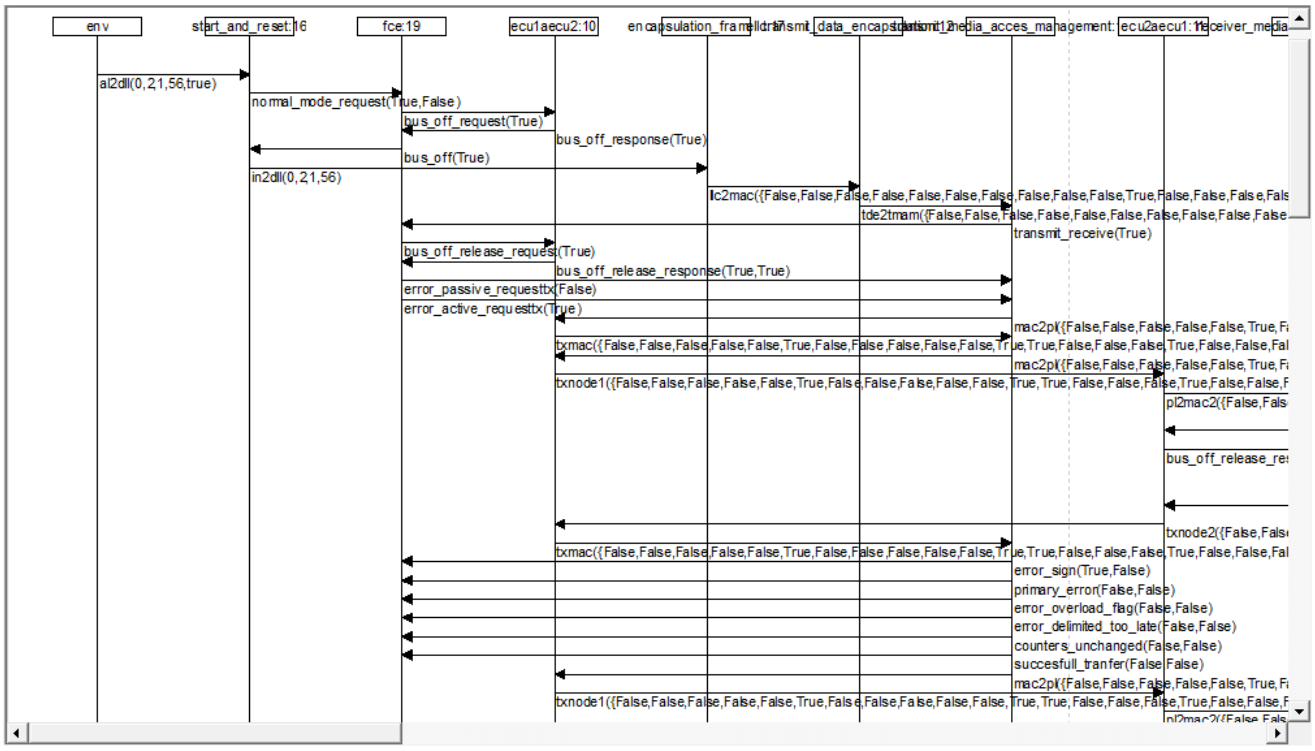

Figura 5.2: Ejemplo de los casos de usos o MSCs generados con Cinderella SDL.

y se fue ascendiendo hasta llegar al nivel de sistema más alto (*top*). En el primer nivel, el comportamiento de cada uno de los procesos y procedimientos SDL que conforman los diferentes bloques del sistema, se verificó como si se tratara de una pieza independiente. Para cada pieza se construyó un sistema SDL con un único bloque conteniendo el módulo bajo prueba. Después de realizar esta tarea con todos los procesos que constituyen un bloque, se procedió a verificar el bloque mismo. Esto se aplicó incluso a bloques con un único proceso, ya que en la verificación de un bloque no sólo se quiere comprobar el comportamiento de sus procesos internos, sino también la correcta conexión de las rutas de señal con los canales externos al bloque. Ascendiendo en niveles de abstracción, los bloques compuestos de otros bloques (subestructuras de bloques), también fueron probados después de verificar cada uno de sus componentes por separado. Finalmente, se verificó el funcionamiento del sistema completo.

