ATTACHMENTS

APPENDICES

TABLE OF TABLES

TABLE OF FIGURES

1.0 INTRODUCTION

1.1 Purpose

The purpose of this document is to standardize provisions of existing Controller Area Network (CAN) standards for use in aircraft.

1.2 Scope

The scope of this document characterizes access and data flow relative to CAN and certain aspects of the data flow across the network boundaries.

1.3 Document Continuity

The intent of this standard is to maintain compatibility with previous supplements and future supplements. Further, it is intended to keep this standard updated and relevant as technology matures.

1.4 Document Organization

This standard is organized into the following sections:

Section 1.0, Introduction, provides the purpose and scope of this document. It refers to other relevant international standards.

Section 2.0, System Overview, describes the context for the use of CAN in avionics.

Section 3.0, Physical Layer, describes the physical layer provisions compliant with ISO 11898.

Section 4.0, Data Link Layer, describes the link layer provisions.

Section 5.0, CAN Communications, describes various methods of sending and receiving CAN messages.

Section 6.0, Gateway Considerations

Section 7.0, Design Guidelines

1.5 Use of Shall in this Document

1.5.1 Definition of Keywords and Imperatives

MUST – This word, or the terms “REQUIRED,” “SHALL,” or “MANDATORY” mean that the definition is an absolute requirement of this specification.

MUST NOT – This phrase, or the phrase “SHALL NOT,” mean that the definition is an absolute prohibition of this specification.

SHOULD – This word, or the adjective “RECOMMENDED,” mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

SHOULD NOT – This phrase, or the phrase “NOT RECOMMENDED,” mean that there may be valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

MAY – This word, or the adjective “OPTIONAL,” mean that an item is truly optional. One supplier may choose to include the item because a particular marketplace requires it or because the supplier feels that it enhances the product while another supplier may omit the same item. An implementation which does not include a

particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

1.6 Reference Documents

ISO 11898-1: 2015, Controller Area Network (CAN), Part 1: Data Link Layer and Physical Signaling

ISO 11898-2: 2016, Controller Area Network (CAN), Part 2: High-Speed Medium Access Unit

ISO 16845 Qualification Test Specification

CAN Specification 2.0 (Bosch)

CAN with Flexible Data-Rate (Bosch)

CANaerospace Specification (Stock Flight Systems)

ARINC Specification 664: *Aircraft Data Network, Part 7 – Avionics Full Duplex Switched Ethernet (AFDX) Network*

ATA Specification 2200 (iSpec 2200) Information Standards for Aviation Maintenance Specification, Revision 2008.1

SAE ARP4754 Certification Considerations for Highly-Integrated or Complex Aircraft Systems, Issued 1996-11

SAE ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, Issued 1996-12

AC 20-156 Aviation Databus Assurance

2.0 SYSTEM OVERVIEW

2.1 General Description

CAN refers to Controller Area Network and CAN FD refers to CAN with a flexible data rate. Unless otherwise noted, the use of the word “CAN” refers to both CAN and CAN FD. CAN is a linear multi-drop bi-directional data bus conforming to international standard ISO-11898. Originally intended to support automotive applications, it is increasingly finding its way into aerospace applications because of its cost effective and efficient networking capability for LRUs that may share data across a common media.

CAN and CAN FD are based on a serial bus with application mainly in mobile systems such as cars, public transports, trucks, military vehicles and ships because of its characteristics (e.g., immunity to EMI, high flexibility, good error detection capability, short latency time, etc.). Furthermore, CAN is a worldwide available and accepted open industrial standard with a large number of standardized low cost but still high-quality components manufactured by different suppliers. The long-term availability of CAN components is likely because it has been selected by the automotive industry worldwide as the current and future communication standard and is one of the major communication standards of the automation industry.

The ability of CAN to transmit data, in a half-duplex mode, across a shared shielded twisted pair media has advantages in terms of weight savings at the aircraft integration level. Additionally, the CAN Physical Layer protocol specification provides a built-in message priority scheme coupled with error recovery and protection mechanisms that make this data bus standard attractive to aviation applications.

While CAN components and technology have served the automotive industry well over the years, there are certain aspects of CAN technology that need to be adapted to the airborne environment. Specifically, the CAN protocol requires definition to control the priority of message delivery across the bus suitable to meet the needs of aerospace applications. While standalone applications may implement unique intra-system protocol requirements, at the airplane level there is a need to standardize aspects of the protocol at the system level to ensure interoperability across system and network domains.

2.2 Basic Concepts

It is desirable to standardize the CAN protocol to ensure CAN device interoperability and to simplify interoperation of CAN subsystems with other airborne networks.

CAN may be used as a primary or ancillary network for air transport, general aviation and other types of aircraft. It is desirable to integrate CAN into a larger networked architecture to ensure:

- Easy connections of local CAN networks to the rest of the airplane network
- Minimal cost of implementation and cost of change over time
- Maximum interoperability and interchangeability of CAN connected Line Replaceable Units (LRUs)

- Configuration flexibility: easy addition, deletion, and modification of bus nodes, without undue impact to other LRUs
- Interconnection of systems
- Traffic to easily cross system and network boundaries for both parametric and block data transfers
- Integrated error detection and error signaling
- System level functions such as on-board data load and airplane health management may be implemented

2.3 ISO Specifications

ISO 11898 specifications are divided in the following parts:

- Part 1: Data link layer and physical signaling
- Part 2: High-speed medium access unit
- Part 3: Low-speed, fault-tolerant, medium dependent interface
- Part 4: Time-triggered communication
- Part 5: 2007 High-speed medium access unit with low-power mode
- Part 6: 2013 High-speed medium access unit with selective wake-up functionality

Only parts 1 and 2 are applicable in this document.

Only ISO 11898-compliant nodes are considered in this specification.

2.4 Role of CAN in Commercial Air Transport Aircraft

Some air transport aircraft system architectures have incorporated CAN as an ancillary subsystem bus to ARINC Specification 664, Part 7 for networked Integrated Modular Avionics (IMA) architectures.

The Controller Area Network (CAN) has been developed to link sensors, actuators, and other types of avionics devices that typically require low to medium data transmission volumes during operation. In this role, CAN complements high capacity networks that support the higher bandwidth systems controlling the flight deck information flow and presentation.

2.5 Role of CAN in General Aviation Aircraft

General aviation system architectures employ CAN as one of the major avionics buses or even as the avionics backbone network. In this role, CAN may have to fulfill all requirements of a flight safety critical network.

2.6 Basic Concepts of CAN

The following figure depicts a typical system communicating via CAN. Any node needing to transmit must wait until the bus is idle. All the nodes needing to transmit information send their proper message identifier pending for transmission when the bus is idle or after an interframe space. Arbitration for the bus is accomplished while the message identifiers are transmitted. Nodes transmitting lower priority messages grant exclusive control to the node transmitting the highest priority message. The receiving nodes validate the received message. After an interframe space, the cycle is repeated with the next

message. As shown in the figure below, the circled node is transmitting while the others receive.

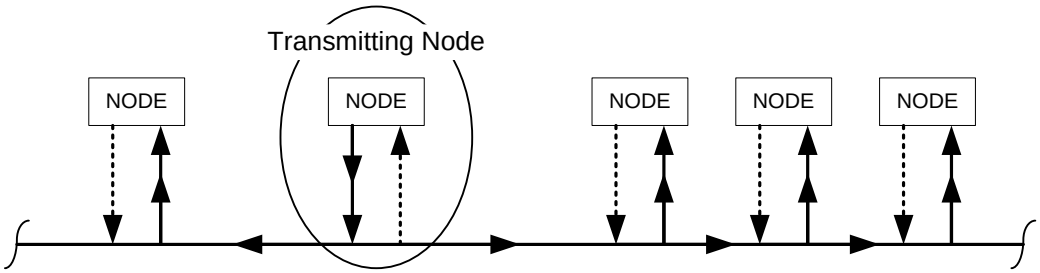


Figure 2-1 – CAN Communication Model

The following are the key characteristics of CAN:

- Multi drop network with bi-directional communication
- Broadcast and acknowledgement by all nodes
- CAN data frame (0 to 8 bytes)
- Error Detection and Error signaling

The following are key characteristics specific to CAN FD:

- CAN FD data frames (0 to 64 bytes)
- Variable data transmission rate for the payload

2.7 Network Domain Description

From an overall system architecture view, there are general network domains that fall under ARINC Specification 664 as Ethernet based networks and ARINC Specification 825 as CAN based system buses. ARINC 664 networks are characterized by IP addressed based protocols and a communication port structure at the host interface. ARINC 825 networks are characterized by CAN based protocols using data identifiers that allow data consumers to select the information they need to process. The scope of this document characterizes access and data flow relative to CAN and certain aspects of the data flow across the network boundaries.

Figure 2-2 shows an example networked system with two separate CAN networks, each with a gateway to a high-speed network. Various intercommunications are available to this system as follows:

- Intra-system communication is defined as communication between CAN nodes forming part of the same system within the same CAN network (e.g., System #1 nodes communicating together on CAN bus #1)
- Inter-system communications are defined as communications between CAN nodes from different systems within the same CAN network (e.g., System #4 nodes to System #5 nodes on CAN bus #2)
- Inter-network communications are defined as communications between systems on or via different networks, for example:
    - High-Speed Network controlling nodes on the CAN buses (e.g., System #7 on the High-Speed Network communicating with System #4 nodes on CAN bus #2)

- Communications between CAN nodes on separate CAN buses, with the High-Speed Network providing the bridging function between the buses (e.g., System #3 nodes on CAN bus #1 to System #3 nodes on CAN bus #2, via the High-Speed Network)
- Communications between CAN nodes on separate CAN buses, with the High-Speed Network providing the bridging function between the buses (e.g., System #2 nodes on CAN bus #1 to System #5 nodes on CAN bus #2, via the High-Speed Network)

It should be noted that all CAN messages are broadcast and received by all other nodes.

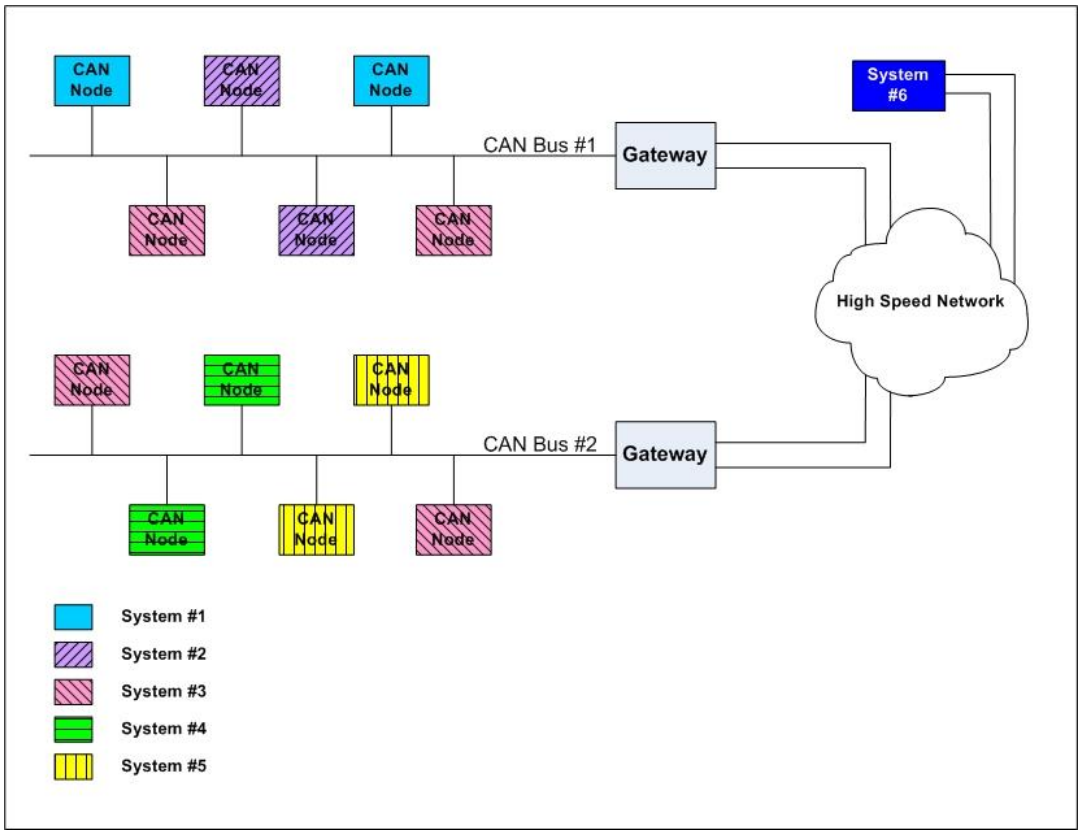


Figure 2-2 – Network Domains

The figure below shows the two data flows of a Controller Area Network integrated with high-speed system network. As shown data transmitted by LRU1 of System 1, in blue, is received by all of the LRUs on the bus and the Gateway. The Gateway selectively forwards received messages to the High-Speed Network. Messages received by the Gateway from the High-Speed Network are selectively forwarded to the CAN connected LRUs, in System 1 and 2 in Figure 2-3.

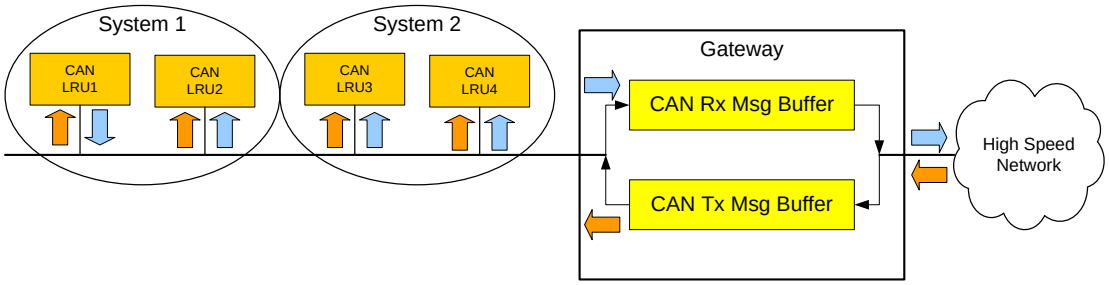


Figure 2-3 – Network Domain and Data Flow

2.8 Data Flow Across Domains

Data flows across domains fall into three categories that encompass different levels of interoperability. The intra-CAN domain only demands interoperability within a system. The inter-CAN domain demands a mutually agreed characterization for the CAN protocol domain to ensure interoperability between systems. Inter-CAN also requires standardization of electrical characteristics and data identifiers (similar to ARINC 429) or an approach that allows CAN systems to configure data identifiers from a loadable file allowing the definition to be specific to a given implementation. The Inter-network domain demands certain aspects of data flow across the network boundaries (going from potentially large addressed ports to/from a small labeled data packet) be characterized. It is recommended that data flow across this network boundary be configurable to allow flexible definition of ARINC 664 ports and ARINC 825 identifiers. The cross-network data flows need to support both parametric and directed block data transfers that ensure a consistent data set. Ideally data payload formats for all domains are the same to eliminate the need for data reformatting. Using consistent data types and formats between the high-speed network and the CAN domain is recommended.

Another aspect of the boundary between the high-speed networks and CAN that requires characterization is the bandwidth capacity differences between the network domains. Mechanisms must be put in place to ensure within capacity data flows for each side of the network boundary without unplanned data loss.

Data latency aspects for the various domains need to be characterized. This is especially true for the inter-network data flows where data buffering and bandwidth throttling mechanism may be employed that dictate certain characteristics in these regards.

Lastly, the mechanisms that allow higher layer protocols such as data load to operate across inter-network boundaries need to be characterized to allow these higher-level system functions to interoperate.

2.9 CAN Topology

CAN buses may be designed as point-to-point communication links or as multi-drop shared linear buses. There may be several CAN local buses interconnected to the larger networked architecture. These different topologies raise different system concerns relative to media access, bus determinism and fault detection/containment mechanisms. Ideally, the characteristics dealing with these system issues are common for all topologies.

2.10 CAN Protocol Architecture

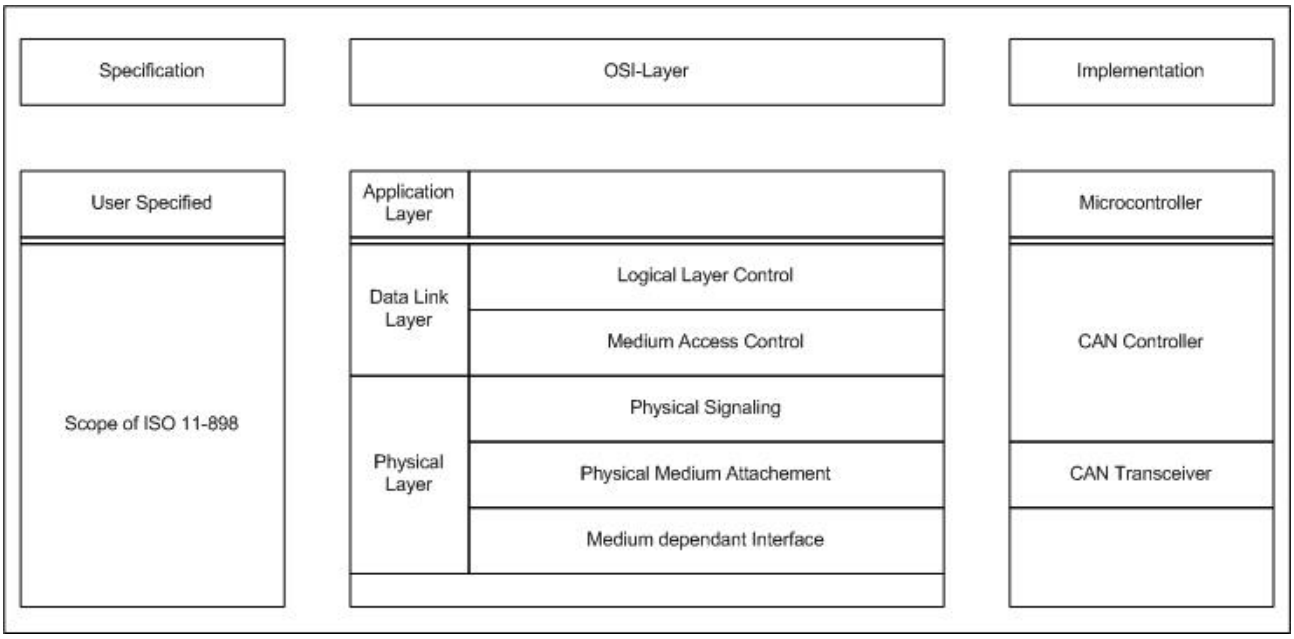


Figure 2-4 – Layered Architecture of CAN

The CAN protocol (ISO 11898-1: 2015) may be implemented in principle on any medium: copper wire, fiber optics, or wireless network. Fiber optics and wireless may have limited use for point-to-point connections. Shielded twisted copper wire is the only physical layer treated in ARINC Specification 825.

CAN nodes shall integrate the following functions:

- Electromagnetic protection (lightning indirect effects, surge, High Intensity Radiated Fields (HIRF), etc.)
- Interface between the controller protocol and the physical bus (e.g., transceiver)
- Data processing
- Health Management and Monitoring
- Data flow and bus arbitration

3.0 PHYSICAL LAYER

A Controller Area Network (CAN) is composed of nodes interconnected through a 120 Ω shielded twisted pair cable. These nodes share the medium and communicate using the CAN protocol. CAN is typically a multi-master network utilizing bi-directional communication. To ensure interoperability and required performance, all nodes must comply with CAN ISO 11898 Part 1 and Part 2 specifications.

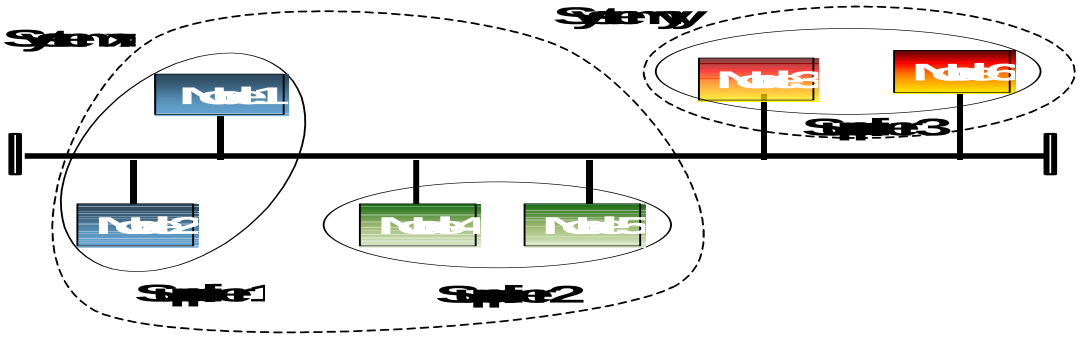


Figure 3-1 – Multi System Network

Networks that incorporate a mix of CAN 2.0 (more commonly known as Classical CAN) and CAN-FD controllers results in minimal differences at the physical layer. However, interoperability at the data link layer will need to be expressly managed by the network integrator. Refer to Section 5.3 for more information on CAN/CAN-FD Interoperability.

If CAN FD controllers are used in a mixed configuration with Classical CAN controllers, the configuration of the CAN FD controller should be statically configured to be Classical CAN compliant.

3.1 CAN Physical Layer Standard

3.1.1 Node Characteristics

The CAN physical layer defines how signals are transmitted on the medium. Signals are treated by ISO 11898-2: 2016 in the following way:

- Bit representation and synchronization
- Electrical signal level and characterization of transmission and reception driver stages
- Transmission medium and connectors

Figure 3-2 shows an example of CAN node. Nevertheless, CAN functions may be implemented in different ways. A CAN controller could be stand alone or integrated in microcontrollers. An Application Specific Integrated Circuit (ASIC) could also contain all functions. Many of these controllers are commercially available.

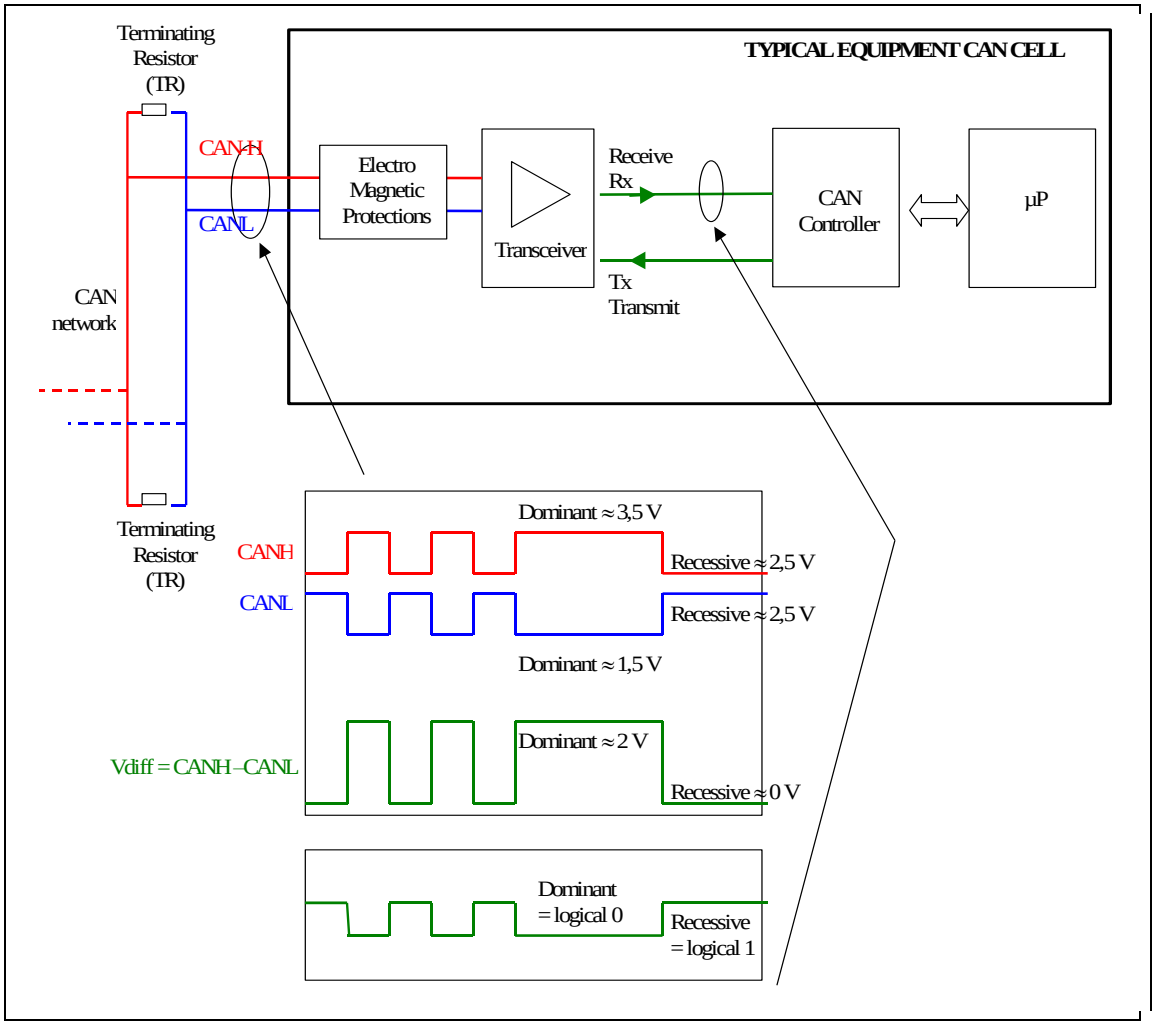


Figure 3-2 – Node Architecture Example

3.1.1.1 Electromagnetic Protection Requirements

Each node shall contain a circuit for electromagnetic protection (e.g., lightning indirect effects, surge, HIRF) consistent with environmental requirements of the design. Electromagnetic protection will lead to additional capacitance. The compromise between signal integrity and electromagnetic protection is given by the requirements specified in Section 3.1.1.2.1.

With electromagnetic protection circuits incorporated, the node must comply with the ISO 11898-2: 2016 High-Speed specification parameters and those defined in this standard.

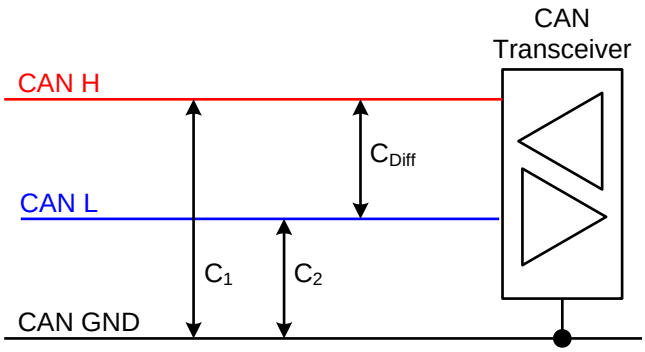


Figure 3-3 – Differential Capacitance

3.1.1.2 Transceiver Requirements

3.1.1.2.1 Electrical Parameters

CAN transceivers shall be fully compliant with ISO 11898-2: 2016 High-Speed specification (electrical parameters and its tolerances; i.e., output voltage, reception threshold).

The Differential transmission (on CAN_H and CAN_L) allows high common mode rejection and thus a high electromagnetic immunity (EMI).

The combined common mode internal capacitance C_1 and C_2 shall not exceed 100pF.

The differential capacitance C_{DIFF} as measured per Figure 3-3 – , shall not exceed 50pF.

ISO 11898-2: 2016 provides guidance on how to measure these capacitance values.

It should be understood that, for CAN-FD, the transmission of the payload at an accelerated rate means the reduction of the bit-time during the payload transmission. The designer should consider the impact of the associated electromagnetic effects.

3.1.1.2.2 Dominant Timeout Protection

Transceivers shall have protection on output lines to avoid the bus setting continuously to a dominant level. After a maximum allowable time in a dominant state, this timeout protection shall disable the transmitter.

3.1.1.2.3 Un-Powered Node

If a node is not powered, it shall not disturb the bus.

An un-powered node shall not deteriorate or degrade output signals of the powered connected nodes.

3.1.1.3 Controller Requirements

CAN controllers shall be compliant with the ISO 11898-1: 2015 and ISO 16845 qualification test specifications.

The next section will discuss bus speeds as related to Classical CAN and CAN FD while Section 4.3 will expand on the differences and the arbitration phase between Classical CAN and CAN FD.

For CAN FD, the data payload may be expanded up to 64 bytes and the data-phase transmitted at a user-defined rate greater than or equal to the Classical CAN bit rates.

3.1.1.3.1 Bit Timing

3.1.1.3.1.1 CAN Bit Timing

The bit timing parameters of a CAN controller convert the analog time domain into the digital domain of the CAN controller. They determine the time of the bit and the decision points for the CAN controller. It is essential to control these parameters to ensure interoperability between CAN controllers on the physical layer.

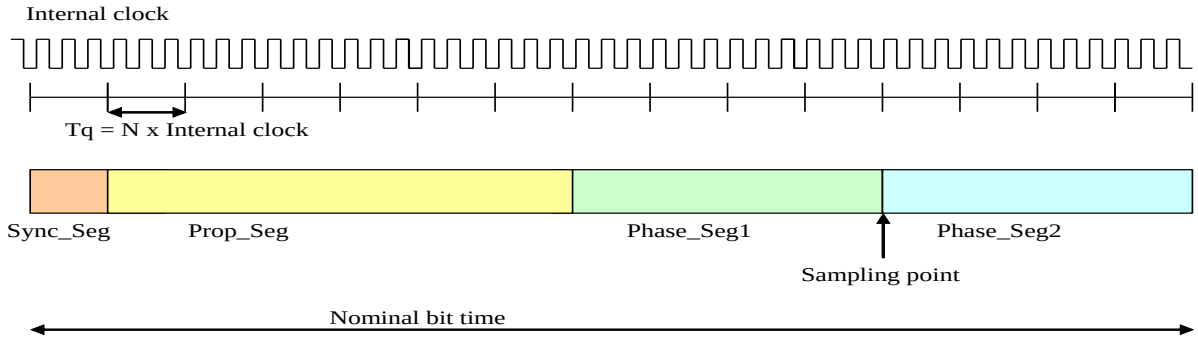


Figure 3-4 – Bit Timing

Prop_Seg

Propagation segment. It is used to compensate physical delay times within the network caused by both the transmission line delay and the impact of node interface impedance.

Phase-Seg (1 / 2)

Phase buffer segments. They are used to compensate for edge phase errors. These segments may be lengthened or shortened by resynchronization.

Sync_Seg

Synchronization segment. The first portion of the bit in which an edge is expected during a transition from recessive to dominant.

Sample point

The point of time at which the bus level is read and interpreted. The sample point position is determined by the Sync_Seg, Prop_Seg, and Phase_Seg 1.

IPT

Information Processing Time is the time it takes a CAN controller to calculate the bit level after the Sample Point.

SJW (Synchronization Jump Width)

As a result of resynchronization due to an edge lying outside the Sync_Seg, Phase_seg1 may be lengthened and Phase_seg2 may be shortened up to a limit given by the SJW. SJW is also sometimes referred to as resynchronization jump width.

Internal delay time

t_{node} of a CAN node is the delay imposed on the signal by the internal transition of the node. Section 7.6.5.4 provides more detail with regard to publisher and subscriber latencies.

Line propagation delay

T_{wire} is the delay caused by the electrical characteristics and length of the wire between the two most distant nodes.

Transmitter Delay Compensation

TDC, introduced for CAN FD, allows a transmitting CAN controller to delay its samples of the bus during the CAN FD data phase to compensate for the transceiver loopback delay. Otherwise, the bits may transmit so fast that the controller samples before its own signal propagates back through the transceiver.

3.1.1.3.1.2 CAN Clock Requirements

The bit rate of a CAN controller is determined by the oscillator and prescaler. Controlling the maximum deviation between CAN nodes is vital to ensuring interoperability.

The oscillator tolerance of a CAN node shall be less than 500 ppm.

The transmission bit rate tolerance shall be less than ± 0.20% of the selected bit time regardless of temperature and ageing in an aircraft environment. This includes the impact of the prescaler resolution, oscillator tolerance (which is less than 500 ppm per requirement), temperature effects, and deterioration over the life of the product.

3.1.1.3.1.3 CAN Bit Time Setting Requirements

Classical CAN and CAN FD require careful control of the bit timing in order to ensure interoperability. The introduction of CAN FD adds several dimensions and much complexity to the bit timing problem. As shown in Figure 3-5, there are many factors that were considered in creating the constraints. Refer to APPENDIX I for a more thorough explanation.

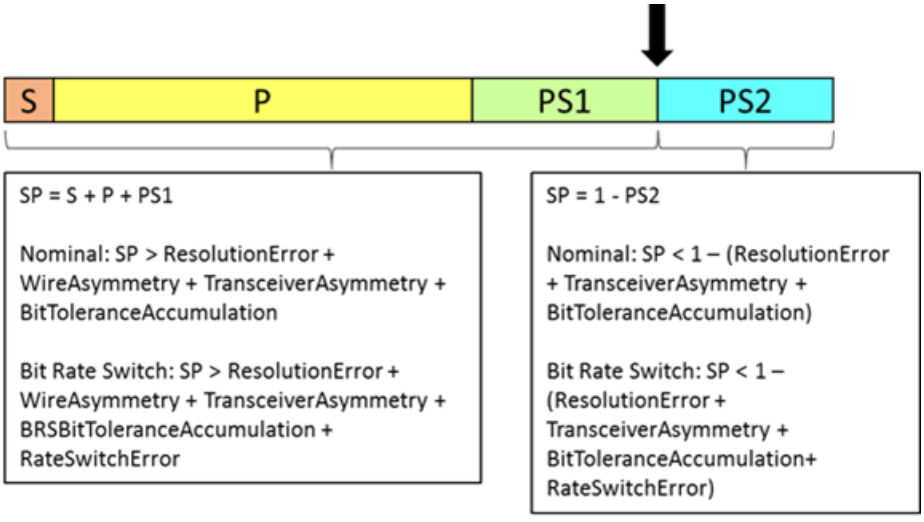


Figure 3-5 – Caption

3.0 PHYSICAL LAYER

The approach used provides design envelope requirements for the LRU designer based on the aerospace environment. Relying on this defined envelope, a simplified approach to bus length and bit rate analysis is enabled and ensures physical layer interoperability.

The design envelope is created by specifying limits on Bit Time Tolerance, Time Quanta (TQ) Resolution, and locations for the sample point. Using these settings, the allowable bus lengths and the bus speeds are calculated using simplified equations.

The following bit rate settings apply for all phases unless otherwise specified: Classical CAN, CAN FD Arbitration Phase, and CAN FD Data Phase.

1. The CAN controller shall be configured to have at least 8 TQ per bit.
    a. This applies to the arbitration phase for Classical CAN or the data phase for CAN FD (since the CAN FD arbitration bit has many more time quanta).
    b. As an example, 8 TQ per bit allows for 2 for Phase_Seg 1, 2 for Phase_Seg 2, 1 for Sync, and 3 for Prop_Seg.
2. For CAN FD, CAN controllers shall configure the same TQ size, in nanoseconds, for Arbitration and Data phases.
3. The SJW shall be at least 1/16th of a bit time to accommodate the worst-case oscillator tolerance error accumulation.
    a. No greater than min (Phase_Seg 1, Phase_Seg 2).
4. IPT of the CAN controller shall be less than Phase Segment 2.
    a. For CAN FD, the Phase Segment 2 of concern is the Data Phase since the arbitration field will have many TQ for each segment.
5. For Classical CAN, the Sample Point location shall be between 75% and 87.5% of the bit time.
6. For CAN FD arbitration phase with a data rate increase of 2x or less and CAN FD data phase, one of the following requirements shall be used:
    a. The Sample Point location for all phases shall be between 75% and 82.5% of the bit time.
    b. The Sample Point location shall be the same for all nodes on the bus.
7. For CAN FD with data rate increase of more than 2 times the arbitration rate, the Sample Point location for arbitration shall be the same for all nodes on the bus.
    a. It is recommended to place the sample point at 80% of the bit time.
8. CAN FD rates should be a multiple of 1, 2, or 4 times the arbitration rate.
9. The CAN controller shall use Single Sampling Mode only.
    a. As opposed to triple sampling options on some CAN controllers.
10. TDC should be used for CAN FD when available.
11. All CAN nodes on a bus should have the same CAN clock root frequency: either 20MHz, 40MHz, or 80MHz.
12. Classical CAN transceivers shall comply with ISO-11898-2: 2003 requirements.
13. CAN FD transceivers shall comply with ISO-11898-2: 2016 requirements and asymmetries.

14. CAN nodes with CAN FD data phase rates up to 2 Mbps shall use 2 Mbps or 5 Mbps capable transceivers, per ISO-11898-2: 2016.
15. CAN nodes with CAN FD data phase rates between 2 Mbps and 5 Mbps shall use 5 Mbps capable transceivers, per ISO-11898-2: 2016.
    a. Higher rates need more specialized equipment and should be assessed carefully using the equations provided.

All other bit timing parameters are derived from the above requirements.

3.1.1.3.2 Bit Encoding/Decoding

As CAN is designed as a multi-master system, each node may put a message on the bus when it is not occupied. Bit-to-bit arbitration (dominant bit supersedes recessive bit) will ensure that there will be no loss of information. This corresponds to wired-AND characteristics.

3.1.1.3.3 Bus Speed

Classical CAN controllers and the CAN FD arbitration phase shall be configured to one of the following transmission rates:

- 83.333 kbit/s
- 125 kbit/s
- 250 kbit/s
- 500 kbit/s
- 1 Mbps

The maximum usable CAN data rate is a function of bus length and physical characteristics of the nodes. The equations below provide the relationship between ARINC 825-bit timing, the baud rate, and the physical layer for the Classical CAN or, for CAN FD, the Arbitration Phase and the Data Phase.

3.1.1.3.3.1 Classical CAN and CAN FD Arbitration Phase

For Classical CAN and CAN FD arbitration, ARINC 825-bit timing parameters yield the following equations. The most constraining places the upper bound on the bit rate.

$$Bit\ Rate\ Max < \frac{0.287}{T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})}$$

$$Bit\ Rate\ Max < \frac{0.605}{2[T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})] + Asym_1}$$

$$Bit\ Rate\ Max < \frac{0.155}{Asym_2}$$

$$Asym_1 = bt_{Reference} - \min\left(t_{bit(Bus)}\right) - \min(\Delta t_{Rec})$$

$$Asym_2 = \max\left(t_{bit(Bus)}\right) - bt_{Reference} + \max(\Delta t_{Rec})$$

For 2 Mbps transceiver, Asym1 = 130ns, Asym2 = 70ns; for 5 Mbps transceiver, Asym1 = 90ns, Asym2 = 25ns per ISO 11898-2: 2016. Max(T_{Rise},T_{Fall}) corresponds to the worst case rise or fall time for the waveform.

3.1.1.3.3.2 CAN FD Data Phase

For the CAN FD data phase, different constraints apply. The three equations below assume that the sample point location is identical for all nodes on the bus. The most constraining places the upper bound on the bit rate.

X is the ratio of the bit rates from arbitration to data phases. $X = bt_A/bt_D$

$$Data\ Bit\ Rate\ Max < \frac{0.585}{\max(T_{Rise}, T_{Fall})}$$

$$Data\ Bit\ Rate\ Max < \frac{0.605}{\max(T_{Rise}, T_{Fall}) + Asym_2}$$

$$Bit\ Rate\ Max < \frac{0.155}{Asym_2}$$

Add the following constraint when the sample point location is allowed to vary (allowed when $X \leq 2$):

$$Data\ Bit\ Rate\ Max < \frac{0.608 - 0.0823X}{\max(T_{Rise}, T_{Fall})}$$

3.2 Design Considerations

High immunity to interference of the CAN physical layer is ensured by the CAN_H and CAN_L symmetry quality.

The transceiver is a major contributor to insure high immunity. In addition, the internal equipment layout (routing) is also important. Every effort should be made to keep the lines to the transceiver (wires and/or traces on the PCB) parallel and as short as possible. Connector pins shall be near each other and separated from any interfering signal and environment as illustrated in Figure 3-6.

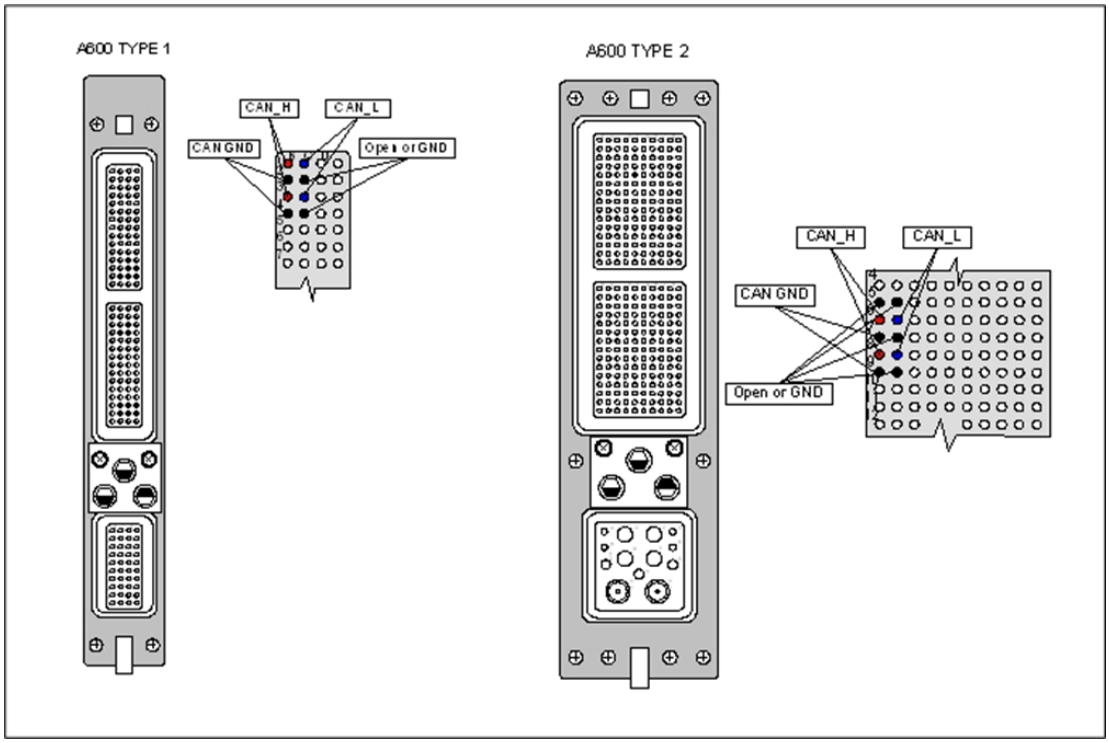


Figure 3-6 – Examples of ARINC 600 Connector Pin Assignments

Wire and connectors should be specified in accordance with air-framer design criteria and with consideration for the least delay and attenuation to meet overall requirements.

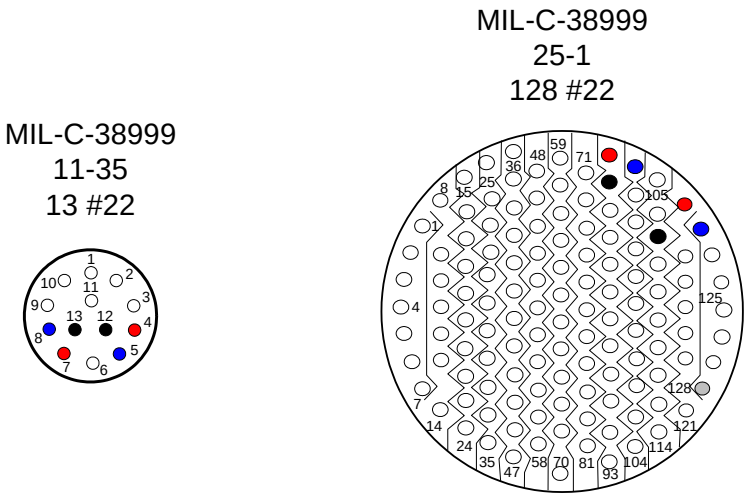


Figure 3-7 – Examples of Circular Connector Pin Assignments

3.2.1 Cabling Characteristics

Cable characteristics shall be in accordance with the ISO specification 11898-2: 2016 (i.e., impedance, length-related resistance, line delay). The cable shall be selected dependent on the installation area, i.e., pressurized, non-pressurized, exposed area. As a minimum, shielded twisted pair shall be used.

3.2.2 Connector/Contact Characteristics

No special type of connector or contact is required for CAN: At least a two-pin connector for the two CAN signals (CAN_H and CAN_L) is needed. A diagnostic plug may be added to the wiring to ease troubleshooting and maintenance on CAN using a standard Sub D9 connector described below.

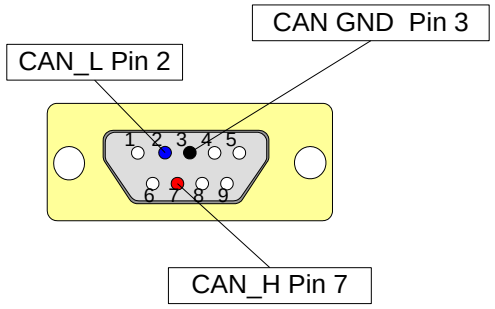


Figure 3-8 – CAN Signal Assignments to a Subminiature D Connector

3.2.3 Installation and Grounding Characteristics

The shielding of the CAN wire shall be connected to the chassis ground.

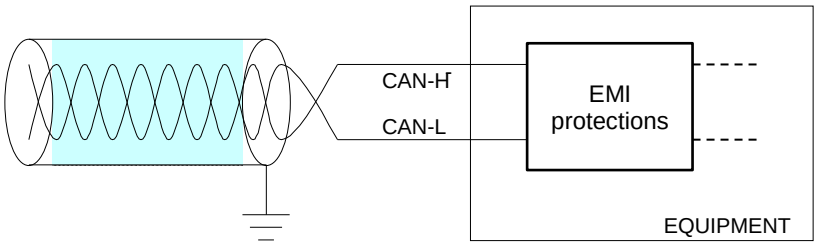


Figure 3-9 – Cabling Shielding

ARINC 825 installations shall only utilize a linear topology as illustrated in Figure 3-10. A star topology is not recommended as it may introduce undesirable wave reflections.