5.2. 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 calculadas, tramas ensambladas erróneamente, códigos de CRC calculados erróneamente, transiciones implícitas, problemas de inicialización de estructuras de datos e interpretación errónea de algún aspecto del protocolo.

La validación general de la especificación del protocolo CAN FD se realizó mediante la verificación de la detección de errores, empleando el cálculo del CRC e inserción de bit, 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 del protocolo.

Durante la verificación de la inserción de bit se encontró un error en la documentación que no se detalla en la especificación del protocolo CAN FD [36], el cual se describe en la siguiente sección.

5.2.1. Verificación y pruebas sobre la inserción de bit

La definición que da la especificación CAN FD [36] acerca de la inserción de bit es la siguiente: “Cuando un transmisor detecte cinco bits consecutivos del mismo valor en el flujo de bits a transmitir, automáticamente se introducirá un bit complementario dentro del flujo de bit a transmitir”.

Al realizar el procedimiento correspondiente a la inserción de bit respecto a la pruebas correspondientes en el sistema CAN FD, cuando se presentaba un caso similar al mostrado en la Figura 5.3-a, la ECU receptora detectaba la trama de bits (véase Figura 5.3-b) como una bandera de error (véase la sección 3.2.2.1.4). Después de analizar y corregir el error, se determinó que la forma correcta de realizar el método de inserción de bit es la que muestra la Figura 5.3-c.

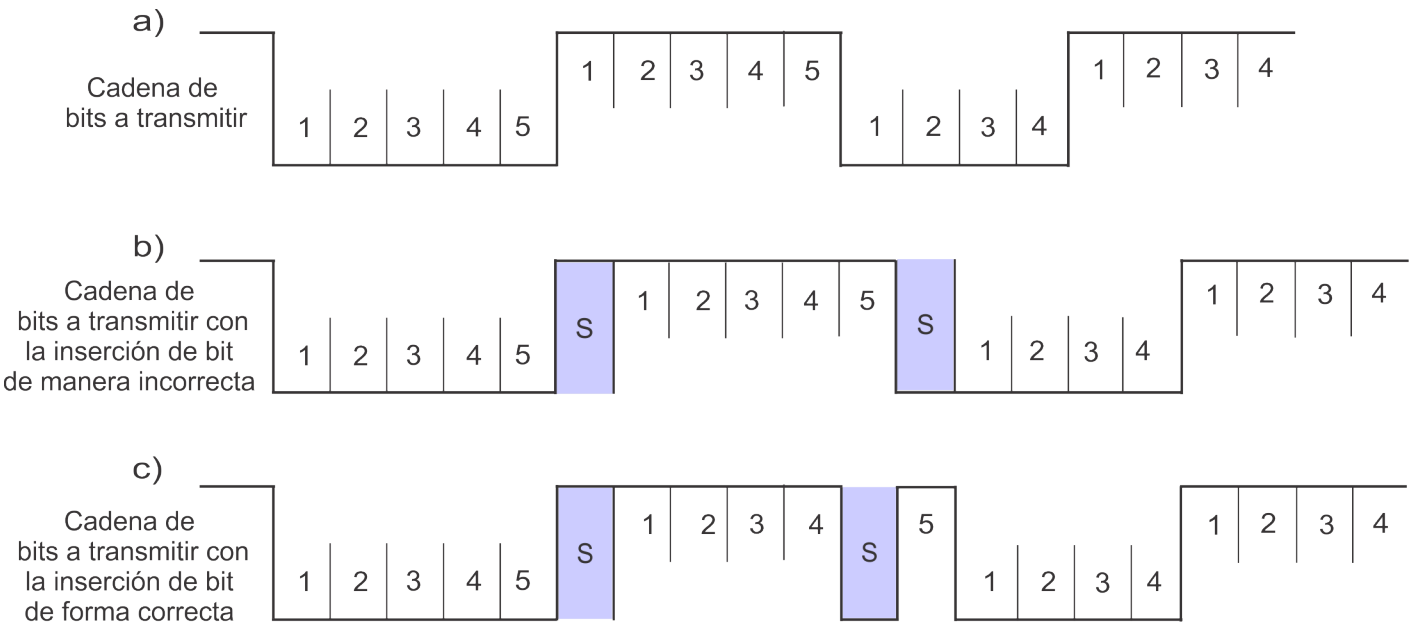

Figura 5.3: Ejemplo del bit-stuffing.