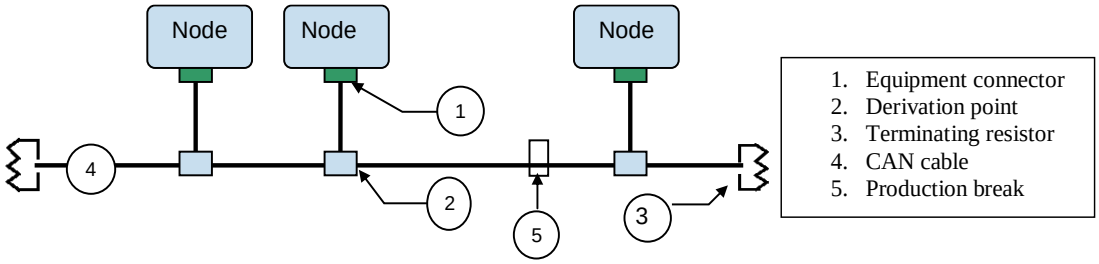


Figure 3-10 – Bus Topology

3.2.3.1 Network Termination

A 120 Ω terminating resistor shall be installed on each end of a network segment for impedance adaptation and to avoid disturbing reflections. Terminating Resistor tolerances are given in the ISO specification 11898-2: 2016 (± 10%).

3.2.3.2 Ground Offset

The voltage drop between the grounding points of two different nodes is called ground offset potential. A node shall continue to communicate in normal mode with a ground-offset up to or equal to ± 2 Vdc.

4.0 DATA LINK LAYER

4.1 ISO Compliance

ARINC 825 CAN interfaces shall be fully compliant with Classical CAN, 29-bit extended identifiers, according to ISO 11898-1: 2015.

If 11-bit identifiers are used for backward compatibility as described in Section 5.1, the Classical CAN passive standard shall be used. Mixing both identifier types on the same network is not recommended. Classical CAN passive controllers are capable of sending and receiving standard (11-bit) identifiers. Classical CAN passive controllers will also acknowledge receipt of extended (29-bit) identifiers but will ignore the messages.

The ISO 11898-1: 2015 specifies the Data Link Layer (DLL) and physical signaling. It provides the characteristics for setting up an interchange of digital information between modules implementing the CAN DLL: Logical Link Control (LLC) and Medium Access Control (MAC) sub layers and are repeated here for convenience. The conformance of the DLL shall be tested according to ISO 16845.

4.2 Frame types

CAN defines the following types of frames:

- Data Frame
- Error Frame
- Remote Frame
- Overload Frame

4.2.1 Data Frame

4.2.1.1 Classical CAN Extended Data Frame Format

Data frames have the purpose to transport information. The Classical CAN extended format data frame is shown in Figure 4-1.

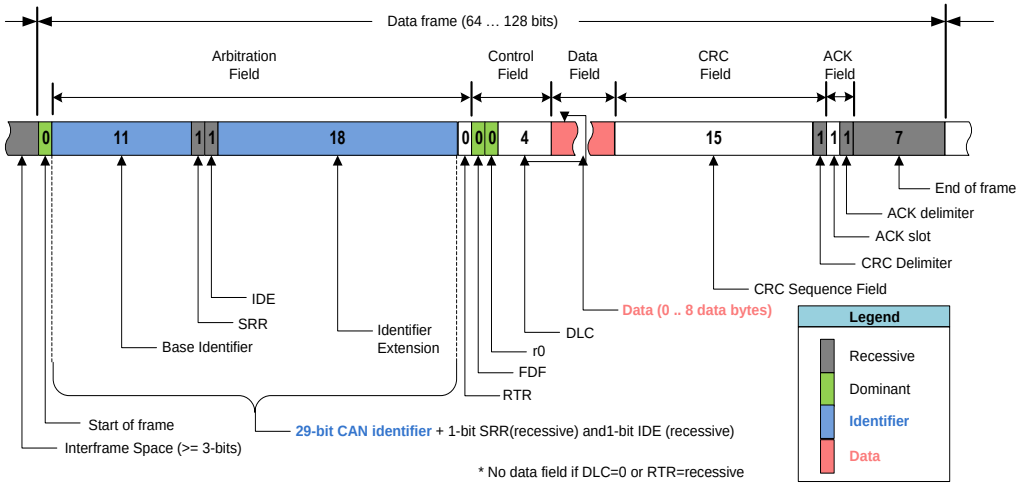


Figure 4-1 – CAN Data Frame Structure

A data frame is divided into several fields (Arbitration Field, Control Field, Data Field, CRC Field, and Acknowledge (ACK) Field) with sub structures and defined bits within each field.

- IFS: Inter-Frame Space (minimum 3 recessive bits)
- SOF: Start Of Frame (1 dominant bit)
- Arbitration Field:
  - ID: Identifier (29 bits = 11-bit base identifier + 18-bit identifier extension)
  - SRR: Substitute Remote Request bit (1 recessive bit)
  - IDE: Identifier Extension bit (1 recessive bit)
  - RTR: Remote Transmission Request bit (1 bit)
- Control Field:
  - FDF (1 dominant bit)
  - r0 (1 dominant bit)
  - Data Length Code (DLC): (4 bit field to specify data byte length)
- DATA Field: 0 to 8 bytes
- CRC Field:
  - CRC Sequence Field (15 bits)
  - CRC Delimiter (1 recessive bit)
- ACK Field:
  - ACK Slot (1 bit)
  - ACK Delimiter (1 recessive bit)
- EOF: End Of Frame (7 recessive bits)

4.2.1.1.1 Arbitration field

The extended format Arbitration Field includes the 29-bit Identifier, the Substitute Remote Request (SRR) bit, the Identifier Extension (IDE) Flag, and the Remote Transmission Request (RTR).

4.2.1.1.1.1 Identifier/ID:

The extended format Identifier field includes the 11-bit base identifier plus the 18-bit identifier extension fields. The combined 29-bit Identifier Field (ID) provides a unique bit pattern used to establish priority of the message and may be used to describe the frame content. Each node will decide to process the frame or not depending on the ID. Priority is accomplished based on the ID field where the lowest value ID indicates the highest priority message. For further description of the identifier and priority scheme, see Section 5.2.

4.2.1.1.1.2 Substitute Remote Request (SRR) Bit

The SRR bit is recessive in extended format frames. This bit is used to ensure an 11-bit base frame message prevails in a collision with a 29-bit extended frame message. Since ARINC 825 uses only the 29-bit extended format frame type this bit will always be sent recessive for ARINC 825 messages.

4.2.1.1.1.3 Identifier Extension Flag (IDE) Bit

The IDE bit distinguishes between 11-bit base format and the 29-bit extended format. The IDE bit is transmitted dominant for base format and recessive for extended format. Since ARINC 825 uses only the 29-bit extended format frame type, this bit will always be sent recessive for ARINC 825 messages.

4.2.1.1.1.4 Remote Transmission Request (RTR) Bit

The RTR bit is used for ISO Classical CAN extended frames. It is set dominant for data frames and recessive for remote frames. Since ARINC 825 discourages the use of Remote Frames, this bit should always be sent dominant for ARINC 825 messages.

4.2.1.1.2 Control Field

The ISO Classical CAN extended format Control Field includes the r1 and r0 reserved bits and the 4-bit Data Length Code (DLC).

4.2.1.1.2.1 FD Format Identifier (FDF) Bit

This bit distinguishes between Classical and FD frames. It is dominant in Classical Frames and recessive in FD frames. The FDF bit corresponds to the r1 bit in 29-bit identifier frames specified in previous versions of ISO 11898.

4.2.1.1.2.2 Reserved (r0) Bit

The r0 reserve bit is transmitted as dominant for Classical CAN extended format frames. However, receiving nodes will accept either dominant or recessive.

4.2.1.1.2.3 Data Length Code (DLC) Field

The DLC field defines the number of data bytes to be transmitted in a Data Frame. ISO Classical CAN provides a maximum of 8 data bytes. When the DLC is zero, there is no data field. Each byte contains 8 bits, which are transferred Most Significant Bit (MSB) first. The coding for the 4-bit field used for Classical CAN format frames is shown in Table 4-1.

Table 4-1 – Data Length Codes

	Number of Data Bytes	Data Length Code			
		DLC_3	DLC_2	DLC_1	DLC_0
Classical CAN	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

Note:

0 = Dominant

1 = Recessive

4.2.1.1.3 Data Field

The Data Field contains the data bytes to be transferred. The number of data bytes to be transferred is defined in the DLC field defined in Table 4-1. Each byte contains 8 bits, which are transferred MSB first.

4.2.1.1.4 Cyclic Redundancy Check (CRC) Field

A Classical CAN format frame uses a 15-bit CRC value based on a CRC generator polynomial intended to provide an error detection Hamming Distance of HD=6.

The relevant bit stream used for CRC calculation includes the SOF bit, the Arbitration Field, the Control Field, the Data Field (if present), and padded zeroes to fill out the polynomial requirements. The CRC Sequence is calculated based on the following polynomial:

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

For further guidance on calculating CAN CRC, refer to ISO 11898-1: 2015.

4.2.1.1.5 Acknowledge (ACK) Field

The ACK Field contains the ACK Slot and ACK Delimiter bits which are sent as recessive by the transmitting station. All nodes on the network send a dominant ACK bit if their CAN controller determines the received frame is a CAN consistent frame. When a frame is acknowledged, it means that AT LEAST 1 node has received the frame. The ACK mechanism is not an indication of data integrity but only frame consistency.

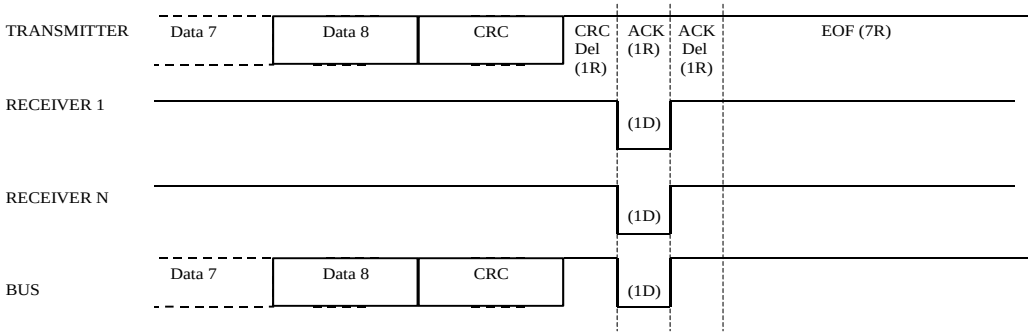


Figure 4-2 – Acknowledgement Mechanism

4.2.1.2 CAN FD Extended Data Frame Format

CAN FD Data frames revise the Classical CAN extended data frame as illustrated in Figure 4-3.

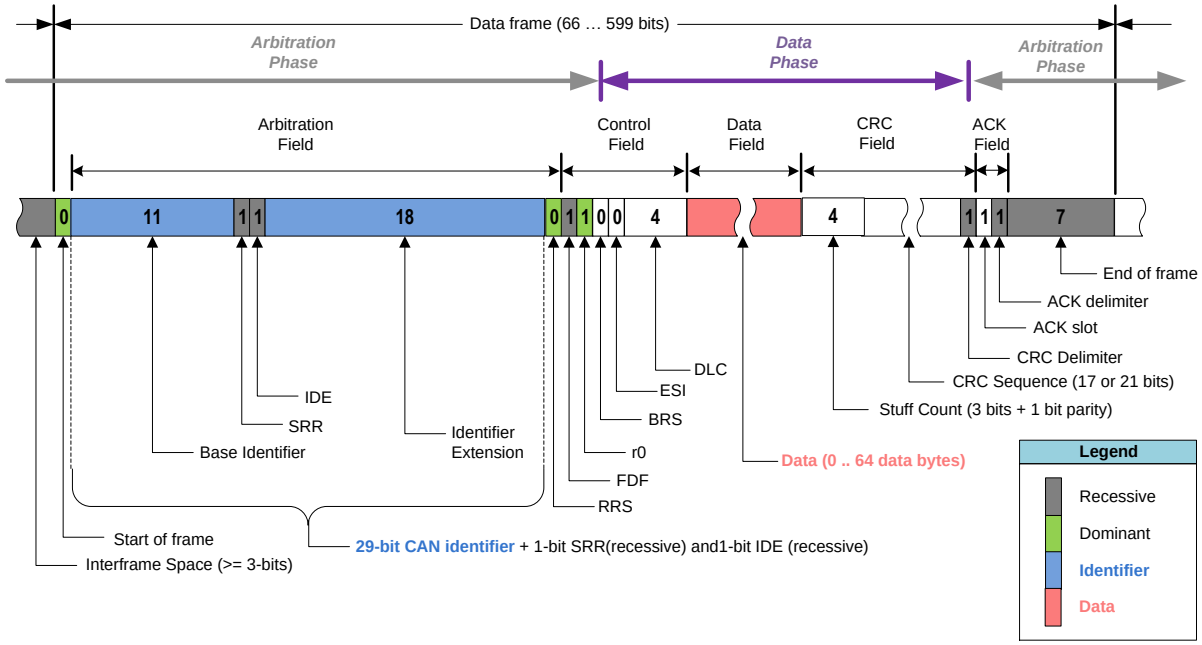


Figure 4-3 – CAN FD Extended Data Frame Structure

A CAN FD data frame is divided into several fields with many common to the previous Classical CAN format (Arbitration Field, Control Field, Data Field, CRC Field, and ACK Field) with sub structures and defined bits within each field, however CAN FD takes several bit fields and repurposes them.

- IFS: Inter Frame Space (minimum 3 recessive bits) – same as Classical CAN
- SOF: Start Of Frame (1 dominant bit) – Same as Classical CAN
- Arbitration Field:
  - ID: Identifier (29 bits = 11-bit base identifier + 18-bit identifier extension)
  - SRR: Substitute Remote Request bit (1 recessive bit)
  - IDE: Identifier Extension bit (1 recessive bit)
  - RRS: Remote Request Subscription (1 dominant bit)
- Control Field:
  - FDF: FD Format Identifier (1 recessive bit)
  - r0 (1 dominant bit)
  - BRS: Bit Rate Switch (1 bit)
  - ESI: Error State Indicator (1 bit)
  - Data Length Code (DLC): (4-bit field to specify data byte length)
- DATA Field: 0 to 64 bytes
- CRC Field: 22 to 26 bits
  - Stuff Count: 3 bits + 1 parity bit

  - CRC Sequence Field (17 or 21 bits)
  - CRC Delimiter (1 recessive bit)
- ACK Field:
  - ACK Slot (1 bit)
  - ACK Delimiter (1 recessive bit)
- EOF: End Of Frame (7 recessive bits)

4.2.1.2.1 Arbitration Field

The CAN FD extended format Arbitration Field includes the 29-bit Identifier, the Substitute Remote Request (SRR) bit, the Identifier Extension Flag (IDE) and the RRS bit. RRS is always transmitted dominant.

The SRR is always transmitted recessive for extended frame format, as required by ARINC 825.

The IDE is always transmitted recessive for extended frame format, as required by ARINC 825

The RRS is always transmitted dominant for extended frame format, as required by ARINC 825

Note: The RTR bit for Classical CAN is no longer used for CAN FD frames as remote frames are not allowed in CAN FD.

4.2.1.2.1.1 Identifier/ID

The extended format Identifier field includes the 11-bit base identifier plus the 18-bit identifier extension fields and remains unchanged from Classical CAN.

4.2.1.2.1.2 Substitute Remote Request (SRR) Bit

The SRR bit is recessive in extended format frames. This bit is used to ensure an 11-bit base frame message prevails in a collision with a 29-bit extended frame message. This is still necessary for CAN FD as both identifiers are supported by CAN FD. However, since ARINC 825 uses only the 29-bit extended format frame type, even for CAN FD, this bit will always be sent recessive for A825 messages.

4.2.1.2.1.3 Identifier Extension Flag (IDE) Bit

The IDE bit distinguishes between 11-bit base format and the 29-bit extended format. The IDE bit is transmitted dominant for base format and recessive for extended format. This is still necessary for CAN FD as both identifiers are supported by CAN FD. However, since ARINC 825 uses only the 29-bit extended format frame type, even for CAN FD, this bit will always be sent recessive for A825 messages.

4.2.1.2.1.4 Remote Request Subscription (RRS) Bit

The RRS bit shall be transmitted in FD frames at the same position that the RTR bit is transmitted in Classical Frames. The RRS bit shall be transmitted as dominant.

4.2.1.2.2 Control Field

CAN FD extended format Control Field includes the EDL, r0, BRS, ESI bits and the 4-bit Data Length Code (DLC).

4.2.1.2.2.1 FD Format Identifier (FDF) Bit

This bit distinguishes between Classical and FD frames. It is dominant in Classical Frames and recessive in FD frames. The FDF bit corresponds to the r1 bit in 29-bit identifier frames specified in previous versions of ISO 11898.

4.2.1.2.2.2 Reserved r0 Bit

The r0 reserve bit is transmitted dominant for both Classical CAN and CAN FD extended format frames. Receiving nodes will accept this bit as either dominant or recessive.

4.2.1.2.2.3 Bit Rate Switch (BRS)

The Bit Rate Switch bit is used to indicate if the transmission rate inside the CAN FD format frame remains the same. If the bit is transmitted recessive, the transmission rate of the data frame changes from the standard bit rate of the Arbitration Phase to a preconfigured alternate bit rate. If the bit is transmitted dominant, the rates remain unchanged.

4.2.1.2.2.4 Error State Indicator (ESI)

The Error State Indication flag communicates the current state of the CAN node. If the bit is set dominant the node is in error active, if transmitted recessive the node is in error passive. See Section 4.5 for further clarification on the error states.

4.2.1.2.2.5 Data Length Code (DLC) Field

The DLC field defines the number of data bytes to be transmitted in a Data Frame. CAN FD provides a maximum of 64 data bytes. When the DLC is zero, there is no data field. Each byte contains 8 bits which are transferred MSB first. The coding for the 4-bit field, used for both Classical CAN format frames and CAN FD, is shown in Table 4-2.

Table 4-2 – Data Length Codes for CAN FD

	Number of Data Field Bytes	Data Length Code			
		DLC_3	DLC_2	DLC_1	DLC_0
Classical CAN and CAN FD Format	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
Classical CAN Only	8	1	0/1	0/1	0/1
CAN FD Format	8	1	0	0	0
	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

Note:

0 = Dominant
1 = Recessive

4.2.1.2.3 Cyclic Redundancy Check (CRC) Field

The CRC field is composed of a Stuff Count and a CRC sequence.

4.2.1.2.3.1 Stuff Count

The Stuff Count uses 4 bits, 3 bits for the count and 1 bit for parity. The 3-bit count shall be the total stuff bits count modulo 8 followed by the parity bit as shown in Table 4-3.

Table 4-3 – Stuff Count Coding

Stuff Count	Coding							
Stuff bit count modulo 8	0	1	2	3	4	5	6	7
Gray-coded with parity bit	000 0	001 1	011 0	010 1	110 0	111 1	101 0	100 1

4.2.1.2.3.2 CRC Sequence

A CAN FD CRC sequence uses either a 17-bit or 21-bit CRC value, dependent on the number of data bytes specified in the DLC field, designed to also provide an

error detection Hamming Distance of HD=6. For DLCs up to 16 bytes, a 17-bit CRC is used, and for DLCs of 20, 24, 32, 48, and 64, a 21-bit CRC is used.

The relevant bit stream used for CRC calculation includes the SOF bit, the Arbitration Field, the Control Field, the Data Field (if present), and padded zeroes to fill out the polynomial requirements. In CAN FD format frames, a stuff bit count is included in the message and covered in the CRC calculation. But, in Classical CAN format frames, stuff bits are not included. In addition, the CAN FD CRC includes fixed stuff bits. The CRC Sequence is calculated based on the following polynomial:

- 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)
  $(x^{21}+x^{20}+x^{13}+x^{11}+x^{7}+x^{4}+x^{3}+1) = (x+1) \cdot (x^{10}+x^{3}+1) \cdot (x^{10}+x^{3}+x^{2}+x^{1}+1)$

4.2.1.2.4 Acknowledge (ACK) Field

The ACK Field remains unchanged for CAN FD, although the system designer must be aware of the potential for delays in bit times in the transceivers and the propagation on the bus, resulting in a potential phase-shift. This phase-shift is the same for Classical CAN and CAN FD, but may be proportionally larger in the phase with the shorter bit time. This may result in all receivers on a network having a different phase-shift to the transmitter due to their distances from the source of the CAN message. To compensate for these phase-shifts when the bit rate is switched back to the nominal (arbitration) rate on additional bit time, tolerance is allowed before and after the edge from recessive to dominant that starts the Acknowledge Slot.

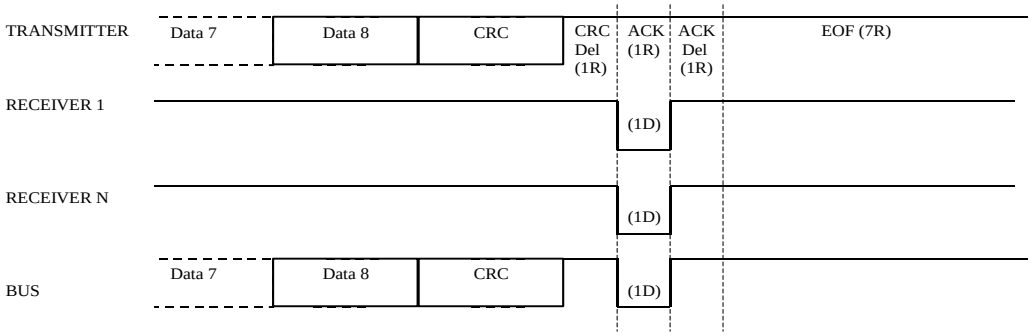


Figure 4-4 – Acknowledgement Mechanism

4.2.2 Error Frame

An Error frame consists of two different fields. The first field is given by the superposition of error flags contributed from different nodes. The second field is the error delimiter.

There are two forms of error flags: the active error flag and the passive error flag. The active error flag consists of six consecutive “dominant” bits. The passive error flag consists of six consecutive “recessive” bits. Some or all bits of a passive error flag may be overwritten by “dominant” bits from other nodes.

Note: A Classical CAN-only controller interface will respond to a CAN FD format frame with an error frame. CAN FD format frames should not be integrated on systems with Classical CAN controllers.

The system designer needs to be aware that, should an error occur during transmission of a CAN FD payload, the error frame will be sent at the arbitration bit rate.

4.2.3 Remote Frame

In Classical CAN format frames, remote frames have the task to request data from a CAN node, which in turn sends the requested data by means of a data frame. The use of remote frames is strongly discouraged. The reason for not using remote frames is the possible incompatibility problems between different controller types caused by the ambiguous definition of Data Length Code (DLC) in ISO 11898-1: 2015. Remote frames should only be used if the other ARINC 825 communication mechanisms are unable to accomplish the same task.

CAN FD format frames do not provide a RTR bit and do not support Remote Frames.

4.2.4 Overload Frame

Overload frames could be sent to tell other CAN nodes that a CAN controller is currently not able to receive frames because it is busy. Overload frames delay the start of the next datagram.

A CAN controller that initiates overload frames shall not be used, because they increase network load and reduce network reliability and availability.

4.3 Arbitration

As CAN is designed as a multi-master system, each node may initiate a message transfer if the bus is idle. Bit to bit arbitration (dominant bit supersedes recessive bit) will ensure that there will be no loss of information and no loss of bandwidth. Bus access management is called “line listen” (Collision Sense Multiple Access – CSMA) and the non-destructive bit-to-bit arbitration phase (Collision Avoidance – CA), referred to as CSMA/CA.

The truth table below illustrates the state of the bus for each bit of the arbitration process given a three-node system.

Table 4-4 – CAN Bus Arbitration Logic

Node 1	Node 2	Node 3	CAN Bus	Controllers Monitoring Status	Comments
0	0	0	0	Bus is in dominant state	Arbitration now continues to the next bit.
0	0	1	0	Node 3 loses arbitration	Node 1 and 2 continue arbitration to the next bit. Node 3 has lost arbitration and backs off.
0	1	0	0	Node 2 loses arbitration	Node 1 and 3 continue arbitration to the next bit. Node 2 has lost arbitration and backs off.
0	1	1	0	Node 2 and 3 loses arbitration	Node 1 has won arbitration and is now in control of the bus.
1	0	0	0	Node 1 loses arbitration	Node 2 and 3 continue arbitration to the next bit. Node 1 has lost arbitration and backs off.
1	0	1	0	Node 1 and 3 loses arbitration	Node 2 has won arbitration and is now in control of the bus.
1	1	0	0	Node 1 and 2 loses arbitration	Node 3 has won arbitration and is now in control of the bus.
1	1	1	1	Bus is in recessive state	Arbitration now continues to the next bit

Note:

0 = Dominant

1 = Recessive

An example is given in Figure 4-5.

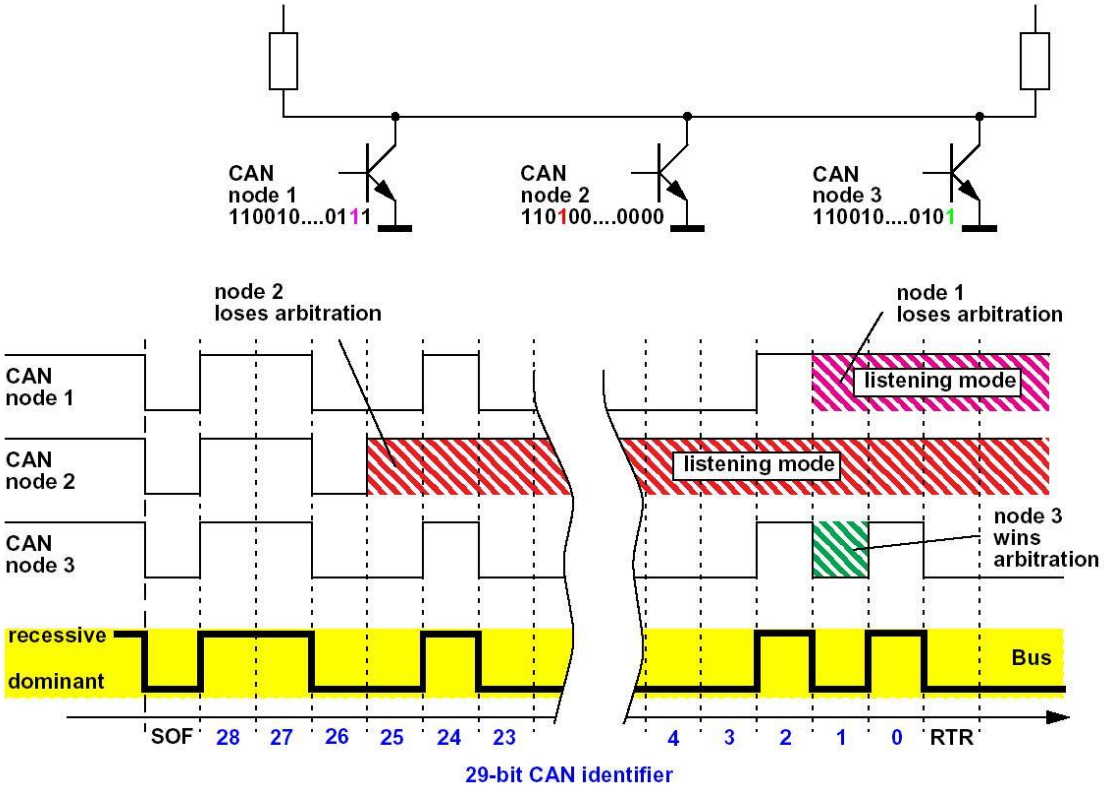


Figure 4-5 – Bus Arbitration

If a node loses the arbitration, the frame pending for transmission will be automatically retried after the next interframe space.

4.4 Hardware Acceptance Filtering

Hardware acceptance filtering is implemented in the CAN controller and utilizes the message identifier. If hardware acceptance filtering is utilized, the processor should handle all messages that have passed the acceptance filter. A well-defined data object code field, as defined in this document and/or by the systems designer, will allow efficient hardware acceptance filtering.

4.5 Error Handling

CAN utilizes different methods to recognize errors during transmission. These methods are divided into three parts: error recognition, error treatment, and error containment. The error recognition and fault tolerance performance as given by ISO 11898 shall be the minimum standard to be achieved.

Note: CAN FD ESI signifies the error state of the transmitting node and can be used for health management.

4.5.1 Error Recognition

There are five different fault types that may be detected by the CAN controller: Bit error, Bit-stuffing error, CRC error, Form error, and Acknowledgement error.

4.5.1.1 Bit Error

Transmitters monitor their own transmissions and are able to detect bit errors by comparing the transmitted and received bits of the message. Bit error means a transmitted bit has not been received with the same logic value as it was originally sent. Exceptions are the identifier bits and the acknowledgement time slot.

4.5.1.2 Bit-Stuffing Error

Bit-stuffing error occurs when six or more consecutive bits with the same logic value have been detected during a message. The only exception is the end of frame bit sequence with 7 recessive bits which is not regarded as an error.

4.5.1.3 CRC Error

A CRC error is generated when a transmitted CRC checksum does not match the value calculated by the receiver.

4.5.1.4 Form Error

A form error means one or more of the pre-defined bits have been found to be incorrect. Examples are a CRC delimiter that is not a recessive bit or the end of frame bit sequence that is not correct. As an exemption, a dominant last bit of the end of frame sequence is not recognized as form error.

4.5.1.5 Acknowledgment Error

An Acknowledgement Error is generated when the transmitter has not received a dominant bit in the acknowledgement slot which means that a frame was not correctly recognized by any CAN node. The transmitter will continue to transmit the same message until it receives an acknowledgement.

4.5.2 Error Treatment

The detection of error active CAN nodes results in an error frame sent by the node, which first detected the error. All other CAN nodes receive that error frame and answer with their own error frames in return provided that they are in the error active state. As a consequence, all nodes will discard the message and the message will be repeated at least once (retransmission). In this way, global data consistency is assured.

4.5.3 Error Containment and Bus Off Management

Local disturbances or failures of one node should not block the bus by its frequently sending active error frames. Each CAN node has therefore an individual error counter for received and transmitted corrupted messages called the Receive Error Counter (REC) and Transmit Error Counter (TEC), respectively. Depending on the error situation, REC and TEC are increased differently. Please refer to the latest edition of the ISO 11898 for further information on the subject.

The following three operation modes depend on error counters and are explained in the figures below:

1. Error active mode, (REC ≤ 127 and TEC ≤ 127)
   This is the normal mode in which the node will send and receive messages. Active error flags consisting of dominant bits will be sent if errors occur.
2. Error passive mode, (REC ≥ 128 or TEC ≥ 128 and TEC ≤ 255)
   Messages will still be received and sent in this mode but errors are signaled to other nodes by passive error flags containing only recessive bits.

3. Bus off mode, (TEC > 255)
   Bus off mode means the node has logically disconnected itself from the bus. In this condition, it neither receives nor sends frames and also does not send any error frames. This function prevents a faulty node from jeopardizing the network.

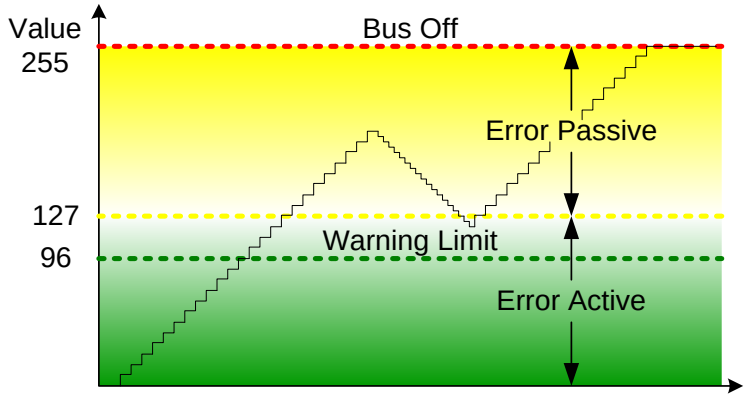


Figure 4-6 – Transmitter Error Count (TEC) Versus Error Mode

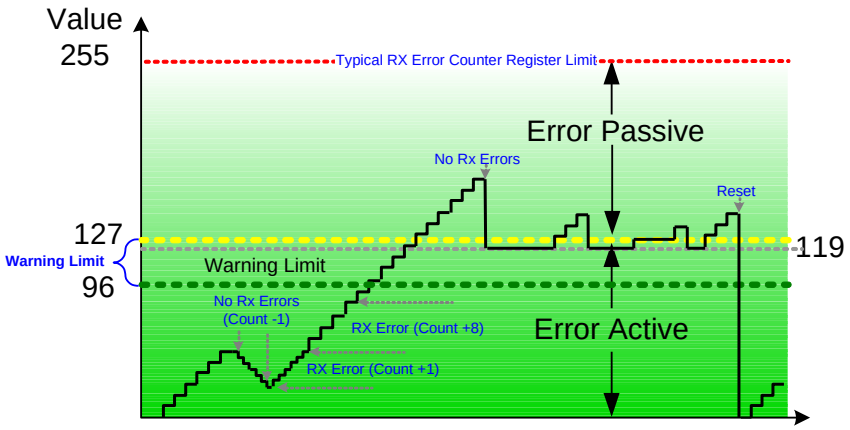


Figure 4-7 – Receiver Error Counter (REC) Versus Error Mode

COMMENTARY

The CAN controller's implementation of the REC may vary as there is room for interpretation of the CAN Specifications. For example, REC may exceed 128. The system designer should be aware of this possibility.

4.5.3.1 Monitoring of Counters

The CAN controller is expected to use the error counters described in Section 4.5.3, Error Containment and Bus Off Management, to manage its state and comply with ISO 11898. Most CAN controllers also make this information available to the Application Layer. These are useful for network health monitoring and fault reporting purposes. In order to create the Periodic Health Status Message (PHSM) (see Section 5.4), the Application Layer should monitor, at a minimum, the TEC and REC counters, the number of entries into Bus Off, and the number of accumulated

Transmit and Receive Errors. Further definition and transmission of these counters is provided in Section 5.4.

4.5.3.2 Management of "Bus-off"

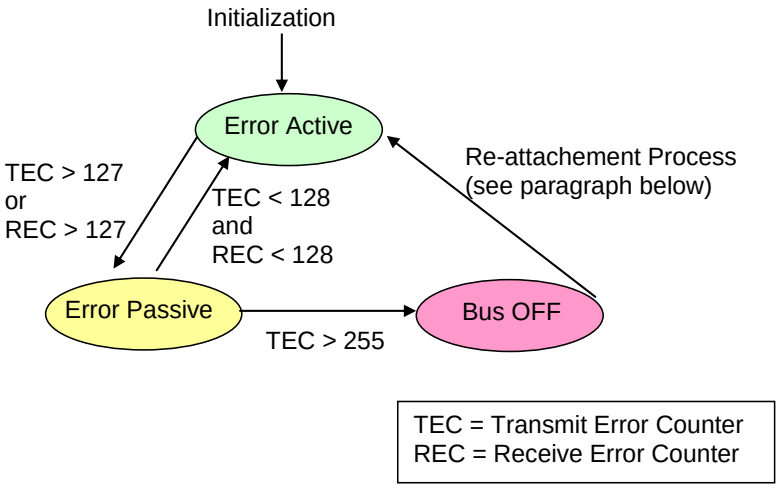


Figure 4-8 – Error Passive – Error Active State Diagram

After bus-off, a node may return to normal mode, i.e., error active. This corresponds to the re-attachment process. ISO 11898-1: 2015 (see Chapter 13, "Description of Supervisor") describes that two requirements are necessary for the re-attachment process:

- REC and TEC both have values of zero (0)
- The node shall have monitored 128 occurrences of 11 consecutive recessive bits. These 11 recessive bits correspond to the ACK delimiter + EOF + IFS bits mean that the bus has recovered with a substantial healthy comportment.

ARINC Specification 825 recommends the following steps for robust behavior:

- The application layer should initiate a Normal-Mode Request of the CAN controller.
- A system defined timeout before Normal-Mode request should be implemented.
- The number of re-attachments allowed should be system defined according to the availability and reliability requirements.

COMMENTARY

Some CAN controllers are not ISO 11898-1: 2015 compliant regarding the re-attachment process. It is the node designer's responsibility to confirm compliance. The system designer should be aware that some controllers include an auto bus-on feature which automatically initiates a normal mode request. This behavior is discouraged in the system design.

4.6 Node State Machine

The CAN controller supports the necessary node states as shown in the following figure. The system designer shall determine the needed transitions according to the safety and reliability objectives.

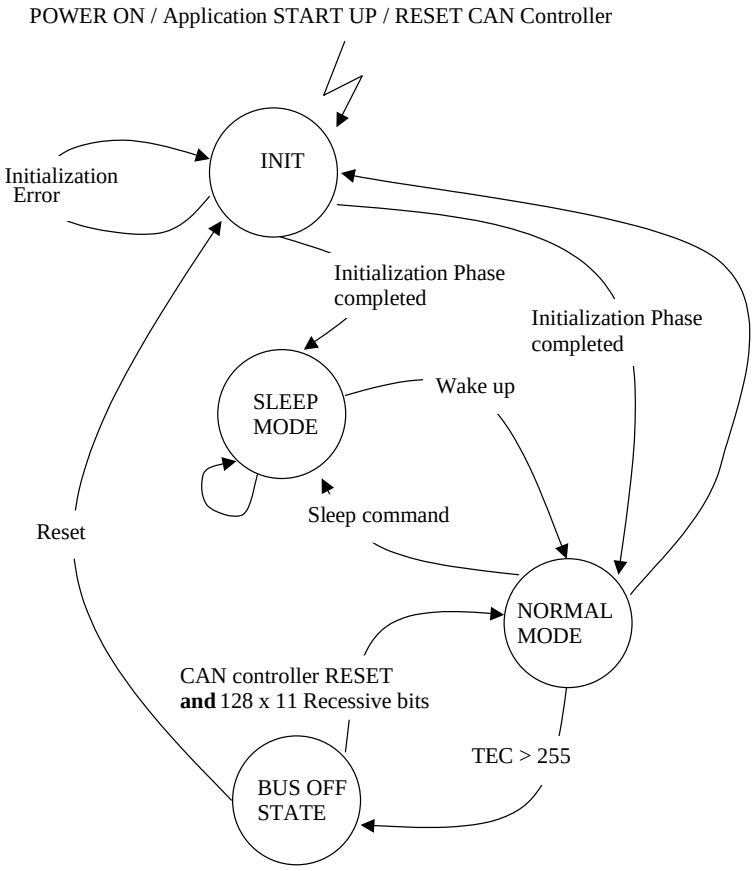


Figure 4-9 – Node State Machine Diagram

The system designer should design the CAN node such that the CAN transceivers are non-disruptive to bus communication until the CAN controller completes the initialization process. The node transceiver should produce dominant bits only during normal mode operations (Error Active or Error Passive). Refer to Section 7.4.3 for additional recommendations.

CAN interface states:

- INIT PHASE: After a POWER ON, the application starts up and CAN interface is initialized.
- SLEEP MODE: The sleep mode is a state in which the node is in low power mode. The transceiver is still active and able to wake up the node by activity on the bus or internal conditions. The sleep mode is currently not used in aircraft environment. This standard does not discuss Sleep mode as this is not recommended for aviation applications.
- NORMAL MODE: The node transmits data and processes received data.
- BUS OFF: The node is removed from actively communicating on the network and has no more influence on the bus. The CAN transceiver is disabled. The

node does not have any influence on the bus and is incapable of sending any frames, will not acknowledge messages, and will no longer send error frames.

4.7 Performance and Robustness Considerations

CAN nodes shall sustain and properly process CAN messages of up to 100% of bus load. Even when the data frames are sent to the same buffer and have the same identifier, the node shall continue its intended function (no reset or abnormal propagation). The controller shall transmit all the frames to the application without losing any data. Performance of the other interfaces shall not be diminished.

5.0 CAN COMMUNICATION

Section 5 defines the ARINC 825 CAN Message ID structures. All ARINC 825 compliant messages **shall** utilize these identifier structures.

5.1 Communication Concept

This standard is based on utilizing the extended frame identifiers specified in ISO 11898. Utilizing the extended frames (29-bit identifiers) provides an adequate number of bits to divide the identifier into several sub-fields. These sub-fields are a key issue in employing the identifier bits not only for the data object identification and transmission prioritization inherent to CAN but also for the purpose of creating a standardized application layer.

ARINC 825, based on Classical CAN, provides provisions for 11-bit CAN identifiers to coexist on an ARINC 825 network. While this use is not encouraged for interoperability reasons, it permits the use of already existing equipment in certain applications, e.g., Frame Migration Channel.

Note: The system designer must consider that Classical CAN controller chips are only interoperable with CAN FD chips operating exclusively in Classical CAN mode. However, CAN FD messages cannot be utilized in these configurations.

COMMENTARY

11-bit identifiers potentially provide additional bandwidth and latency capability benefits for both CAN FD and Classical CAN. However, the added complexity of a new identifier may not justify expanded use of 11-bit identifiers at this time. Expanded use of 11-bit identifiers can be considered for a future supplement.

5.1.1 CAN and the ISO/OSI Model

Complex system networking requires multiple layers of communication. Lacking any definition of the Open Systems Interconnect (OSI) layers 3-7 and without an additional application layer, the basic CAN specification provides only pure object-oriented broadcast data transmission:

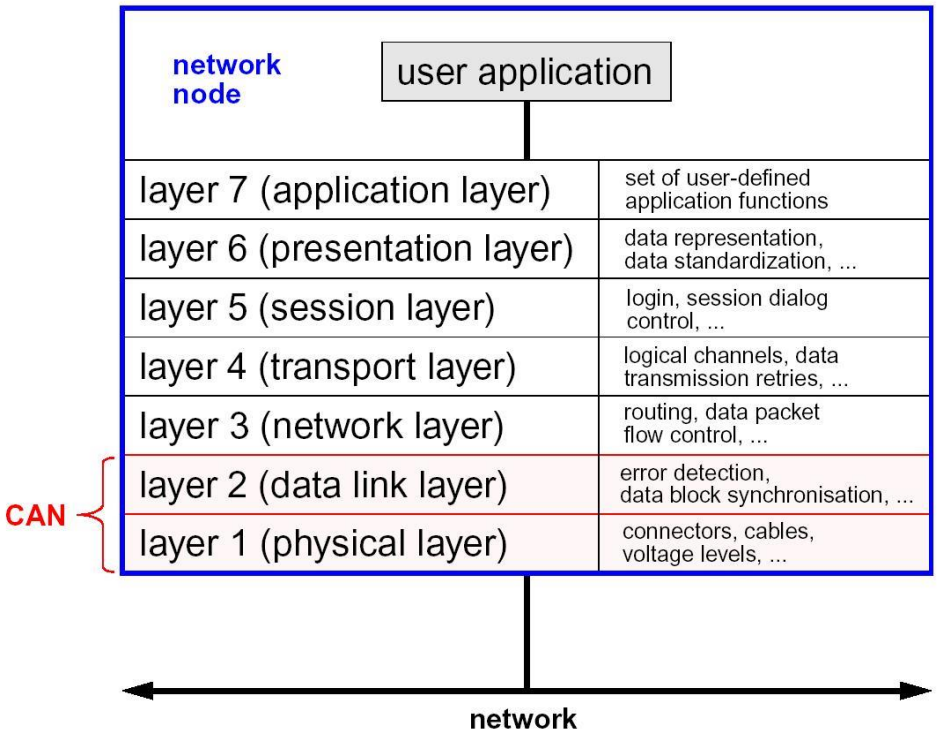


Figure 5-1 – ISO/OSI Layer Model for CAN

Consequently, some OSI layer 3-7 functions have to be provided to turn CAN into a network with multiple communication layers supporting one-to-many and peer-to-peer communication. While one-to-many is readily available with “raw” CAN, peer-to-peer communication is not available and requires node identification and node addressing. Both are defined by ARINC Specification 825.

5.1.2 ARINC 825 Communication Types

CAN is a half-duplex, multi-drop network using “broadcast communication.” The advantage is that it creates inherent data consistency between all nodes in the network. Both periodic and aperiodic data transmission during normal operation is possible.

ARINC Specification 825 CAN messages are based on one-to-many, Directed Message, and peer-to-peer communication.

One-to-many communications requires only little overhead and makes effective use of available bandwidth. Nevertheless, to prevent a receiving node from processing data that it does not need, hardware acceptance filtering within the CAN controller may be used to block incoming messages not directed to the affected node from being passed upward into layers implemented in software, thereby saving precious CPU time.

Directed Message communication provides the ability for a node to explicitly target another node with a message. The resulting node-to-node communication enables targeted monologues or dialogs between two end points on a bus. In addition, it enables multiple dialogs between the same pair of nodes through the use of node identifiers and ports. These effectively create dialogs similar to UDP/IP communication on a subnet.

Peer-to-peer communication allows client/server type interactions between all nodes in the network and is necessary to request certain actions from a specific node. The idea behind this concept is that any node in the network may be client for one task and server for another task at the same time. By this concept, functions may be distributed over the network, unleashing the real power of distributed systems. Using peer-to-peer communication involves logical communication channels that are created for that purpose by ARINC Specification 825. Peer-to-peer communication distinguishes between connectionless (no response transmitted) and connection-oriented (handshake type) communication (similar to UDP/IP and TCP/IP).

5.2 CAN Identifier Usage

In order to provide multiple network layers, the 29-bit message identifier field of the CAN messages is divided into several sub-fields. The sub-field structure implements logical communication ports and supports one-to-many as well as peer-to-peer communication in an effective way to minimize the software to implement this ARINC Specification 825. This is beneficial as many CAN nodes will be cost-driven implementations with limited computing power, ruling out a bulky communication layer with lots of overhead. By specifying user-defined communication channels, the system designer is given a high level of freedom to make use of the channels according to the designer's needs.