En la Figura 5.4 se muestran mediante el explorador de procesos de la herramienta Cinderella, dos cadenas de bits; la primera cadena de bits (`cadetx`) corresponde al flujo de bits de la Figura 5.3-a, mientras que la cadena `stuffbit` contiene el flujo de bits después de realizar la inserción del bit de relleno y tiene el mismo formato que el de la Figura 5.3-c.

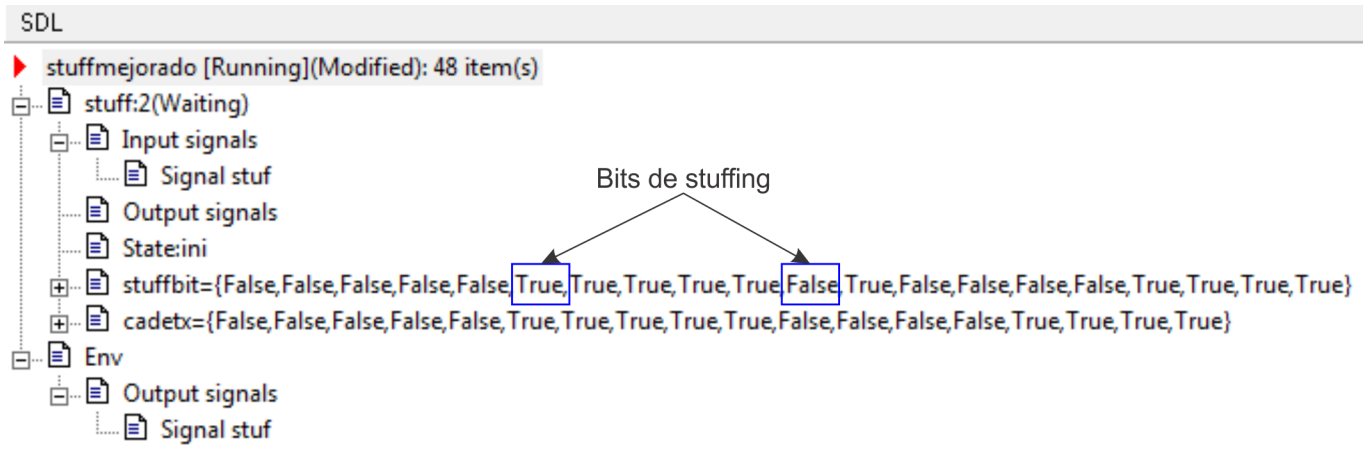

Figura 5.4: Ejemplo de la inserción del bit mediante el explorador de Cinderella.

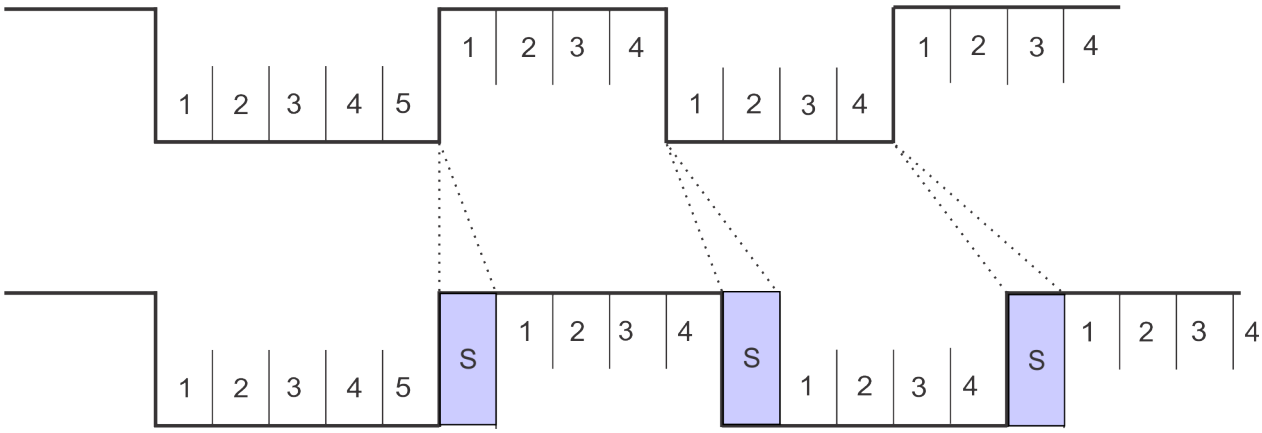

Figura 5.5: Ejemplo del peor caso para la inserción de bit.