5.2.1 Logical Communication Channels

The Logical Communication Channels (LCCs) serve the purpose of creating independent network layers that support different functions while isolating them from each other. They are coded into the upper three bits of the identifier fields (bits 29-31 are shown for alignment with respect to 32-bit CPU registers only):

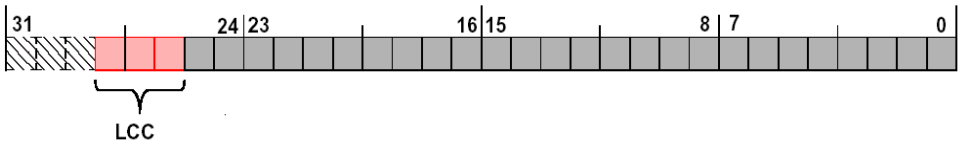


Figure 5-2 – Logical Communication Channel (LCC)

The value of the LCC bits has the highest impact on message prioritization. Accordingly, the channels are arranged according to their importance with respect to the overall system.

Table 5-1 – Logical Communication Channel Assignment

Channel Number	Channel Acronym	Description	LCC Bits	Message Priority
0	**EEC**	Exception Event Channel	**000**	Highest
1		Reserved	**001**	
2	**NOC**	Normal Operation Channel	**010**	
3	**DMC**	Directed Message Channel	**011**	
4	**NSC**	Node Service Channel	**100**	
5	**UDC**	User-defined Channel	**101**	
6	**TMC**	Test and Maintenance Channel	**110**	
7	**FMC**	CAN Base Frame Migration Channel	**111**	Lowest

The usage of the Logical Communication Channels is as follows:

- The Exception Event Channel (EEC) shall only be used for fast and high priority transmission to override other message transfers. These events will usually require some sort of immediate action (i.e., system degradation, shifting functions to other units or communicating the events to higher order systems). This channel is used for one-to-many communication.
- The Normal Operation Channel (NOC) shall be used for the transmission of aircraft operational periodic or aperiodic data based upon one-to-many communication. All data that is not assigned to another channel shall be transmitted over the NOC.
- The Directed Message Channel (DMC) shall only use the Directed Message protocol and is intended for targeted messaging and dialogs through the use of node addressing and port definitions. Keeping these messages on LCC 3 automatically gives them lower priority than parametric broadcast messaging (one-to-many) so that they have less impact on timing. These messages may be periodic or aperiodic depending on system design.
- The Node Service Channel (NSC) is for peer-to-peer communication for client/server type services. These services may be connectionless (no response transmitted) or connection-oriented (handshake type communication).
- The User-Defined Channel (UDC) is available for user-defined communication using CAN extended frames that do not fit into one of the other channels for any reason. As long as the LCC code is retained, the rest of the sub-field structure may be modified for this purpose. These channels may be used for one-to-many communication only as required. Whenever possible, however, it is strongly recommended the other defined channels be used (EEC, NOC, DMC, NSC, TMC).
- The Test and Maintenance Channel (TMC) supports the communication for test and maintenance functions. The communication is peer-to-peer for client/server interactions and may be connectionless (no response transmitted) or connection oriented (handshake type communication).
- The CAN Base Frame Migration Channel (FMC) is available for CAN application layers using CAN base frames with 11 bit identifiers (e.g., CANaerospace). As long as the LCC code is retained, the rest of the sub-field structure may be modified for this purpose. These channels may be used for peer-to-peer as well as for one-to-many communication as required. However, it is highly discouraged to use this channel for application layers which are not absolutely free of potential deadlock scenarios caused by single source bus masters, etc. Also, the communication within this channel shall be included in the bandwidth management calculations.
- Channels marked as “reserved” are intended for future revisions of this standard and shall not be used for other purposes.

5.2.2 ARINC 825 CAN Identifier Structure

Besides the Logical Communication Channel assignment, the identifier field is structured further to support message source identification, routing, and data integrity checking. In this manner, ARINC 825 defines three communication models: a one-to-many format used for broadcast messaging, Directed Messaging used for targeted node-to-node communication, and a peer-to-peer format used for node services defined later in this standard.

Each of these defined communication types have different requirements with respect to node addressing and communication which are addressed in the following sections.

5.2.2.1 One-to-Many Identifier Structure

The One-to-Many Identifier is used with broadcast-style communication and is ideal for parametric data transfer. The intended use is for the LCCs 0 and 2 (EEC and NOC); however, this ID structure may be used on the UDC and the FMC.

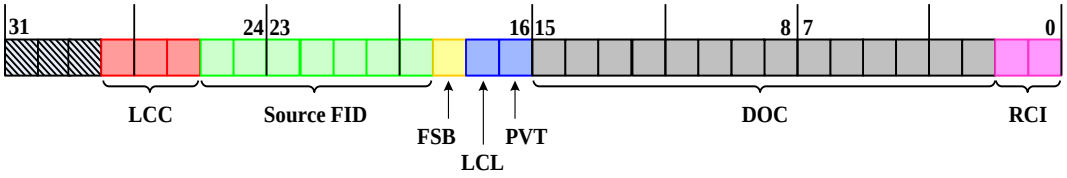


Figure 5-3 – ARINC 825 One-To-Many CAN Identifier Structure

The usage of the sub-fields is as follows:

- The Function Code Identifier (Source FID) specifies the system and/or subsystem the message originates from and is specified through system design. It should reflect the importance of the system with respect to the entire aircraft. FID number 127 is pre-assigned by this standard and may only be used for Test and Maintenance systems that are temporarily plugged into the network. Likewise, FID number 0 is pre-assigned and refers to all FIDs at the same time. The FIDs shall be used on all defined LCCs except UDC and FMC. The FIDs in Table 5-4 shall be used where applicable.
- The Functional Status Bit (FSB) is used to communicate the validity of the data contained in the payload. More information on the use of this bit is provided in Section 5.3.3, Message Level Functional Status, and Section 7.5.7, Parametric Functional Status.
- The “Local” bit (LCL) is set to a one (1) for messages which are designated only for the network the transmitting node resides in. Gateways should not transfer these messages to other network segments.
- The “Private” bit (PVT) identifies messages which have no meaning to nodes other than those which are specifically programmed to use them. Messages with this bit set may have no published description and are for private use only.
- The Data Object Code (DOC) allows specification of 2^{14} different data objects per function. For one-to-many communication, the relationship between the data specification and the DOC is given in the profile specified by system design (refer also to Section 5.3 of this document). For peer-to-peer communication, the DOC is structured further to support node addressing.
- The Redundancy Channel Identifier (RCI) allows the users to identify redundant messages and is optional. Up to four redundant channels (0, 1, 2, and 3 corresponding to the redundant channels A, B, C, and D) are supported. If redundancy is not used in a given application, the RCI may be assigned by the designer as part of the DOC.

5.2.2.2 Directed Message Identifier Structure

The Directed Message identifier is used for directly addressing unique node pairs and creating dialogs. It is restricted to LCC 3 (DMC).

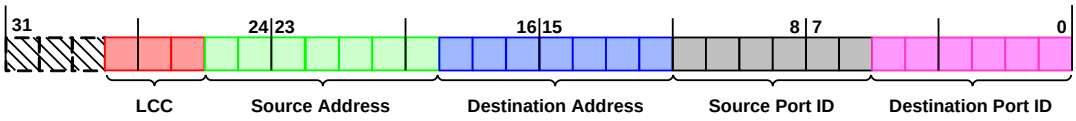


Figure 5-4 – ARINC 825 Directed Message CAN Identifier Structure

In order to use the Directed Message communication channel, each node on a bus shall be assigned a unique 7-bit node address. This allows for up to 128 unique node addresses per bus. Address 0 is reserved and not assigned to any node.

COMMENTARY

Unlike the one-to-many message structure, the Directed Message concept defined here moves the focus of the arbitration to the node interface, thus all messages from a node with a higher priority address will out arbitrate all messages from a node of lower priority. During arbitration between messages on the DMC, the node assigned the lowest Node ID will win arbitration. The System Designer is cautioned to consider priority inversion on data transmitted on the DMC.

The usage of the sub-fields is defined as follows:

- Source Address represents the unique address of the source node of that particular message.
- Destination Address represents the unique address of the destination node of that particular message.

In a dialog between two nodes, the Source and Destination addresses will switch back and forth depending on the direction of the message.

- Source Port ID represents the ephemeral port that the message source node has assigned for that particular message or dialog. Note that the Source Port ID will always be an ephemeral port.
- Destination Port ID represents the port that the message is intended for.

Similar to the Source and Destination addresses, these Port IDs will switch back and forth depending on the direction of the dialog. A node starting a new Directed Message communication addresses a Well Known Port (WKP) of the receiving node to communicate the purpose of the message. The WKPs are defined below. In any subsequent messages in that dialog, this WKP will be replaced with the node's ephemeral port.

Table 5-2 – Directed Message Port Definitions

Port Number	Purpose
0-23	Used as Ephemeral Ports to be assigned during dialogs
24-31	Reserved for future Well Known Port Definition
32-47	Reserved for future definition
48-63	User Defined Functions

The concept of Port is introduced from Internet Protocol. In essence, a port is a key that the nodes use to determine which dialog a message is a part of. There are two types of ports. The first is a Well Known Port (WKP): a port with a predefined meaning that communicates the purpose of the targeted message. Following the identification of the WKP in the first message, nodes move the dialog onto ephemeral ports. An ephemeral port is a port assigned by each node for a particular dialog. It is unique from other ports currently being used for that sender/destination pair (although the same port number can be used when targeting other nodes) and lasts the duration of the dialog. Once the dialog is finished, the port can be reused.

Concept of Operation:

A Directed Message is used similarly to UDP/IP communication. To initiate a message, the sender targets a node and informs the node of the purpose of the message by using the corresponding WKP as the Destination Port field. The initiator indicates its desired ephemeral port for the conversation in the Source Port field. If this interchange is a dialog (both sides sending), the initial destination node will communicate its desired ephemeral port in the response message. The two ephemeral ports are now assigned for the remainder of the dialog.

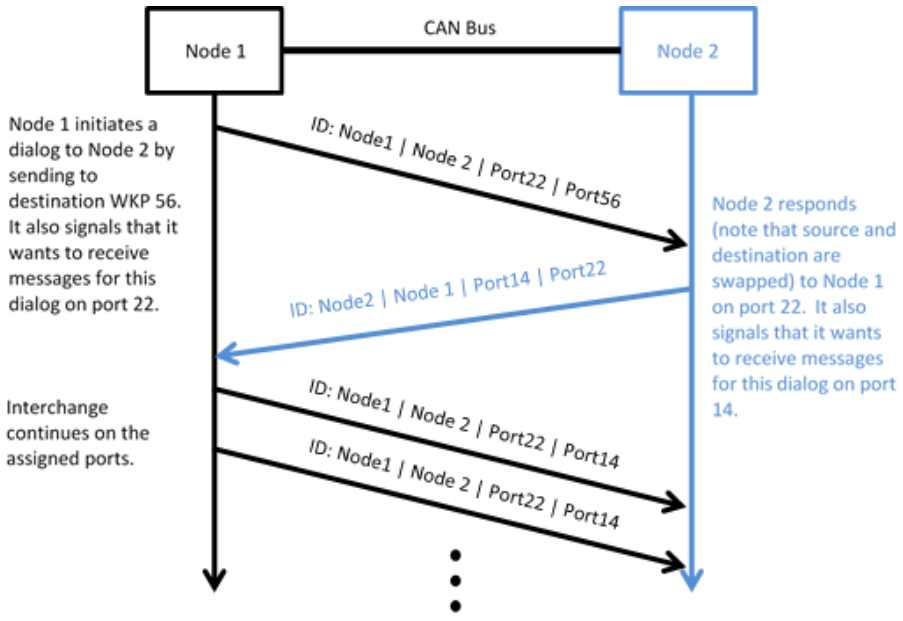


Figure 5-5 – ARINC 825 Directed-Node-Message Example

Multicast Service:

Using Directed Message, a multicast is also possible. To do this, the message structure is the same except that the initiating message has the Destination address set to 0. The initiator sets its own ephemeral port to be used in any responses, as appropriate, from all nodes (this port must not be in-use for any other conversation that the initiating node is currently having). The initial multicast message uses the Destination Port representing the desired function. This is intended to be used for functions like requesting all nodes on the bus to do the same action, such as send their configuration info. The responses to this message will use the responder's address as the Source address and the initiator's address as the Destination address.

5.2.2.3 Peer-to-Peer Identifier Structure

The following defines a specific structure for peer-to-peer communication built off the one-to-many communication type. This peer-to-peer format (used with the NSC/TMC) allows a client node to initiate a peer-to-peer dialog to another uniquely addressed server. This communication type expands the DOC field from the one-to-many type into a specific sub-structure for peer-to-peer communication. This sub-structure allows identification of the addressed nodes by specifying their Node ID (NID), comprising of Function ID (Server FID) and Server ID (SID).

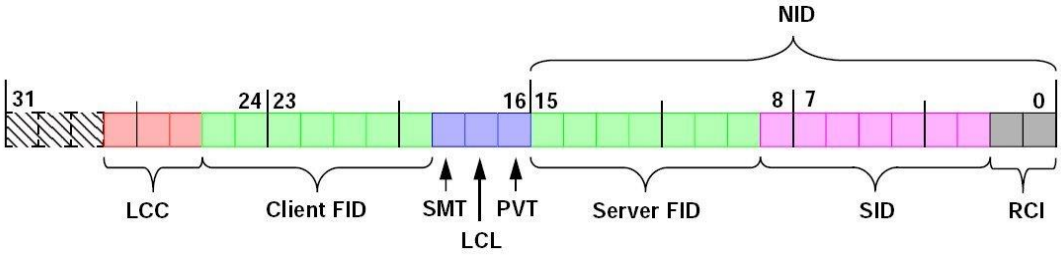


Figure 5-6 – Identifier Field Structure for Peer-to-Peer Communication

The NID field supports up to $2^7 – 1 = 127$ (0 is used for multicast) functions (or systems) with either $2^7 – 1 = 127$ (0 is used for multicast) or a maximum $2^9 – 1 = 511$ (0 is used for multicast) individual CAN nodes (or servers), depending on the use of the RCI. The "Local" (LCL) bit and the RCI field are used in the same manner as with the one-to many identifier field structure, while the "Private" (PVT) bit indicates opaque data transfer during the Data Upload/Download Services (DUS/DDS). Individual nodes within a system are addressed using their personal FID and SID.

Multiple nodes within a function are addressed simultaneously using the multicast SID number 0. Likewise, all nodes within all functions are addressed using the multicast FID number 0 in conjunction with the multicast SID number 0. FID number 0 and SID number 0 are pre-assigned for this purpose. The Service Message Type (SMT) bit indicates the direction of the data flow between client and server. SMT is set for Node Service Request messages and cleared for Node Service Response messages.

COMMENTARY

Servers always respond with their own Node-ID, consisting of Server FID and Server ID.

5.2.2.4 Eleven-Bit Identification Format

The following defines a specific structure for utilizing the eleven-bit id for the Frame Migration Channel. The User-Defined Field (UDF) provides 256 values for the system designer to specify

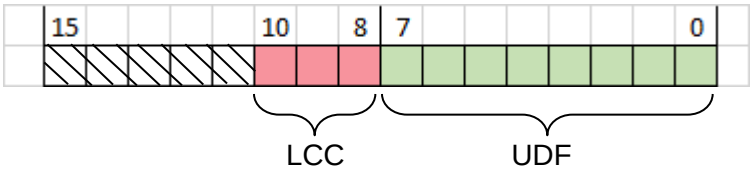


Figure 5-7 – Identifier Field Structure for Eleven Bit Id Format

5.3 Interoperability

Aside from a standardized communication mechanism as described before, interoperability also requires standardized data formats and a known Data Object Code (DOC) distribution also referred to as a “profile” (not to be confused with the definition of Profile in ARINC Specification 664).

As most of the embedded systems use processors adhering to network byte order Big Endian CPU architectures, Big Endian encoding shall be used exclusively. According to the Big Endian definition, the Most Significant Bit (MSB) of any datum is arranged leftmost and transmitted first as shown in Figure 5-8.

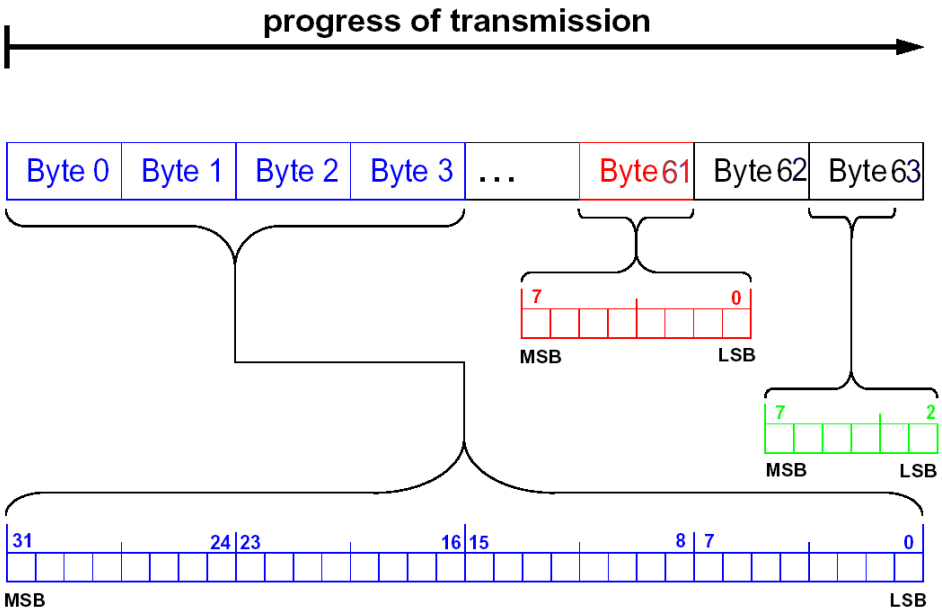


Figure 5-8 – Message Payload Data Arrangement Example

5.3.1 Data Formats

For interoperability reasons, this standard provides data format definitions. The payload data of a CAN message will consist of one or more of the following types (using up to 8 bytes of data, various combinations of these types may be transmitted within a single CAN message):

- Signed Integer
- Unsigned Integer
- Floating-Point
- Enumerated
- ASCII
- Opaque

The definition of these data types is defined in Table 5-3.

Table 5-3 – Permissible Data Types

Data Type Code (DTC)	Data Type Name	Range	Bits	Explanation
0	NODATA	n.a.	0	"No data" type
1	ENUM	n.a.	1-64	Enumerated data
2	CHAR	-128 to +127	8	Two's complement char integer
3	UCHAR	0 to 255	8	Unsigned char integer
4	ACHAR	0 to 255	8	ASCII character
5	SHORT	-32768 to +32767	16	Two's complement short integer
6	USHORT	0 to 65535	16	Unsigned short integer
7	LONG	-2^{31} to $+2^{31} - 1$	32	Two's complement Integer
8	ULONG	0 to $2^{32} - 1$	32	Unsigned integer
9	FLOAT	$+/-(2^{-1}/2^{22})2^{-126 \ldots +127}$	32	Single precision floating-point value according to IEEE-754-1985
10	LONG64	-2^{63} to $+2^{63} - 1$	64	Two's complement Integer
11	ULONG64	0 to $2^{64} - 1$	64	Unsigned integer
12	DOUBLE	$+/-(2^{-1}/2^{51})2^{-1022 \ldots +1023}$	64	Double precision floating-point value according to IEEE-754-1985
13	OPAQUE	n.a.	1-64	Data type with transparent content
14	BOOL	0 to 1	1	Logical
15	BCD	0 to 9	4	Binary Coded Decimal

COMMENTARY

Every effort should be made to use only the data primitives listed in Table 5-3 for ARINC 825 networks to ensure interoperability. When circumstances require that some data structure other than those listed must be used, it should be placed in the opaque data type. Use of the opaque data type, however, should be avoided as much as possible.

For the purposes of ARINC Specification 825, opaque is defined as "obscure and unintelligible in meaning" which is exactly how opaque data may be viewed by other users. Opaque data may not be visible or difficult to interpret for System Integration activities, Flight Test, the Aircraft Interface Control Document (ICD), simulation tools as well as Commercial Off the Shelf (COTS) analyzers, integration, and test tools. Therefore, data that needs to be made visible to other users, when taking into account the life cycle of the product, should avoid the use of "opaque" data primitives as much as possible.

5.3.2 Profiles

The ARINC 825 communication profile specifies the network traffic and constitutes the basis for interpretation of all data (or parameters) on the network.

Communication profiles define the interface to the ARINC 825 network and ensure interoperability. The profile definition must be known by each node in the network to effectively participate in the ARINC 825 network communication. Using the Source Function Code Identifier (FID) and the Data Object Code (DOC) as an index to the communication profile database will define what physical value is represented by the parameter and provides additional information, i.e., data type, data range and scaling. A communication profile description shall be defined for any ARINC 825 device. Use of the communication profile database file format as described in ATTACHMENT 1 is highly recommended. An example of the file format is given in APPENDIX B.

To aid communication profile development by system designers, a sample DOC assignment list covering some important aircraft data is provided in APPENDIX D.

5.3.2.1 Aircraft Functions

Grouping parameters based on the ARINC 825 Function Code Identifier (FID) provides an approach for system designers to logically separate individual functions and provides functional priority to the ARINC 825 network. ARINC Specification 825 has reserved FIDs to facilitate future growth. The FIDs with label *Reserved for System Integrator* may be allocated based on the system application requirements. To facilitate interchangeability, the pre-assigned FIDs in Table 5-4 shall be adhered to.

Table 5-4 – ARINC 825 Function Code List

FID	Function [1]
0	Multicast Function Code ID
1	Reserved for Future Growth
2	Reserved for Future Growth
3	Reserved for Future Growth
4	Flight State
5	Reserved for System Integrator
6	Reserved for System Integrator
7	Reserved for System Integrator
8	Indicating/Recording Systems: Central Warning Systems
9	Indicating/Recording Systems: Central Display Systems
10	Flight Controls
11	Engine Controls
12	Engine Indicating
13	Electrical Power
14	Auto Flight
15	Integrated Modular Avionics
16	Engine Fuel And Control
17	Fire Protection
18	Fuel

5.0 CAN COMMUNICATION

FID	Function [1]
19	Indicating/Recording Systems: Central Computers
20	Stabilizers
21	Wings
22	Propellers/Propulsion
23	Rotors Flight Control
24	Rotor(S)
25	Rotor Drive(S)
26	Tail Rotor
27	Tail Rotor Drive
28	Exhaust: Thrust Reverser
29	Air
30	Power Plant
31	Oil
32	Ignition
33	Ice And Rain Protection
34	Landing Gear
35	Landing Gear: Wheels And Brakes
36	Doors: Landing Gear
37	Doors: Monitoring And Operation
38	Doors: Passenger/Crew
39	Indicating/Recording Systems
40	Reserved For System Integrator
41	Reserved for Future Growth
42	Reserved For System Integrator
43	Defensive Aid System
44	Engine Turbine/Turbo Prop Ducted Fan/Unducted Fan
45	Turbines
46	Propulsion Augmentation
47	Accessory Gear-Boxes
48	Hydraulic Power
49	Water Ballast
50	Communications
51	Communications: Speech Communications
52	Navigation
53	Air Conditioning: Pressurization Control
54	Indicating/Recording Systems: Instrument & Control Panels
55	Indicating/Recording Systems: Independent Instruments
56	Fuselage

FID	Function [1]
57	Air Conditioning: Temperature Control
58	Communications: Data Transmission And Automatic Calling
59	Communications: Static Discharging
60	Payload Handling (Military)
61	Communications: Integrated Automatic Tuning
62	Lights: Flight Compartment
63	Lights
64	Lights: Emergency Lighting
65	Oxygen
66	Information Systems
67	Air Conditioning
68	Doors: Emergency Exit
69	Doors: Cargo
70	Nacelles/Pylons
71	Doors: Service And Miscellaneous
72	Communications: Audio Integrating
73	Starting
74	Reserved For System Integrator
75	Reserved for Future Growth
76	Reserved For System Integrator
77	Airborne Auxiliary Power
78	Pneumatic
79	Vacuum
80	Doors
81	Windows
82	Exhaust
83	Folding Blades/Pylon
84	Exhaust: Collector/Nozzle
85	Air Conditioning: Heating
86	Air Conditioning: Distribution
87	Air Conditioning: Compression
88	Air Conditioning: Cooling
89	Communications: Satcom
90	Communications: Interphone
91	Communications: Passenger Address, Entertainment And Comfort Available
92	Cabin Systems: Cabin Core System
93	Communications: Audio & Video Monitoring
94	Indicating/Recording Systems: Automatic Data Reporting Systems

FID	Function [1]
95	Lights: Passenger Compartment
96	Exhaust: Noise Suppressor
97	Exhaust: Supplementary Air
98	Water Injection
99	Reserved For System Integrator
100	Reserved for Future Growth
101	Reserved For System Integrator
102	Central Maintenance System (CMS)
103	Cargo And Accessory Compartments
104	Indicating/Recording Systems: Recorders
105	Lights: Cargo And Service Compartments
106	Water/Waste
107	Cabin Systems
108	Cabin Systems: Cabin Mass Memory System
109	Cabin Systems: External Communication System
110	Cabin Systems: Inflight Entertainment System
111	Doors: Fixed Interior

FID	Function [1]
112	Doors: Entrance Stairs
113	Vibration And Noise Analysis (Helicopter Only)
114	Air Conditioning: Moisture/Air Contaminant Control
115	Equipment/Furnishings
116	Lights: Exterior
117	Cabin Systems: Cabin Monitoring System
118	Cabin Systems: Miscellaneous Cabin System
119	Standard Practices And Structures - General
120	Reserved For System Integrator
121	Reserved For System Integrator
122	Reserved For System Integrator
123	Galley Inserts
124	Reserved for Future Growth
125	Periodic Health Status Message (PHSM)
126	Reserved for ARINC 826 SDL Target
127	Temporary Test & Maintenance

Note: Equipment type descriptions are found in the Air Transport Association “Information Standards for Aviation Maintenance Specification,” ATA Specification 2200 see Table 3-1 - 3.5 “Definitions of Aircraft Groups, Systems, and Subsystems” in Section 1.8.

5.3.3 Message Level Functional Status

Unlike typical avionic networks, the ISO Classical CAN protocol does not have an inherent ability to indicate functional status for data in a message. This is particularly imperative for complex aircraft employing highly integrated systems. Communication of data validity is an important part of the operation and reliability for airborne data networked systems.

Functional Status is used to communicate the validity of the data contained in the payload. ARINC 825 provides the Functional Status Bit (FSB) in the one-to-many format type (refer to Section 5.2.2.1, One-to-Many Identifier Structure), which is intended to communicate the validity of the data in the message payload. This makes ARINC 825 compatible with other avionic network protocols that use Functional Status, such as ARINC 664 and ARINC 429.

Note: ARINC 825 Directed Message and ARINC 825 peer-to-peer messages do not require validity.

There are four states needed to adequately address functional status:

- Normal Operation
- Functional Test

- No Computed Data
- Fail/Warn (FW)

To communicate that the data is valid, Normal Operation and Functional Test are used. To communicate invalidity, Fail/Warn and No Computed Data are used.

ARINC 825 nodes shall utilize the method in Table 5-5 to communicate message-level functional status. This applies for all CAN nodes and gateways.

Table 5-5 – ARINC 825 Functional Status Definitions

Functional Status	Meaning	Description
Normal Operation (NO)	Normal operation is encoded by the source and indicates the data is valid.	The FSB is set to 0 and the DLC is greater than 0.
Functional Test (FT)	FT is encoded by the source when the associated data is for test.	The FSB is set to 1 and the DLC is greater than 0.
No Computed Data (NCD)	The source is unable to compute or send valid data from reasons other than failure conditions.	The FSB is set to 0 and the DLC is 0.
Fail/Warn (FW)	Fail/Warn is sent by the source when a fault is detected that prevents it from calculating or outputting valid data. It is encoded by the source and analogous to ARINC 664's No Data (ND).	The FSB is set to 1 and the DLC is 0.

If a periodic message is not received within a defined timeframe, it is at the discretion of the system architect to determine how the system responds. If a periodic message is received whose DLC does not match the DLC defined in the message profile, then it is at the discretion of the system architect to determine how the system responds.

CAN is inherently different from other network types in that no functional status is provided. As such, mapping to other common Avionics networks is not a straight forward exercise. The defined mapping below is intended to prevent ambiguity between the functional validity of ARINC 825 and the functional status of three more common network types and provides subscribers an accurate interpretation of the data validity. Table 5-6 compares the ARINC 825 Functional Status, as determined by the combination of the ARINC 825 Functional Status Bit (FSB) and the value of the Data Length Code (DLC), across the two more prevalent avionic networks. ARINC 653 provides additional information of functional validity alignment across network standards.

Table 5-6 – Functional Status Comparison

Indicator	ARINC 825 FSB	ARINC 825 DLC	ARINC 664P7 FSS	ARINC 653P2 FS	ARINC 429 SSM
Normal Operation	0x0	>0	NO	0x03	Normal
Functional Test	0x1	>0	FT	0x0C	FT
No Computed Data	0x0	= 0	NCD	0x30	NCD
Fail/Warn	0x1	= 0	ND	0x00	FW

Functional Status may be needed at a parametric level either in addition to, or as an alternative to, the Message level functional status presented above. Parametric level

Functional Status is supported by ARINC 825 but is transparent. It is up to the designers to define it in the payload according to their needs, see Section 7.5.7, Parametric Functional Status, in the Design Guidelines for additional guidance on ways to embed functional status information within the payload of an ARINC 825 message.

COMMENTARY

Changes to the Functional Status will result in different CAN Identifier for a given message because the Functional Status Bit (FSB) is included in the One-to-Many Identifier. For a given message, the payload of the message should match the definition in the profile when the Message Level Functional Status is either Normal Operation (NO) or Functional Test (FT). A message with a Message Level Functional Status of No Computed Data (NCD) or Fail/Warn (FW) should not contain payload and the DLC should be set to zero (0).

5.4 Periodic Health Status

The ISO 11898-1: 2015 defines mechanisms for error detection implemented in each CAN controller (such as bit error, stuff error, CRC error, from error, ACK error). However, the understanding and the interpretation of permanent or intermittent faults need more than the ISO 11898-1: 2015 has defined.

To enhance fault detection, ARINC Specification 825 provides an additional mechanism for health monitoring on CAN by:

- Analysis of parameters associated with detected failures
- Actions for maintenance by indicating the failed node or wiring

To enable these functions, all connected nodes shall transmit a generic message with predefined data content, referred to as a Periodic Health Status Message (PHSM). To limit bandwidth consumption, this message shall be emitted at low periodicity: such as once a second.

5.4.1 Health Status Message Identifier

The Periodic Health Status Message is emitted in a special format of a connectionless peer-to-peer message with the LCC field set to TMC (6). For this purpose, the Client FID uses the predefined Periodic Health Status Message (PHSM) FID 125, as defined in Table 5-4. The unique Node ID (NID) is a concatenation of the Server FID of the emitting node, a unique server (SID), and the RCI as shown in Figure 5-9. The Server ID (SID) is a unique ID specific to each node of a server FID on a bus. The SMT bit of Periodic Health Status Messages is always set (1) indicating a Node Service Request. This request does not require a Node Service Response.

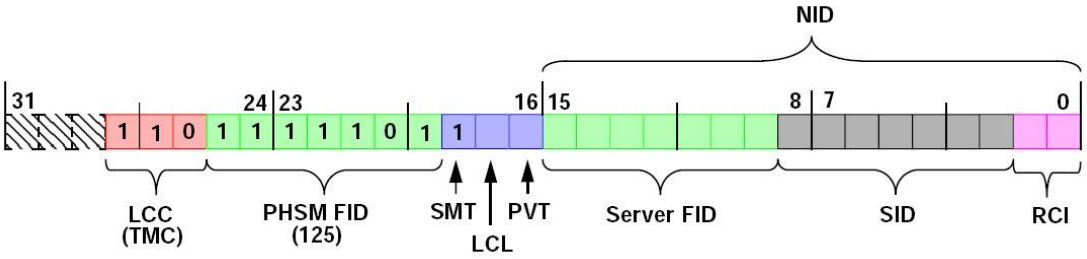


Figure 5-9 – Periodic Health Status Message ID Structure

Note: ARINC 825-3 revised the data payload of the PHSM from previous versions. In order to accommodate the need to differentiate between the different formats, it is recommended that the PVT bit be defaulted to a zero (0) value for Supplement 2 or below implementation and set to one (1) for Supplement 3 implementation.

5.4.2 Data Payload Content

To help correctly identify failed CAN units, and to provide the user or other nodes with an efficient diagnostic, a CAN status health message is defined as follows:

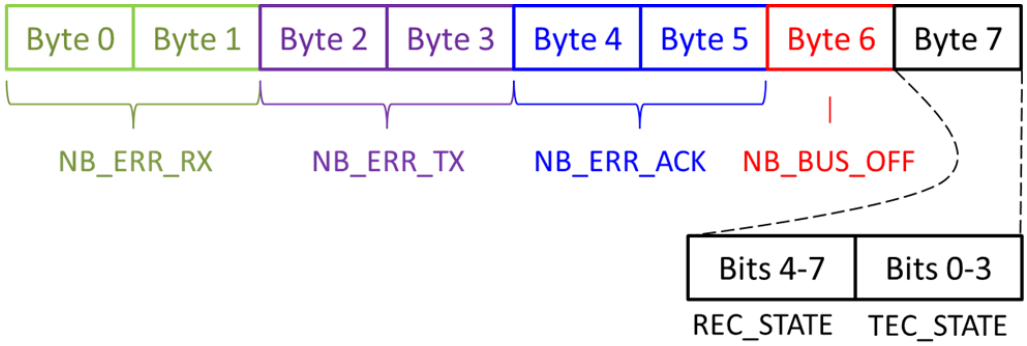


Figure 5-10 – Periodic Health Status Message Payload

With:

- NB_ERR_RX: Total number of detected receive errors since the processor was powered on (USHORT)
  - A Receiver error is an error that occurs while a node is a receiver. As defined by ISO 11898-1: 2015, a receiver is a “node when it is not a transmitter and the bus is not idle.”
- NB_ERR_TX: Total number of detected transmit errors, other than acknowledgement errors, since the processor was powered on (USHORT)
  - A Transmit error is an error that occurs while a node is a transmitter. As defined by ISO 11898-1: 2015, a transmitter is a “node originating a data frame…which stays transmitter until the bus is idle again or until the node loses arbitration.”
- NB_ERR_ACK: Total number of detected Acknowledgment Errors since the process was powered on (USHORT)

- NB_BUS_OFF: Total number of times the CAN controller has entered BUS_OFF since the processor was powered on (USHORT)
- ERR_STATES: This parameter consists of two fields: Receiver Error State (bits 4-7) and Transmit Error State (bits 0-3). Table 5-7 shows how these states are communicated.

Table 5-7 – PHSM REC/TEC State Definitions

REC/TEC Value (n)	REC_STATE/TEC_STATE
Controller Init/Disabled	0b0000
Bus Off	0b0001
0 ≤ n ≤ 95	0b0010
96 ≤ n ≤ 127	0b0100
128 ≤ n	0b1000

For example, a REC of 56 and a TEC of 145 results in the value 0b00101000 for PHSM Byte 7.

The management of these counters is under control of the Application Layer/CAN drivers. The information for updating these counters is available in most CAN controllers. However, if it is impossible to compute a parameter, the corresponding value will be replaced by 1's (0xFF or 0xFFFF).

In the case of Bus Off, the TEC and REC state will both be set to Bus Off (this avoids the potential ambiguity when the REC value is used to count down the 128 occurrences of 11 recessive bits).

In the PHSM, the NB_ERR_RX, the NB_ERR_TX, the NB_ERR_ACK, and the NB_BUS_OFF fields are counters. These counters shall start at zero (0) upon initialization. For each specified event, the counter shall be incremented by a count of one (1). Once the counter reaches its maximum possible value, the counter shall rollover and restart at zero (0).

Note: During Bus Off a node interface will not have the ability to transmit on a single bus, but it is possible to utilize a redundant interface to report the PHSM of the interface currently in Bus Off. In that case, the RCI of the PHSM will reflect the redundant bus number associated with the node with the BUS OFF condition rather than the redundant bus number of the transmitting node.

5.4.2.1 Value of the TEC and the REC

All CAN controllers have two internal counters that are updated upon detected errors or upon correct transmissions or receptions. These two counters are named Transmit Error Counter (TEC) and Receive Error Counter (REC). These counters provide an indication of the health status of the CAN interface. These counters may be used to determine the value of the TEC/REC states communicated in the PHSM.

5.4.2.2 CAN Bus-Off Event Counter (NB_BUS_OFF)

The number of Bus-off events is not managed by the CAN controller. The application layer must maintain a count of the number of transitions from error passive state to bus off state. Moreover, according to the system requirements, the

system integrator shall define the number of allowed bus re-attachments for a CAN node, refer also to Section 4.5.3.1. This information may be used to determine if a single node has a fault or the entire bus is corrupted, because multiple nodes indicate a high number of faults.

Note: Some CAN controllers offer an auto bus_on feature which does not always require a reset of the controller.

5.4.2.3 Network Quality Performance Indicator

TEC and REC are not sufficient to evaluate the quality of a CAN bus. Indeed, both are permanently updated – they may be incremented and decremented. A global number of errors detected during a given period (NB_ERR_TX and NB_ERR_RX) are necessary to gain an indicator that is equivalent to a long term network quality status. This statistical value may be used to reflect a long-term health status of a single node. If all the nodes indicate a high number of faults, then a single faulty node will not be identified.

For safety reasons, it may be necessary to decide when the CAN bus must be considered as faulty (temporarily or permanently).

Consequently, internal counters shall be defined in the application layer that are incremented each time an error occurs. Internal Counters shall be reinitialized to zero each time the host processor is initialized.

5.5 Node Service Interface

ARINC 825 introduces a Node Service concept for targeted communication services between CAN nodes. It is a mechanism for peer-to-peer communication on LCC 4 Node Service Channel (NSC), although the service may be implemented on LCC 3 Directed Message Channel (DMC). The PHSM is a special case of Node Service with a fully defined data frame that is transmitted on LCC 6. The following sections describe these services as they would be implemented on the Node Services Channel (LCC 4).

5.5.1 Node Service Concept

Node services may run in parallel to other data transfers, thereby providing an independent network layer for peer-to-peer communication. This independent network layer implements command/response type connections between two nodes for specific operations, i.e., data load or other client/server actions. Node service requests requiring action but no response are possible as well. These “connectionless” requests may be sent to a specific node (unicast) or to all nodes (broadcast), while connection-oriented requests require a unique destination. ARINC 825, in general, allows node services to be initiated by any node in the network. However, system designers may choose to restrict the use of node services according to their requirements.

A node service is initiated by a node service request message, transmitted on the corresponding communication channel (i.e., DMC, NSC, and TMC). Each node attached to the network is obliged to continuously monitor these channels and check if a received message identifier’s NID sub-field (consisting of Server FID and Server ID) matches its personal node-ID. If a match is detected, and the node is capable of supporting the requested service, the addressed node has to react by performing the required action and transmitting a node service response message within a system specified time (if it was a connection-oriented request). The Service

Function Code (SFC) specifying the requested action is coded into the first two bytes of the message payload. A node service request message therefore has a minimum byte count of two. Use of the remaining payload data bytes 2-7 for legacy Classical CAN or data bytes 2-63 for CAN FD messages is optional. The content of the remaining payload data bytes is entirely user-defined and may be used to transfer any data related to the service. The SFC is identical for Classical CAN and CAN FD.

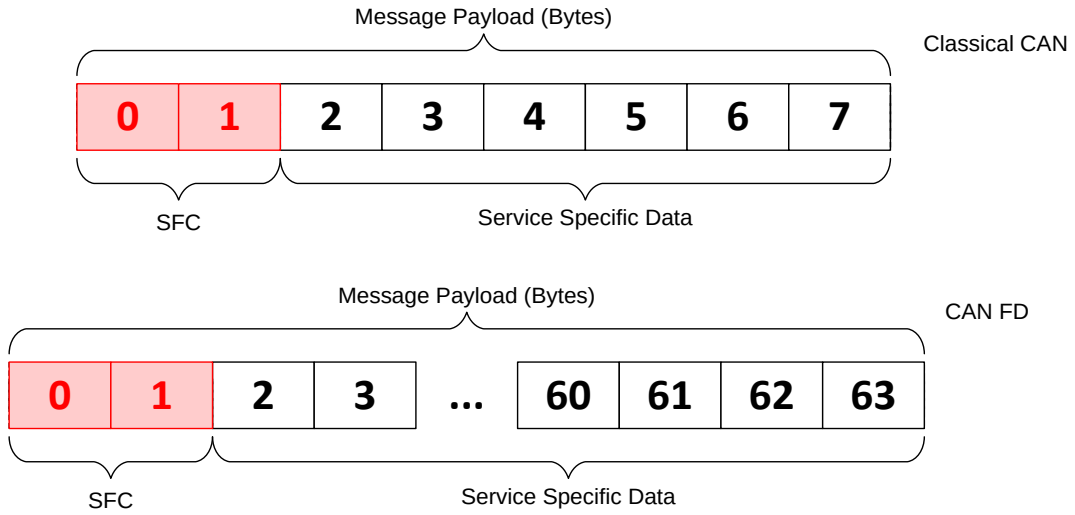


Figure 5-11 – Node Service General Message Payload Structure

For connection-oriented Node Services, a timeout should be implemented on both the client and server side to prevent deadlocking on a broken connection. For connectionless Node Services comprised of more than one message, a timeout should be implemented on the server side. The measuring of this time should be conducted on a message by message basis rather than the entirety of the transfer.

5.5.2 Node Service Codes

Node services and node service code regions are defined within Table 5-8. SFC 49152-65535 may be used for user-defined service implementations:

Table 5-8 –Service Function Codes

Node Service	Service Function Code	Response required	Action
Node Identification Service (IDS)	0	Yes	Requests configuration information from the addressed node.
Node Synchronization Service (NSS)	1	No	Performs network-wide time synchronization for all addressed nodes.
Data Upload Service (DUS)	2	Yes	Sends a block of data to another node.
Data Download Service (DDS)	3	Yes	Requests a block of data from another node.
BIT Control Service (BCS)	4	Yes	Triggers internal built-in test functions of the addressed node.

Node Service	Service Function Code	Response required	Action
Non-Volatile Storage Service (NVS)	5	Yes	Programs settings made through other services into internal non-volatile memory.
Node-ID Setting Service (NIS)	6	Yes	Sets the Node-ID of the addressed node.
Service Control Service (SCS)	7	Yes	Allows enabling/disabling of other Node Services.
Reserved	8 – 1023		Reserved for future use.
ARINC 812A Node Services	1024 – 2047		Reserved for ARINC 812A.
ARINC 826 Node Services	2048 – 2304		Reserved for ARINC 826.
Reserved	2305 – 49151		Reserved for future use.
User-defined	49152 – 65535		User-defined services.

5.5.2.1 Identification Service

The Identification Service is a connection-oriented service. The service may be used to obtain a “sign-of life” indication from the addressed node and check if its profile is compliant with the network that it is attached to. The addressed node returns its Profile ID and its LRU code. The Profile ID is a string in the format ID.Sub-ID. ID shall be the number to the left of the decimal point of the Profile ID attribute of the communication profile database ARINC 825 element as described in Attachment 1. The Profile Sub-ID shall be the number to the right of the decimal point of the Profile ID attribute of the communication profile database ARINC 825 element as described in ATTACHMENT 1. The LRU code is a 16-bit value that corresponds to the name of the LRU, which is unique to the CAN system or network.

Table 5-9 – Node Identification Service Format

Message Payload Bytes	Node Service Request	Node Service Response
0-1 (SFC)	Identification Service Function Code 0	Identification Service Function Code 0
2-3	unused	ID of Profile ID (USHORT)
4-5	unused	Sub-ID of Profile ID (USHORT)
6-7	unused	LRU Code (OPAQUE)

All ARINC 825 compatible nodes shall support the Identification Service on the Node Service Channel. This guarantees that any ARINC 825 network may be scanned for attached nodes to check their presence and determine their Profile-ID and unique LRU code. The Identification is statically defined in the communication profile.

5.5.2.2 Node Synchronization Service

The Node Synchronization Service may be implemented as a connectionless or connection-oriented service. It is used to perform limited precision time synchronization between some (or all) nodes attached to the CAN network.

Table 5-10 – Node Synchronization Service Format

Message Payload Bytes	Node Service Request	Node Service Response
0-1 (SFC)	Node Synchronization Service Function Code 1	Node Synchronization Service Function Code 1
2-7	unused	ACK

5.5.2.3 Data Upload Service

For the node service defined here, the target of the data upload service is referred to as the "server," while the node initiating the request for service is the "client." This service provides a mechanism for a network user to send more data than will fit in a single CAN message. This service allows data transfers to a node on the network from an external source node as well as between nodes within the system, so both source and target may be airborne or one may be a piece of ground based equipment. In either case, the transfer of data shall be called "uploading" when the sender of the data initiates the transfer.

COMMENTARY

The Upload and Download Services in ARINC Specification 825 are for transferring blocks of data that exceed CAN maximum message size. For loadable software aircraft parts, see **ARINC Report 665:** *Loadable Software Standards*, **ARINC Report 615A:** *Software Data Loader Using Ethernet Interface,* and **ARINC Specification 826:** *Software Data Loader Using CAN Interface.* These software data loading standards are designed to transfer data, referred to as Loadable Software Aircraft Parts, to Aircraft Target Hardware.

The data upload is a connection oriented service and is used to send data from one node to another. The maximum number of bytes in an upload may be 2^{32}. The upload process depicted in Figure 5-12 is defined in two phases, Upload Setup and Data Transfer. During the first phase, Upload Setup, the client and server exchange information on the data transfer size and timeouts to be used, these messages are referred to as Upload Control Messages. The second phase is where the client sends the data to the server. As shown in Figure 5-12, the data may be transmitted in blocks, where each block is composed of multiple CAN messages. The data being transferred shall include a 32-bit checksum as the last four bytes of the data for the block transfer. The CAN message ID shall be as defined in Section 5.2.2.3, Peer-to-Peer Identifier Structure, for peer-to-peer communications. In the CAN message ID, the client sets to a one (1) the Service Message Type (SMT) bit for all messages it sends and the server shall set the SMT bit to zero (0) for all of the messages it sends. This avoids the potential problem when both client and server are assigned the same FID value. By using the SMT bit, the unacceptable scenario where two units on the bus could be sending messages with the same CAN ID is avoided. The "Private" (PVT) bit in the CAN message ID is set to zero (0) for all service control messages (non-data transfer messages) and set to one (1) for all data transfer messages.

COMMENTARY

Common with all node services the Data Upload service may be peer-to-peer (unicast) or one to many (multicast) by using the

multicast SID and/or FID defined in Section 5.2.2 (ARINC 825 CAN Identifier Structure). Servers always respond with their own Node-ID, consisting of Server FID and Server ID. The data block size for CAN FD may be up to 64 bytes.

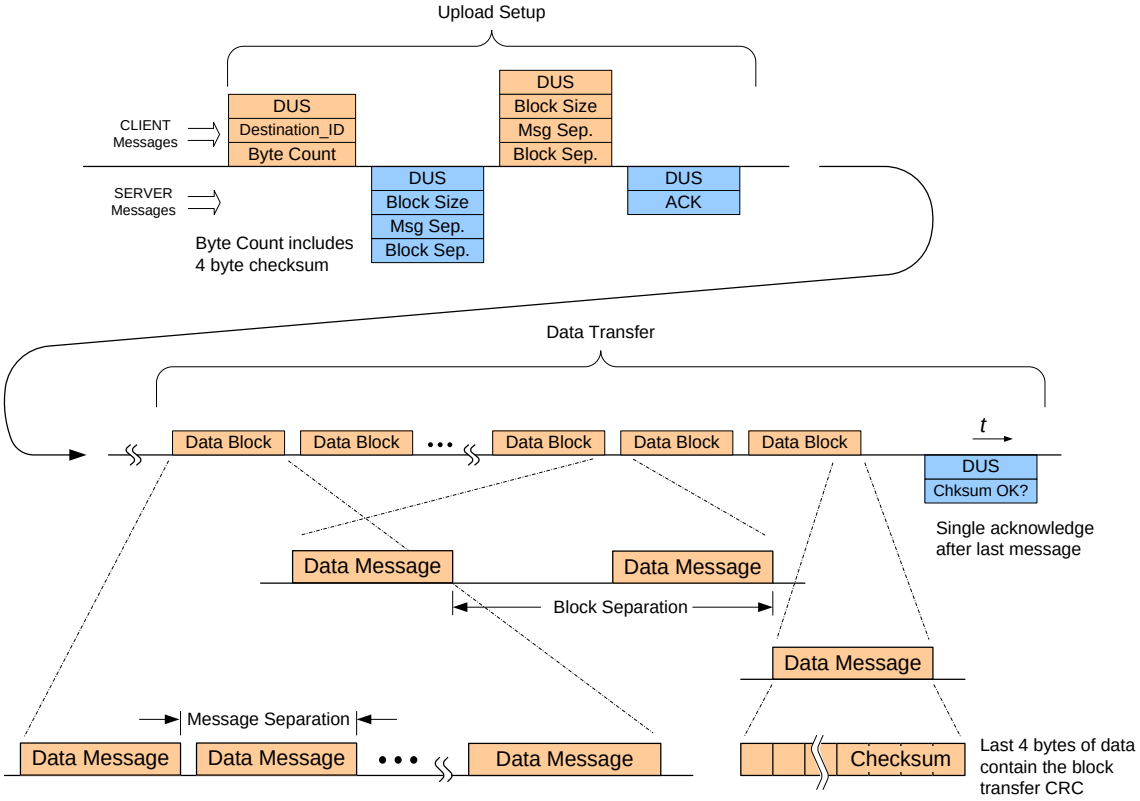


Figure 5-12 – Data Upload Message Sequence

Data Upload Setup

As shown in Figure 5-12, to initiate the service, the client (sending node) transmits the “Data Upload Function Service Code” request to the addressed node, using the server’s (target’s) NID in the message ID. The message data contains the DUS function code (0x0002), a Destination Identifier (Destination_ID), and the Byte Count for the data that will be uploaded. The Destination Identities 0-511 are predefined or reserved, see Table 5-11. The Byte Count includes the four bytes for the data transfer checksum. If the Byte Count is zero (0) the number of bytes in the transfer shall be 2^{32}. The client then waits for the server to respond, see Table 5-12 for server responses.

Table 5-11 – Source and Destination Identifiers

Identity	Description
0	**Default**
1	**Unit Fault Log**
2	**Unit Identification**
3	**Unit Configuration**
4	**Basic Software/Bootstrap**
5	**Operating System**
6	**Application Program**
7-15	**SDL via ARINC 826**
16-255	**Reserved**
256-511	**Must Never Be Used**
512-65535	**User-defined**

Table 5-12 – Data Upload: Initial Message

Message Bytes	Node Service Request
0-1	Data Upload Service Function Code 2
2-3	Destination_ID (USHORT) - see note
4-7	Byte Count (ULONG): Number of bytes to be transferred including 4-byte checksum

Message Bytes	Node Service Response
0-1	Data Upload Service Function Code 2
2	Response Code (CHAR): -3 = OUT OF SPACE -2 = INVALID_DESTINATION -1 = ABORT 1 = ACK
3	Minimum Message Separation time in mSec. (UCHAR)
4-5	Block Size as number of bytes (USHORT)
6-7	Minimum Block Separation Time in mSec. (USHORT)

Note: Destination_IDs 0-511 are predefined or reserved.

The server must respond with the data defined in Table 5-11. If the Response Code (data byte 2) is not an ACK, the upload shall be terminated. If the Minimum Message Separation Time is zero (0), the server does not require any additional separation between messages. The block size shall be in multiples of 8 bytes, with 16 being the lowest allowed value. If the block size is zero (0), then the server does

not require a specific block size (that is, the data may be sent as a single block). If the Minimum Block Separation Time is zero (0), the server does not require any additional time between blocks of data.

As shown in Table 5-11, both the Minimum Message Separation Time and the Maximum Block Separation Time are measured from the end of the previous message to the end of the next message. All timeouts used in the Upload service are measured the same way, from the end of the previous message to the end of the next message.

In the second Upload Control Message, the client shall set the Minimum Message Separation Time to a value that is not less than the value received from the server. The Block Size shall be set to the value received from the server unless it is zero. If the server responded with a Block Size of zero, the client may either send the data as a continuous block or select a Block Size value. If the client is going to send the data as a single block, it shall set the Block Size to zero (0) in the second Upload Control Message. The Maximum Block Separation Time shall not be set to a value less than the Minimum Block Separation Time sent by the server. If the server Minimum Block Separation Time response was zero (0), the client must select a value to be used for timeouts during the block transfer. It is the client's responsibility to set the Minimum Message Separation Time and Maximum Block Separation Time consistent with bandwidth allocated for the data transfer. The Maximum Block Separation Time must be greater than the Minimum Message Separation Time. The client then sends the second Upload Control Message with the Minimum Message Separation, selected Block Size, and Maximum Block Separation Time that will be used during the transfer and waits for the server to respond. The server's response shall be as shown in Table 5-12.

Table 5-13 – Data Upload Service: 2nd Message

Message Bytes	Node Service Request
0-1	Data Upload Service Function Code 2
2	1 (constant)
3	Minimum Message Separation Time in mSec. (UCHAR)
4-5	Block Size as number of bytes (USHORT)
6-7	Maximum Block Separation Time in mSec. (USHORT)

Message Bytes	Node Service Response
0-1	Data Upload Service Function Code 2
2	Response Code (CHAR): -4 = INVALID_DATA -3 = OUT_OF_SPACE -2 = INVALID_DESTINATION -1 = ABORT 1 = ACK

Table 5-13 lists five possible responses to the 2nd Upload Setup message. The server should respond with ACK if the separation times and the block size are acceptable to proceed with the data transfer. The server should respond with INVALID_DATA if the 2nd Upload Setup message contains unacceptable values to proceed with the data transfer. If the server is out of space, the OUT_OF_SPACE response should be used or if the Destination_ID is not recognized then the INVALID_DESTINATION should be used. If there are any other conditions that will prohibit the data transfer the ABORT response should be used. If any response is received other than ACK, the Upload service shall be stopped and the nodes should revert to normal operation.

Upload Data Transfer

When the client receives an ACK from the server in response to the second Upload Control Message, it shall begin sending the sequence of messages that contain the data to be uploaded. While sending the data the client may monitor for messages from the server that terminate the Upload Service prior to completion, see responses in Table 5-12, where any of the negative responses may terminate the Upload service. During the data transfer phase, the only required server response is after the last data message from the client. The system integrator shall define the maximum time (Time_Out) the client or server must wait for the next message or response before aborting the upload sequence (for example, 4 times the Maximum Block Separation Time). If there is a separation time (gap) between messages from the client greater than system integrator defined Time_out during the data transfer phase, the server shall stop accepting the data and return to normal operating mode.

After each data block is sent, the client waits at least Maximum Block Separation Time and then continues with the next block of data. Data blocks are all the same size except the last block which might be smaller (if the amount of data is not a multiple of block size used). The last data message(s) shall contain the data transfer checksum. After the last data message, the server shall respond with one of the responses also shown in Table 5-13 within the system integrator defined Time_out, unless it has already terminated the transfer, in which case it need not respond. If the server does not respond in less than the system integrator defined Time_out, the client may assume the transfer failed. This completes a data upload.

Table 5-14 – Data Upload: Acknowledge

Message Bytes	Node Service Response
0-1	Data Upload Service Function Code 2
2	Response Code (CHAR): -1 = Checksum Error 1 = Checksum Okay

Cyclic Redundancy Check Generation

The checksum, in the last message(s) of the data transfer, includes all of the data that was part of the upload. Depending on the size of the data transfer the checksum may be split between the last two CAN messages. The Upload and Download Service functions use the same 32-bit CRC for the checksum. If the data has been received by the server (receiver) without error, the result of CRC computation on the data and checksum value will produce a zero result.

For the data upload and download node service functions, the checksum is a CRC-32 defined by the polynomial: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$. This polynomial is sometimes represented by a 32-bit hexadecimal value, 0x04C11DB7. The CRC-32 generator/checker shall use 0xFFFFFFFF for the initial CRC value. This is the same CRC polynomial defined for IEEE 802.3 (Ethernet) and the calculation method should result in equivalent results. The generator/checker shall reflect the input bytes about their center and reflect the final result about its center before exclusive-or'ing (XOR) the final value with 0xFFFFFFFF. A generator/checker that evaluates the following character string, '123456789' to 0xCBF43926 calculates the CRC correctly.

5.5.2.4 Data Download Service

For the node service defined here, the source of a data for the download service is referred to as the “server,” while the node initiating the request for the service is the “client.” This service provides a mechanism for a network user to request more data than will fit in a single CAN message. This service allows transfers from a node on the network to an external node as well as between nodes within the system, so both computers may be airborne or one may be ground based. In either case, requesting data from another node shall be called “downloading” when the receiver initiates the transfer.

COMMENTARY

The Upload and Download Services in ARINC Specification 825 are for transferring blocks of data that exceed CAN maximum message size. For loadable software aircraft parts, see **ARINC Report 665:** *Loadable Software Standards*, **ARINC Report 615A:** *Software Data Loader Using Ethernet Interface*, and **ARINC Specification 826:** *Software Data Loader Using CAN Interface.* These software data loading standards are designed to transfer data, referred to as Loadable Software Aircraft Parts, to Aircraft Target Hardware.

The data download is a connection oriented service and is used to request data from one node by another. The maximum number of bytes in download may be 2^{32}. As shown in Figure 5-13, the download process has two phases, Download Setup and Data Transfer. During the first phase, Download Setup, the client and server exchange information on the data transfer size and timeouts to be used, these messages are referred to as Download Control Messages. The second phase is when the server sends the data to the client. As shown in Figure 5-13, the data may be transmitted in blocks, where each block is composed of multiple CAN messages. The data being transferred shall include a 32-bit checksum as the last four bytes of the data. The CAN message ID shall be as defined in Section 5.2.2 (ARINC 825 CAN Identifier Structure) for peer-to-peer communications. In the CAN message ID, the client sets to a one (1) the Service Message Type (SMT) bit for all messages it sends and the server sets the SMT bit to zero (0) for all of the messages it sends. This avoids the potential problem when both client and server are assigned the same FID value. By using the SMT bit, the unacceptable scenario where two units on the bus could be sending messages with the same CAN ID is avoided. The “Private” (PVT) bit in the CAN message ID is set to zero (0) for all service control messages (non-data transfer messages) and set to one (1) for all data transfer messages.

COMMENTARY

Common with all node services the Data Upload service may be peer-to-peer (unicast) or one to many (multicast) by using the multicast SID and or FID defined in Section 5.2.2 (ARINC 825 CAN Identifier Structure). Servers always respond with their own Node-ID, consisting of Server FID and Server ID. The data block size for CAN FD may be up to 64 bytes.

5.0 CAN COMMUNICATION

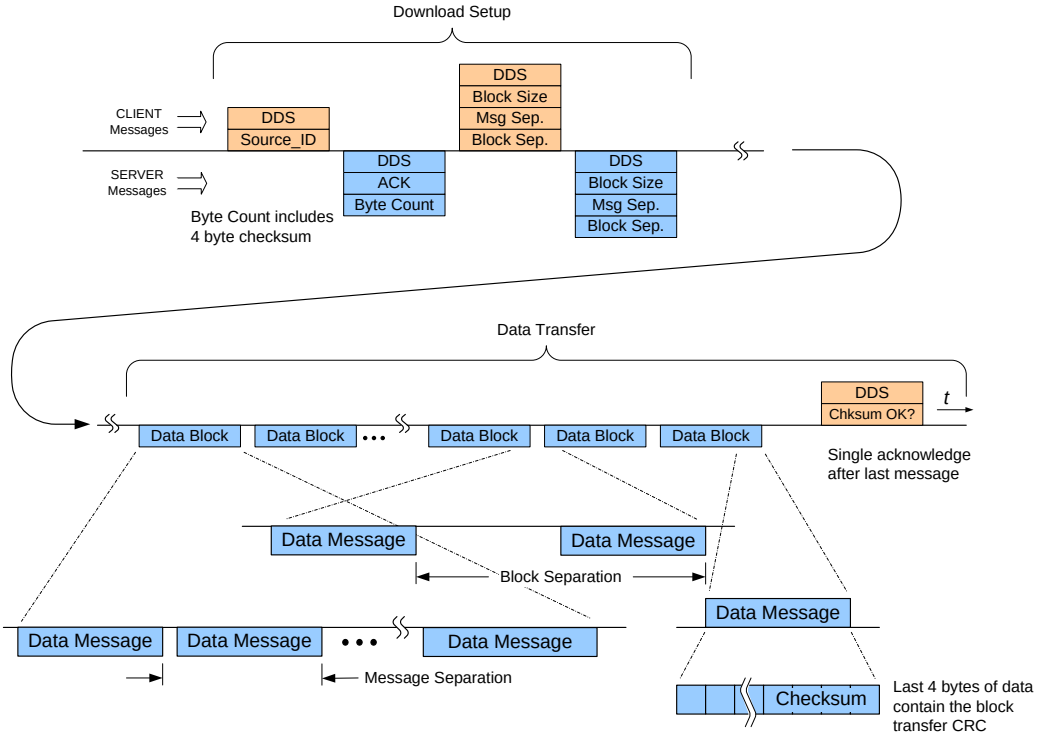


Figure 5-13 – Data Download Message Sequence

Data Download Setup

As shown in Figure 5-13, to initiate the service, the client (requesting node) transmits the “Data Download Function Service Code” request to the addressed node, using the server’s (data source’s) NID in the message ID, the message data contains the DDS function code (0x0003) and a Source Identifier (Source_ID). The Source Identities 0-511 are predefined or reserved (see Table 5-9). The client then waits for the server to respond with one of the responses shown in Table 5-14. The server responds with a message containing the response code and the Byte count including the four Byte data transfer checksum. If the Byte Count is zero (0), the number of bytes in the transfer shall be 2^{32}. If the server response code is not ACK, the download shall be terminated.

Table 5-15 – Data Download: Initial Message

Message Bytes	Node Service Request
0-1	Data Download Service Function Code 3
2-3	Source_ID (USHORT) [1]
4-7	Source_ID_Extension [2]

Message Bytes	Node Service Response
0-1	Data Download Service Function Code 3
2	Response Code (CHAR): -3 = INVALID_SOURCE_EXT -2 = INVALID_SOURCE -1 = ABORT 1 = ACK
3	1 (constant)
4-7	Byte Count (ULONG): Number of bytes to be transferred including the checksum

Notes:

1. Source_IDs 0-511 are predefined or reserved.
2. Source_ID_Extension is optional. Size and layout may vary but should be bound to the Source_ID

Table 5-14 lists possible responses to the 1st Download Setup message. The server shall respond with ACK if the Source_ID is acceptable to proceed with the data transfer. Likewise, the ACK will include the optional Source_ID_Extension, if used. The server shall respond with INVALID_SOURCE if the Source ID in the 1st Download Setup message contains an unacceptable value to proceed with the data transfer. If there are any other conditions that will prohibit the data transfer, the ABORT response should be used. If any response is received other than ACK, the Download service shall be stopped and the nodes should revert to normal operation. When the server responds with an ACK it shall also include the constant, 1, and the byte count for the data including the required 4 bytes for the checksum.

The client shall send the 2nd Download Control Messages shown in Table 5-16. If the Minimum Message Separation Time is zero (0), the client does not require any additional separation between messages. The block size shall be in multiples of 8 bytes, with 16 being the lowest allowed value. If the Block Size is zero (0), then the client does not require a specific Block Size. If the Minimum Block Separation Time is zero (0), the client does not require any additional time between blocks.

As shown in Figure 5-13, both the Minimum Message Separation Time and the Maximum Block Separation Time are measured from the end of the previous message to the end of the next message. All timeouts used in the Download service are measure the same way, from the end of the previous message to the end of the next message.

Table 5-16 – Data Download Service: 2nd Message

Message Bytes	Node Service Request
0-1	Data Download Service Function Code 3
2	1 (constant)
3	Minimum Message Separation Time in mSec. (UCHAR)
4-5	Block Size as number of bytes (USHORT)
6-7	Minimum Block Separation Time in mSec. (USHORT)

Message Bytes	Node Service Response
0-1	Data Download Service Function Code 3
2	Response Code (CHAR): -4 = INVALID_DATA
	-1 = ABORT 1 = ACK
3	Minimum Message Separation Time in mSec. (UCHAR)
4-5	Block Size as number of bytes (UCHAR)
6-7	Maximum Block Separation Time in mSec. (USHORT)

In the response to the 2nd Download Control Message, the server shall set the Response Code to ACK if the server will proceed with the data transfer. If the 2nd Download Control Message contains incorrect data, the server should respond with INVAILD_DATA. If any other condition will not let the download proceed, the server should set the Response Code to ABORT. If the Response Code is ACK, the server sets the Block Size to the value received from the client unless it is zero. If the client sent a Block Size of zero, the server may either send the data as a continuous block or select a Block Size value. If the server is going to send the data as a single block, it shall set the Block Size to zero (0) in response to the second Download Control Message. The Minimum Message Separation Time shall be set to a value that is not less than the value received from the client. The Maximum Block Separation Time shall not be less than the Minimum Block Separation Time received from the client. If the client's Minimum Block Separation Time was zero (0), the server must select a value to be used to timeout the transfer in the event the sequence is interrupted. It is the server's responsibility to set the Minimum Message Separation Time and the Maximum Block Separation Time consistent with bandwidth allocated for the data transfer. The Maximum Block Separation Time must be greater than the Minimum Message Separation Time. The server sends its response to the second Download Control Message with the Response Code, Minimum Message Separation Time, Block Size, and Maximum Block Separation Time that will be used during the data transfer phase.

Download Data Transfer

The server shall wait at least one Maximum Block Separation Time before beginning to send the data to be downloaded. The server shall insert an additional separation time between messages as defined in the 2nd data Download Control Message. While sending the data the server may monitor for messages from the client that terminate the Download service prior to completion, see responses in Table 5-16, where any of the negative responses may terminate the Download service. During the data transfer phase, the only required response from the client is after the last message. The system integrator shall define the maximum time (Time_Out) the client or server must wait for the next message or response before aborting the download sequence (for example, 4 times the Maximum Block Separation Time). If there is a separation time (gap) between messages from the server greater than

system integrator defined Time_out, during the data transfer phase, the client shall stop accepting the data and return to normal operating mode.

After each block is sent, the server waits at least the Maximum Block Separation Time and then continues with the next block of data. Data blocks are all the same size except the last block which might be smaller (if the data block is not a multiple of block size used). The last data message(s) shall contain the data transfer checksum. After the last data message the client shall respond with one of the responses also shown in Table 5-17 within at least the system integrator defined Time_out, unless it has already terminated the transfer, in which case it need not respond. If the client does not respond within the system integrator defined Time_out, the server may assume the transfer failed. This completes a data download.

Table 5-17 – Data Download: Acknowledge

Message Bytes	Node Service Response
0-1	Data Download Service Function Code 3
2	Response Code (CHAR): -1 = Checksum Error 1 = CRC Okay

The checksum used in the last message(s) is a 32-bit CRC, includes all of the data that was part of the download. The CRC used for the data download service is the same as defined for the Data Upload Service; please refer to Section 5.5.2.3, Cyclic Redundancy Check Generation.

5.5.2.5 BIT Control Service

The BIT control is a connection-oriented service and is used to control built-in test functions of the addressed node. Such functions are usually implemented to support in-service testing of the equipment and vary widely between systems. Consequently, the service control data itself is user-defined.

Table 5-18 – BIT Control Service Format

Message Payload Bytes	Node Service Request	Node Service Response
0-1 (SFC)	BIT Control Service Function Code 4	As in request
2-7	User-defined	User-defined

5.5.2.6 Non-Volatile Storage Service

The Non-Volatile Storage Service is a connection-oriented service used to store settings made through other node services into internal Non-Volatile Memory.

Table 5-19 – Non-Volatile Storage Service Format

Message Payload Bytes	Node Service Request	Node Service Response
0-1 (SFC)	Non-Volatile Storage Service Function Code 5	As in request
2-3	Password (USHORT)	-1 = INVALID PASSWD 0 = DATA STORED

5.5.2.7 Node-ID Setting Service

The node-ID setting service is a connection-oriented service used to set the personal ID of the addressed node. The new ID becomes effective immediately after the service response has been transmitted and is used by the addressed node to identify itself in the NID field for node service requests.

Table 5-20 – Node-ID Setting Service Format

Message Payload Bytes	Node Service Request	Node Service Response
0-1 (SFC)	Node-ID Setting Service Function Code 6	As in request
2-3	New Node-ID (USHORT)	-1 = INVALID NODE-ID 0 = NODE-ID CHANGED

This message must be addressed to a specific node, which by definition has an undefined Node ID. The message may be sent using the broadcast command (with FID=0 and SID=0). To prevent all nodes on the bus acting on the message, a mechanism needs to be in place to isolate the desired node.

5.5.2.8 Service Control Service

The Service Control Service (SCS) allows one to individually enable or disable specific Node Services for the addressed node. The idea behind this service is that important functions should require a two-step approach to be accessible. Node Services selected for the two-step approach should neither respond nor perform the requested action when called unless they have been previously enabled through the SCS.

Table 5-21 – Service Control Service Format

Message Payload Bytes	Node Service Request	Node Service Response
0-1 (SFC)	Service Control Service Function Code 7	As in request
2-3	Target Service Code (code of the function to be controlled)	As in request
4	Control Code (user-defined)	Response Code: 0 = OK: -1 = FUNCTION CODE ERROR -2 = CONTROL CODE ERROR -3 = SCS NOT SUPPORTED

The response codes of the SCS have the following meaning:

- OK: The request has been performed.
- FUNCTION CODE ERROR: The Target Service Function Code is wrong.
- CONTROL CODE ERROR: The Control Code is wrong.

- SCS NOT SUPPORTED: The server node does not support the SCS (SFC=7).

The Function Code specifies the service to be controlled. The Control Code is user-defined and specifies the type of control to be performed. This may include enabling/disabling services fully or partially.

Functions being controlled through SCS should be locked by default when the node is powered up. Additionally, a timeout should be implemented to revert to locked mode if a service has not been called for a given amount of time after it has been unlocked.

5.5.3 Test and Maintenance Support

Basically, node services may be defined by the system designer. To support test and maintenance, however, a basic service to interrogate all nodes in a particular network is useful. The advantage of such a service is that test systems plugged into the network (without having any knowledge about the attached nodes) may quickly obtain valuable information supporting follow-on test and maintenance procedures.

All ARINC 825 compatible nodes shall support the “Identification Service” on the Test and Maintenance Channel (TMC). This guarantees that any ARINC 825 network may be scanned for attached nodes to check their presence and determine their Profile-ID, Profile Sub-ID and unique LRU code.

Test systems may interrogate individual nodes by a specific NID, all nodes within a FID by using the multicast SID number 0, or all nodes within all functions by using the multicast SID number 0 and the multicast FID number 0.

5.6 Bandwidth Management

An important issue concerning CAN networks is message prioritization and bandwidth management. Message prioritization steps in as soon as multiple nodes try to access the network at the same time for transmission. CAN uses CSMA/CA arbitration (Carrier Sense Multiple Access/Collision Avoidance). This means that whenever multiple nodes try to access the network at the same time, the node that tries to transmit the message with the lowest numeric identifier (equaling highest priority) will be granted access for transmission. The others have to withdraw and perform a retry at the beginning of the next transmission window (start of frame). The advantage of this concept is that it does not sacrifice bandwidth for arbitration. The system designer is still required to perform bandwidth management, as the number of nodes (and messages) increases, higher priority messages may block out other messages causing excessive latency. Bandwidth management is used to make sure that the bus loading on the network is – within certain limits – evenly balanced over time and some growth potential exists. This is achieved by controlling the transmission rate of each node in the network so that no single message is delayed beyond acceptable limits.

The bandwidth management concept presented in this section describes one acceptable method. This method is based on a minor/major time frame concept where all the nodes share the same minor/major time frame period, and the minor time frames are even fractions of a major time frame (i.e., 1/2, 1/4, 1/8, and so forth).

Major time frames are defined as the time period for all the periodic messages to be transmitted at least once, and minor time frames are defined as the time period required for the most frequent messages to be transmitted once.

The method defined in this section provides an effective means to balance the message loading on the bus. If the bus loading is not balanced via scheduling, greater message latency may occur.

5.6.1 Bus Load Calculation

To determine the percentage of bus loading for an identified time period, it is necessary to take into account the possibility to have a different data bit-rate between the arbitration phase and data phase (see frame representation).

Two cases are taken into account:

1. The same bite-rate, for example message sent at 1 Mbps.
2. The different bit-rate, for example Arbitration phase at 1 Mbps and the Data phase send at 4 Mbps.

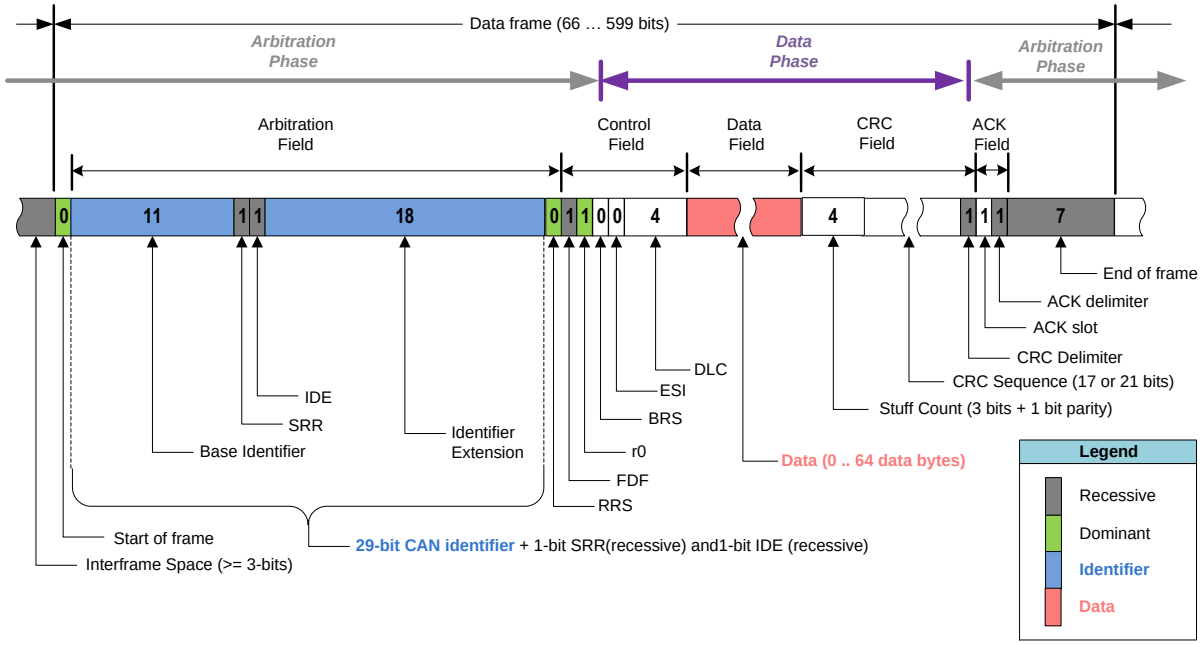


Figure 5-14 – CAN FD Data Frame

5.6.1.1 Bus Load Calculation for a Same Bit Rate

If the bit-rate of the Arbitration phase and Data phase is the same, to determine the percentage of bus loading for an identified time period, the following equation is presented:

$$PercentAverageBusLoad = \left\{ \frac{\Sigma(MessageLength \; X \; NumberOfMessages)}{TransmissionInterval \; X \; DataRate} \right\} X \; 100$$

Where:

- *MessageLength* is the length in bits of each message sent within the defined *TransmissionInterval* including stuff bits, see Table 5-22.
- *NumberOfMessages* is the maximum occurrence of messages of the selected length in the identified *TransmissionInterval* (i.e., how many times does this message appear in the *TransmissionInterval*).
- *TransmissionInterval* is the duration of the selected time interval in ms (e.g., minor time frame duration).
- *DataRate* is the bus speed in bits/second**.**

CAN FD allows for different data rates for the arbitration phase and the data phase. Both of these rates need to be accounted for in the bus load calculation. Additionally, the change to the increased data rate for the data phase of the message is controlled by setting the Bit Rate Switch (BRS) to recessive. Since the BRS can be turned on or off on the fly, the bus load calculation should take into consideration the worst case where the BRS is not set recessive and the data phase rate is equivalent to the arbitration phase rate.

The Bandwidth Usage equation only addresses periodic data usage. Aperiodic data must also be included by converting it to its equivalent periodic rate by using the data's minimum and maximum occurrence rates.

To properly analyze a CAN bus bandwidth utilization, all the data transmitted by the nodes on the bus (whether there are any receivers of the data or not) must be included in the analysis. It is also important to choose the *TransmissionInterval* wisely as variations in the calculation of average bus load are possible depending upon the chosen interval.

Table 5-22 provides a list of data frame sizes in a message with an identified maximum message length including stuff bits. The following equations provide the values in Table 5-22.

$$MaxNrOfStuff[bits] = \frac{54 + Payload - 1}{4}$$

$$MessageLength[bits] = Overhead[bits] + Payload[bits] + MaxNrOfStuff[bits] + IFS[bits]$$

The values in Table 5-22 shall be considered for the bus load calculation.

Table 5-22 – Values for Bus Load Calculation

DLC	Data Field (Bytes)	*MessageLength* Maximum frame size (bits)
0	0	91
1	1	101
2	2	111
3	3	121
4	4	131
5	5	141
6	6	151
7	7	161
8	8	171
9	12	211
10	16	251
11	20	296
12	24	336
13	32	416
14	48	576
15	64	736

The below example illustrates the concept utilizing a Minor Time Frame as the *TransmissionInterval*.

As we see the message identified as “C” in the figure occurs in Minor Time Frame #1, but not in Minor Time Frame #2. Therefore, determining the percentage bandwidth loading would provide a more realistic view utilizing Minor Time Frame #1 as the *TransmissionInterval*.

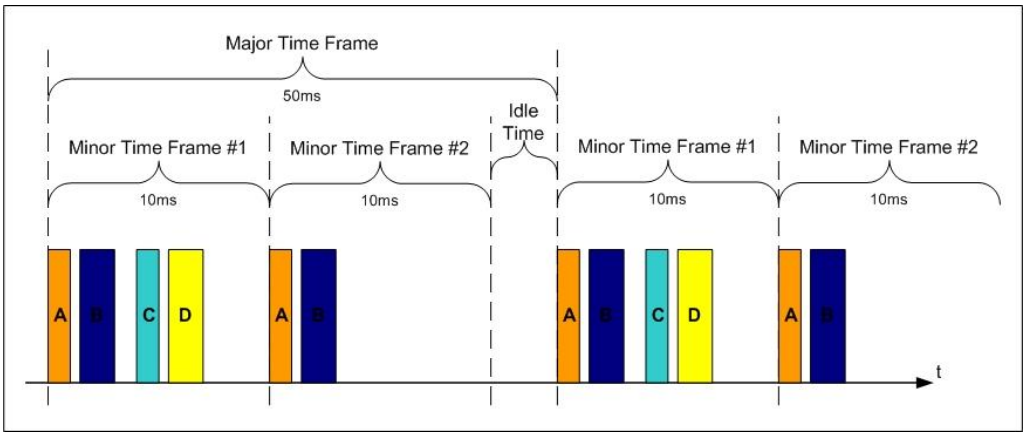


Figure 5-15 – Detailed View of Messages within Major and Minor Time Frames

In this specific example case,

- A is a 4-byte frame = 128 bits
- B is an 8-byte frame = 168 bits
- C is a 3-byte frame = 118 bits

- D is an 8-byte frame = 168 bits (same as B)

Using the equation provided above with a *DataRate* (bus speed) of 125 kbit/s (125,000bit/s):

$$\left\{\frac{\left[(A\times 1)+(B\times 2)+(C\times 1)\right]}{(MinorTimeFrame)\times(DataRate)}\right\}\times 100$$

$$\left\{\frac{\left[(119\,bits\times 1)+(158\,bits\times 2)+(109\,bits\times 1)\right]}{(0.01s)\times(125{,}000\,bits/s)}\right\}\times 100$$

$$\left\{\frac{534bits}{1250bits}\right\}\times 100 = 46.5\%$$

Therefore, the worst-case percentage bandwidth utilization is 46.5%.

If the bus load would have been calculated on the Major Time Frame, the value would have been far lower.

5.6.1.2 Bus Load Calculation for Different Bit Rates

If the bit-rate of the Arbitration phase and Data phase is different, the same equation is used to determine the percentage of bus loading for an identified time period, but a new formula is used to compute the *MessageLength*.

$$PercentAverageBusLoad = \left\{\frac{\Sigma(MessageLength\ X\ NumberOfMessages)}{TransmissionInterval\ X\ DataRate}\right\}\ X\ 100$$

Where:

$$MessageLength = MessageArbitrationLength\ \mathrm{X}\left(\frac{DataRate\ DATA}{DataRate\ Arbitration}\right) + MessageDataLength$$

- *MessageLength* is the sum of Arbitration phase plus Data phase including stuff bits.

  Note: To take into account the difference between the bit rate of arbitration phase and Data phase, it is necessary to multiply the arbitration phase length by the ratio Data rate/Arbitration rate.
  For the length of the Arbitration and Data, see Table 5-23.

- *NumberOfMessages* is the maximum occurrence of messages of the selected length in the identified *TransmissionInterval* (i.e., how many times does this message appear in the *TransmissionInterval*).
- *TransmissionInterval* is the duration of the selected time interval in ms (e.g., minor time frame duration).
- *DataRate* is the bus speed in bits/second of data phase.

Table 5-23 provides a list of data frame sizes in a message with an identified maximum message length including stuff bits. The following equations provide the values in Table 5-22.

Message Arbitration Phase:

The arbitration length is constant, the value 56 bits is the sum of 12 bit without stuffing and 35 bits with 9 bits stuffing.

The 9-bit stuffing is obtained by the formula:

$$MaxNrOfStuff[bits] = \frac{35}{4} + 1 * Rounded\_down$$

* This formula compensates the splitting of the stuffing bit on the 2 phases Arbitration/data and the rounded effect.

Message Data phase:

$$MaxNrOfStuff[bits] = \frac{6 + 8 * DataFieldByte - 1}{4}$$

$$MessageLength[bits] = Payload + MaxNrOfStuff + CRC\ Header\ (parity, variable\ CRC, specific\ stuffing\ bit)$$

The values in Table 5-23 shall be considered for the bus load calculation.

Table 5-23 – Values for Bus Load Calculation

DLC	Data Field Bytes	Arbitration Phase Length (bit)	Data Phase Length (bit)
0	0	56	35
1	1	56	45
2	2	56	55
3	3	56	65
4	4	56	75
5	5	56	85
6	6	56	95
7	7	56	105
8	8	56	115
9	12	56	155
10	16	56	195
11	20	56	240
12	24	56	280
13	32	56	360
14	48	56	520
15	64	56	680

In the specific example of Figure 5-15, the concept utilizing a Minor Time Frame with an Arbitration phase at 1Mbit/s, and a Data Phase at 4 Mbit/s.

In this specific example case,

- A is a 4-byte frame = 72 bits
- B is an 8-byte frame = 112 bits
- C is a 3-byte frame = 62 bits
- D is an 8-byte frame = 112 bits (same as B)

- A is a 4-byte frame = 56-bit Arbitration phase + 72-bit Data phase
  A= 56x4+72= 296
- B is an 8-byte frame = 56-bit Arbitration phase + 110-bit Data phase
  B= 56x4+110= 336
- C is a 3-byte frame = 56-bit Arbitration phase + 60-bit Data phase
  C= 56x4+60= 286
- D is an 8-byte frame = 56-bit Arbitration phase + 110-bit Data phase (same as B)
  D= 56x4+110= 324

Using the equation provided above with a *DataRate* (bus speed) of Mbit/s (4 000 000 bit/s):

$$\left\{\frac{\left[(A\times 1)+(B\times 2)+(C\times 1)\right]}{(MinorTimeFrame)\times(DataRate)}\right\}\times 100$$

$$\left\{\frac{\left[(296\,bits\times 1)+(336\,bits\times 2)+(286\,bits\times 1)\right]}{(0.01s)\times(4.000.000\,bits/s)}\right\}\times 100$$

$$\left\{\frac{1254\,bits}{40{,}000\,bits}\right\}\times 100 \;= 3.1\%$$

Therefore, the worst-case percentage bandwidth utilization is 3.1%.

5.6.2 Bandwidth Management Example

The bandwidth management example in this section (for Classical CAN) is applicable only if all the nodes on the bus utilize this common message schedule.

The following bus scheduling method provides a means of computing the bus load based on the number of messages in the network and adjusting their transmission times to minimize peak load scenarios. Note also that the described example does not include large aperiodic blocks of data which may be generated, i.e., through use of the Data Upload/Download Services. This data will require reasonable throttling of the message flow to avoid unacceptable bus load variations. Good practice is to link large data block transmissions to the minor time frame chosen for the respective application and include it in the bus load calculation described hereafter. Be aware that this may substantially increase **the duration of the block data transfer!**

The maximum length of a 29-bit identifier CAN data frame (including stuff bits) is 158 bits. In order to establish a universal guideline, however, the following example assumes an average of 19 stuff bits, resulting in a total frame length of 150 bits.

Table 5-24 – Example 29-Bit Identifier CAN Data Frame

29-Bit Identifier CAN Data Frame Field Name	Length in bits
Start of Frame (SOF)	1
Identifier Field A	11
Substitute Remote Request (SRR)	1
Identifier Extension (IDE)	1
Identifier Field B	18
Remote Transmission Request (RTR)	1
Reserved	2
Data Length Code (DLC)	4
Data Field (0-8 Bytes)	64
Cyclic Redundancy Check (CRC)	15
CRC Delimiter	1
Acknowledge Slot (ACK)	1
Acknowledge Delimiter	1
End of Frame (EOF)	7
Assumed number of Stuff Bits (above average)	19[1]
Minimum inter-frame space	3
Total	150

Note: [1]The minimum number of stuff bits is 0 (zero) and the maximum for an ARINC 825-compatible message is 29. In this example, 19 was chosen as it is roughly 2/3 of the maximum and it makes the math for this example simpler.

To demonstrate how a CAN bus schedule for a specific system is developed, it will be assumed that the data rate is 1 Mbps, resulting in a maximum transmission time of 150 µs per CAN frame. Additionally, the maximum transfer rate of any message on the bus is assumed to be 66.6 messages per second, resulting in a time frame of 15 ms. This means that 100 frames transmitted each 15 ms, or approximately 6666 frames per second, would consume 100% of the potential bus load.

In theory, all messages could be transmitted at 66 times per second but this would make no sense unless the rate of change of the associated signal dictates this and there is equipment installed in the network which may make use of data at this rate. A minor time frame is defined as the time interval for the highest rate message on the bus and a major time frame is the period to transmit all the (periodic) messages at least once. The bus scheduling concept uses a “minor time frame” (15ms in this example) and takes advantage of the fact that not all messages in a given system have to be transmitted at this interval. Specifying multiples of the minor time frame transmission interval and associated “transmission slots” allow a substantially larger number of messages to be transmitted via a single network. Note that this approach is valid even if all nodes in the network run asynchronously. This means that the nodes all have to use the same minor time frame but do not necessarily have to start their minor time frames at the same point in time.

Table 5-25 – Bus Scheduling

Transmission Interval	Messages/ Transmission Slot	Number of Transmission Slots (equaling 100% bus load)	Transmission Slot Identification
15ms (66.7 Messages per second)	1	50	A0 – A99
30ms (33.3 Messages per second)	2	100	B0[0] – B99[1]
60ms (16.7 Messages per second)	4	200	C0[0] – C99[3]
120ms (8.3 Messages per second)	8	400	D0[0] – D99[7]
240ms (4.2 Messages per second)	16	800	E0[0] – E99[15]
480ms (2.1 Messages per second)	32	1600	F0[0] – F99[31]
1000ms (1.0 Message per second)	66	6666	G0[0] – G99[65]

An example of a transmission schedule based on the bus-scheduling concept is shown below:

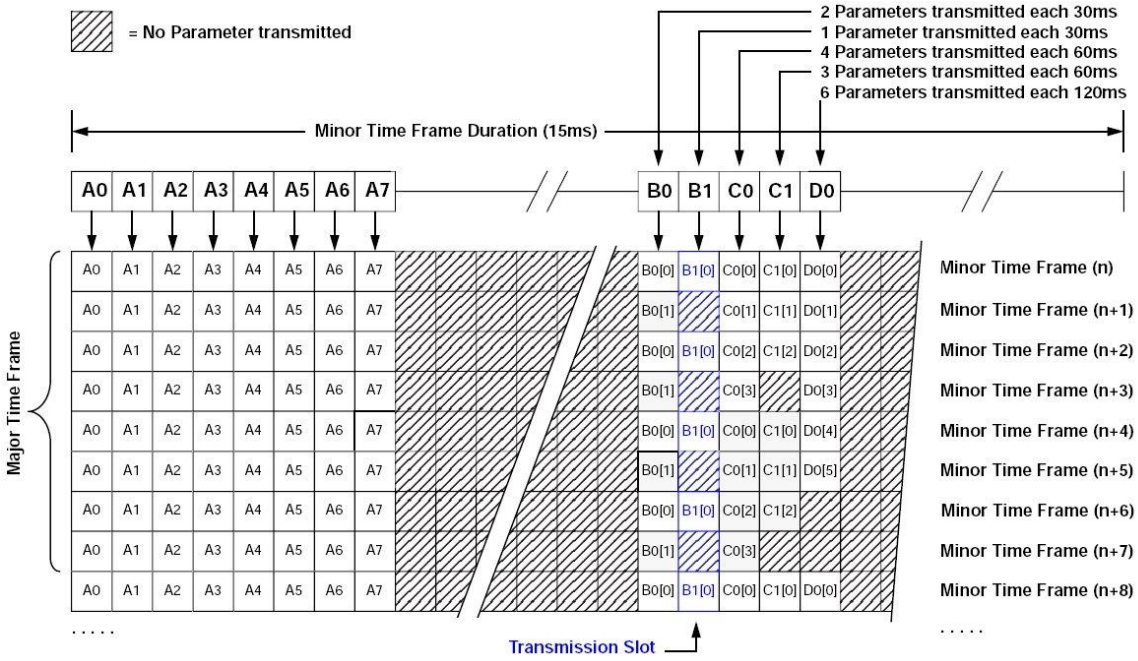


Figure 5-16 – Bandwidth Management Example Using Major and Minor Time Frames

The bus load calculation of the example above delivers the following result:

Table 5-26 – Bus Scheduling Example Bus Load Calculation

Transmission Interval	Number of Messages	Required Transmission Slots	Max. Number of Messages per Transmission Slot
15ms	8	8	1
30ms	3	2(1.5)	2
60ms	7	2(1.75)	4
120ms	6	1(0.75)	8
total	24	13(12.0)	

As the example uses 13 out of 100 available transmission slots, the corresponding average bus load is 13% (note the temporary bus load may be higher).

5.6.3 Maximum Bus Load

Keeping in mind that a reasonable number of event-driven messages and error frames in case of data corruption may require additional bus capacity, a system designed according to this standard should not continuously exceed 50% average bus load during normal operation. For implementations requiring very small transmission point jitter it may even be appropriate to limit this value to 30% average bus load. If more bandwidth is required in this case, the use of a higher baud rate or a second ARINC 825 network should be considered.

5.7 High Integrity Protocol

As defined, the standard CAN protocol has excellent data integrity due to built-in fault and error detection mechanisms. However, due to the operating environments of equipment used in avionic applications, they may be adversely affected by high energy particles that may cause single or multiple bit upsets. These effects may be mitigated by utilizing additional protection mechanisms within the CAN message data payload. This added protocol adds a sequence number and an additional integrity check to each high integrity message. These additional elements allow the receiver to detect missing messages and verify the correct sender of the message. The added protocol is transparent to nodes that do not require the added assurance and may co-exist on a CAN bus with normal messages without interference.