El peor caso para la inserción de bit se da cuando se está transmitiendo una secuencia como la que se muestra en la Figura 5.5, en donde el mensaje original contiene secuencias de cuatro bits idénticos. Esto hace necesario que se inserten más bits de relleno con mayor frecuencia.

De acuerdo a la ambigüedad descrita, se propone que la definición acerca del método del bit de relleno sea la siguiente: "Cuando un transmisor detecte cinco bits consecutivos de la misma polaridad en el flujo de bits a transmitir, incluyendo el bit de relleno, automáticamente se introducirá un bit de polaridad apuesta dentro del flujo de bit a transmitir".

Capítulo 6

Conclusiones y trabajos futuros

El trabajo que se ha desarrollado durante la realización de esta tesis representan un conjunto de aportaciones a los sistemas de comunicaciones automotrices e industriales y a los campos de los métodos formales. Resultado de dicho trabajo es la especificación ejecutable del protocolo de comunicaciones CAN FD como caso de estudio, objetivo central de la presente tesis, y se deriva un conjunto de aportaciones que se enlistan a continuación:

- Para la presentación del estado del arte del protocolo de comunicaciones CAN del Capítulo 2, se realizó un estudio detallado sobre la historia del protocolo CAN, su estandarización, cuál es su objetivo y cuales son sus principales aplicaciones en la industria.

- A partir de las especificaciones, estándares y otros textos escritos del protocolo CAN y CAN FD, se realizó un estudio en profundidad de la capa de supervisor, capa física y capa del nivel de enlace de datos (Capítulo 3). Durante este estudio se han encontrado algunas ambigüedades y se han propuesto las soluciones.

- Se estudió la norma SDL detallada en la recomendación Z.100, el cual fue fundamental para lograr el desarrollo de la especificación del protocolo de comunicaciones CAN FD.

- Fue necesario estudiar el lenguaje ASN.1, detallado en la recomendación Z.105, durante las etapas de pruebas y desarrollo, con la manipulación correcta de esos tipos de datos se

redujo de manera significativa el tiempo empleado en el ejecución de los procedimientos como inserción de bit, cálculo y verificación de CRC.

- Se obtuvo una especificación estática del protocolo de comunicaciones CAN FD que permite analizar las señales, datos, bloques y procesos que conforman el sistema.
- Se obtuvo una especificación dinámica del protocolo de comunicaciones CAN FD que permite 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 FD. La especificación formal contribuye a la introducción de las FDT en la especificación de buses de campo para comunicaciones industriales.
- Se amplió la especificación formal de una ECU para obtener la especificación de un sistema distribuido correspondiente a una red CAN completa. De esta forma se verificó el funcionamiento de las estaciones en una red CAN FD, puesto que las tramas transmitidas por una ECU eran recibidas por otra ECU distinta. Esto permitió detectar errores debidos a las propiedades emergentes de este tipo de sistemas.
- Con esta especificación se obtuvo una aplicación, que permite a un usuario observar e interactuar con el funcionamiento de una red CAN. La aplicación es útil como herramienta de enseñanza en la docencia de materias relacionadas con los sistemas de comunicaciones industriales.

Durante el desarrollo de esta tesis, la experiencia adquirida en la utilización del lenguaje SDL como herramienta de especificación formal ha sido muy 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 que funciona correctamente, 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 está bien adaptada al lenguaje 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, 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 con el entorno.

6.1. 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.

- Extender la especificación formal para incluir algún estándar de capa de aplicación, correspondiente al nivel de aplicación OSI.
- Realizar en estudio detallado sobre la tolerancia a fallos del protocolo CAN FD.
- Realizar un estudio detallado de la eficacia del CRC implementado por el protocolo de comunicaciones CAN FD respecto a los CRC utilizados en protocolos similares.
- Realizar la simulación de la especificación con más de dos ECUs.
- Realizar una valoración de la verificación usando herramientas de software con objeto de obtener una mayor rigurosidad.
- Utilizar SDL para especificar otros sistemas de comunicaciones industriales.
- Realizar una evaluación comparativa (*benchmarking*) entre las distintas topologías de comunicación que permite el protocolo de comunicaciones CAN FD

Bibliografía

[1] P. Fonseca i Casas. SDL, a graphical language useful to describe social simulation models. *Proceedings of the 2nd Workshop on Social Simulation and Artificial Societies Analysis.*, 2008.