Another aspect of data integrity is temporal, the time when the data is generated compared to when it is received. If the user's application requires this additional assurance, a timestamp may be added to the message, but timestamps are not included in the standard high integrity protocol.

These high integrity messages are intended to be used for one-to-many communication on the NOC, UDC, and EEC. High integrity messages may be used for peer-to-peer communication channels (NSC, TMC) employing user-defined Node Services only. This protocol provides the following capabilities:

1. Impersonation detection (random errors in the CAN message ID)
2. Detection of missing messages (message sequence numbers enable the receiver to detect when messages have been lost)
3. Bit upsets in data bytes (bit upset that occur in the controller, before the CAN CRC is generated, may be detected)

The receivers of high integrity messages should maintain a log of faults detected and respond when the fault records are requested over the bus.

5.7.1 High Integrity Messages

To guard against single or multiple bit upsets causing faults in the CAN message ID or data field, the high integrity protocol uses a Sequence Number (SNo) and a Message Integrity Check (MIC) incorporated into the message payload before it is queued for transmission. Figure 5-17 depicts the High Integrity Protocol Message format.

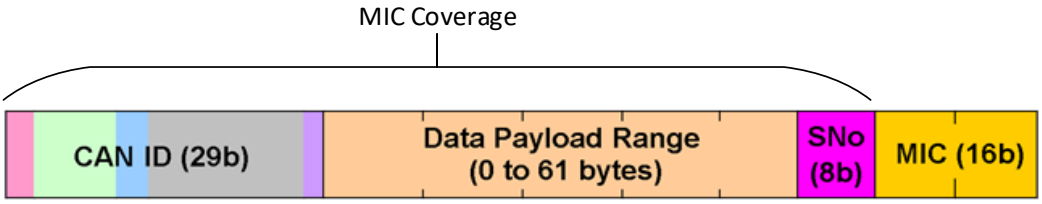


Figure 5-17 – High Integrity Message Protocol Format

When a message is defined as using the High Integrity Protocol, the parametric data shall be aligned in the message according to the offset defined in Table 5-27.

Table 5-27 – SNo and MIC Offsets for High Integrity Messages

	DLC	Data Field Bytes	SNo Offset (msb, lsb)	MIC Offset (msb, lsb)
Classical Sizes	0	0	N/A	N/A
	1	1	N/A	N/A
	2	2	N/A	N/A
	3	3	0.7, 0.0	1.7, 2.0
	4	4	1.7, 1.0	2.7, 3.0
	5	5	2.7, 2.0	3.7, 4.0
	6	6	3.7, 3.0	4.7, 5.0
	7	7	4.7, 4.0	5.7, 6.0
	8	8	5.7, 5.0	6.7, 7.0
FD Sizes	9	12	9.7, 9.0	10.7, 11.0
	10	16	13.7, 13.0	14.7, 15.0
	11	20	17.7, 17.0	18.7, 19.0
	12	24	21.7, 21.0	22.7, 23.0
	13	32	29.7, 29.0	30.7, 31.0
	14	48	45.7, 45.0	46.7, 47.0
	15	64	61.7, 61.0	62.7, 63.0

5.7.2 High Integrity Message ID

High integrity messages use message IDs that are assigned in the same manner as any CAN message that is compliant with standard high integrity data.

Data payload in a high integrity message is limited to 488 bits (61 bytes) as the overhead associated with the protocol required 3 bytes. If the data is longer than 5 bytes (Classical CAN) or 61 bytes (CAN FD), two or more messages may transport parts of the data, to be reassembled by the receiver. High integrity messages may be less than 64 bytes but not less than 3 bytes.

5.7.3 Sequence Number (SNo)

Sequence numbers (SNo) shall be maintained on a per message ID basis. The SNo indicates the number of the current message for a particular message ID. The sequence numbers shall initialize to zero (0) and increment by one (1). When the SNo reaches 255, the next increment shall rollover back to one (1). A SNo of zero indicates this is an initial sequence number. It is the responsibility of the sender to increment the SNo before each transmission of a particular message ID. For messages utilizing Message Level Functional Status, the SNo shall only be incremented when the Message Level Functional Status is Normal Operation (NO) or Functional Test (FT). If the SNo is zero, the receiver accepts the data and resets

its message counter for that message ID. It is the responsibility of the receiver to check the SNo of the received data to ensure no messages were missed.

5.7.4 Message Integrity Check (MIC)

High integrity messages shall use a 16-bit cyclic redundancy code for the Message Integrity Check (MIC), defined by the following polynomial:

$x^{16}+ x^{15}+ x^{12}+ x^{7}+ x^{6}+ x^{4}+ x^{3}+ 1$ (also known as 0x90D9 without the leading '1bit').

The MIC shall include the CAN message ID, the data payload, and the SNo.

The following shall be used to compute the CRC:

1. The initial CRC value shall be 0xFFFF.
2. Each input byte shall be reflected about its center.
3. Each input byte shall be calculated Most Significant Bit first.
4. The final CRC shall be reflected about its center and XOR with 0xFFFF.
5. The message ID is assumed to be 32 bits with the leading 3 bits set to zero.
6. An algorithm that properly computes the CRC shall produce 0x4084 when it computes the CRC on the following test string of ASCII characters, "123456789" with no null character.

COMMENTARY

The reason for the complicated CRC algorithm is to simplify the computation and reduce the time needed to compute the checksum. The message ID is a constant for any particular message; the CRC may be computed for the message ID and saved to use as a constant to initialize the CRC generator for subsequent computations. If the CRC computation uses a lookup table, the table would be based on the reflected inputs. This avoids extra shifting of the input values to align them with the CRC register. Also, if the table is based on the reflected inputs and reflected CRC polynomial, the results do not need to be reflected at the end of the computation. If the receiver computes its message CRC on the entire message, including the received CRC, the result should be zero if the CRCs match.

5.7.5 High Integrity Health Status

Each receiver should maintain a record of the faults detected in high integrity messages. The fault record should include a summary of faults detected for any high integrity message and a record should be maintained based on the message ID. Each error detection record should be as follows:

1. MIC errors (USHORT)
2. Messages missing (USHORT)
3. Messages with SNo = zero (USHORT)

If the error counters reach their limit before they are read out or reset, they should stop counting at their maximum values. It is intended that a node on the bus should periodically read the contents of the fault counters. Transmitting the error counter values over the bus should reset their contents to zero. Note, additional system bandwidth is needed to monitor the error records over the network.

5.7.6 High Availability Message

When a message type is defined as High Availability a sequence number (SNo) shall be encoded into the message data field. The SNo can be used to provide ordinal integrity information as well as context to messages sent redundantly across two or more buses. The sequence number is implemented as an unsigned eight-bit parameter that is aligned to the data field according to the alignment values listed in Table 5-28 below. The transmitter of a message shall encode the SNo value as defined in Section 5.7.3. The receiver of High Availability messages can utilize the values defined in the XML profile, **txp** and **sno_valid_period**, in order to make decisions about ordinal integrity, message source selection, redundancy management, or voting. The LRU XML profile message attributes **txp** and **sno_valid_period** shall be defined by the LRU designer or system integrator.

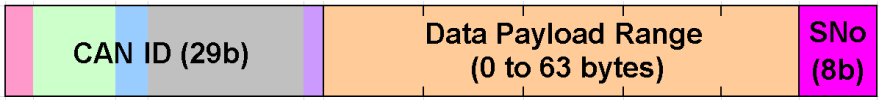


Figure 5-18 – High Integrity Message Protocol Format

Table 5-28 – SNo Offsets for High Availability Messages

	DLC	Data Field Bytes	SNo Offset (msb, lsb)
Classical Sizes	0	0	N/A
	1	1	0.7, 0.0
	2	2	1.7, 1.0
	3	3	2.7, 2.0
	4	4	3.7, 3.0
	5	5	4.7, 4.0
	6	6	5.7, 5.0
	7	7	6.7, 6.0
	8	8	7.7, 7.0
FD Sizes	9	12	11.7, 11.0
	10	16	15.7, 15.0
	11	20	19.7, 19.0
	12	24	23.7, 23.0
	13	32	31.7, 31.0
	14	48	47.7, 47.0
	15	64	63.7, 63.0

6.0 CAN GATEWAY CONSIDERATIONS

This section describes CAN gateways and the primary considerations that need to be addressed in their implementation. The focus is on gateways to and from CAN.

In this context, a gateway is a CAN node equipped for interfacing with another network or digital bus, such as ARINC 429. A gateway provides physical interface such as impedance matching, rate converters, fault isolators, or signal translators as well as functional protocol translations necessary for networks to interoperate. A gateway must also provide mechanisms to handle administrative procedures and health management between the networks.

Although the description here refers to an interface between two networks, these concepts may be extended to any number of networks that use a gateway via a single CAN node. The gateway requirements described here are for a single unit, providing the gateway for one CAN bus, to one higher performance network, such as ARINC 664. The gateway could be between the CAN bus and other serial buses (e.g., ARINC 429, RS232C, RS422/485). A gateway may also be used between two CAN buses to provide isolation or to adapt to different data rates or extend the length of the combined CAN buses.

A gateway does not need to have all the physical or functional attributes described here to be considered a gateway. Nor does the node performing the CAN gateway function need to be limited to only the gateway function.

A CAN gateway as a minimum shall provide the following functional capabilities:

1. Comply with the protocol requirements on transmitting or receiving networks include the physical interfaces.
2. Isolate faults on one network from the other network.
3. Forward (or not) messages from one network to the other.
4. Convert from one network protocol to another network protocol.
5. Limit the flow of data from higher speed to lower speed networks.
6. Monitor and make available the health and status of the connected networks.
7. Provide repeatable and predictable operation.

6.1 General Requirements

Gateways are required to perform one or more of the following functions:

1. Adapt one network protocol to another, where protocol includes:
   a. Transmission media (fiber versus copper)
   b. Electrical signal levels (voltage or light wave lengths)
   c. Data encoding method (e.g., NRZ, Manchester)
   d. Data rate (e.g., 250 kbit/s, 500 kbit/s, 1,000 kbit/s, 10,000 kbit/s)
   e. Message encoding (addressing, identifiers, and check codes)
   f. Message sizes (number of bytes per message)
2. Addressing or message ID conflicts (where two networks use the same identifiers or addresses for different purposes)

3. Filtering
    a. Limit the message traffic that is passed from network to network
    b. Reduced parameter update rates

An ideal gateway would be transparent to the network users, but typically this ideal may not be achieved. In this context, the term ‘transparent’ should be taken to mean ‘the same as if transmitted or received directly from a device on the destination network’. There are a number of different strategies that may be used by a gateway unit depending on the system and its requirements. Message buffering is required between networks due to different data rates, message sizes, and network access timing. Even when networks that use the same protocol are gatewayed, buffering is required due to network access constraints. Buffering in itself forces the gateway from the ideal by introducing additional latency.

6.2 Gateway Model

The general networking model in Figure 6-1 is used to illustrate where gateways are used. The figure depicts a system with 3 network types: A, B, and CAN. Between Network A and B there is a gateway and another one between Network A and the CAN Network. The units on Network A may communicate with units on the CAN Network or Network B but only to the degree the gateways are configured to allow that communication. This is one of the functions of a gateway, to filter or limit traffic between the connected networks. Similarly, units on Network B may communicate with units on Network A and the CAN Network, but again only to the extent the gateways are configured to allow this communication. Gateways must be capable of selectively forwarding messages between the interfacing networks.

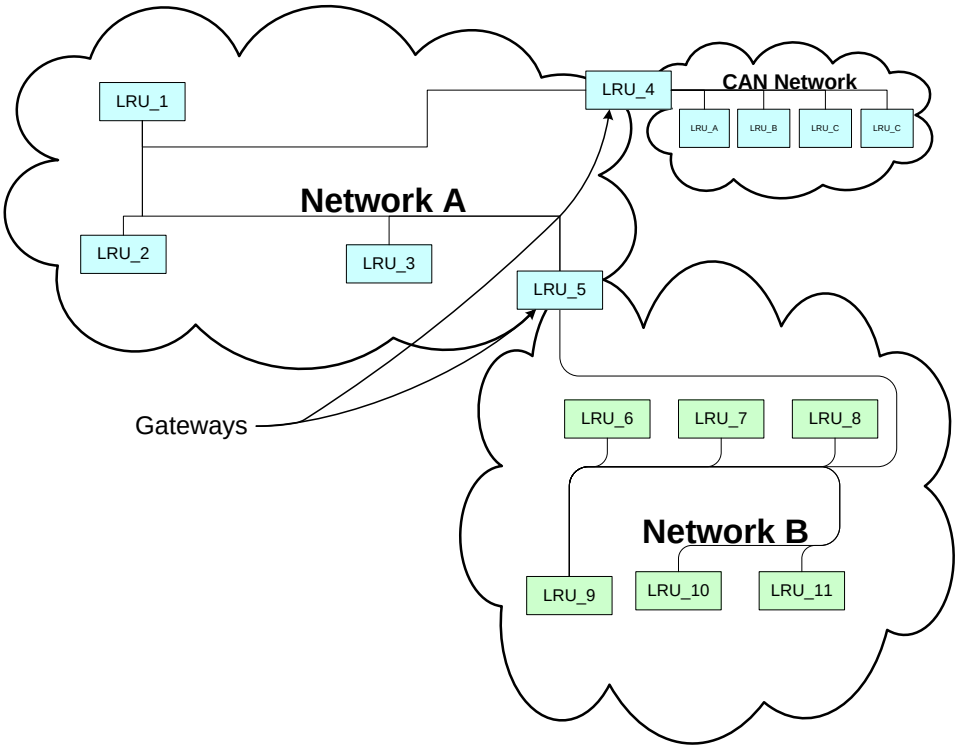


Figure 6-1 – General Networking Model

6.0 CAN GATEWAY CONSIDERATIONS

When networks of different protocols are connected via a gateway, the gateway is required to add or possibly remove some, or all of one of the network header information. How the gateway manages protocol translations depends on the network protocols involved and the system requirements for data security, integrity, and data latency.

Figure 6-2 illustrates the essential resources for a gateway device, such as physical interfaces to the networks, protocol engine and CAN controller, non-volatile storage for configuration and control programs, buffers for receive and transmit data, and a programmable gateway controller. Gateways may be simple units that translate one network connection to another network, or more complex units translating many types of networks to other types of networks. For the purpose of this discussion, we will only address gateways that interface one network type to another network, but these same concepts may be applied to one-to-many or many-to-many type gateways. Figure 6-2 illustrates the essential functions for a gateway device and an actual implementation may integrate some or all of the separate elements shown. The interface to CAN is shown in the lower half of the figure, and in the top half, the interface to the other network. In the middle is the gateway controller. Key points here are the network, CAN controller, and gateway controller have storage for configuration information while the network interface and CAN interface also have storage buffers for data.

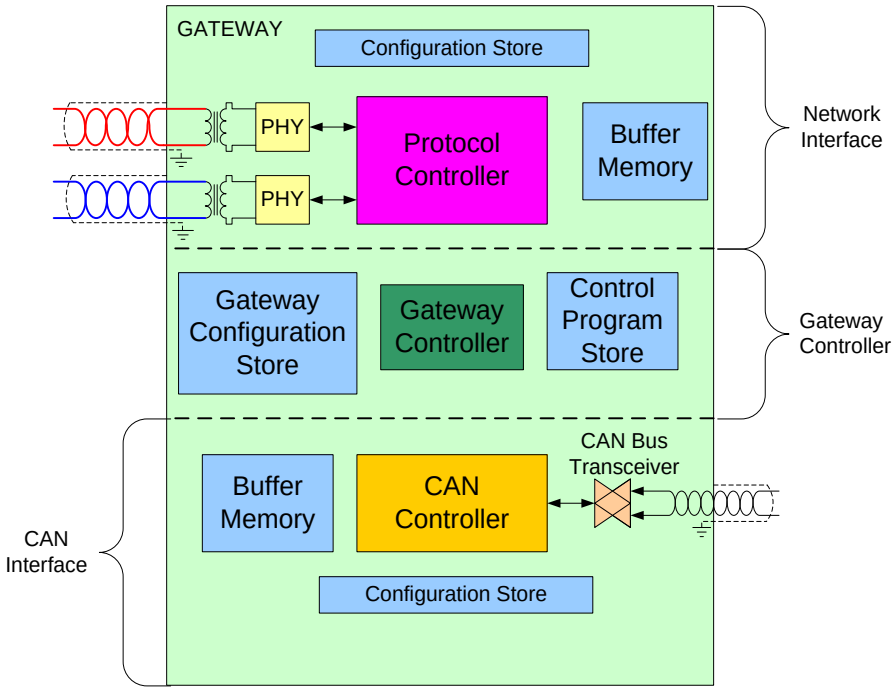


Figure 6-2 – Generic CAN Gateway

6.3 Primary Gateway Functions

The primary gateway functions are required to provide the functionality for proper system operation.

6.3.1 Comply with Protocols

A gateway shall comply with the network protocols of the receiving and the sending side of the data transfers.

6.3.2 Gateway Bandwidth

The gateway design shall ensure it does not use more bandwidth than has been allocated to it. See Section 5.6 for CAN bandwidth management.

6.3.3 Gateway Fault Isolation

Faults that might occur on one of the connected networks should not be propagated from the one network to the other.

6.3.4 Gateway Message Forwarding

The gateway shall provide a routing function of the network messages, per configuration, to the other network(s).

6.3.5 Protocol Conversion

Messages received from one network may require conversion in order to be forwarded. Conversions described here are limited to protocol specific conversions, such as message header, addressing, or identification of messages. If the data itself is modified, this is not considered a gateway function except potentially in the following cases:

1. Data loading
2. Health monitor and status gathering
3. Message integrity translation

It is the required function of a gateway to add necessary header, addressing, and trailers to messages to ensure they are routed to the proper destination. CAN messages have relatively small data payload capability. In order to reasonably and efficiently transfer messages between CAN and Ethernet based networks, the gateway must add appropriate protocol headers and checksums.

Due to the dynamic nature of system designs during development, it is recommended that gateway configuration and control be on-board loadable. It is beyond the scope of this standard to define the parameters a gateway should include in its programmable configuration.

6.3.6 Data Rate Metering

It is essential that a gateway meter its output of data to a CAN bus. This requirement needs to be balanced against the need to meet data latency requirements. Gateways, like other CAN nodes, must be allocated bandwidth on the bus. The gateway should spread the data being output to the CAN bus over a frame of time. The time frame is more of a concept than an actual entity. The time frame should be consistent for all users on the CAN bus and documented in the CAN Specification, see APPENDIX F.

6.3.7 Health and status

Gateways by their nature isolate the networks they interface with from each other. An important function of a gateway is to make available the health and status of the networks they interface to. The required health reporting is dependent on the types of networks and the systems connected. The health monitoring and reporting should be captured in the CAN Specifications, see APPENDIX F.

6.3.8 Repeatable and Predictable Operation

A key attribute of avionic systems is their operation is both repeatable and predictable. The operation of a gateway must also be repeatable and predictable.

6.4 Gateway Resources

For a gateway to provide the functionality previously described, it will need to have a set of resources appropriate for the networks or buses it is providing the gateway between.

6.4.1 Data Buffering

The gateway should be capable of buffering data in order to be compatible with the data rates of the networks the gateway interfaces with. There are several acceptable data buffering methods and any one or more may need to be employed by a gateway.

1. Buffer queue
2. Sample buffer
3. FIFO (First In, First Out)

6.4.2 Time Based Scheduler

In order to perform the required data rate metering, the gateway shall utilize a time-based executive or scheduler.

6.4.3 Filtering

An essential function of a gateway is to selectively route data from one network to the other. The gateway shall be configurable to selectively route some data while filtering out other data.

6.5 Data Transfer via a Gateway

Gateways shall be capable of accepting data and program loads from one network and performing the require program loading protocol on the other. The load protocols used in some more complex systems may not be supported by equipment on a CAN bus. The gateway must manage the system data transfer (data load) protocol and forward the data to the devices.

6.6 Gateway Redundancy Management

Highly available systems frequently utilize redundancy to mitigate the reliability of units in the system. The network that connects these systems must also mitigate the effect of component failures.

The redundancy required for each application must be determined by the system safety assessment.

7.0 DESIGN GUIDELINES

7.1 Introduction

7.1.1 Purpose of the Guidelines

This section is intended to document industry experience and rationale in preparing this document. What follows are general network design guidelines to be considered when designing a Controller Area Network. These guidelines are recommendations to avoid potential design traps a network designer may encounter. It should also be noted that this section is not an exhaustive discussion of these potential design issues nor are they necessarily the most important issues a designer may face. Each implementation is unique and, as such, unique issues will arise that may not be discussed below.

7.1.2 Use of the Design Guidelines

The Design Guidelines attempt to follow a standard format. Each of the major positions discussed in ARINC Specification 825 has a corresponding section in the Design Guidelines. The purpose for this structure is to support the efforts of any design engineer in a balanced way such that the primary subject matter of interest has an easily found supporting guideline that may be referred to as a parallel activity.

In addition, these guidelines provide additional design considerations that do not conveniently fit into the sections of the ARINC Specification 825 format.

7.1.3 Organization of These Guidelines

The Design Guidelines section is organized as follows:

- Section 7.1, Introduction
- Section 7.2, Overview
- Section 7.3, CAN Physical Layer
- Section 7.4, CAN Data Link Layer
- Section 7.5, CAN Communication
- Section 7.6, Bus Load Management
- Section 7.7, Redundancy Management
- Section 7.8, Data Load
- Section 7.9, Safety, Reliability, and Certification Awareness Considerations

7.1.4 Design and Development Aids

The Design Guidelines provide significant information for Equipment Suppliers and System Integrators. Equipment Suppliers and System Integrators may also find the information in the following appendices useful as well:

- Appendix E, Design Checklist, provides a list of topics with references to the sections stating what every bus needs to address. The list may be used to ensure the key topics have been addressed during the developing and implementation by developers and integrators.
- Appendix F, CAN Specification Template, provides a starting point for creating a specification for implementing a CAN bus on an aircraft.

- Appendix G, FAA AC 20-156 Cross Reference, may assist the developer and integrator to meet certification requirements.

7.2 Overview

The network designer will need to understand which of the following three network designs most closely resembles the design they intend to implement.

The first, and simplest, is the network pictured as CAN Bus #1. In this situation, a designer needs to consider how the nodes on this bus will operate together and what individual characteristics each node will contribute to the whole. In this situation, there is a possibility of more than one supplier providing a node to the bus, each of which may meet the defined requirements, but due to various factors, the node(s) are not interoperating in the shared media effectively. The requirements presented in the Physical Layer section of ARINC Specification 825 are intended to mitigate this potential, while the corresponding sections below will discuss the rationale behind these requirements.

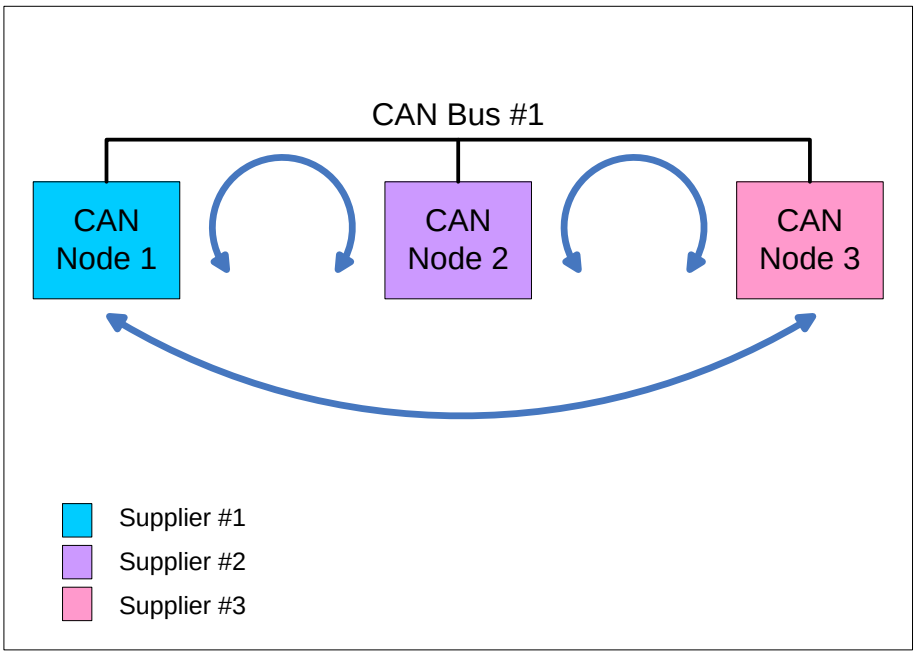


Figure 7-1 – Simple CAN

The second consideration would be communication from the network identified as CAN Bus #1 and one identified as a High-Speed Network. In this case, data from CAN Bus #1 is destined for some other network and, as such, the data will be converted to another network protocol specific to this other network. This conversion will need to be considered in the initial network design so that any potential impacts are clearly understood. While the initial release of ARINC Specification 825 does not specifically address gateway functionality, certain attributes are discussed in the Communications section that will facilitate these functionalities.

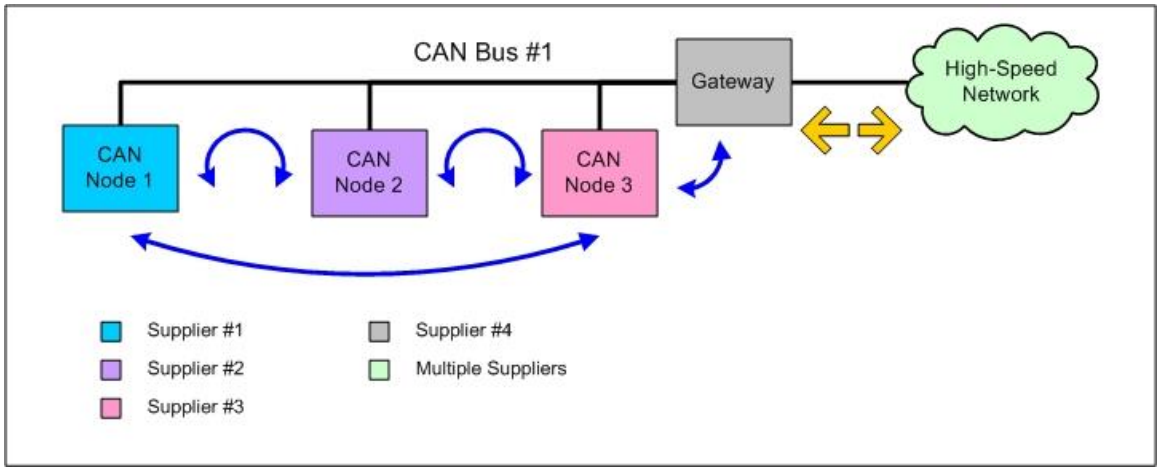


Figure 7-2 – CAN Communication to High-Speed Network

The third consideration is a variation of the second with communication between the network identified as CAN Bus #1 and that identified as CAN Bus #2. In this situation, the task of the network designer is greatly increased as there will be nodes supplied from multiple suppliers as well as the increased complexity of converting data to accommodate the intermediary network protocols before being converted back to CAN data. Without a strict definition of the major network components (Physical Layer, Data Link, and Communication) anarchy would reign and the system as a whole would not function as anticipated.

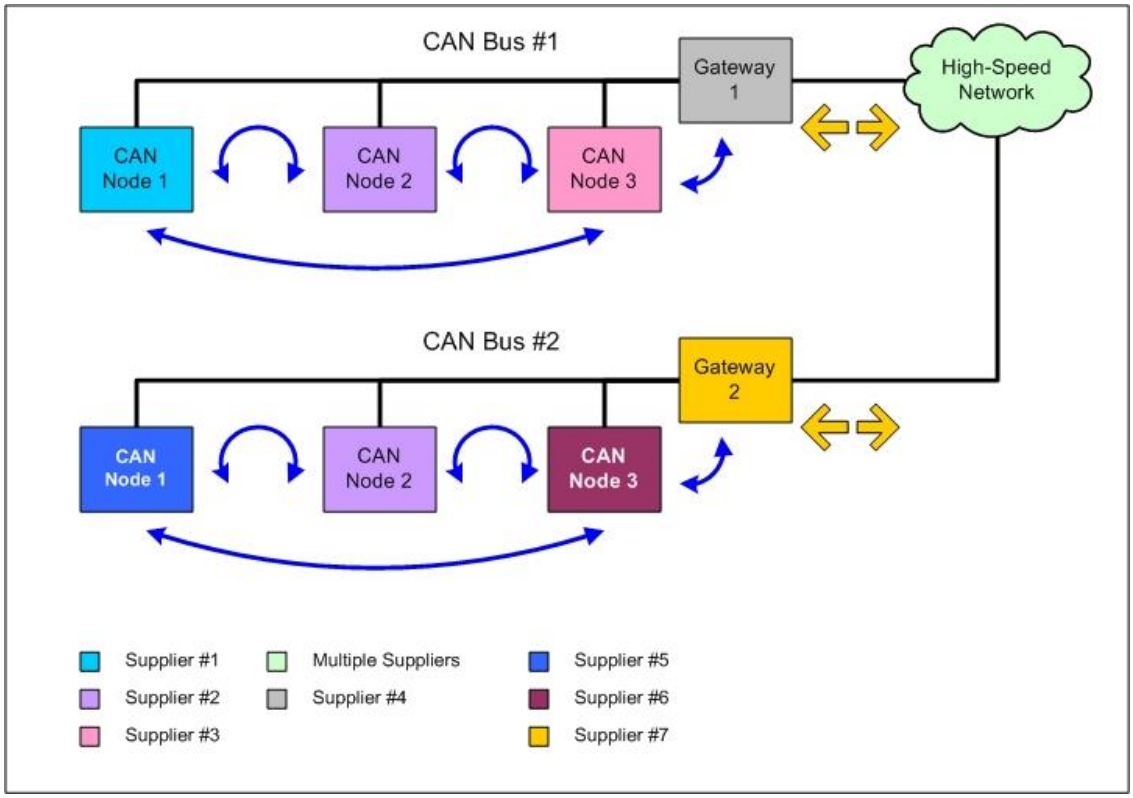


Figure 7-3 – CAN to CAN Communication Through Gateways

The nature of CAN makes efficient management of shared bus resources critical to the successful integration of CAN based systems on a shared transmission medium.

Successful CAN integration depends on effective bus management in the early design phase. Effective bus management establishes the design constraints and rules that CAN nodes use to ensure interoperability when integrated as a complete system onto the shared bus.

The overall CAN design should consider interoperability issues ranging from the physical interfaces to the communication layer of the system architecture.

7.3 CAN Physical Layer

Compliance to the requirements defined in the Physical Layer section will ensure interoperability of the nodes connected to the same CAN bus. To ensure a robust design, the following parameters should be considered.

7.3.1 Basic Considerations

- Ensure that CAN controllers and transceivers are fully compliant to ISO 11898-1: 2015 and ISO 11898-2: 2016.

  Note: CAN Controllers and transceivers need to be fully compliant not just compatible.

- Ensure nodes have the same bit timing (sample point, etc.).
- Ensure the capacitance added by the EMI protection does not exceed the capacitive load limits as set forth in Electromagnetic Protection Requirements.

COMMENTARY

Ensure all electrical characteristics are met without exception; otherwise, there may be intercommunication incompatibilities. The intercommunication incompatibilities may be subtle and difficult to detect initially but become more obvious as the system matures.

CAN transceivers are Commercial Off The Shelf (COTS) components used by other industries are generally interoperable and pin compatible. Nevertheless, some transceivers do not offer the required aviation safety requirements. Examples are dominant time-out protection and “invisible when not powered” functions. Some transceivers have better characteristics than others, such as more electromagnetic robustness and wider common mode range. Thus, the designers should carefully review the data sheet in the light of their design requirements.

To produce a robust network with minimum wave reflections, the system designer must follow the guidelines in the Installation Considerations, Section 7.3.5. For CAN, the bus length is defined as the distance between the two terminating resistors and all other connections being stubs, the following should be considered:

- Ensure that the cable has the correct characteristic impedance.
- Ensure that a bus has exactly two CAN Bus terminators, one at each physical end of the cable run.
- Ensure that the wire harness design does not accidentally create a Star Topology due to wire routing and any wire integration panel design.
- Limit stubs to the minimum possible length.

Ignoring the above considerations could result in a weak or marginal physical layer affecting the data bus communication by increasing the Bit Error Rate (BER), causing sporadic failures which are difficult to detect and to solve. This weak or marginal physical layer will reduce the functional and EMI margins.

The signal quality is mainly affected by:

- CAN bus length
- Number, position, and length of stubs
- Shielding quality
- Transceiver characteristics
- EMI protection components
- Bit timing (sample point, etc.)

7.3.2 Bus Loading Constraints

When identifying the maximum number of nodes, a CAN segment may support, the designer must also consider the design lifecycle and operating environment of the system. Typically in avionics, the design environments are severe and life cycles are long. A certain amount of “capability de-rating” is usually applied to ensure adequate performance margin to cover the design lifecycle. Table 7-1 illustrates a suggested typical maximum number of CAN nodes per single segment for the selected data transfer rate of the bus given the criteria above. The table represents successful systems experience and extrapolated values. This table may only be used as a rough guideline because other constraints, i.e., bus length, may cause variation.

Table 7-1 – Typical Maximum Number of Nodes

Data Transferred Rate (kbit/s)					
Number of CAN Nodes (Typical Maximum)	**83.333**	**125**	**250**	**500**	**1000**
	60	50	40	35	30

7.3.3 Guidance for Determining Physical Bus Loading

The maximum number of nodes on the bus is determined by the minimum load resistance a transceiver is able to drive $R_{L\_min.}$ The overall bus load is defined by the termination resistance R_T, the bus line resistance R_W and the transceiver’s differential input resistance R_{diff}. For worst case consideration, the bus line resistance R_W is considered to be zero.

$$n_{\max} < R_{diff\_\min} \times \left(\frac{1}{R_{L\_\min}} - \frac{2}{R_{T\_\min}} \right)$$

The maximum achievable bus line length for CAN is determined by the maximum available round trip propagation delay t_{prop} (CAN bit timing), the effective loop delays t_{node} of the connected bus nodes (CAN controller, clock tolerance, transceiver, isolator, etc.), the specific line delay of the bus t_p, and the signal amplitude drop due to the series resistance of the bus cable and the input resistance of bus nodes. As the signal drop is only significant for very long lengths, the voltage drop may often be neglected.

$$L < \frac{\frac{t_{prop}}{2} - t_{node}}{t_p}$$

It follows that the choice of CAN transceivers and the internal node parameters may make a difference between 30 nodes and 60 nodes or between 50m and 100m bus length.

7.3.4 Electromagnetic Environment Specifications

The CAN interface(s) should be, at a minimum, compliant with the airframe and regulatory specification(s) such as RTCA DO-160 concerning lightning, radio frequency susceptibility, and on-board system environment levels consistent with the installations. Equipment protection has to be at the highest level of exposure of the equipment or the cable routing for the connected buses.

7.3.4.1 Exposed Area

The system designer must establish the correct Lightning and EMI environmental test categories early on in the design. The applicable environmental category will, in part, be based on the equipment location, its aircraft cable routing and its functional criticality.

The EMI and lightning threat levels of the node on the same CAN wire are determined by the most constraining node or cable exposed area. For example, when a system topology is from the electronic bay to the landing gear location, the electronic bay CAN interface will have to sustain the same EMI environmental constraints as the landing gear CAN interface.

7.3.4.2 Electromagnetic Interference Testing

The goal of EMI testing should be that no error frames occur during the test.

The following conditions for EMI testing should be met:

- Normal Operation;
- Monitoring for errors frames is available during the test;
- The test set up should be representative of the real aircraft installation and topology.

COMMENTARY

The no error method of testing facilitates assurance of a no bus off condition.

7.3.4.3 Lightning Indirect Effects

The CAN physical layer uses a DC coupled differential signal with a reference to ground. Therefore, all threats applied to ground will apply to the CAN interface. As a result, bi-directional Transient Voltage Suppression (TVS) or equivalent devices are necessary on each CAN interface connection (see Capacitance and Lightning Protection section of the Guidelines for more information). The high voltage and current levels induced by a lightning strike will lead to a temporary total loss of communication on the bus. During such an event, the CAN nodes:

- Should not be damaged
- Should recover automatically to full capacity after the lightning event

7.3.4.3.1 Lightning Suppression

A major difference between the Automotive and Avionics implementations of CAN is operation in the presence of Lightning. On average, each air transport aircraft is struck by lightning a couple of times each year. Dependent on the location of the node or its wiring route, the induced lightning voltage levels are significant (for example: > 1000 Volts).

Traditionally, avionics have employed the use of voltage and current limiting, lightning suppression devices, such as TVS. While these devices work very well, they have some drawbacks which designers should be aware of. The most notable drawback of lightning suppression devices is the inherent internal capacitance of this technology.

Bus performance is susceptible to large capacitance on the physical layer. The following sections outline some example methods, but not the only methods, that provide lightning transient protection while reducing the undesirable effects of large induced capacitances on the physical layer.

7.3.4.3.2 Transient Voltage Suppression

A CAN node may be referenced to the local ground via the use of Transient Voltage Suppression (TVS) devices to prevent LRU damage when subjected to lightning transients, see Figure 7-4. Care should be used in the selection of TVS devices to ensure low capacitance units are used. Additionally, in order to reduce the capacitive loading of the lightning suppression circuit even further, blocking diodes should be employed to reduce the apparent intrinsic TVS capacitance.

The breakdown voltage of the TVS should be selected to be less than the common mode voltage level of the transceiver.

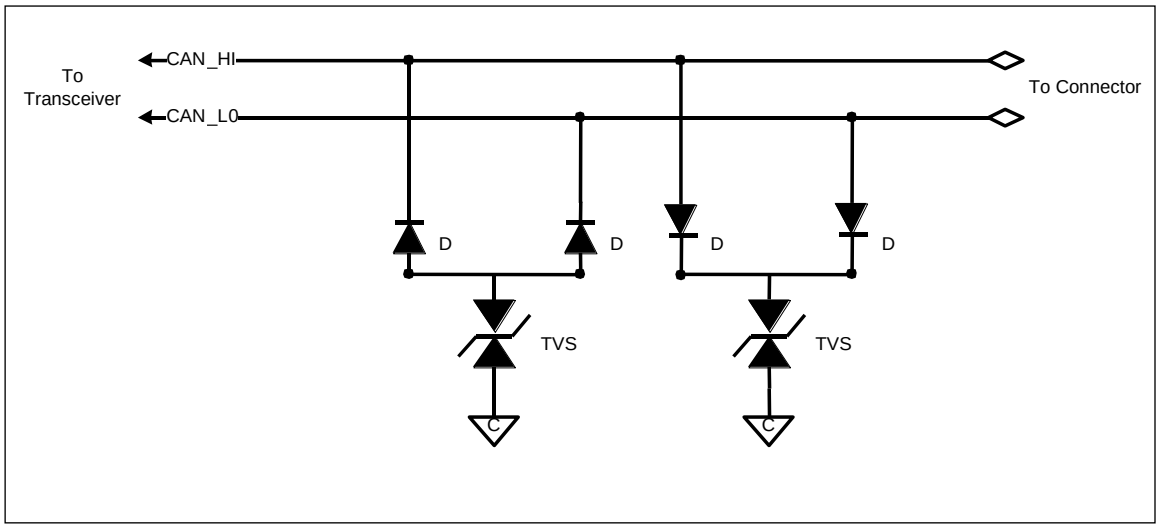


Figure 7-4 – Example of a CAN Bus Lightning Suppression Circuit

7.3.4.3.3 Isolation Barrier

An isolation barrier may be applied between the CAN transceiver and the CAN controller if the TVS solution is not possible because of very high lightning constraints. Then, both the CAN transceiver and the isolation barrier are isolated from the chassis ground and from the local CPU/controller power supply. The isolation barrier may be either external or internal to the CAN transceiver.

The tradeoffs of this technique are that:

- It adds components on the printed board (CAN isolation barrier + power supply isolation). It should be compared with the TVS size.
- It increases the t_{node} loop delay: the compromise of baud rate/bus length/number of nodes must be validated.

If this technique is chosen, it must be applied to all nodes (no TVS).

Figure 7-5 illustrates an isolation barrier example for CAN (on-chip magnetic isolation with 25 kV/s common mode rejection and 2500 V_{rms} dielectric withstand):

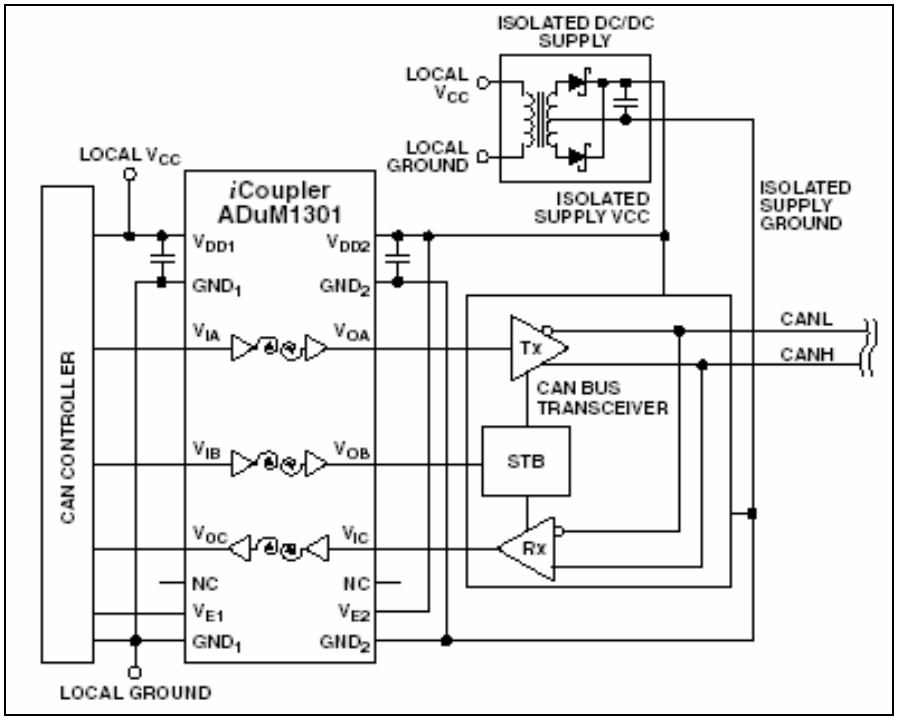


Figure 7-5 – Isolation Barrier Example for CAN

7.3.4.3.4 Filter Pins

The use of EMI Filter Pins on the LRU connector(s) should be avoided. The capacitance varies and low pass effect causes additional propagation delay. The effectiveness of the filtering depends on large capacitance and these filter elements are so small they only work at high frequencies.

7.3.5 Guidance for Installation

This section defines the CAN interconnect system including physical geometry with wiring, connectors, and splices. This physical media (wiring and connectors) are to be selected to support the ARINC 825 selected data rates. The interconnect system should be designed to meet the requirements of the areas of the aircraft where it will be installed such as pressurized or un-pressurized areas. The following sections describe the characteristics of the cable, termination, and physical geometry of the CAN layout.

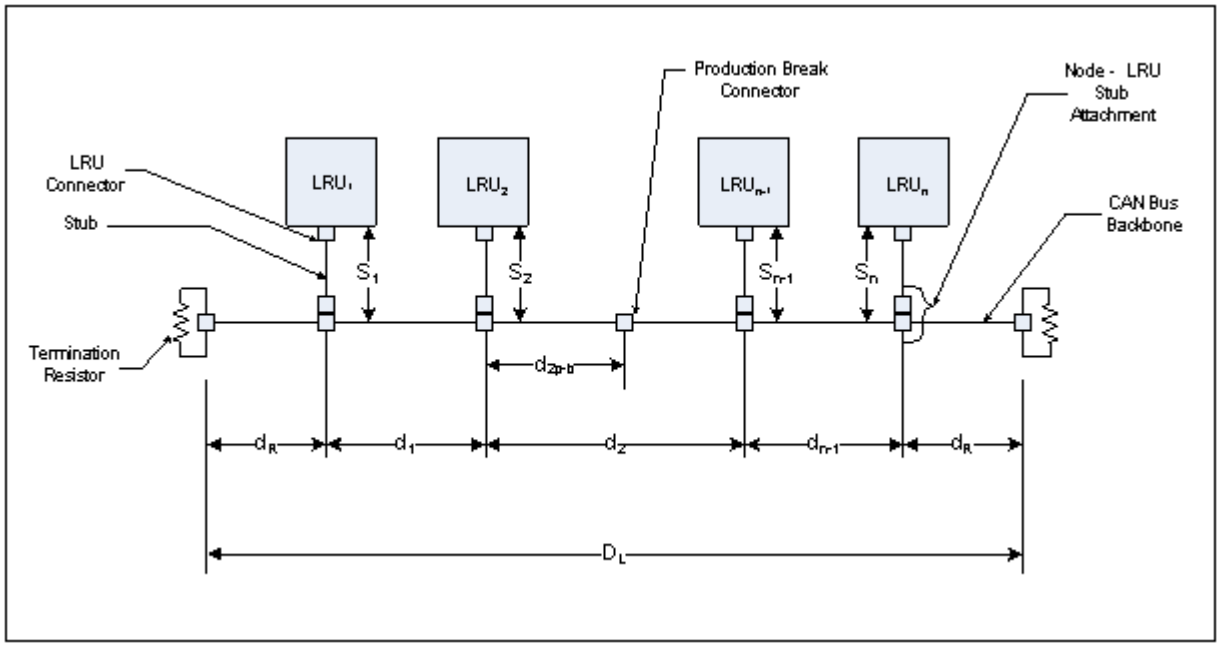


Figure 7-6 – Linear Bus Topology

7.3.5.1 Data Rate Versus Distance

It is important for system designers to understand the relationship between CAN data rate versus distance when setting up the system architecture. The physical location of CAN nodes in the aircraft may affect the maximum CAN data rate achievable, consequential CAN node throughput and thus system performance (refer also Section **Error! Reference source not found.**, Bus Speed).

If the wire length exceeds the electrical limit, the only options are to reduce the data rate to one compatible with the wire distances or to adjust the system architecture accordingly to extend the bus segment via the use of a repeater or a gateway. Either way, finding out these limitations late in a program is a painful experience.

7.3.5.1.1 CAN Wiring Length Constraints

In order to have an operational system with functional margins at a selected data rate, there are constraints on lengths (the bus and stubs) and number of nodes. Conservative relationships among the baud rate, number of nodes, and lengths are presented as guidance in Table 7-2. The user is cautioned that only test and/or simulation will assess the physical layer signal quality and the topology acceptability.

Table 7-2 – Classical CAN Wiring Length Guidance

PARA-METER	SYMBOL (See Figure 7-6)	83.333 kbit/s		125 kbit/s		250 kbit/s		500 kbit/s		1000 kbit/s		UNIT
		MIN	MAX[5]	MIN	MAX[5]	MIN	MAX[5]	MIN	MAX[5]	MIN	MAX[5]	
Bus length[1]	DL	3.5	200+	3.5	160+	3.5	80+	3.5	40/80	3.5	20/ 40	m
Stub length[2]	Sn	-	1.0	-	1.0	-	1.0	-	1.0	-	1.0	m
Node distance[3]	dn	0.5	197	0.5	157	0.5	77	0.5	37	0.5	17	m
Number of nodes	N	2	60	2	50	2	40	2	35	2	30	
Minimum Distance from Closest Node to Terminating Resistor[4]	dR	1.5	-	1.5	-	1.5	-	1.5	-	1.5	-	m

1 Length of the bus as measured between the two terminating resistors (excludes stub lengths) Stubs and/or bulkhead connectors contribute to impedance mismatch may substantially reduce this optimum length.
2 Measured wire length between controller chip and node attachment point on the main bus.
3 Measured distance between nodes on the bus.
4 The minimum distance needs to be 1.5 meters greater than the stub length of the closest node.
5 The maximum values may be extended when stub lengths are minimized and the number of nodes is less than the maximum. For speeds of 500 and 1000 kbit/s, the shorter length is applicable to maximum number of nodes and maximum stub lengths, and the longer length is applicable if the stub lengths are minimal and number of nodes is reduced.

7.3.5.2 Capacitance and Lightning Protection

The designer is cautioned that CAN performance may be degraded in the presence of high capacitive loads on the bus. The capacitive load may be due to a single non-conforming LRU or may be a cumulative effect of multiple LRUs on the bus. The system level effects of high capacitance range from random anomalous behavior to complete bus failure.

It is important to establish early on in the system development cycle the design constraints of an integrated CAN system at the physical layer in order to avoid interoperability problems later on. A mismatched CAN bus may not interoperate correctly and it may be very difficult to isolate faults. This is particularly important in systems that interconnect LRUs from multiple suppliers.

The internal capacitance C_{in}, for a LRU that attaches to CAN is defined as the capacitance seen between CAN_H or CAN_L and transceiver ground during the recessive state, with the LRU powered on and disconnected from the bus.

The differential internal capacitance, C_{diff}, of an LRU is defined as the capacitance seen between CAN_H and CAN_L during the recessive state, with the LRU powered on and disconnected from the bus.

Capacitance and Differential Internal Capacitance should be in accordance with the values described in Section 3.1.1.1, Electromagnetic Protection Requirements.

7.3.5.3 Termination Resistors

The termination resistors are located at the physical ends of the CAN main bus as shown in Figure 7-6. The practice of placing the termination resistors internal to the LRU should be avoided. The termination resistor needs to be 120Ω ±10% for correct bus operation. The resistor should have an effective series inductance of less than 0.37 μH and an effective parallel capacitance of less than 1.5 pF.

The termination resistor should have an installed length of d_n = 1.5 m beyond the adjacent CAN node stub attach point.

7.3.5.4 Guidance for Wire Implementation

The following sections describe good wiring practices and potential design deficiencies.

7.3.5.4.1 Acceptable Wire Methods

There are essentially three methods of connecting LRU(s) to the bus as follows:

1. Connector Splice Method (refer to Figure 7-7):
2. This method is highly recommended as the preferred method for aircraft environments.
    - In this method the bus is spliced at the LRU connector. The two twisted pair wire shields are connected to the connector backshell.
    - The advantage is short stub lengths.
    - A disadvantage is larger wire distance and weight dependent on installation because the bundle must be routed to and from the LRU location.
3. Daisy Chain Method (refer to Figure 7-8):
4. This method is deemed to be highly undesirable in an aircraft environment and should be avoided.
    - In this method, the bus is run through the LRU on separate input and output pins or connectors. Note that there is only one CAN transceiver inside the node.
    - This method provides for short stub lengths but larger wire distance than the splice method. It has the undesirable effect of splitting the bus upon LRU removal: The remaining nodes beyond the bus break may fail, as the characteristic impedance of the bus is broken to one termination resistor. This will degrade the characteristic impedance of the bus by breaking the connection to one termination resistor.
5. Bundle Splice Method (refer to Figure 7-9):
6. This method is less desirable in an aircraft. As such, the system designer should take extra precautions when utilizing this method.
    - In this method, the stubs are spliced and routed from a main trunk bundle.
    - An advantage is less wire weight.
    - A disadvantage is longer stubs that may bring discrepancies due to wire routing. A deep validation is needed to ensure physical layer quality and robustness. Use this method with caution.

- Care must be taken to avoid accidental star wiring configurations, particularly where Wire Integration Panels (WIPs) are utilized.

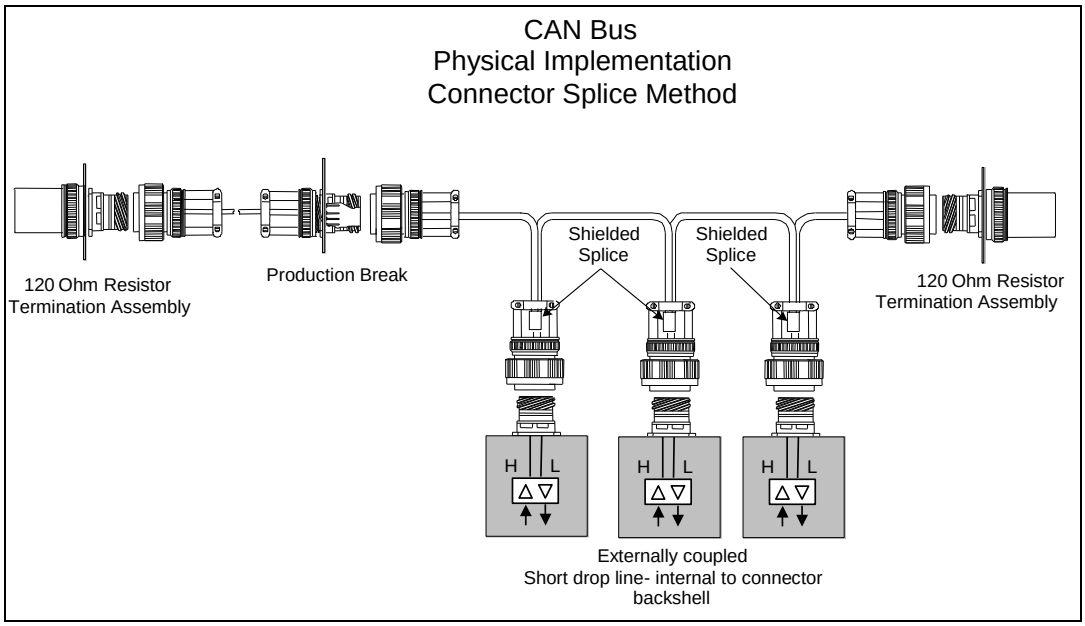


Figure 7-7 – Connector Splice Wire Method (Preferred Method)

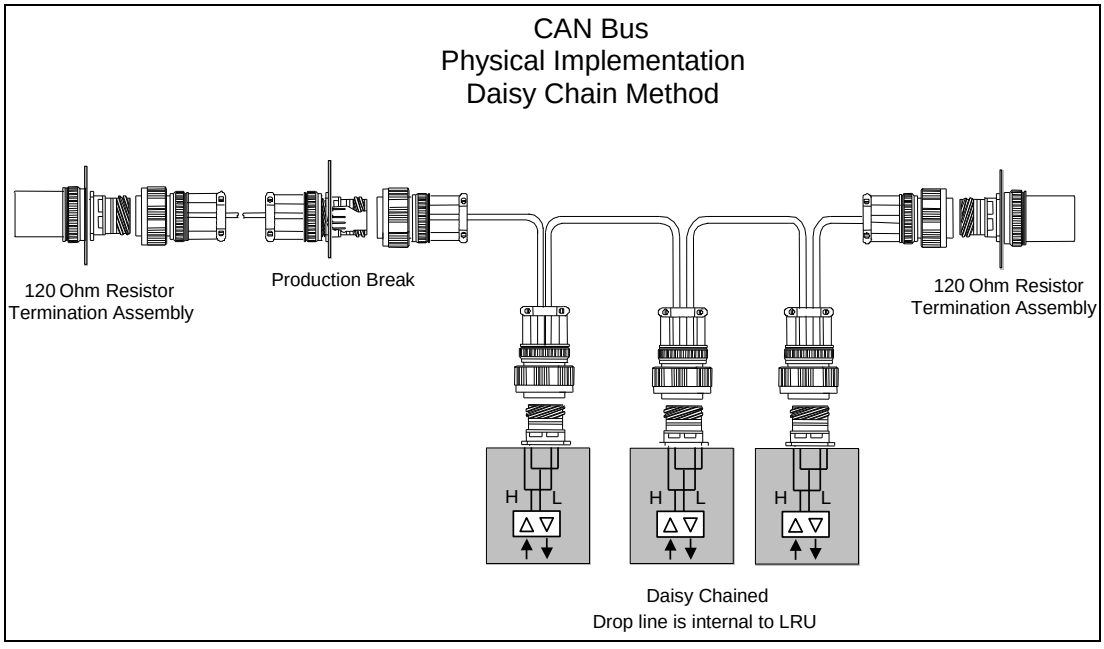


Figure 7-8 – Daisy Chain Wire Methods (To Be Avoided)

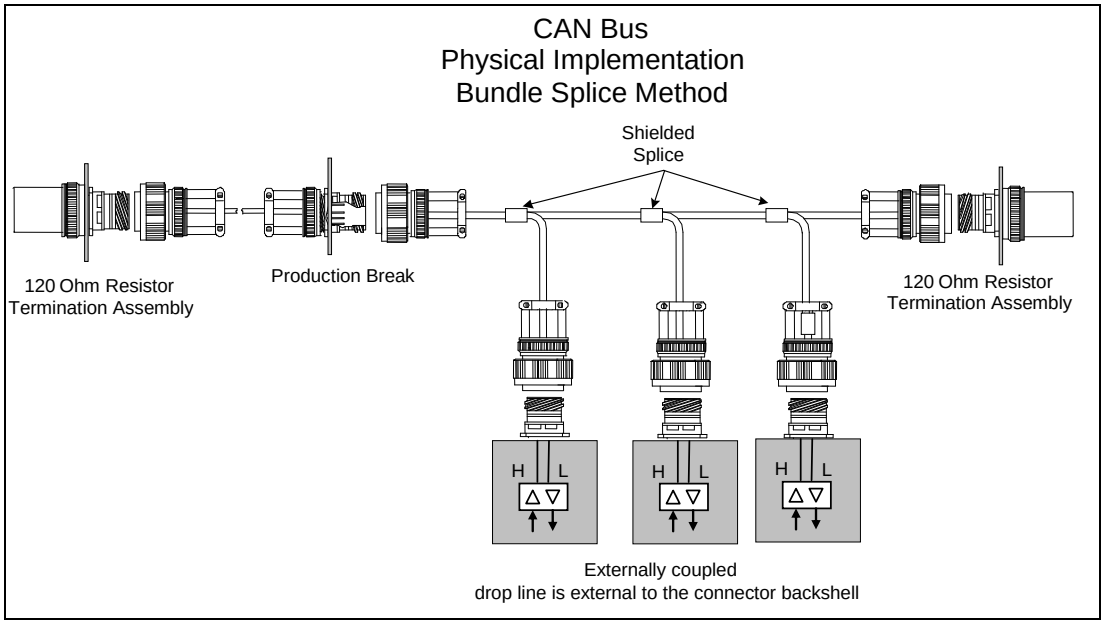


Figure 7-9 – Bundle Splice Wire Method

7.3.5.4.2 Wiring Methods to Be Avoided

COMMENTARY

It should be noted that anyplace where wire distribution is concentrated, such as in Wire Integration Panels (WIPs), there is a tendency to create situations where the wiring issues discussed below may be inadvertently implemented. As such, ARINC Specification 825 recommends avoiding the use of WIPs.

7.3.5.4.2.1 Star Configuration

The wiring implementation should avoid producing a star wiring configuration: no more than one stub or drop line should be connected to the same derivation point of the main line. A star wiring configuration may degrade signal characteristics of the CAN physical layer. Star wiring configurations may accidentally be produced by the use of splices or Wire Integration Panels (WIP). Recall that CAN must look electrically like a linear bus at all times.

Figure 7-10 illustrates a star wiring configuration. In this example, four nodes are spliced together at a common point.

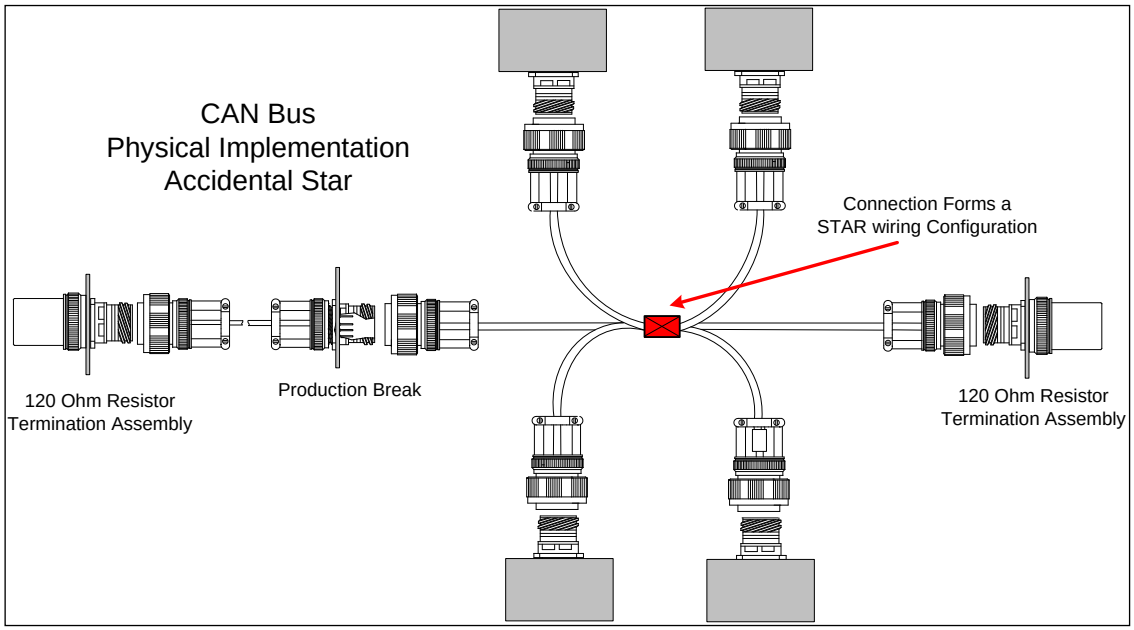


Figure 7-10 – Example CAN Star Wiring Configuration (To be avoided)

7.3.5.4.2.2 Long Stub Lengths

The CAN wire design needs to avoid long stub lengths that may sometimes happen as a result of the implementation. In systems that use Wire Integration Panels (WIP), for instance, long stub lengths may be accidentally created as a result of the WIP physical layout design and location of the CAN nodes.

Figure 7-11 shows an example implementation using a WIP. The example shows an area of caution where the stub lengths may become large due to physical layout of the WIP, location of the CAN node and WIP itself.

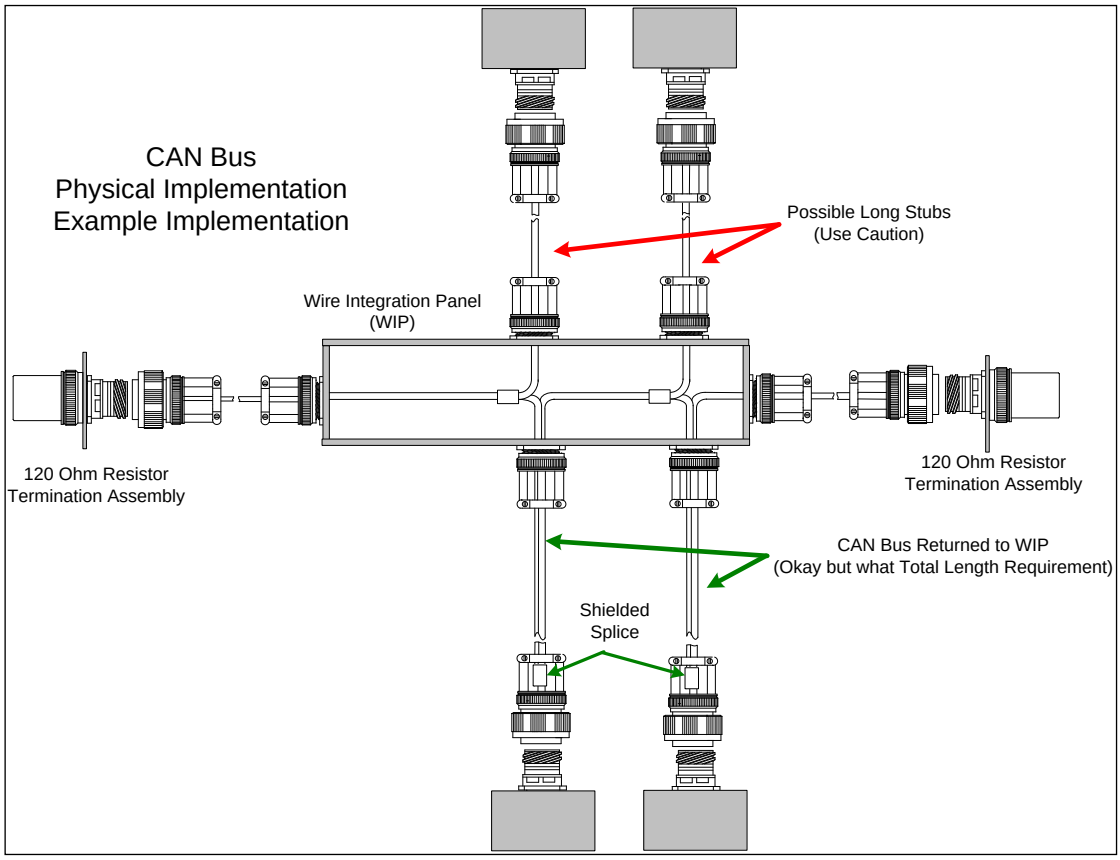


Figure 7-11 – Example of Bad Wire Integration Panel Design (To Be Avoided)

7.3.5.4.3 CAN Bus Length Design Estimate

The following method uses wire lengths to avoid discrepancies due to wire routing. A straight-line model between LRUs will yield a different length than the actual wire design model. A straight-line model may not account for:

- Wire raceway locations
- Wire feed through locations
- Separation channel routing
- Threat zone routing
- EME separation routing
- System separation rules
- Curvature of the airframe

The actual bus wire length is a sum of the following key design elements:

- Wire length between nodes including the distance between LRU connectors and internal CAN transceivers.
- It must be noted that routing in Wire Integration Panels must be included in the total bus length calculations.
- Wire length to and from the Wire Integration Panel(s) and a node (if applicable); wires returned to the Wire Integration Panel(s), from a node, also count in the total length calculation.

There are many factors that affect the actual wire length as described above. The initial system design may not have access to a final wire bundle routing. For this reason, adequate margin should be applied to the CAN physical parameters at the initial system design phase to accommodate changes due to any wire design routing or installation issues that may arise.

7.3.5.4.4 Shielding Inline Connectors

The shielding should be maintained through the inline connector(s), such as production break or disconnect brackets, by connecting the shields to the connector shielding backshells. EMI and lightning protection should be further enhanced, if necessary, with this method by providing an over shield for the wire bundle (braid sock).

7.3.5.4.5 Shield Termination at Connectors

Cable Shields should be terminated at the LRU connectors using near zero length shield grounds wherever possible. The following shield termination methods are listed in descending order of preference:

1. Shield termination to a grounded backshell
2. Shield termination utilizing a grounded shield sock, as illustrated in Figure 7-12
3. A pin through connection to ground for the shield

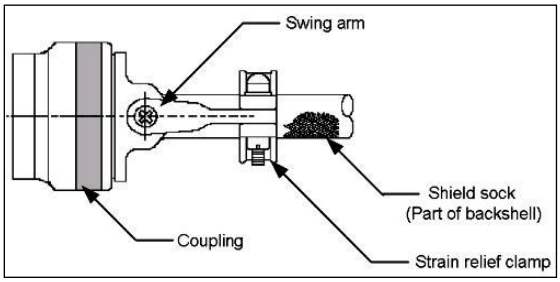


Figure 7-12 – Circular Connector Shield Example

7.3.5.5 Bit Timing

7.3.5.5.1 Bit Timing Requirements Explanation

1. The CAN controller shall be configured to have at least 8 TQ per bit.
   a. This applies to the arbitration phase for Classical CAN or the data phase for CAN FD (since the CAN FD arbitration bit has many more time quanta).

b. As an example, 8 TQ per bit allows for 2 for Phase_Seg 1, 2 for Phase_Seg 2, 1 for Sync, and 3 for Prop_Seg.

Comment: This reduces the impact of resolution error. Some implementations chose to have fewer than 8 TQ per data phase bit, such as 5 TQ. This may be acceptable but increases the chance for error and requires re-evaluation of the provided equations (see Appendix XX).

2. For CAN FD, CAN controllers shall configure the same TQ size, in nanoseconds, for Arbitration and Data phases.

Comment: This design decision is advocated by the industry to prevent phase errors from being introduced during the bit rate switch.

3. The SJW shall be at least 1/16th of a bit time in order to accommodate the worst-case oscillator tolerance error accumulation.
   a. No greater than min (Phase_Seg 1, Phase_Seg 2).

Comment: An SJW of 1/16th of the bit time is adequate to accommodate the 0.20% bit timing error allowed in the standard (combination of clock, bit setting, and aging/temperature effects).

4. IPT of the CAN controller shall be less than Phase Segment 2.
   a. For CAN FD, the Phase Segment 2 of concern is the Data Phase since the arbitration field will have many TQ for each segment.
5. For Classical CAN, the Sample Point location shall be between 75% and 87.5% of the bit time.

Comment: This is the setting allowed in ARINC 825-3. The accuracy of the sample point is less critical in Classical CAN than for CAN FD.

6. For CAN FD arbitration phase with a data rate increase of 2x or less and CAN FD data phase, one of the following requirements shall be used:
   a. The Sample Point location for all phases shall be between 75% and 82.5% of the bit time.
   b. The Sample Point location shall be the same for all nodes on the bus.

Comment: This is essential whenever the bit rate switch is used. The closer the sample point setting is between nodes, the better. CAN root clocks should be selected early on that enable these settings (and preferably 80%, see next requirement).

7. For CAN FD with data rate increase of more than 2 times the arbitration rate, the Sample Point location for arbitration shall be the same for all nodes on the bus.
   a. It is recommended to place the sample point at 80% of the bit time.

Comment: This is essential whenever a large change in bit rate is used because of the bit rate switch error. CAN root clocks

should be selected early on that enable setting the sample point to 80% of the bit time.

8. CAN FD rates should be a multiple of 1, 2, or 4 times the arbitration rate.

   Comment: In theory, the rate increase options are continuous. However, in the interest of interoperability and consistency, the standard presents the suggested multiples.

9. The CAN controller shall use Single Sampling Mode only.
   a. As opposed to triple sampling options on some CAN controllers.

   Comment: There is no exact definition for how to use triple sampling mode. To ensure consistency and determinism only single sampling point is allowed.

10. TDC should be used for CAN FD when available.
11. All CAN nodes on a bus should have the same CAN clock root frequency, if possible: either 20MHz, 40MHz, or 80MHz.

    Comment: This allows nodes to have consistent TQ sizing and use the same sample point locations. These values are the norm in the industry.

12. Classical CAN transceivers shall comply with ISO-11898-2: 2003 requirements.
13. CAN FD transceivers shall comply with ISO-11898-2: 2016 requirements and asymmetries.
14. CAN nodes with CAN FD data phase rates up to 2 Mbps shall use 2 Mbps or 5 Mbps capable transceivers, per ISO-11898-2: 2016.
15. CAN nodes with CAN FD data phase rates between 2 Mbps and 5 Mbps shall use 5 Mbps capable transceivers, per ISO-11898-2: 2016.
    a. Higher rates need more specialized equipment and should be assessed carefully using the equations provided.

7.3.5.5.2 Setting the Bit Timing

The method to assess CAN bit timing is time-based, defining the size (in seconds) of each bit and the relative fraction of a bit that is taken up by the various timing segments. The bit timing is driven by the physical layer and thus must be handled in terms of a physical layer unit, time. Notice how in the equations in Section 3.1.1.3.1, all of the variables are in terms of time (or can be specified as a percentage of a bit).

When implementing these specifications in a CAN controller, one must cross the analog to digital boundary. The resolution that each CAN controller has for a bit time (determined by the source oscillator frequency and the implementation of the prescaler) is irrelevant as long as, in the time domain, they all meet the required factors to interoperate on the physical layer. An example for either a Classical CAN controller or for a CAN FD controller data phase is provided below. The same method of translating the analog domain into the digital domain TQ holds true for CAN FD arbitration even though there are many more TQ per Arbitration bit.

Table 7-3 – Example Bit Timing Parameters

Parameter	Example Requirement	Controller A	Controller B
Oscillator Frequency		20 MHz	40 MHz

Oscillator Tolerance		100 ppm	500 ppm
Prescaler		5	8
Bit Time	2µs +/- 0.2% including oscillator tolerance (500 kbps)	2µs +/- 0.01%	2µs +/- 0.05%
TQ per bit	Greater than or equal to 8 (to minimize resolution error)	8	10
Sample Point	75% - 82.5%	75% (6 TQ)	80% (8 TQ)
SJW	At least 1/16 bit (max = PS2=1-SP)	12.5% (1 TQ)	20% (2 TQ)

A set of bit time requirements could be given in terms of Time Quanta. This ensures sameness but can only be reasonably achieved if the requirements also force the same CAN controller, oscillator, and prescaler settings on all nodes. This is an option that is available but not practical in all cases.

7.4 Data Link Layer

7.4.1 CAN Controller Types

When selecting a CAN controller, the designer should be aware of three generic types of controllers.

- Standalone
- Integrated with a microcontroller
- Intellectual Property (IP) module

When selecting a CAN controller, the designer should be aware that the message buffer management concept varies widely.

7.4.2 CAN Controller Interface

Typical host CPU to CAN controller interfaces are either interrupt driven and/or polled at fixed rates. Depending upon your application, either or both methods may be used.

Typically, CAN controllers offer message identifier acceptance filtering. Use of this feature may reduce the number of messages that need to be processed. This relaxes the throughput requirements for the CPU.

If the interface is interrupt driven, then ensure the CPU is able to process CAN controller interrupts at the maximum anticipated rate plus a margin for jitter. Also, protect against abnormal message rates or other anomalies that may cause undesired behavior. Additionally, ensure exception handlers are present for any error interrupts that may be generated by the CAN controller.

If the interface is polled, ensure the poll rate is consistent with anticipated message rates. If every message needs to be received, then the poll rate would be greater than the message rate of the bus.

7.4.3 Start-Up Sequence

Start-up sequences are categorized as either active or passive. In general, the Passive Start-up Sequence is more complex than the Active Start-up Sequence. The Passive Start-up Sequence implies architectural design constraints, bus behaviors and failure modes, of which the system designer needs to be aware. The following sections detail the recommended procedures for each type of Start-up

Sequence. For additional information regarding potential safety issues, reference Section 7.9.

COMMENTARY

One should be aware that the passive start-up sequence relies on a master node to start all network communication, thereby creating a potential single point of failure. The passive start-up sequence should be limited to systems where a total loss of communication is non-critical. Alternatively, if multiple master nodes are used to avoid a single point of failure condition, these multiple master nodes will require well designed and tested control transfer mechanisms to avoid conflicts.

An active start-up sequence has four parts described below:

1. Stabilization of electronics in each node:
   At power up, the electronics should be stabilized before any initialization of the CAN interface is performed. When the node is powered-up, the first CAN frame should not be sent before the electronics are fully configured. Otherwise, inconsistent frames could be sent. Additionally, the bus could be set to a continuous dominant level causing other nodes to bus-off. Specific transceiver silent-mode (or inhibit) should be used for this purpose.
2. Node is ready to process message:
   Before one enables the CAN controller to acknowledge frames, the node must be ready to process received messages.
3. First powered node communicates:
   At power-up, the first node completing steps 1 and 2 above will communicate alone on the bus sending its first frame. This frame will not be acknowledged leading to an ACK error. To understand the error mechanism at start-up, see schematics below. When using FIFO buffers, the first powered node could go into a FIFO overload.
4. Start-up sequence of the bus communication is completed:
   When the other nodes come up on the bus, initialization of the system may be performed.
   At this point, system level initialization, normal operation messages, and self/health test frames may be sent and must be taken into account for the bandwidth calculation.

COMMENTARY

FIFO at start-up:

During startup, for systems that have opted to utilize a FIFO as part of the CAN output a second message should not be placed into the queue until the previous message transmitted on the bus has been acknowledged. Additionally, nodes that are powering up should not enable their CAN receiver until they are ready to process received messages. The rationale behind this recommendation is to reduce the potential for any node flooding the bus with data during startup and causing a denial of service for other nodes on the bus.

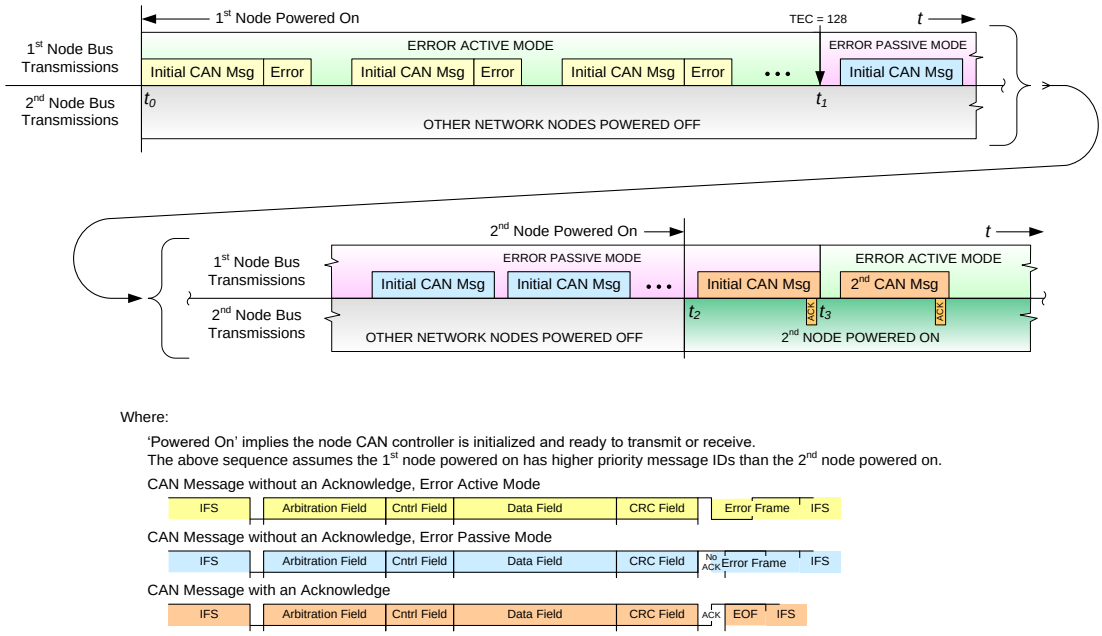
Consider the following: A system with multiple nodes that are normally powered off but connected to a gateway device. Since the nodes in this system are not powered they are unable to provide acknowledgement to any transmitted CAN message. However, it is reasonable to assume that the gateway node is powered and attempting to transmit data on the bus. The first message queued for transmission would essentially be “stuck” in the transmit buffer of the CAN controller awaiting an acknowledgement.

Meanwhile, the internal FIFO of the gateway LRU is being filled with messages that are not able to transmit. Depending on the size of the queues, it is possible that the amount of data stored could represent a significant amount of processing for any node that should power up. In this event that CAN controller of the gateway would attempt to clear the queue at full line rate, effectively sending a tsunami of CAN data across the bus that would swamp the receiving unit’s CAN controller. It is not a stretch to assume that, for the aviation industry, any system that is usually not powered is a safety related system. As such, when the system is activated it would typically be for some emergency event.

At this time, it is reckless to overwhelm the unit with data that most likely is not relevant to the current airplane situation.

Additionally, the data that is transmitted on the bus must ultimately not pose a safety concern. If data is allowed to sit in a register or transmit buffer for a prolonged period of time, it may no longer reflect the current state of the airplane but instead represent an airplane configuration from some indeterminate point in time.

7.0 DESIGN GUIDELINES



Where:

'Powered On' implies the node CAN controller is initialized and ready to transmit or receive.
The above sequence assumes the 1st node powered on has higher priority message IDs than the 2nd node powered on.

CAN Message without an Acknowledge, Error Active Mode

CAN Message without an Acknowledge, Error Passive Mode

CAN Message with an Acknowledge

Figure 7-13 – First Node Communication

A Passive start-up has the following steps:

1. Electronics stabilization of each node:
2. Same as active start-up sequence.
3. Node is ready to process message:
4. Same as active start-up sequence.
5. Master node communicates:
6. At power-up, the nodes completing the two steps above will wait to receive a frame from the master. In the event the master is the first node to complete the steps above, the master will communicate alone on the bus sending its first frame. If the master is first to transmit, it will follow the same sequence defined for active start-up described previously.
7. Start-up sequence of the bus communication is completed:
8. Same as active start-up sequence.

Figure 7-13 depicts the Active Start-Up Sequence where:

t_0: The first node on the network is; powered on, stabilized, CAN controller initialized.

t_1: The Master Node Powered On and sends the first message.

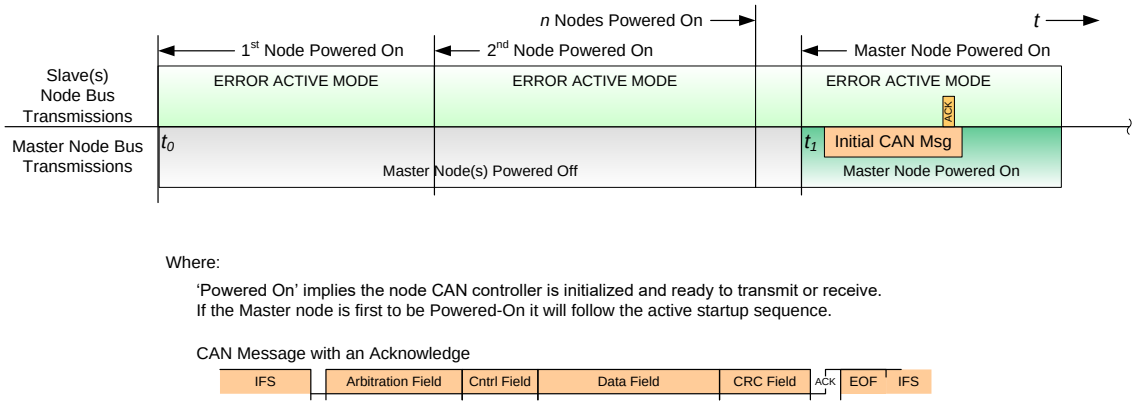


Figure 7-14 – Passive Start-up

7.4.4 Bus Off Management

CAN Messages may be corrupted during transmission. The CAN protocol contains mechanisms to detect faults on every transmitted and received message. Error detection is handled by the CAN controller hardware. When a node considers itself as being faulty, it enters the Bus Off state and disconnects itself from the network. This mechanism is implemented in the CAN protocol state machine.

The bus specification (See APPENDIX F) must set forth rules how to handle Bus Off conditions, such as reattaching or resetting the CAN interface. The number of allowed re-attachments should be defined in accordance with the system requirements.

7.5 Communication

7.5.1 LCC Assignment and Usage

Logical Communication Channels (LCCs) have been introduced with ARINC 825 in order to quickly distinguish message priority, separate independent levels of communication from each other, and make it easier to listen to specific data on the bus through acceptance filtering. The LCC field is the highest priority field in the identifier. The priority of the messages is associated with the assigned channel.

The designer should limit the assignment of messages to communication channels EEC, NOC, DMC, NSC, and TMC whenever possible. UDC and FMC have been defined for purposes only where the use of the other channels is impossible without violating ARINC Specification 825. Reserved channels are there for future enhancements of ARINC Specification 825 and must not be used under any circumstances.

It is recommended that nodes report aircraft and system critical conditions or extraordinary system defined events through the exception event messages over the EEC in an appropriate manner. The designer should be aware that these messages will receive highest priority for transmission. It is extremely important to limit and bound the frequency at which exception event messages are transmitted

by a particular node to avoid excessive bus load. Consequently, the occurrence of multiple or recurrent failures must be planned for. The designer must ensure that such scenarios do not lead to a burst or continuous flow of messages over the EEC, adversely affecting the flow of other messages. Exception event messages transmitted over the ECC should be the exception. EEC traffic is to be included in the bandwidth management calculation.

Peer-to-peer communication is not possible over the Emergency Event Channel (EEC) and the Normal Operation Channel (NOC). The EEC and NOC are dedicated to the classic, broadcast type message transfer during operation. The Node Service Channel (NSC) and the Test and Maintenance Channel (TMC) support only peer-to-peer communications. Note that ARINC Specification 825 allows operation over all channels to be in effect at the same time. It is the system designers' responsibility, however, to manage parallel operations over the independent channels such as the NOC and TMC.

7.5.2 Node Identifier Assignment

The Node Identifier (NID) used for peer-to-peer addressing comprises of Function Identifier (FID) and Server Identifier (SID). The idea behind the FID/SID combination is that a specific node (server) is able to be used in different applications (functions) and still be uniquely addressed through the peer-to-peer communication mechanism. Likewise, this concept allows multiple virtual servers within one physical entity by changing the SID field and makes sure that groups of nodes belonging to the same function may be identified through the FID part of the NID.

The NIDs used by each ARINC Specification 825 node must be unique within the entire aircraft installation. If multiple units of the same type are used within the same function, it is the system designer's responsibility to further subdivide the 9-bit SID to provide the required structure.

7.5.3 Communication Profile Design

The ARINC Specification 825 communication profile specifies the network traffic for a given network node. The communication profile database combines the communication profiles of all nodes in a given network and constitutes the basis for the interpretation of all data in the network.

To support the communication profile development, an example DOC assignment list covering some aircraft data is provided in APPENDIX D. The Appendix covers a range of parameters common to all aircraft in general and defines the DOC as well as the unit and data type for each of them.

In the example, each parameter is assigned a unique DOC and the FLOAT data type. Although ARINC Specification 825 allows multiple parameters in a single message, this example does not demonstrate this capability. The example may serve as a basis to create unique profiles.

Note: ARINC Specification 825 nodes may be interrogated for their communication profile ID and sub-ID through the mandatory Node Identification Service.

7.5.4 Node Health Status Message

The mandatory Periodic Health Status Message (PHSM) (see Section 5.4) has been introduced in ARINC Specification 825 for two reasons:

1. Locating network interface problems is difficult for CAN due to the fact that multiple stations participate in the error processing inherent to the CAN protocol at the same time. The PHSMs deliver information about the internal error status of each individual unit on the bus. In many cases, combining this information from all units and evaluating it allows to identify and/or locate the source of erratic behavior.
2. Failures in CAN-based systems may remain undetected if units on the network transmit messages with a very low repetition rate or event-driven messages only. These units remain silent during extended periods of time, increasing the probability of undetected failures and thereby reducing system integrity. The PHSM ensures that all units in the network signal their presence in regular intervals. Monitoring the PHSM traffic provides this information for the entire network.

It is the system designer's responsibility to specify how received PHSMs are to be processed. A typical application is to program a central maintenance computer to scan received PHSMs, extract those messages that indicate errors and store them together with a corresponding time stamp for long-term health monitoring.

7.5.5 Design of Node Services

Node services are client/server type interactions between two (or more) nodes in an ARINC Specification 825 network. In addition to the node services already defined in ARINC Specification 825, the system designer may create other node services according to the specific needs of the system. The only restrictions that apply are that the peer-to-peer communication mechanism is retained and that the additional bus loading must be included in the bandwidth management calculation.

Before implementing a node service, the designer may choose one-to-one or one-to-many addressing. If one-to-one addressing is chosen, the communication may be either connectionless or connection-oriented. However, if one-to-many addressing is chosen, only connectionless communication may be implemented.

Node services are always initiated by the client sending a node service request message to one or more servers (nodes). Depending on the selected NID of the request message, the service request addresses one of the following; a single node, all nodes within a function, or all nodes in the entire system (aircraft). Using connection-oriented communication, a response from the server is required which may either be a single acknowledge message or a complex response delivering more information. Using connectionless communication, the request is processed "silently" by the server without sending a response.

Having implemented the node services, they must be published to others using the network. The node service description table format in Section 5.5.2 from ARINC Specification 825 is recommended to describe these node services in a uniform and known way.

7.5.6 Standardization and Interoperability

In an effort to better support interoperability of ARINC 825 developments, this standard looks to utilize standardized definitions for key aspects. The following attributes should be considered when coordinating the system design between the designer, implementer, and/or verifier.

7.5.6.1 Axis Definitions and Sign Convention

To further ensure interoperability, this standard uses SI units according to ISO 1000 and axis systems as set forth in ISO 1151 (EN 9300). The axis systems provide the sign conventions for geodetic and aerodynamic data common to all aircraft. The basic definitions are as follows:

Symbol	Definition
x, y, z	Body axis system
x_a, y_a, z_a	Airpath axis system
k_1, k_2, k_3	Flightpath axis system
x_e, y_e, z_e	Intermediate axis system
V	Velocity vector
L	Lift vector
Y_e	Transverse force vector
D_e	Drag vector
V_w	Wind vector
α	Angle of attack
β	Angle of sideslip
Θ	Pitch angle
Φ	Roll angle
Ψ	Heading angle

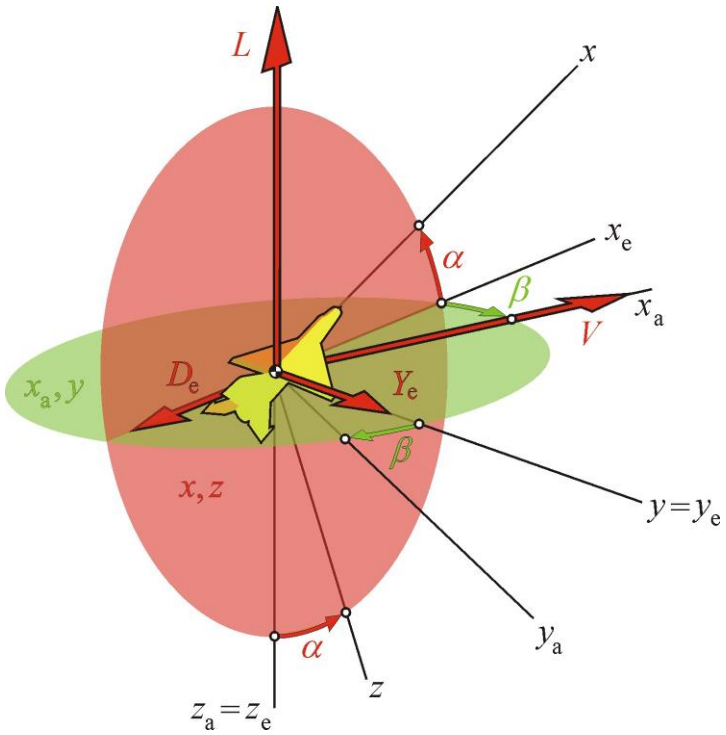


Figure 7-15 – ISO 1151 (EN9300) Aerodynamic Body-Fixed Coordinate System

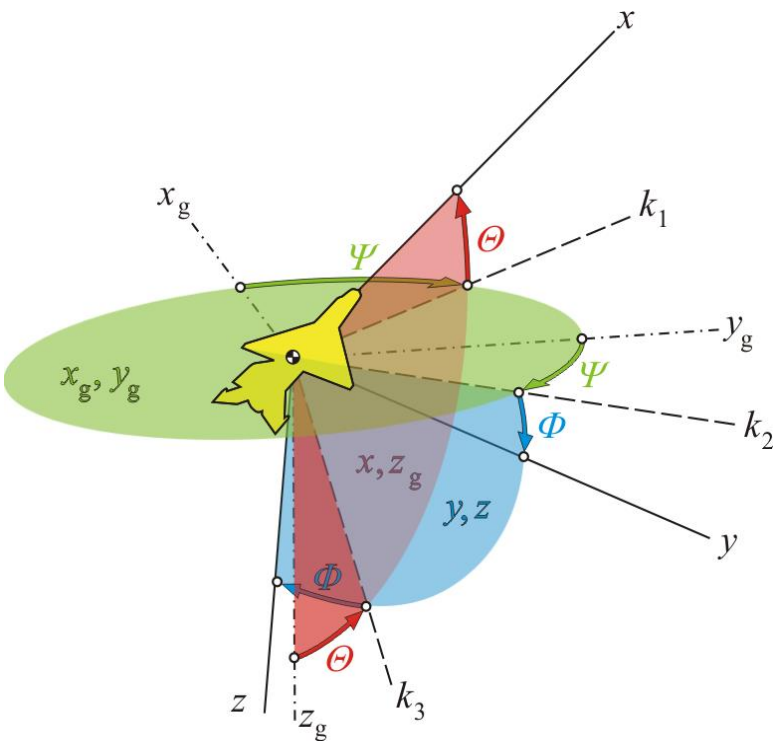


Figure 7-16 – ISO 1151 (EN9300) Earth and Body - Fixed Coordinate System

7.5.6.2 Physical Units

To support interoperability, an ARINC 825 profile should use SI units, SI derived units, or units accepted for use with SI according to the ARINC 825 unit code list.