[2] ISO/IEC Standard 9074. Information processing systems, Open System Interconnection, STELLE: A formal description technique based on an Extended Transition Model, 1989.

[3] ISO/IEC Standard 9074. Open System Interconnection, LOTOS: A formal description technique based on an Temporal Ordering of the Observational Behaivor, 1989.

[4] ITU-T Implementer's guide Version 2.0.1. Formal Description Techniques (FDT), Specification and Description Language (SDL), 2013.

[5] ITU-T Recommendation Z.100. Specification and Description Lenguaje (SDL), December 2011.

[6] IBM developerWorks: IBM Rational SDL Suite. http://www.ibm.com/developerworks/rational/products/sdlsuite/. Fecha de consulta: 21 de Octubre 2013.

[7] PragmaDev Real Time Developer Studio. http://www.pragmadev.com/. Fecha de consulta: 21 de Octubre 2013.

[8] Safire World. http://www.safire-world.com/. Fecha de consulta: 21 de Octubre 2013.

[9] Dafocus web site. http://www.dafocus.com/. Fecha de consulta: 21 de Octubre 2013.

[10] Sandrila - SLD,MSC,TTNCN,URN,UML and UML2. http://www.sandrila.co.uk/visio-sdl/index.php. Fecha de consulta: 21 de Octubre 2013.

[11] Cinderella Software. http://www.cinderella.dk/. Fecha de consulta: 21 de Octubre 2013.

[12] C. A. Chamú Morales. *Desarrollo de un Sistema Educativo para Enseñanza del Protocolo de Comunicaciones CAN, Tesis de Licenciatura,*. Universidad Tecnológica de la Mixteca, Abril 2005.

[13] R. Gonzáles Salinas. *Especificación del Protocolo FLEXRAY Utilizando un Lenguaje de Descripción Formal, Tesis de Licenciatura*. Universidad Tecnológica de la Mixteca, July 2008.

[14] D. D. Pérez Ortiz. *Especificación del Protocolo DNP3 Utilizando un Lenguaje de Descripción Formal, Tesis de Licenciatura*. Universidad Tecnológica de la Mixteca, Julio 2011.

[15] R. M. Sousa and G. D. Putnik. Formal Description Technique SDL for manufacturing systems specification and description. *Congress of Production and Systems Engineering Department University of Minho*, 1999.

[16] A. Olsen, D. Demany, E. Cardoso, F. Lodge, M. Kolberg, M. Björkander and Richard Sinnott. The pros and cons of using SDL for creation of distributed services. *Lecture Notes in Computer Science*, 342-354, 1999.

[17] A. A. Khwajaa and J. E. Urban. A property based specification formalism classification. *The Journal of Systems and Software*, 83:2344 – 2362, 2010.

[18] K. Verschaeve. *UML - SDL round-trip engineering through incremental translation of changes*. PhD thesis, Vrije Universiteit Brussel, 2001.

[19] A. A. Khwajaa and J. E. Urban. A synthesis of evaluation criteria for software specifications and specification techniques. *International Journal of Software Engineering and Knowledge Engineering*, 12:581 – 599, 2002.

[20] R. Reed. The Evolution of SDL-2000. *Congress of Languages for Telecommunications Applications*, Volume 96, 2000.

[21] J. Ellsberger, D. Hogrefe and A. Sarma. *SDL, Formal Object-oriented Language for Communicating Systems*. Prentice Hall, 1997.

[22] L. Doldi. *SDL Illustrated Visually desing executable models*. British Library Cataloging in Publication Data, 2001.

[23] A. Mitschele-Thiel. *Systems Engineering with SDL, Developing Performance-Critical Communication Systems*. Jonh Wiley & Sons, Ltd, 2001.

[24] F. Xiaoming and D. Hogrefe. Modeling Soft State Protocol with SDL. *Technische Berichte des Instituts für Informatik an der Georg-August-Universität Göttingen*, Augost 2004.

[25] J. B. Nogueira Nine. *Metodologías de Especificaciíon Formal Aplicados a Modelos Normalizados de Protocolos de Comunicaciones Industriales, Tesis Doctoral*. Universidad de Vigo, Departamento de Tecnología Electrónica, Octubre 2000.