Table 7-4 – ARINC 825 Unit Code List

Code	Base Quantity	Name	Unit Symbol	Unit Acronym
0	No Physical Unit	n.a.	n.a.	nounit
1	Length	meter	m	m
2	Length	millimeter	mm	mm
3	Length	kilometer	km	km
4	Mass	kilogram	kg	kg
5	Mass	ton	t	t
6	Mass	gram	gm	gm
7	Volume	liter	l	l
8	Time	second	s	s
9	Time	minute	min	min
10	Time	hour	h	h
11	Time	day	d	d
12	Time	year	y	y
13	Current	ampere	A	A
14	Temperature	kelvin	K	K
15	Luminous Intensity	candela	cd	cd
16	Area	square meter	m^2	m2
17	Volume	cubic meter	m^3	m3
18	Liquid Flow	liter/minute	l/min	l/min

Code	Base Quantity	Name	Unit Symbol	Unit Acronym
19	Speed, Velocity	meter per second	m/s	m/s
20	Rotational Speed	reciprocal second	s^{-1}	1/s
21	Mass Flow	kilogram per hour	kg/h	kg/h
22	Acceleration	meter per second squared	m/s^2	m/s^2
23	Wave Number	reciprocal meter	m^{-1}	1/m
24	Mass Density	kilogram per cubic meter	kg/m^3	kg/m^3
25	Specific Volume	cubic meter per kilogram	m^3/kg	m^3/kg
26	Current Density	ampere per square meter	A/m^2	A/m^2
27	Magnetic Field Strength	ampere per meter	A/m	A/m
28	Luminance	candela per square meter	cd/m^2	cd/m^2
29	Angle	radian	rad	rad
30	Frequency	hertz	Hz	Hz
31	Force	newton	N	N
32	Pressure, Stress	pascal	Pa	Pa
33	Energy, Work	joule	J	J
34	Power	watt	W	W
35	Electric Charge	coulomb	C	C
36	Electric Potential Difference	volt	V	V
37	Capacitance	farad	F	F
38	Electric Resistance	ohm	Ω	ohm
39	Electric Conductance	siemens	S	S
40	Magnetic Flux	weber	Wb	Wb
41	Magnetic Flux Density	tesla	T	T
42	Inductance	henry	H	H
43	Temperature	degrees Celsius	^oC	degC
44	Angle	degree	^{o} (deg)	deg
45	Moment of Force	newton meter	Nm	Nm
46	Surface Tension	newton per meter	N/m	N/m
47	Angular Velocity	radian per second	rad/s	rad/s
48	Angular Acceleration	radian per second squared	rad/s^2	rad/s^2
49	Heat Flux Density	watt per square meter	W/m^2	W/m^2
50	Heat Capacity	joule per kelvin	J/K	J/K
51	Thermal Conductivity	watt per meter kelvin	W/mK	W/mK
52	Energy Density	joule per cubic meter	J/m^3	J/m^3
53	Electric Field Strength	volt per meter	V/m	V/m
54-99	Reserved			
100-999	User-defined			

User-defined physical units (Unit codes 100-999) should only be used for units that are not represented in the list.

7.5.7 Parametric Functional Status

ARINC Specification 825 introduced Message level Functional Status in Section 5.2.2.1, One-to-Many Identifier Structure, in order to be compatible with common avionic networks. However, this implementation indicates that all parametric data in the message is related with a common validity. There are designs where the parameters in the data frame may have dissimilar validity. In this instance, a parameter level functional status must be applied. When the parametric functional status is needed, the Functional Status Bit (FSB) is cleared (value – 0) to remain compatible with other CAN messages on the bus (see Figure 7-17).

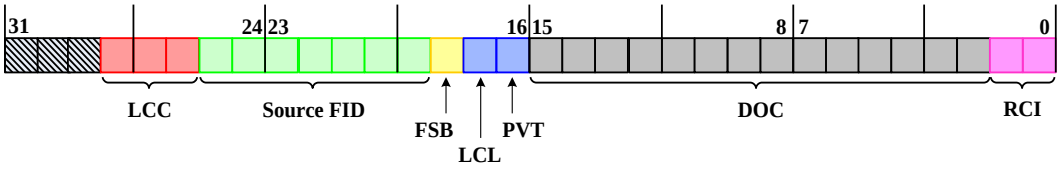


Figure 7-17 – ARINC 825 One to Many Identifier

Parameter Validity Status (PVS) is a two-bit enumeration, which describes validity of the associated parameter(s). The PVS is defined in the table below. In contrast to message-level validity, the PVS enables the transmitting system to individually flag a parameter or a subset of parameters within a given payload.

Table 7-5 – ARINC 825 Parameter Validity Status

Enumeration	Value (bits)
Failure Warning	00
No Computed Data	01
Functional Test	10
Normal Operation	11

The PVS may be implemented in two ways, either adjacent to the parameter or grouped at the front of the data field.

Adjacent PVS

Adjacent to the parameter that is directly related to the PVS for all parameters in the data frame.

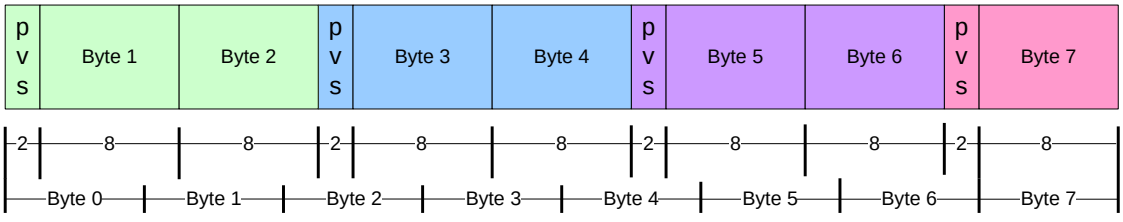


Figure 7-18 – Adjacent PVS

In this example, the PVS is a two-bit field preceding and adjacent to the covered parameter in the data field.

Grouped PVS

Grouped at the beginning of the data frame the PVS resembles the ARINC Specification 664, Part 7, Functional Status Set (FSS).

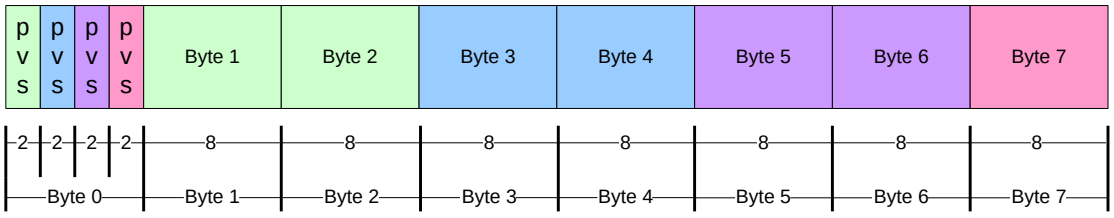


Figure 7-19 – Grouped PVS

In this example, the two-bit PVS fields are grouped into a single byte at the beginning of the data frame, although this is for illustration purposes only. This standard does not limit the size of the PVS fields to one byte; if there is a need, a designer may expand the PVS as needed to effectively communicate the validity of the embedded data in the CAN data frame.

7.6 Bus Load Management

For CAN, effective message management is critical to the successful system integration. This management is of even greater importance in mixed supplier systems or systems that employ a gateway.

7.6.1 Data Throughput Capabilities of CAN

For the same data rate for arbitration and data phase, the method used to calculate data throughput is determined by the maximum data frame size (512 bits), Inter-Frame Space (IFS 3 bits) plus the maximum amount for bit stuffing (145 bits).

$$CANFramesPerSecond\left[\frac{1}{s}\right] = \frac{DataRate\left[\frac{bits}{s}\right]}{Overhead[bits] + Payload[bits] + Stuff[bits] + IFS[bits]}$$

Where the Overhead is 76 bits and Payload is between 0 and 512 bits.

Table 7-6 provides an estimate of the data throughput of CAN for the usable maximum bus load as recommended by ARINC Specification 825:

Table 7-6 – CAN Frames Per Second (Worst Case)

					Data rate (kbit/s)				
Bus Load	Overhead Bits	Payload Bits	Stuff Bits (max)	IFS Bits	83.333	125	250	500	1000
100% (Unusable)	76	512	145	3	113	170	340	679	1358
50% (Recommended)	76	512	145	3	57	85	170	340	679

Table 7-7 illustrates the CAN data throughput at various data rates in bits per second (based on 50% bus load for comparison).

Table 7-7 – CAN Payload Throughput (Bits Per Second)

Data Rate (kbit/s)					
Payload Size (Bits)	**83.333**	**125**	**250**	**500**	**1000**
512	28 985	43 478	86 957	173 913	347 826

The CAN throughput is shared amongst all nodes on the segment. Therefore, the system designer is advised to allocate throughput budgets to the key CAN system features.

For different data rates between arbitration and data phase, the formulas are the following:

$$CANFramesPerSecond\left[\frac{1}{s}\right] = \frac{DataRate\left[\frac{bits}{s}\right]}{MessageLength[bits]}$$

$$MessageLength\,[bit] = MessageArbitrationLength\ \mathrm{X}\left(\frac{Datarate\ DATA}{Datarate\ Arbitration}\right) + MessageDataLength$$

With:

$$MessageArbitrationLength\ [bit] = OverHead\,[bit] + Stuff[bit] + IFS\,[bit]$$

.

This formula results in a fixed value:

MessageArbitrationLength [bit]= 34 +9+3= 56 bits

$$MessageDataLength\ [bit] = OverHead\,[bit] + Data\ Payload + Stuff[bit] + CRC\ [bit]]$$

Notes: CRC length takes into account the Stuff bit counter, parity bit, the CRC delimiter, and the specific stuff bit.
Depending of the payload the size of the CRC, for the payload ≤ 128, the CRC length is 28 bits and 33 bits for the payload > 128.
The Overhead is 6 bits and the Payload is between 0 and 512 bits.

If the Arbitration phase is 1 Mbit/s and the Data phase 1Mbit/s to 5Mbit/s, Table 7-8 provides an estimate of the data throughput of CAN for the usable maximum bus load as recommended by ARINC Specification 825:

Table 7-8 – CAN Frames Per Second (Worst Case)

						Data rate Data phase Mbit/s				
	Arbitration Phase	Data Phase								
Bus Load		Over-head Bits	Payload Bits	Stuff Bits	CRC Bits	**1 Mbit/s**	**2 Mbit/s**	**3 Mbit/s**	**4 Mbit/s**	**5 Mbit/s**
100% (Unusable)	56	6	512	129	33	1359	2525	3538	4425	5208
50% (Recommended)	56	6	512	129	33	679	1263	1769	2212	2604

Table 7-9 illustrates the CAN data throughput at various data rates in bits per second (based on 50% bus load for comparison).

Table 7-9 – CAN Payload Throughput (Bits Per Second)

Data Rate (kbit/s)					
Payload Size (Bits)	**1 Mbit/s**	**2 Mbit/s**	**3 Mbit/s**	**4 Mbit/s**	**5 Mbit/s**
512	347 826	646 465	905 660	1 132 743	1 333 333

The CAN throughput is shared amongst all nodes on the segment. Therefore, the system designer is advised to allocate throughput budgets to the key CAN system feature.

7.6.1.1 Periodic and Event Driven Data

There are really three possible forms of data in a system: rate based data, state based data, and block data. Rate based data is typically periodic while state based data is typically aperiodic in nature. This form of data typically resides in the Normal Operation Channel (NOC), but may be communicated on the Directed Message Channel (DMC) as well. Block data is aperiodic and typically transferred using peer-to-peer communication.

1. Rate based data is typically associated with a change in a parameters value over time. The value of a parameter over a time interval may be used in transfer equations for control and indication functions of the system. Examples of this data would be velocity or acceleration data used in auto flight equations.
2. State based data is typically associated with a change in a parameters value at a specific time. The rising edge or falling edge of a parameters state at a specific point in time may invoke certain key features or mode transitions in the system. Typical examples of this data type would be Air/Ground transition parameters.
3. Block data is typically used for file transfers of large blocks of data.

The amount of rate based, state based, and block data on a CAN segment will impact the overall performance and possibly system architecture. It is therefore important for system designers to scale bandwidth allocation according to the anticipated CAN node utilization of each data type.

CAN message update rates (also referred to as refresh rates) will influence system latency and performance requirements.

COMMENTARY

Message rates: System designers should specify their message refresh and transmit rates logically and consistently with the intended function. For instance, there is no purpose in sending data out faster than is required by the end user(s) thus tying up valuable bandwidth needlessly.

7.6.1.2 Transmission Intervals

System designers should minimize any tendency to burst data onto the bus. It is better to have data transmitted as periodic data at regularly scheduled intervals rather than stack messages up in a queue and then schedule the queue to be transmitted. Transmitting an entire queue at once consumes instantaneous bandwidth, which may cause bus contention and temporary denial of service to other CAN nodes. It also creates performance issues for other CAN nodes in that they must now be capable of handling the burst without error which in turn may drive CAN node design issues (i.e., more memory).

7.6.2 Bandwidth Management

An essential characteristic of all flight safety critical systems is that their behavior has to be precisely defined, analyzed, and tested to meet formal certification requirements. This characteristic is often misinterpreted as timing determinism but is in fact predictability. The degree of precision required for timing is specific to each application and has to be quantified by system analysis. The ultimate target to be reached, however, is that it may be demonstrated to certification authorities that a safety critical system behaves predictably under foreseeable circumstances. Using CAN, this predictability may be achieved.

ARINC Specification 825 sets forth a concept of managing the available bandwidth of a multi-drop CAN network to ensure predictable behavior for one-to-many and peer-to-peer communication. This management concept is based on a limitation of the number of CAN messages that any node in the network may transmit within a “minor time frame.” The minor time frame is defined during initial system design. The maximum number of messages transmitted within one minor time frame may differ from node to node and contain growth potential if granted by system design.

It is crucial to the bandwidth management concept that every node in the network adheres to its transmission schedule at all times when generating network traffic. On the other hand, it is neither required nor prohibited that nodes in the network synchronize to other nodes concerning their message transmission order or transmission times. Error frames may lead to unpredictable behavior if the bandwidth is consumed by error frames resulting from faults of the network or the nodes attached to it. However, ARINC Specification 825 limits the bandwidth usage to 50% of the maximum bandwidth so that unpredictability is mitigated. While this management concept requires margins and does not optimize network bandwidth usage, it provides a safe and straightforward approach to build certifiable (predictable) systems. Applying the bandwidth management concept, it may be demonstrated that an ARINC Specification 825 compliant network behaves predictably.

Shown in Figure 7-20 is the transmission schedule of an ARINC Specification 825 compliant network with two nodes transmitting their messages asynchronously, in

alternating order and at random times within their minor time frames (worst case scenario). This example utilizes 50% of the maximum bandwidth.

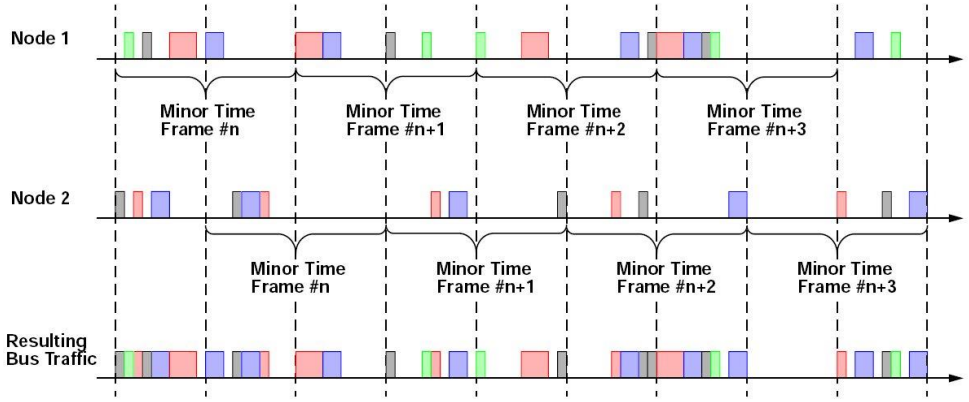


Figure 7-20 – ARINC 825 Transmission Schedule Example

Using the ARINC Specification 825 bandwidth management concept, no message in this transmission schedule has a latency exceeding three times the duration of one minor time frame plus the duration of the longest message. The ARINC Specification 825 bandwidth management reduces the impact of message priority because the nodes on the network are required to meter their message transmissions.

Local oscillator tolerances and lack of time synchronization between the nodes will result in minor time frames drifting away from each other. This does not adversely affect message latencies as long as the duration of the minor time frame in all nodes matches closely.

To ensure predictability, all aperiodic messages must be included in the bandwidth management calculations.

To support the implementation of the ARINC Specification 825 bandwidth management concept, the node software should be able to handle CAN message transmission/reception in the background. This implementation decouples the node application software from the timing requirements of the CAN interface. In addition, all nodes in the network should transmit their messages in descending message priority order (highest priority messages first) in order to minimize jitter due to priority inversion.

While sacrificing some bandwidth, the ARINC Specification 825 bandwidth management concept provides an effective approach based on simple math as described in Section 5.6. The bandwidth management concept also ensures adequate flexibility for increasing network traffic during the lifetime of the system if growth potential is planned. As an example, system design will allow nodes to be integrated into the network without affecting the existing nodes.

Furthermore, the predictable behavior enforced by the ARINC Specification 825 bandwidth management concept allows systems with different criticality levels to coexist on the same network.

7.6.2.1 Bandwidth and Bus Contention

The relationship between bandwidth and bus contention is such that as bandwidth utilization increases then bus contention similarly increases which also increases

the probability of lower priority messages being denied access to the bus for extended periods of time.

7.6.3 Message Optimization

7.6.3.1 Message Consolidation

In a system that employs many CAN nodes on a common bus, bandwidth may be saved by eliminating duplicate signal data on the bus. The objective of the bus integrator is to save bandwidth by avoiding duplication of messages and examining messages for potential consolidation. It is up to the bus integrator to ensure the CAN Interface Control Document (ICD) is up to date, accurate, and the bus is not over allocated.

Some areas to consider are:

- Data types on the bus
  - Rate (periodic data)
  - State (event driven)
- Block (file transfer)
- Refresh rates
- Functional criticality
- Message priority
- Spare capacity
  - Future growth
  - Development margin

7.6.3.2 CAN Identifier Prioritization

CAN is built upon a multi-master half duplex asynchronous serial data transmission medium. CAN nodes contend for media access based upon a priority scheme built into the CAN Identifier where the lowest CAN Identifier value represents the highest priority on the bus.

A CAN Identifier of “0” has the highest priority. In an integrated environment where many nodes share the same bus, the tendency is for each unique node type to prioritize its data as the most important on the bus. From a system architecture perspective, this is clearly not acceptable. Typically, the System Integrator must assign and manage CAN Identifier assignments in order to avoid conflicts and priority issues in the integrated environment.

ARINC Specification 825 provides mechanisms to assist designers in prioritizing messages based upon the CAN Identifier structure. The purpose of this guideline is to stress the coordination and control of CAN Identifiers to ensure an appropriate message priority scheme is implemented which is consistent with the criticality of the data. Wherever possible, higher criticality data should be given the highest CAN Identifier priority.

7.6.4 Transmit Buffers

Preferred processes for implementing CAN software are as follows:

1. Any CAN node is able to send out a stream of scheduled messages without releasing the bus between the two messages. Such nodes arbitrate for the

bus immediately after sending the previous message and only release the bus in case of lost arbitration.

2. The internal message queue within any CAN node is organized such that the highest priority message is sent out first, if more than one message is ready to be sent.

The above behavior cannot be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message is sent. This loading process lasts a finite amount of time and has to be completed within the Inter-Frame Space (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU react with short latencies to the transmit interrupt.

A double buffer scheme de-couples the reloading of the transmit buffer from the actual message sending and, as such, reduces the reactiveness requirements on the CPU. Problems may arise if the sending of a message is finished while the CPU re-loads the second buffer. No buffer would then be ready for transmission and the bus would be released.

7.6.5 Determinism and Timing Behavior

This section defines the timing information essential to describe the end-to-end timing behavior of distributed systems interconnected by CAN. The timing concept constitutes the basis for performing determinism analysis on selected portions or the entire system.

Aircraft systems are typically required to demonstrate predictable operation. These aircraft systems have top-level functional requirements that events must happen in a timely manner. The timing requirements may range from long duration (for example, data loading should be finished within 15 minutes), or may be fractions of a second (for example, a fast control loop, less than few milliseconds).

Signals may be produced and transported on different media (e.g., CAN, ARINC 664, analog) and through gateways. All of these elements (i.e., node, gateway, buses, etc.) must be considered when determining compliance of timing requirements.

7.6.5.1 Determinism

A system is deterministic when the end-to-end timing behavior is known and bounded. The signal, expected in a bounded slot, is effectively available during this slot: the signal has a guaranteed validity. A network is deterministic when the frame is bounded for transmission and reception.

The Timing Concept defined in the following sections is a toolbox to be used to demonstrate determinism. Figure 7-21 provides a graphical representation of the timing concept.

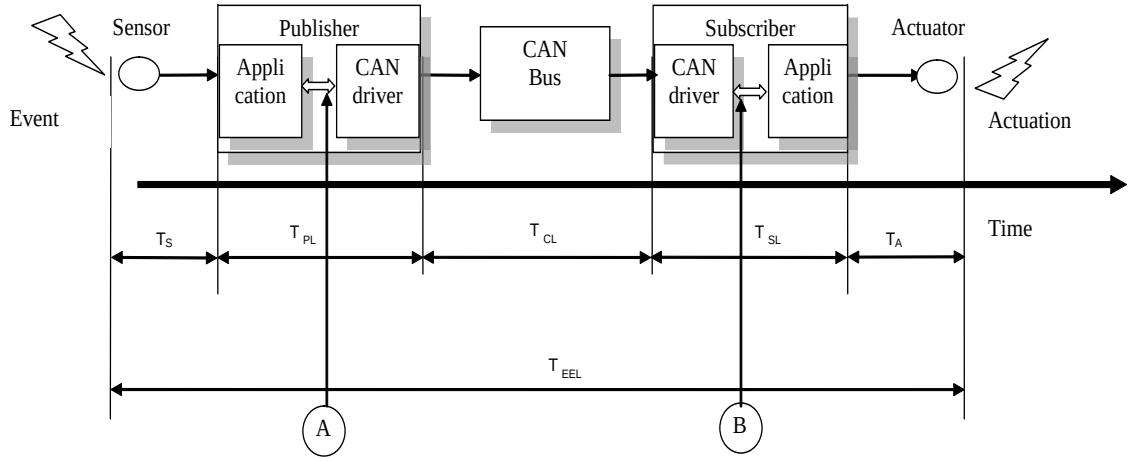


Figure 7-21 – Timing Concept

7.6.5.2 Definitions

Signal: A signal is the smallest piece of identifiable information which serves two purposes: From the “functional” perspective, it is an input or output parameter of a function representing a physical or logical value of a specific data type. From the “communication” point of view, it is a contiguous bit-field of defined length.

Publisher: LRU making a signal available to others, by transmitting it on the bus.

Subscriber: LRU(s) receiving and processing a specific signal published on the bus.

Message: A message is a protocol dependent container capable of carrying data (signals). Messages may use two different transport mechanisms for transmission: PERIODIC or SPORADIC (EVENT_TRIGGERED).

7.6.5.3 Latencies

The end-to-end latency between the generation of an event and the dedicated action may be divided in the following segments:

- Sensor latency (T_S): delay from an Event input to the Publisher.
- Publisher latency (T_{PL}): delay from the input to where the data is ready to be transmitted on the bus and arbitrated.
- CAN bus latency (T_{CL}): delay from the beginning of arbitration until input to the subscriber.
- Subscriber latency (T_{SL}): delay from the input of the signal to the transmission to the actuator.
- Actuator Latency (T_A): delay from transmission from the Subscriber to the response by the actuator.
- End-to-end latency (T_{EEL}): system latency from an event to the dedicated action.

In the system specification, it is imperative that the end-to-end latency must be determined for each signal, i.e., what is the maximum allowable Signal age that the system will tolerate. During the implementation of the system design, the designer needs to be aware of this end-to-end latency requirement and the inherent physical limitations of the design to insure the end-to end latency meets the functional requirements. Grouping signals with common latency requirements reduces bus loading which then decreases bus latency (T_{CL}), thus increasing the possibility of meeting the end-to-end latency requirements (T_{EEL}).

The worst case end-to-end latency (maximum delay) may be determined from the sum of the maximized component latencies. Comparing this value to the end-to-end latency requirement (T_{EEL}) will show the designed system solution meets the design requirement.

7.6.5.4 Publisher and Subscriber Latencies

Both publisher and subscriber embedded software may be partitioned into two major parts:

- Application program (realizing a number of distributed functions)
- Application independent, communication driver that formats the data for transmission

The publisher application program processes the incoming discrete or analog (or any media type) signal. It may be an event triggered signal, such as a switch being pressed or a software timeout signal. The publisher latency may include multiple processes like de-bouncing a switch state, average value calculation, and/or the processing rate (interrupt or pooling period).

The communication processing time of the communication driver (COM stack or other) will depend on the internal architecture of the processor as sampling or queuing data transmission and reception. Processing variations may cause timing delay differences; jitter, which depends on the microprocessor type; and execution of other computations. Therefore, the jitter should also be included in the worst-case latency calculation.

The message containing the signal is queued for transmission and begins arbitrating for the bus.

It is the responsibility of the equipment supplier to be able to define the publisher or subscriber latencies for all processed signals.

7.6.5.5 Transport of a Signal Through a Gateway

A signal is often transported over a number of mediums (e.g., CAN, ARINC 664, ARINC 429) and routed via gateways. The same time, points as described above may be identified for gatewayed signals. The only difference is that the time between points A and B (in Figure 7-21 above) are repeated for each of the buses involved in the chain plus delays introduced by the gateway nodes.

Timing Analysis

To perform a timing analysis and demonstrate determinism, provide validation that the system solution meets the end-to-end latency requirements on selected portions or the entire system, and ensure the expected signal arrives within the required time, the system designer will consider the following:

- All signals have clearly defined end-to-end latency (worse case latencies) requirements.
- The end-to-end latencies of each component are known.
- The assigned DOC field reflects the relative urgency of the data carried.
- All CAN nodes are susceptible to the priority inversion condition. If the implementation is not able to avoid this condition, the signal publisher must account for the potential additional signal delay caused by priority inversion in the CAN communication software.

7.7 Redundancy Management

Redundancy may be used for availability, integrity, or both (such as in triple or quad redundant systems). To support systems that require redundancy, ARINC Specification 825 optionally reserves the least significant two bits of the CAN Identifier to indicate source of redundant information. The two bits provide for up to quad redundancy. The CAN Identifier shown in Figure 7-22 identifies the optional Redundancy Channel Identifier (RCI) field. If a system requires more than four redundant channels, it is recommended that the RCI field be expanded as needed.

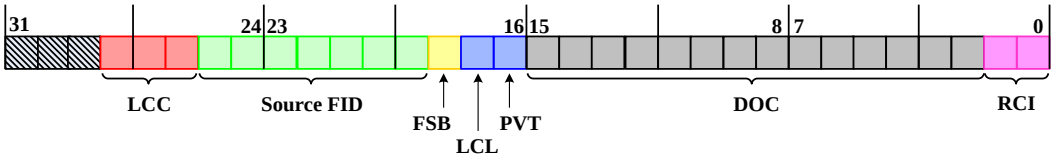


Figure 7-22 – CAN Identifier with RCI Bits

There are additional requirements to manage data on systems with redundant connections. Each unit should set the RCI value for the bus the data is being transmitted on or the bits may be used to distinguish redundant or similar nodes on the same bus. Receiving units must check for valid data on multiple channels (RCIs) and the data may need to be validated for freshness. There are several acceptable uses of the RCI field as shown in the following three figures. Figure 7-23 depicts a partial system with four different units, each interfacing with all four redundant buses.

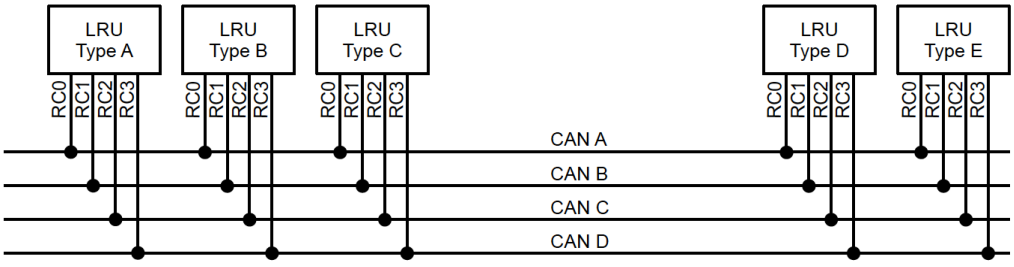


Figure 7-23 – RCI Application Example - Case A

Figure 7-24 depicts a part of a system with four redundant equivalent nodes.

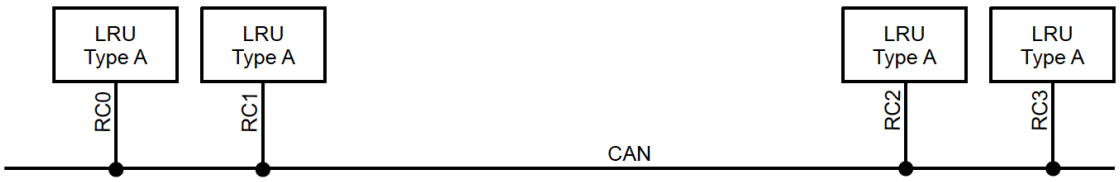


Figure 7-24 – RCI Application Example - Case B

Figure 7-25 depicts a partial system where all nodes are identical but connected to different networks.

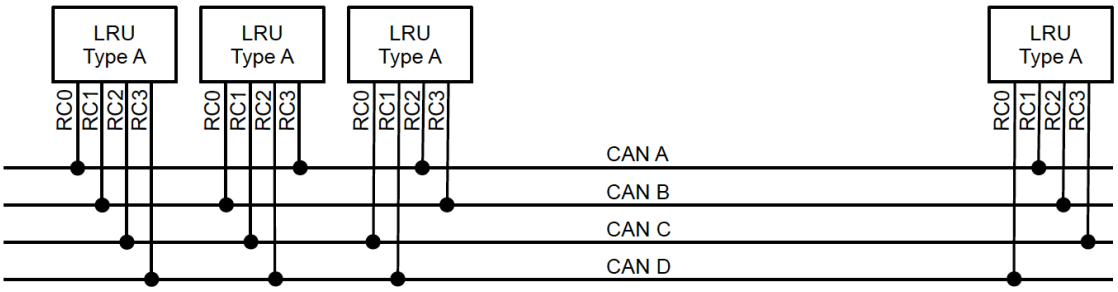


Figure 7-25 – RCI Application Example - Case C

Several examples of multiple subsystems interconnected with CAN are shown in Figure 7-26, where 'network' is used to refer to multiple buses. CAN Network 1 has multiple units connected via a dual redundant CAN. CAN Network 2 is an independent subsystem using dual redundancy while CAN Network 3 is an example of a triple redundant connected subsystem. The same topologies used here may be used to represent non-redundant networks (i.e., systems that have different data on each bus of the CAN Networks), but are used here to represent redundancy. It is possible to define a system that has some different data on each of the multiple buses with the essential or critical data duplicated on all the buses. Systems that allow mixed use of multiple buses may lose some non-essential functionality if one or more buses are lost due to failure.

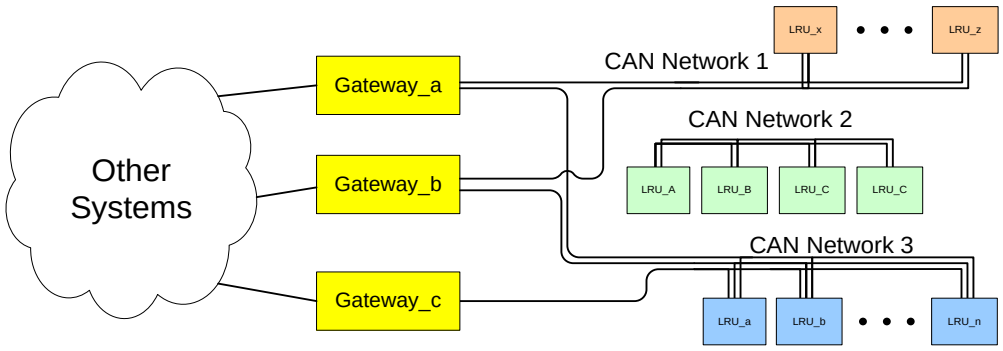


Figure 7-26 – CAN System with Redundancy

7.7.1 Dual Redundancy

For a dual redundant interconnected system such as CAN Network 1 in Figure 7-26, LRUs x through z and Gateways a and b are interconnected. Each of these units should be aware there is redundant information on the two networks. Each unit needs to be designed to set the RCI field as assigned for the bus on which the data is transmitted. Figure 7-27 illustrates a simplified diagram of an LRU with dual redundant CAN connections. The bus connections are labeled A and B. The system integrator assigns the RCI value for each of the buses. For a dual redundant connected system, RCI values 01 and 10 should be used (the RCI assignment should be consistent across the system). Receiving units may need a mechanism to determine whether the data is stale or has been refreshed. Determining if the data is fresh may be complicated in a redundantly interconnected system because the redundant information probably will not be received at the same time from all buses depending on external conditions. The mechanism employed by the LRU to manage redundant information is a function of the LRU requirements and the types of data it is receiving.

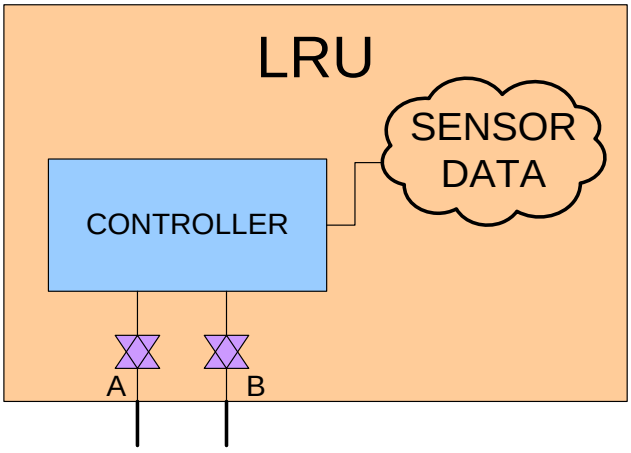


Figure 7-27 – Dual Redundant CAN Interface

For units receiving information from a dual redundant connection and if a message with a particular CAN Identifier is not present or has not been refreshed, the second source should be checked. If neither value has been received, either the sending unit has failed or the receiver is checking for data prior to the maximum latency for the data.

7.7.2 Non-Gateway Subsystem

Subsystems such as CAN Network 2, shown in Figure 7-26, have the same requirements as discussed above. Because the system is not interconnected with other subsystems, the assignment of the RCI values could be delegated to the subsystem developer.

7.7.3 Triple Redundant Subsystems

CAN Network 3, shown in Figure 7-26, is a triple redundant connected subsystem. The requirements for a triple redundant system are similar for to those for a dual redundant system. Each of the units transmitting messages should set the RCI field as assigned by the system integrator for the bus being used for transmission. Receiving units need to manage the received data depending on the system

requirements and the data types being received. The developers of the node managing the redundant data need to be aware that the same redundant data may be skewed in time when received or may not be received at all.

7.7.4 Managing Redundant Data

One of the potential problems with a redundantly connected system is verifying the data has been refreshed. Depending on the system requirements, a system with a single connection merely needs to mark data as read. Assuming the unit performs cyclic processing on the data, on the next iteration the processing checks to verify the data has been refreshed before proceeding. When there are multiple data paths, as in a redundantly connected system, data from only one connection may be required. Once the data is read it should be marked as used. The redundant data on the other CAN data channel may be discarded. For system health monitoring reasons, the data on the redundant channel needs to be monitored to verify the redundant channel is operating properly. Monitoring of the redundant data channel is necessary to avoid latent system faults. If data is used from only one channel and that channel is lost, the system would switch to the second (or third) data channel and it is important to know the backup channels are operating. As mentioned earlier, it is fairly easy to mark the used data as having been used but the redundant data on the 2nd, 3rd, or other channels may not have been received yet. The unit developers need to be aware that a significant time skew may occur when receiving redundant messages over CAN. The time skew may be caused by several normally occurring events and should be expected. Intermittent electrical interference may corrupt one or more messages or single event upsets may corrupt individual messages. Regardless of the bus error cause, it will trigger retries which will force the separate buses of a redundantly connected system to be out of synchronization. The more heavily loaded the buses are, the greater the potential time skew between them. Based on the nature of the subsystem, there are multiple mechanisms for managing the redundant data.

As mentioned above, there are multiple types of bus errors that may cause two buses carrying the same data to be out of sync. For simplicity, in Figure 7-28 we have shown signal interference for causing the delay of messages on bus N_B in a dual redundant connected system. In this example, bus N_B picks up some outside interference that causes a delay in transmitting message M_1. The result is a delay when the receiving controllers receive the completed messages on bus N_B shown as $Skew\_Time_1$ and $Skew\_Time_2$.

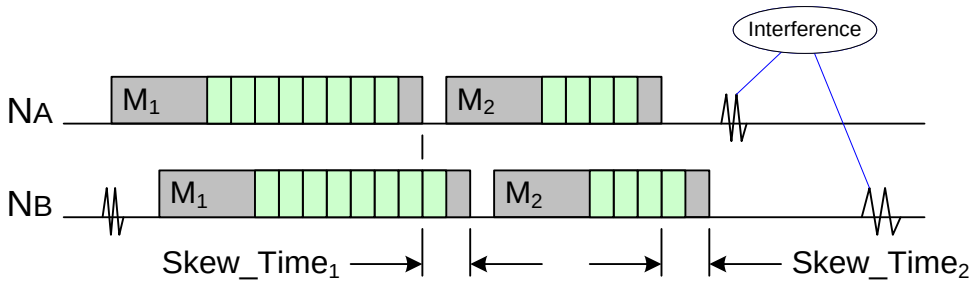


Figure 7-28 – Receiving Time Skew

The message order may be different between redundant buses due to noise on the bus or other bus errors. Figure 7-29 shows a situation where the order of the messages on the two buses in a dual redundant system is reversed. In this

example, a unit is ready to transmit message M_a at time t_1. At this time, there happens to be some interference on bus NB so the message is only sent on bus NA. During the delay, another unit with a higher priority message (M_b) is ready to send at time t_2. Bus NA is busy so the second unit arbitrates for bus NB and begins to transmit message M_b. Once message M_b is complete, message M_a may be sent if another higher priority message is not pending.

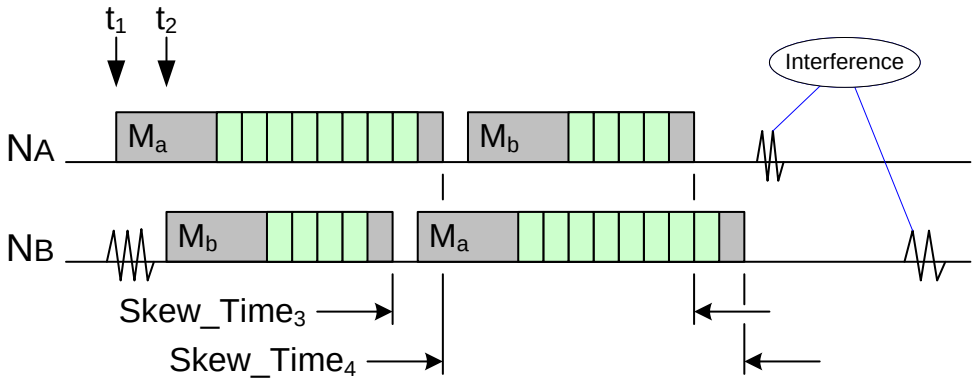


Figure 7-29 – Message Order Changes

Referring to Figure 7-30, sending messages through the gateway from network N to network M is similar. But in this case, the redundant information on network N may either be filtered out or passed through to the units on the network M.

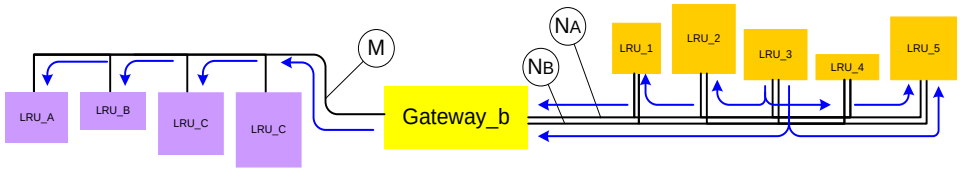


Figure 7-30 – Redundant Bus to Single Bus Gateway

7.8 Data Load

7.8.1 Block Data

Block Data, Directed Messages, or peer-to-peer data, is that data used for such services as data load (ARINC Report 615), maintenance data (ARINC Report 624), or client server services such as FTP and TFTP. This data is usually contained in large files, or blocks, and requires a high-level protocol or handshake mechanism as part of the data transfer process. In ARINC Specification 825, this form of data resides in the Node Service Channel (NSC).

Block Data is normally limited to, but not restricted to, ground based operations. It usually consists of large files that may require a handshake protocol to initiate, run, and complete plus often impose specific timeouts on data transfers. This form of data is usually operator initiated as well. It is important for the system designer to understand the amount of Block Data transfers required on a CAN Bus and allocate enough bandwidth to provide for this form of data.

7.8.2 File and Data Load Rates

This section provides an example for calculating File and Data Load rates for Classical CAN.

Users of CAN that wish to transfer blocks of data for data load should be aware of the time required transfer data via CAN. Based on Section 5.6, Message Prioritization and Bandwidth Management, the maximum theoretical data transfer is 53,333 bytes/second with 1 Mbps data rate at 100% utilization. Section 5.5.4 also limits the maximum bus load to 50% but recommends 30% to minimize data jitter. Using these limits, the maximum bandwidth of CAN is 25.3 kBytes/s at 1 Mbps data rate. Furthermore, if the data transfer protocol is based on blocks where a positive acknowledge is required for data blocks, then the achievable bandwidth is somewhat less.

If, while a file or data load is occurring, the CAN is being actively used by other devices, their bandwidth use must be subtracted from the available bandwidth. As an example, consider a 1 Mbps network with 10 subsystems connected via a CAN network. For a well-designed network, assuming each of these subsystems uses an equal share of the bus bandwidth; one subsystem's bandwidth would be 1.6 kBytes/s (assuming maximum recommended network loading). The bus use may be temporarily pushed up to 50% during a data load. The following equation indicates available bandwidth for data load would be 23%. The available data load bandwidth assume normal operation with the subsystem is suspended during the data load, freeing its allocated bandwidth to be used for the data load, for this example, 1/10 of the allocated bandwidth of 30%, or 3%.

$$BW_{\max} - BW_{used} + BW_{subsystem} = 50\% - 30\% + 3\% = 23\%$$

The expected bandwidth available for the data load would be 12.3 kBytes/s. Table 7-10 provides a summary of the data load times at different data rates based on recommended bandwidth assuming a 10 user network.

Table 7-10 – File Transfer Times

Transfer Time at 23% CAN Usage					
CAN Data Rate	**File Sizes**				
	1 kByte	**10 kBytes**	**50 kBytes**	**100 kBytes**	**1 MByte**
83.333 kbit/s	1.05s	10.5s	52.5s	1.8 min	17.5 min
125 kbit/s	700 ms	7s	35s	1.2 min	11.7 min
250 kbit/s	350 ms	3.5s	17.5s	35s	5.8 min
500 kbit/s	175 ms	1.75s	8.75s	17.5s	2.9 min
1 Mbps	88 ms	875 ms	4.38s	8.75s	1.5 min

If the CAN is disabled for normal use during a data load, such as during maintenance operations the transfer times may be significantly reduced. theoretical network bandwidth. Table 7-11 summarizes the data transfer times based on using 50% of the theoretical network bandwidth.

Table 7-11 – Maximum Recommended Transfer Times

Transfer Time at 50% CAN Usage					
CAN Data Rate	**File Sizes**				
	1 kByte	**10 kBytes**	**50 kBytes**	**100 kBytes**	**1 MByte**
83.333 kbit/s	480 ms	4.8s	24s	48s	8 min
125 kbit/s	320 ms	3.2s	16s	32s	5.3 min
250 kbit/s	160 ms	1.6s	8s	16s	2.7 min

500 kbit/s	80 ms	800 ms	4s	8s	1.3 min
1 Mpbs	40 ms	400 ms	2s	4s	40s

Keep in mind that certain high-level protocols such as **ARINC Report 615A** require handshake signals and have timeouts if responses are not forthcoming in a timely manner. On busy CAN buses, it may be possible to violate these timeouts if not careful. In that context, it should be mentioned that CAN is not really meant to be used in systems that require huge block data or data load services. Services that require large Megabyte file transfers and data load services should consider alternate mediums such as ARINC Specification 664 for this service.

Equally, certain aspects of **ARINC Report 624**: *Onboard Maintenance System (OMS) Protocol* may not be practical on direct connected CAN buses due to message sizes involved. An intermediate process may need to be employed to satisfy certain features of this protocol.

See also Section 7.6.1, Data Throughput Capabilities of CAN.

7.9 Safety, Reliability, and Certification Awareness Considerations

This section classifies types of hardware and software failures and faults. Also included are proposals to deal with the failure modes; how to detect them, report, and manage the failures; and how to design the CAN interface for increased reliability.

Additional information is available in AC 23/25-1309, System Design and Analysis, Advisory Circular; ARP 4754, Certification Considerations for Highly-Integrated or Complex Aircraft Systems and ARP 4761; Guidelines, Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.