[26] F. Ngani Noudem and C. Viho. Modeling, Verifying and Testing Mobility Protocol from SDL Language. *Springer-Verlag Berlin Heidelberg*, pages 198 – 209, 2005.

[27] M. Bagic Babac, M. Kunstic and D. Jevtic. Describing Layered Communication Architecture in SDL Markup Language. *Journal of Information and Organizational Sciences*, 34:1–16, 2010.

[28] W. Lawrenz. *CAN Systems Engineering From Theory to Practical Applications*. Springer, second edition, 2013.

[29] L. Heng, A. Jian-Ping and Y. Jie. Vehicle Network Communication Protocols - Comparison and Case Study. *Technical Article, Dept. of E.E. Beijing Institute of Technology*, 2008.

[30] N. Navet, Y. Song, F. Simonot-Lion and C. Wilwert. Trends in Automotive Communication Systems. *Embedded Systems Handbook: Networked Embedded Systems*, 2 ed., Diciembre 2009.

[31] R. Wróbel. *Trends in Vehicle Electronics*. Wroclaw University of Technology, Automotive Engineering, 2011.

[32] Robert Bosch GmbH. CAN specification version 2.0, 1991.

[33] ISO Standard 11898. Road Vehicles - Interchange of Digital Information, Controller Area Network (CAN) for High Speed Communication, 1992.

[34] ISO Standard 11519-1. Road Vehicles - Low Speed Serial Data Communication, Part 1: Low Speed Controller Area Network (CAN), 1993.

[35] G. Leen and D. R. Hefaman. Expanded Automotive Electronic Systems In-Vehicle Networks. *IEEE Computer*, 35:88–93, 2002.

[36] Robert Bosch GmbH. Specification of CAN with Flexible Data Rate, Abril 2012.

[37] M. Felser. The Fieldbus Standards: History and Structures. *Congress University of Applied Science Berne*, 2005.

[38] F. Seidel. X-by-wire. *Hauptseminar Transportation Systems*, 2009.

[39] D. Rishvanth, Valli and K. Ganesan. Design of an In-Vehicle Network (Using LIN, CAN and FlexRay), Gateway and its Diagnostics Using Vector CANoe. *American Journal of Signal Processing*, 1(2):40 – 45, 2011.

[40] J.P. Thomesse. Fieldbus Technology in Indutrial Automation. *Proceedings of the fieldbus Conference*, 2000.

[41] W. Xing, H. Chen and H. Ding. The application of controller area network on vehicle. *Vehicle Electronics Conference, Proceedings of the IEEE International*, 1999.

[42] S. Tsugawa. Issies and Recent Trends in Vehicle Safety Communication Systems. *The Computarizacion of Transportation: Sophisticated Systems Incorporating IT in the Mobility of People and Goods*, 29:7–15, 2005.

[43] Driving Change Project, editor. *Automotive Technology: Greener Vehicles, Changing Skills*. Report of Center Automotive Research, 2011.

[44] D. Paret. *Multiplexed Networks for Embedded Systems CAN, LIN, FlexRay, Safe-by-Wire...* Jonh Wiley & Sons, Ltd, 2007.

[45] K. H. Johansson, M. Torngren and L. Nielsen. Vehicle Applications of Controller Area Network. *In SAE International Congress No. 860391*, 2006.

[46] CAN in Automation. http://www.can-cia.org/. Fecha de consulta: 25 de Octubre 2013.

[47] N. Kumar, R. Sharma and B. Ramya. Design and development of fault tolerant can-lin gateway for in-vehicle communication. *SASTECH, Automotive Engineering Centre*, 9, 2010.

[48] ISO Standard 11898-4. Road Vehicles - Controller Area Network (CAN) - Part 4: Time-triggered communication, 2004.

[49] R. I. Davis, A. Burns, R. J. Bril and J. J. Lukkien. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Springer Science*, Real Time Sys, 2007.

[50] F. Helmut and S. Uwe. Vehicle Diagnostics - The whole Story. *Vector Congress 2006*, 2007.

[51] CNL Online. Growing car sales increase CAN business. *CAN Newsletter*, Julio 2014.

[52] ISO Standard 11898-1. Road Vehicles - Controller Area Network (CAN) - Part 1 : Data link layer and physical signalling, 2003.

[53] ISO Standard 11898-2. Road Vehicles - Controller Area Network (CAN) - Part 2: High-speed medium access unit, 2003.

[54] ISO Standard 11898-3. Road Vehicles - Controller Area Network (CAN) - Part 3: Lowspeed, fault-tolerant, medium-dependent interface, 2006.

[55] ISO Standard 11898-5. Road Vehicles - Controller Area Network (CAN) - Part 5: Highspeed medium access unit with low-power mode, 2007.

[56] ISO Standard 11898-6. Road vehicles - Controller Area Network (CAN) – part 6: High-speed medium access unit with selective wake-up functionality, 2013.

[57] MOBA AG. http://www.moba.de/es.html. Fecha de consulta: 08 de Septiembre 2014.

[58] Company Moba AG. More than devices for road construction machines. *CAN Newsletter 3/2013*, March 2013.

[59] TTTech. Protocols for Aerospace Control Systems, A Comparison of AFDX, ARINC 429, CAN, and TTP. *Technical Article, TTTech Computertechnik AG*, 2005.

[60] Teletronics Technology Corp. www.ttcdas.com. Fecha de consulta: 10 de Julio 2014.

[61] Stock Flight Systems. www.stockflightsystems.com. Fecha de consulta: 10 de Julio 2014.

[62] S. Palm. With CAN network across the seven seas. *CAN Newsletter 2/2012*, 2012.

[63] Company Selectron Systems AG. CANopen in rail vehicles. *CAN Newsletter 3/2012*, 2012.

[64] K. Szczeciak. CAN-based distributed control of a breakstone cleaner. *CAN Newsletter*, pages 46–49, 2013.

[65] European Agency Space. Extract of SMART-1 Section. *ESA's Report to the 37th COSPAR Meeting*, 2008.

[66] B. Ljung. Payloads on-board the smart-1 spacecraft. *Swedish Space Corporation*, 2004.

[67] Galileo satellite navigation program. *CAN Newsletter*, 2007.

[68] H. Zeltwanger. Controller Area Network and CANopen in Medical Equipment. *Technical Paper, Development Process Desing & Development*, 2002.

[69] M. Merkel and C. Schlegel. CANopen in X-Ray Systems. *IXXAT Automation GmbH*, 2008.

[70] P. Decker. CAN Gets Even Better: Ways to transition from classic CAN to the improved CAN FD. *Technical Article, Vector*, April 2013.

[71] K. Etshberger, R. Hofmann, J. Stolberg, C. Schlegel and S.Weiher. *Controller Area Network: Basic, Protocols, Chips and Applications*. IXXAT Automation GmbH, 2001.

[72] ISO Standard 7498. Information processing systems - Open Systems Interconnection - Basic Reference Model., 1984.

[73] ISO Standard 8802-2. Information processing systems -Local Area Networks - Part 2: Logical link control, 1989.

[74] Kvaser AB. A CAN kingdom, Rev. 3.01, 1996.

[75] CANaerospace. http://www.canaerospace.net/. Fecha de consulta: 25 de Octubre 2013.

[76] H. Keith Henry. CANaerospace/AGATE databus goes flying again. *CANaerospace News, NASA*, 2003.

[77] P. Liebscher. Timing, memory protection and error detection in OSEK systems. *Technical Article, DEVELOPMENT TOOLS, Vector Informatik*, 2006.

[78] B. L. Tejaswini, A.S. Poornima and S. Dhruti. AUTOSAR for Local Interconnect Network Management in Basic Software Modules. *International Journal of Scientific Engineering and Technology*, Vol. 2 , Issue No.7:733 – 736, 2013.

[79] ITU-T Recommendation Z.105. Use of SDL with ASN.1, 2000.

[80] A. Restrepo and W. E. Wong. Validation of SDL-based architectural design models using communication-based coverage criteria. *Information and Software Technology*, pages 1418–1431, 2012.

[81] W. E. Wong, A. Restrepo and B. Choi. Validation of SDL specifications using EFSM-based test generation. *Information and Software Technology*, 51:1505–1519, 2009.

[82] J. Diaz, D. Arroyo and F. B. Rodriguez. A formal methodology for integral security design and verification of network protocols. *The Journal of Systems and Software*, 89:87–98, 2014.