7.9.1 Failures Modes

The associated aircraft functions are determined by the standard safety/reliability studies at system level (i.e., Functional Hazard Analysis (FHA)). The system designer is responsible for resolving the consequences and classifications from the FHA. Also, the systems designers are responsible for performing the Failure Modes and Effects Analysis (FMEA) for their areas of responsibility. CAN is a shared resource, and as such, faults will affect multiple units. The following failure conditions are used for classification: Loss, Erratic, and Erroneous.

These failure modes are defined as system effects with respect to a single bus:

- Loss
  - Partial loss: loss of a node's capability to receive or transmit on the bus.
  - Total loss: loss of the communication on the bus.
- Erratic: a fault that causes an intermittent increase in Bit Error Rate (BER) degrading bus operation and performance.
- Erroneous: Undetected message corruption.

7.9.1.1 Partial Loss of Communication

Software ability to send or receive messages and hardware failures need to be considered. Possible causes for the loss of the ability to transmit on or receive from the bus are:

- A node with an internal open circuit, caused by either a dominant time-out protection, a bus-off state, or a power supply loss. The probability of a loss of

the CAN interface is determined by the FMEA calculation provided by the equipment supplier. A typical failure rate for a CAN interface is $1x10^{-7}$/Flight Hour (FH).

- A stub wire or connector defect that results in an open circuit. A typical value for this type of defect is $1x10^{-8}$/FH. For wiring defects, see the Hardware Failures section.

7.9.1.2 Total Loss of Communication

Possible causes of the total loss of communication on one bus are:

- A node with an internal short circuit. An example is an EMI protection circuit failure. An FMEA should give a typical value of $1x10^{-8}$ / FH for that type of failure.
- A wiring or connector short between any two of the following signals: CAN_H/CAN_L/GND/shield, as shown in Section 7.9.2.2 and Figure 7-31. This type of failure typically has a rate of $1x10^{-7}$/FH.
- All nodes are in Bus-Off mode.

The probability of loss of a single bus is typically between $1x10^{-7}$ and $1x10^{-5}$/FH. The consequence of a single bus loss must not be more than a major hazard (as defined in SAE ARP4754).

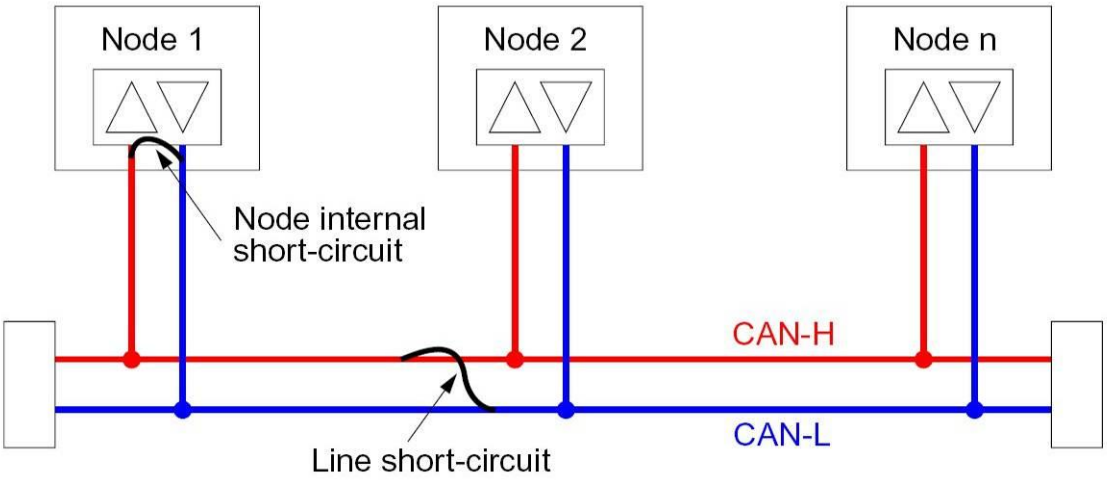


Figure 7-31 – Wire and Short Circuit Example

7.9.1.3 Erratic Behavior

Erratic behavior on the bus will increase the Bus Error Rate (BER). An increased BER will cause error messages and retransmissions. In turn, this condition leads to increased data latency and a reduction of efficiency. This degraded performance could be due to the hardware of the node or a wiring defect. These failures are difficult to troubleshoot and should be minimized through design and test.

Signal quality must be verified during the design phase to eliminate marginal designs which may cause the bus to exceed a typical BER (~$1x10^{-7}$/FH). The system designer should consider the following design criteria described in this document:

- Topology (lengths, stubs, shielding connection, ground shift, etc.)

- Bit timing (sample point, propagation delay, etc.)
- Internal circuitry (layout, choice of EMI components, transceiver common mode rejection level, etc.)

Once the design is validated, the following installation design or production errors may occur:

- An unacceptably high BER may be caused by wiring errors (i.e., loss of a termination resistor) or other kind of wiring errors as described in the Hardware Failures, Section 7.9.2.2.
- Signal degradation caused by external EMI may increase the BER.

7.9.1.4 Erroneous Behavior

The CAN protocol has a high degree of built in error detection such as: bit monitoring, stuff error, and CRC. The CRC insures that up to 5 randomly distributed errors in a message, burst errors shorter than 15 bits in a message, or errors of any odd number are detected in a message.

The probability of a corrupted message going undetected is 4.7×10^{-11} x BER (/bit) which corresponds to the CRC robustness where:

- The Bus Error Rate per Flight Hour (BER/FH) is associated with signal quality bus load and data rate.
- The standard BER/FH should be better than 10^{-5}/FH.

Then, CRC robustness is better than 4.7×10^{-16}/FH and we can consider there is no integrity issue due to the transport of a CAN frame.

The system designer should consider that corruption may occur either in the microprocessor (µP), the CAN controller, in the buffers, or at the sensor I/O. For example, with a Single Event Upset (SEU) or an internal failure, as illustrated in Figure 7-32, the microprocessor (µP) and the CAN controller may be susceptible to an external upset.

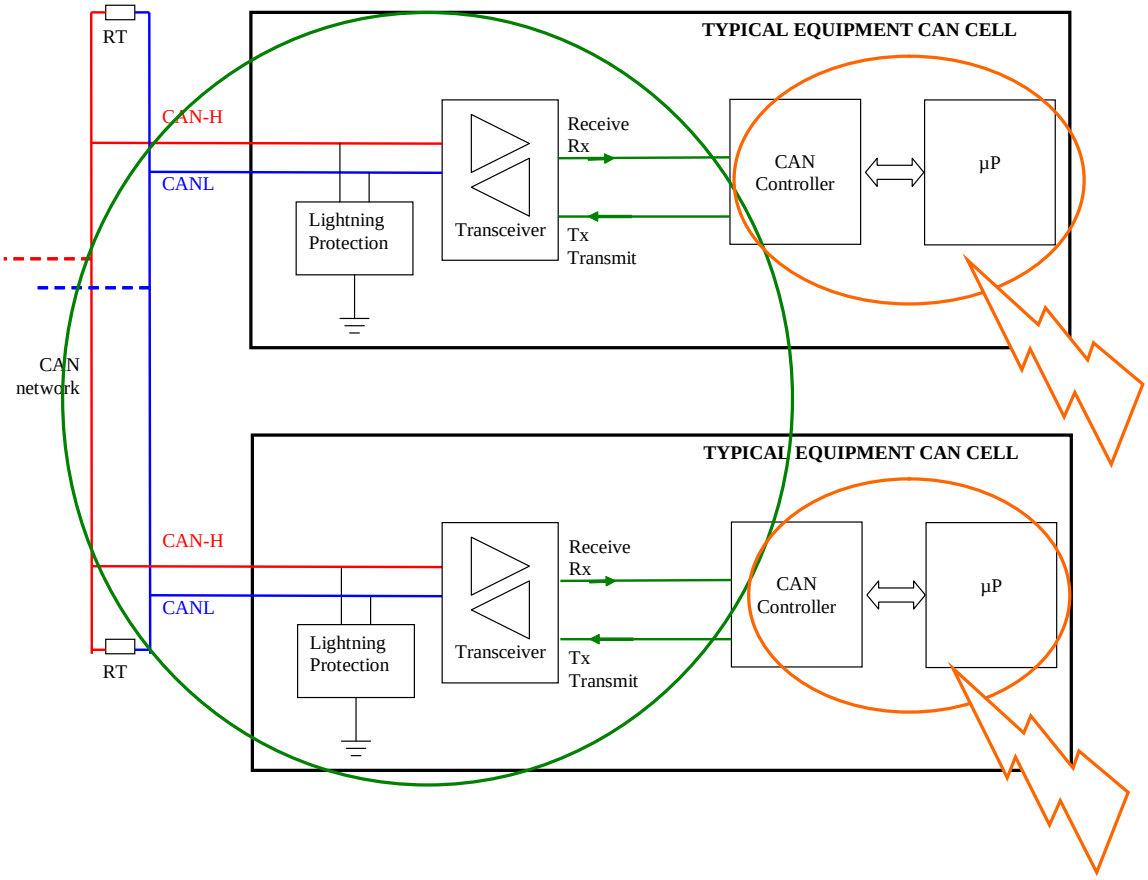


Figure 7-32 – Integrity Issues During Frame Transport

In the above figure, failures in the green circle may be detected by the CAN protocol, failures in the orange are not detected by the CAN protocol, and the system designer is responsible to identify these failures by other means.

7.9.2 Failures

7.9.2.1 Software Failures

7.9.2.1.1 Buffer Overflow

A buffer overflow is an anomalous condition where a process, or device, attempts to store data beyond the boundaries of a fixed-length buffer.

The associated buffer is effectively undersized for the chosen data rate and will “overflow” because data is being written to it faster than it is being read from it. Once the buffer “overflows,” data may be overwritten or in some cases may become “stale.” To any application receiving data from this type of buffer structure, it may appear that data is either getting lost or becomes indeterminate due to what appears to be variations in values. This type of fault may occur at either the sending or receiving nodes.

For example, in a CAN Bus implementation, a buffer overflow might be caused by the data bus message transmission rate being much greater than an applications sample rate.

7.9.2.1.2 Buffer Underflow

A buffer underflow is an anomalous condition that occurs when a fixed length buffer used to communicate between two devices is fed with data at a lower speed than the data is being read from it.

The associated buffer is effectively oversized for the chosen data rate and underflows because data is being read from it faster than fresh data is being written to it. To any application using this sort of buffer structure, it may look like data is stale (old).

For example, in a CAN Bus system a buffer underflow might be caused by an application sample rate that is much faster than the data transmit rate.

7.9.2.1.3 Jabber Frames

A Jabber Frame is the transmission of a packet (or frame) onto a network that is larger than the network's Minimum Transmit Unit (MTU). Such transmissions use excessive bandwidth and create congestion on the network. Additionally, these frames typically will fail a CRC check.

7.9.2.1.4 Babble Error

A Babble Error (or Babbling Idiot) is the continuous, un-controlled, and un-commanded transmission onto the data bus physical layer. These transmissions are out of specification in relation to signal level characteristics and violate the protocol and timing constraints used on the data bus.

A Babble Error is similar to a Jabber Frame error except that Babble usually means the device is having problems electrically and is sending out indeterminate states that violate timing and protocol requirements. It may well deny service to all shared data bus segment users.

7.9.2.1.4.1 Denial of Service

A Denial of Service situation is where a node on the network is saturating the other nodes with data bus transmissions, such that the other nodes are not able to respond to legitimate traffic, or respond so slowly as to be rendered effectively unavailable.

7.9.2.1.5 Priority Inversion

A priority inversion is the scenario where a low priority task holds a shared resource that is required by a high priority task. This causes the execution of the high priority task to be blocked until the low priority task has released the resource, effectively "inverting" the relative priorities of the two tasks.

This will happen if a low priority message is not acknowledged or message priorities and refresh rates are not set correctly. A lower priority message with high refresh rates may starve out higher priority low refresh rate message if not careful.

7.9.2.2 Hardware Failures

The following figures illustrate potential failure modes and consequences that may occur on a bus.

As the media is shared, a single failure may bring the total loss of the bus. Given this, designers are encouraged to design their systems with adequate redundancies (or defined default modes) such that a single failure will not result in the loss of the

system functionality. This is especially important for higher criticality systems and to general performance and reliability considerations of the overall system(s) connected via a shared CAN physical layer.

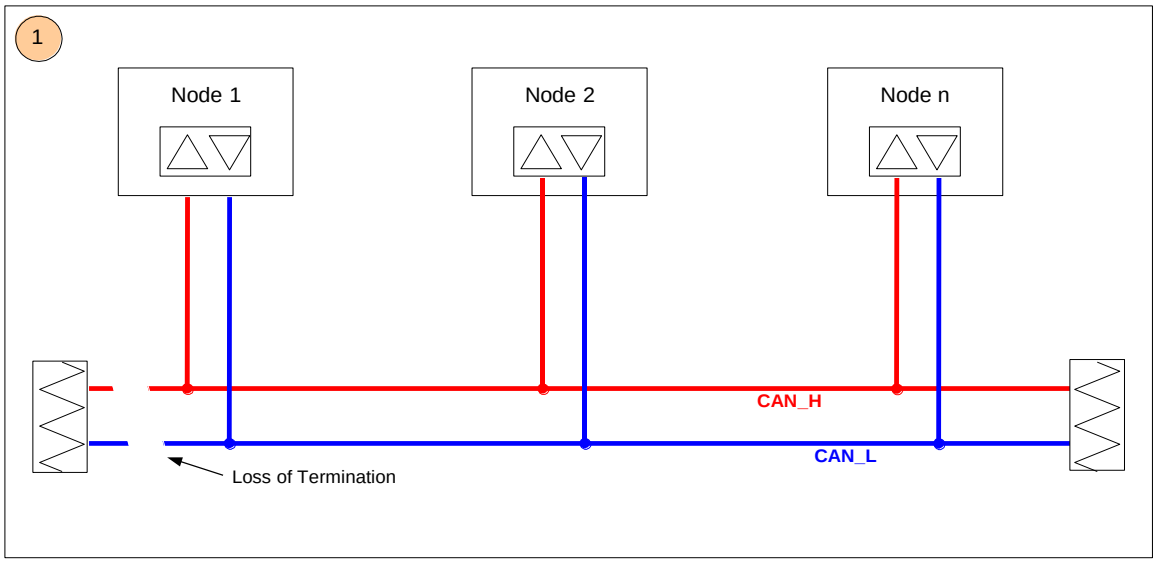


Figure 7-33 – Loss of Termination: Wiring Fault

Figure 7-33 illustrates an open-circuit occurring, on the CAN_H or CAN_L lines of the physical bus: one termination resistor has been disconnected from the bus. In this situation, data communication may be possible with reduced signal to noise ratio, i.e., increased Bit Error Rate (BER): the bus may be fully functional or with an erratic behavior, depending on the environment, the topology, and the node characteristics.

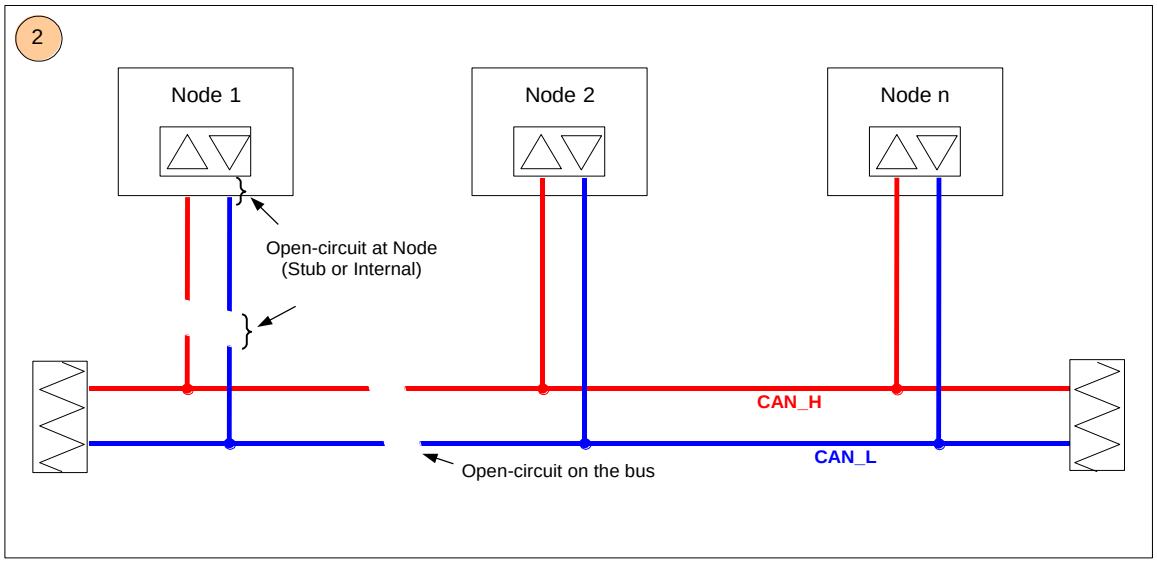


Figure 7-34 – Open Circuit: Wiring Fault

Figure 7-34 illustrates an open-circuit occurring at three places: on both the CAN_H and CAN_L lines of the physical bus, on the stub connecting the node to the bus, and an internal open-circuit inside the node.

For an open-circuit across both the CAN_H and CAN_L lines of the physical bus, communications will be lost with node 1; however, communication may be possible between node 2 and any nodes on the same side of the break (node 3, node 4…, node n), even with the loss of termination (as described above on Figure 7-33). If the open-circuit occurs on the stub, or internal to a particular node, communication with that node is lost while communication between the other nodes on the bus is not impacted.

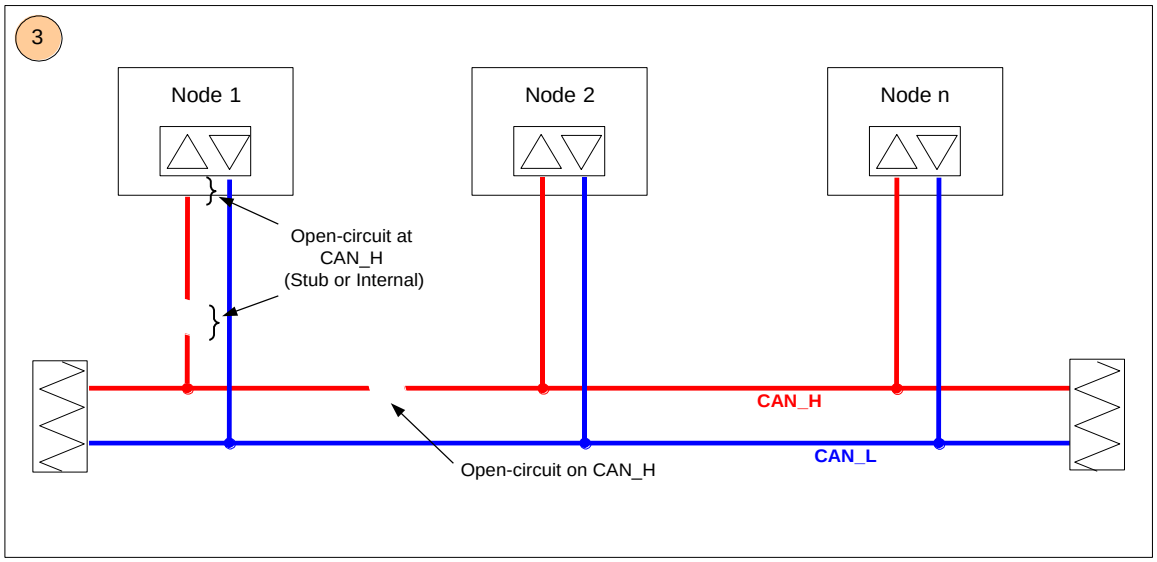


Figure 7-35 – CAN_H Open Circuit: Wiring Fault

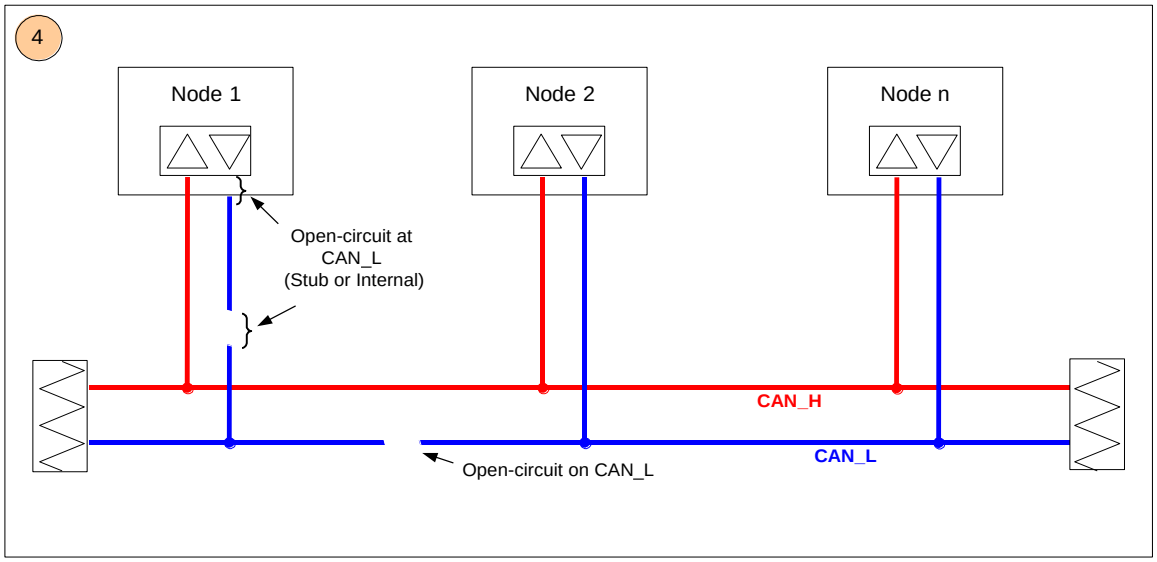


Figure 7-36 – CAN_L Open Circuit: Wiring Fault

Figure 7-35 and Figure 7-36 illustrate a single open line (either CAN_H or CAN_L), affecting either the physical bus, the stub connecting the node to the bus, or internal to the node. In this instance, data communication between LRUs on opposite sides of an interruption may be lost. Data communication between LRUs on the same side of an interruption may be possible with increased BER (as Terminating resistor is lost as described on Figure 7-33) leading to erratic behavior.

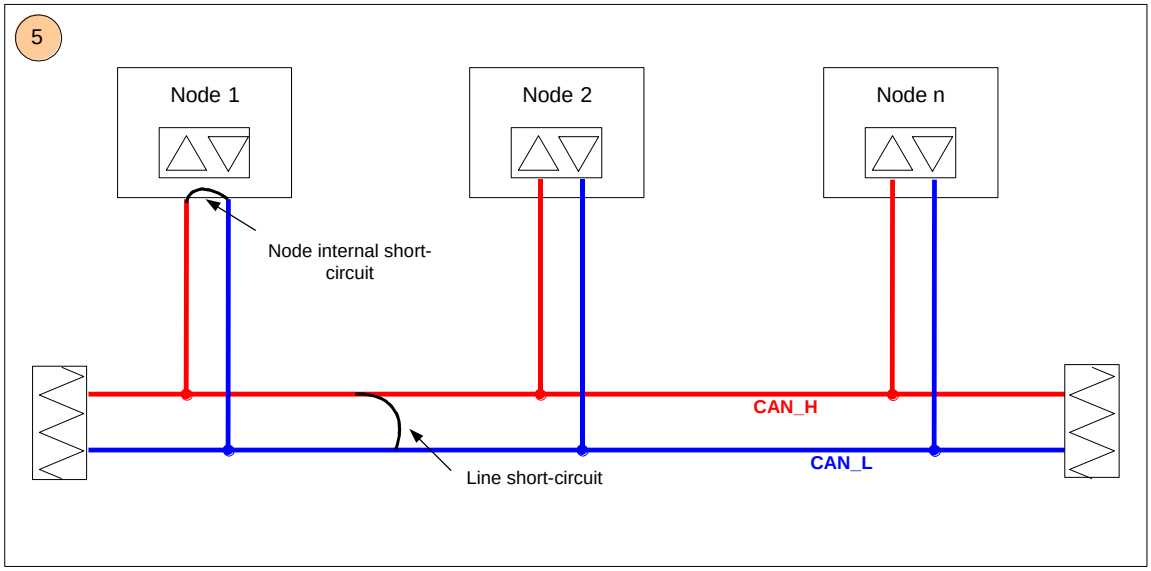


Figure 7-37 – Short Circuit: Wiring Fault

Figure 7-37 shows a short-circuit between CAN_H and CAN_L on the physical bus as well as an internal short across CAN_H and CAN_L. In both conditions, the communication is lost: the fault is total loss.

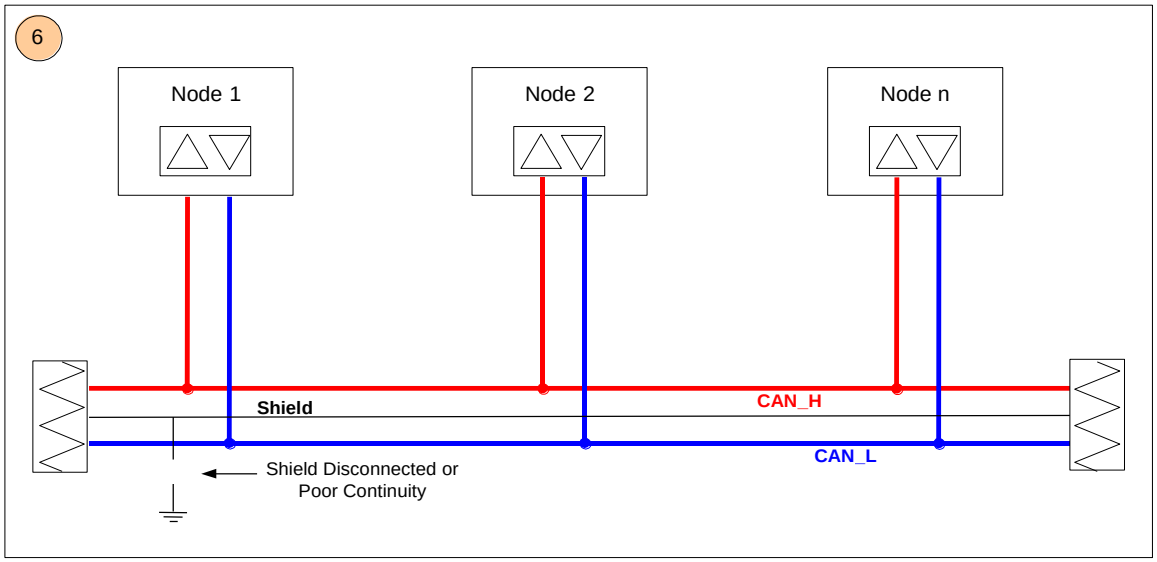


Figure 7-38 – Shield Continuity: Wiring Fault

In Figure 7-38, the shielding for the bus is broken. In this situation, data communication may be possible but with increased BER, depending on the environment noise: EMC margins are reduced.

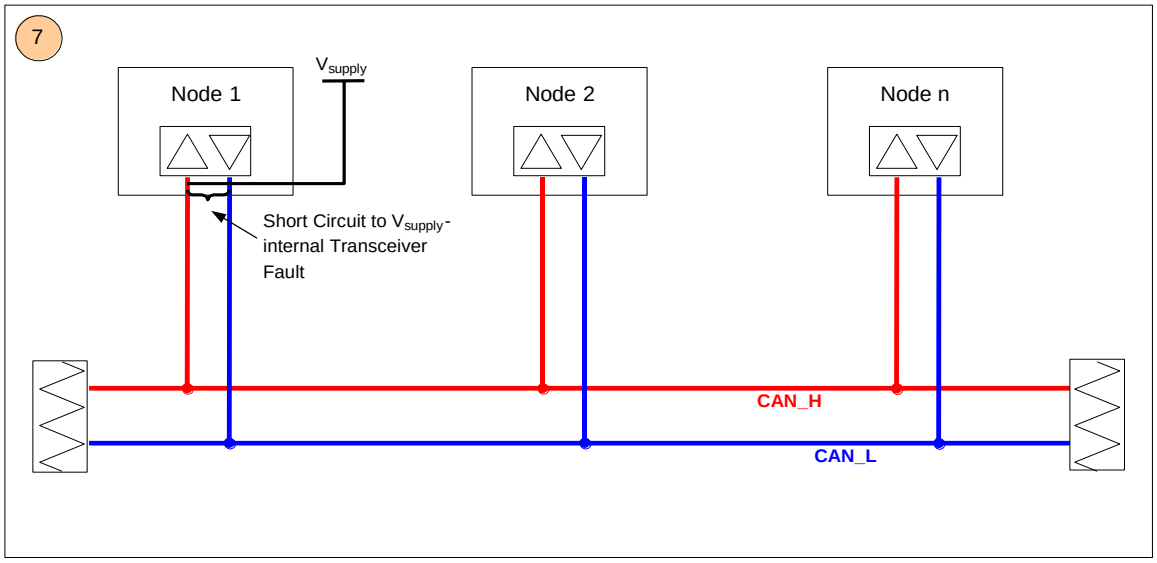


Figure 7-39 – CAN_H or CAN_L Short to V_{supply}: Interface Fault

Figure 7-39 shows either CAN_H or CAN_L shorted to a DC power supply (V_{supply}). If CAN_H is shorted to this DC supply, then data communication will be lost if $V_{supplyt}$ is greater than the maximum allowed common mode voltage. In contrast, if CAN_L is shorted to a DC power supply, then data communication will be lost.

If the DC supply is above the components maximum ratings, all the nodes are damaged.

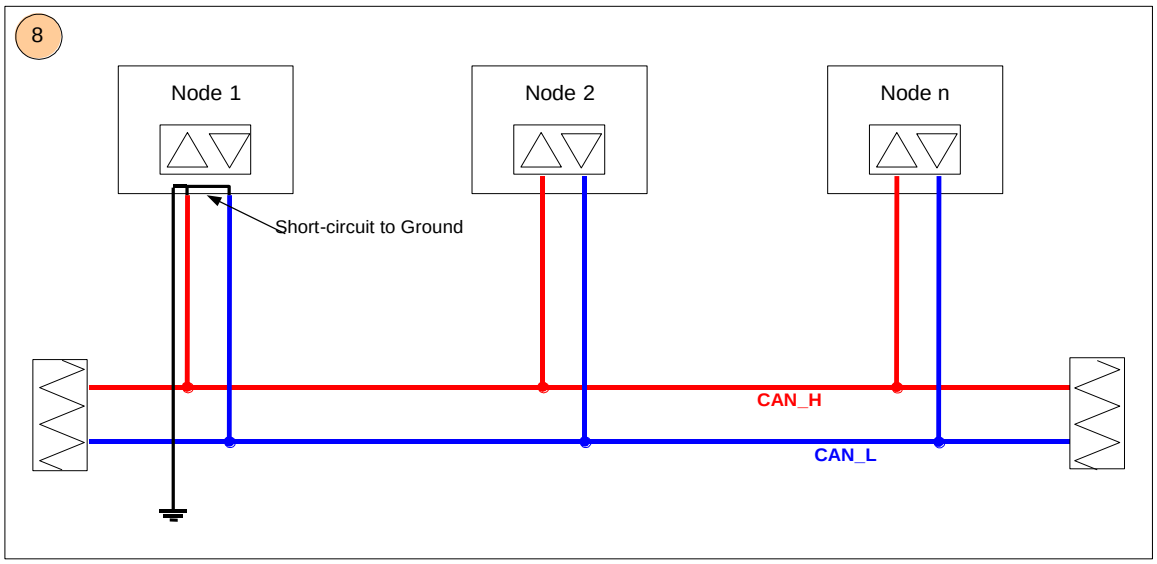


Figure 7-40 – CAN_H or CAN_L Short to Ground: Interface Fault

In Figure 7-40, a short between CAN_H and Ground or CAN_L and Ground is presented. For a CAN_H to Ground short, data communication is lost. For a CAN_L to Ground short, data communication may be possible with increased BER.

Table 7-12 – Bus Fault Behavior

Description of Failure	Communication Behavior	Ref. Fig. #
LRU loss of power or ground (includes low battery condition)	Remaining LRUs continue to communicate with no degradation.	N/A
CPU goes into reset, while its physical layer and IC is still powered	Remaining LRUs continue to communicate with no degradation.	N/A
Loss of one termination	Data communication may be possible with increased BER.	Figure 7-33
One LRU becomes disconnected from the bus	Remaining LRUs continue to communicate with no degradation.	Figure 7-34
CAN_H wire open	Data communication between LRUs on opposite sides of an interruption may be lost. Data communication between LRUs on the same side of an interruption may be possible with increased BER (as Terminating resistor is lost) leading to erratic behavior.	Figure 7-35
CAN_L wire open	Data communication between LRUs on opposite sides of an interruption may be lost. Data communication between LRUs on the same side of an interruption may be possible with increased BER (as Terminating resistor is lost) leading to erratic behavior.	Figure 7-36
CAN_H and CAN_L open	In a Point to Point bus configuration all data communications will be lost. In a Multi-drop bus configuration, data communications will be unreliable.	Figure 7-34
CAN_H shorted to DC power	Data communication will be lost if V_{supply} is greater than the maximum allowed common mode voltage.	Figure 7-39
CAN_L shorted to DC power	Data communication will be lost on the bus.	Figure 7-39
CAN_H shorted to ground	Data communication will be lost on the bus.	Figure 7-40
CAN_L shorted to ground	Data communication may be possible with increased BER.	Figure 7-40
CAN_H shorted to CAN_L	Data communication will be lost.	Figure 7-37
Loss of wire shield continuity	Data communication may be possible with probable increased BER.	Figure 7-38

7.9.3 Overcoming Failures

7.9.3.1 Failure Detection

The following failures should be detected:

- Partial loss (e.g., loss of occasional messages) should be detected at bus level by a proper BITE implementation with the health status message as defined in Section 5.4.1. Nodes on the bus may concentrate the health status messages and transmit to the centralized maintenance the missing node.
- Total loss (i.e., no message traffic on the bus) should be detected at system level. The bus redundancy is generally the best way to overcome the total loss of one bus. The troubleshooting to find the faulty node is very difficult.

- Erratic behavior (i.e., intermittent message loss) may possibly be detected if periodic frames only are defined.
- Erroneous behavior: In case of critical data transport, redundancy of sensors is preferred.

7.9.3.2 Babbling Transceiver Protection

A babbling transceiver fault is a failure that produces data traffic on the network either carrying no useful information or carrying correct or incorrect messages more frequently than planned. The consequence on the system is either loss of the bus, or increase of bus loading, up to 100%. With a high bus loading data could be lost or delayed.

The failure could occur either in the host microprocessor (µP) or the CAN controller. Probabilities for the occurrence of this type of fault are application specific.

A faulty node may degrade the operation of the entire bus because higher priority messages are granted bus access over lower priority messages. However, even very low priority message may consume sufficient bus bandwidth to force higher priority messages to wait.

The easiest way to manage babbling transceiver faults in aircraft applications is through the use of bus redundancy. If redundancy is undesirable, there are other alternatives, such as:

- Limit the use of event-trigger messages using only periodic messages or creating transmission slots at the application layer level.
- Bus guardian implementation adding a transceiver and a CAN controller (listen only mode) for monitoring.
- Using a loop-back frame with low priority expected within a defined time out. In case the loop-back frame is not received by the bus-guardian, the transmission may be inhibited.

In the last two examples, because the medium is shared, the node could be inhibited when the faulty node is elsewhere. The major drawback is that communication could be stopped on healthy nodes.

7.9.4 CAN Security

This standard relies on physical isolation of CAN nodes and the interconnecting wiring to provide security. This standard does not provide mechanisms to validate the source of messages or the validity of the data. It is the responsibility of the system implementer to ensure that controlled processes are used in the installation and maintenance of the system. The system design should be reviewed for potential unauthorized access to the CAN interconnect or nodes.

Normally, nodes on an avionics CAN bus only generate prescribed messages. CAN nodes (i.e., gateways) that provide an interface between CAN and other systems or networks may need to provide security between them.

Depending on the system design and implementation, it is possible that items in Section 7.9.2, Failures, could introduce security vulnerabilities. CAN system designers need to be aware of security standards and regulations (Refer to ARINC Report 811 for guidance).

COMMENTARY

In addition to the above, this standard assumes CAN is a wired network that connects nodes and is secured through physical means. The physical isolation provides protection for this network. Thus, there are only two security considerations for CAN. First, is the connection of an external device to the network (e.g., dataloader). Second, is the connection of a gateway to CAN from another network. External security measures should be implemented to secure this environment.

7.9.5 System Design Assurance Level (DAL) Using CAN

The Design Assurance Level (DAL) for CAN nodes and the respective interconnect is based on the function of these nodes in the aircraft. The design criticality is determined by the Functional Hazard Analysis (FHA) such as described in:

- SAE ARP4754 “Certification Considerations For Highly-Integrated Or Complex Aircraft Systems”
- SAE ARP4761 “Guidelines and Methods for conducting the Safety Assessment Process on Civil Airborne Systems and Equipment”
- CS-25 “Certification Specifications for Large Airplanes”

Even critical systems (with a DAL of A) may use CAN assuming the loss of a single bus is no more and a major hazard.

7.9.6 Certification Awareness

The use of digital networks on aircraft has generated some unique guidance from the certification authorities. Equipment developers and system integrators should be aware of the certification guidance material available in AC 20-156 *Aviation Databus Assurance.* This material provides guidance to certify systems that use digital data buses such as CAN. To aid system developers and integrators Appendix G provides a cross reference between this standard and FAA AC 20-156.

ATTACHMENT 1 COMMUNICATION PROFILE DATABASE

The XSD files can be found at
[update link to new support files].

1.1 System-level Schema

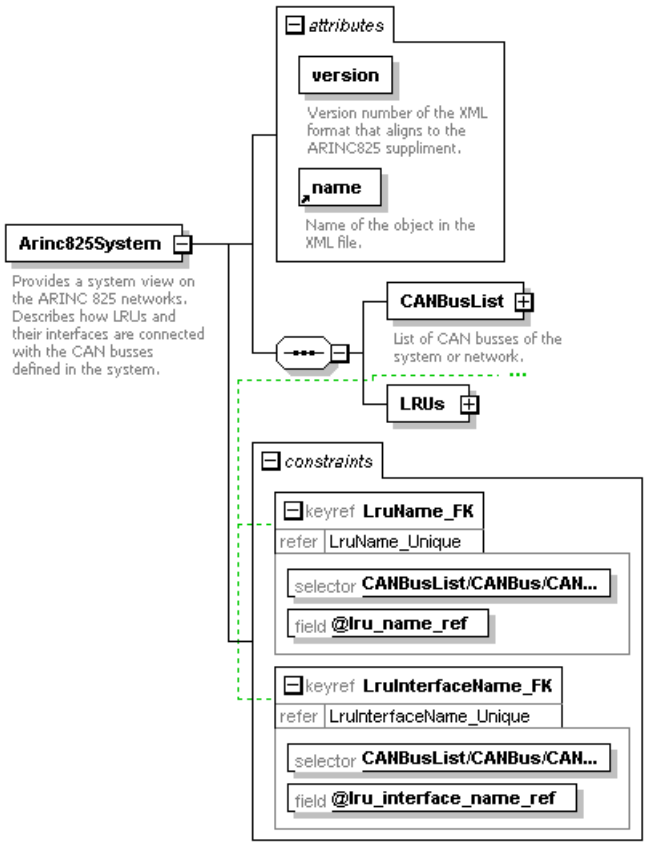
Schema **ARINC825-4-System.xsd**

schema location: **ARINC825-4-System.xsd**
attributeFormDefault: **unqualified**
elementFormDefault: **Qualified**

Elements	Attributes
Arinc825System	**name**

element **Arinc825System**

diagram



properties: content Complex

children: **CANBusList** **LRUs**

attributes

Name	Type	Use	Default	Fixed	Annotation
version	**xs:double**	required			documentation Version number of the XML format that aligns to the ARINC825 supplement.
name	**xs:string**	required			documentation Name of the

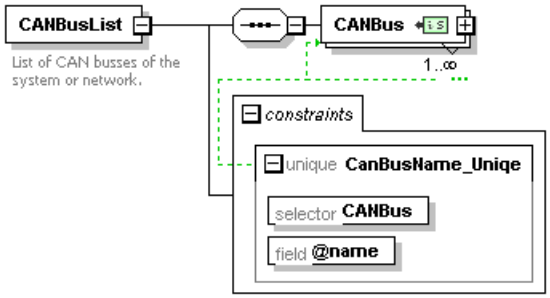
ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

	object in the XML file.
identity constraints	Name: LruName_FK; Refer: LruName_Unique; Selector: CANBusList/CANBus/CANBusStub; Field(s): @lru_name_ref Name: LruInterfaceName_FK; Refer: LruInterfaceName_Unique; Selector: CANBusList/CANBus/CANBusStub; Field(s): @lru_interface_name_ref
annotation	Documentation Provides a system view on the ARINC 825 networks. Describes how LRUs and their interfaces are connected with the CAN busses defined in the system.

attribute **Arinc825System/@version**

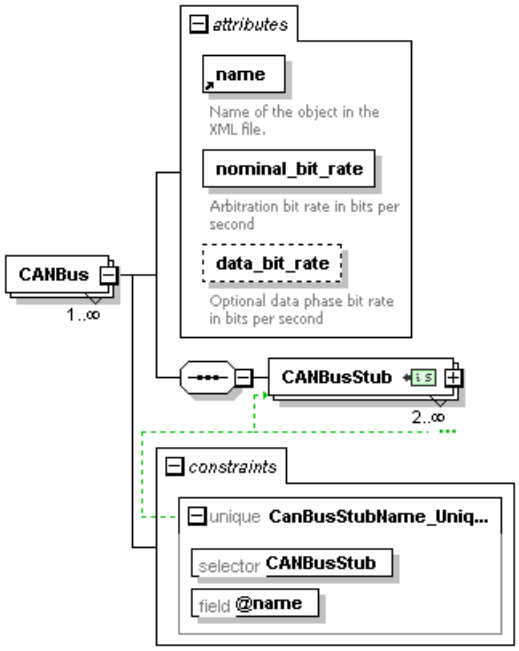
type	**xs:double**
properties	use required
annotation	documentation Version number of the XML format that aligns to the ARINC825 supplement.

element **Arinc825System/CANBusList**

diagram	
properties	content complex
children	**CANBus**
identity constraints	Name: unique; Refer: CanBusName_Uniqe; Selector: CANBus; Field(s): @name
annotation	documentation List of CAN busses of the system or network.

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **Arinc825System/CANBusList/CANBus**

diagram	
properties	minOcc 1 maxOcc unbounded content complex
children	**CANBusStub**

attributes	Name	Type	Use	Default	Fixed	Annotation
	name	**xs:string**	required			documentation Name of the object in the XML file.
	nominal_bit_rate	**xs:unsignedInt**	required			documentation Arbitration bit rate in bits per second
	data_bit_rate	**xs:unsignedInt**	optional			documentation Optional data phase bit rate in bits per second

identity constraints	Name	Refer	Selector	Field(s)	Annotation
unique	CanBusStubName_Unique		CANBusStub	@name	

attribute **Arinc825System/CANBusList/CANBus/@nominal_bit_rate**

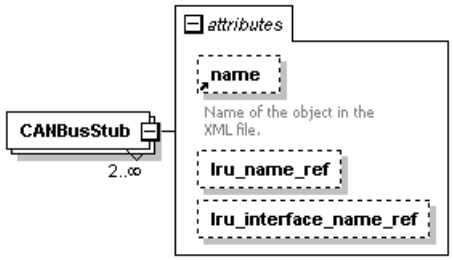
type	**xs:unsignedInt**
properties	use required
annotation	documentation Arbitration bit rate in bits per second

attribute **Arinc825System/CANBusList/CANBus/@data_bit_rate**

type	**xs:unsignedInt**
properties	use optional
annotation	documentation Optional data phase bit rate in bits per second

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **Arinc825System/CANBusList/CANBus/CANBusStub**

diagram	
properties	minOcc 2 maxOcc unbounded content complex

attributes

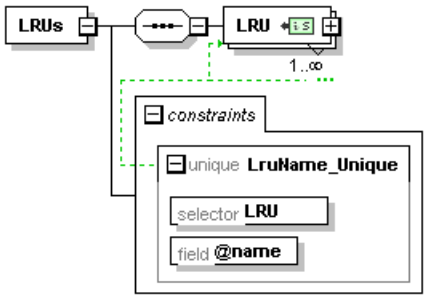
Name	Type	Use	Default	Fixed	Annotation
name	**xs:string**				documentation: Name of the object in the XML file.
lru_name_ref	**xs:string**	optional			
lru_interface_name_ref					

attribute **Arinc825System/CANBusList/CANBus/CANBusStub/@lru_name_ref**

type	**xs:string**
properties	use optional

attribute **Arinc825System/CANBusList/CANBus/CANBusStub/@lru_interface_name_ref**

element **Arinc825System/LRUs**

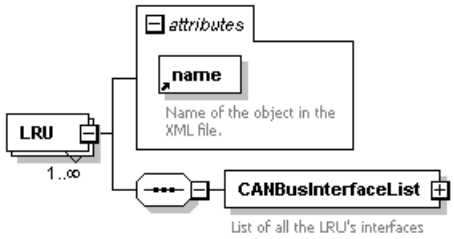
diagram	
properties	content complex
children	**LRU**

identity constraints

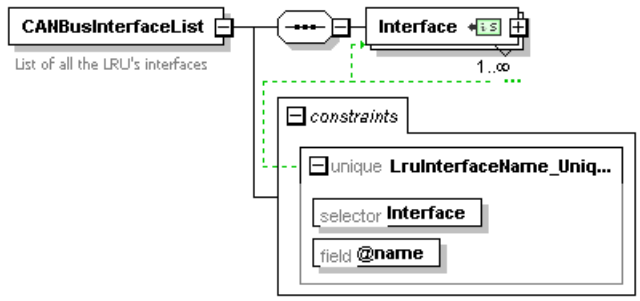
	Name	Refer	Selector	Field(s)	Annotation
unique	LruName_Unique		LRU	@name	

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

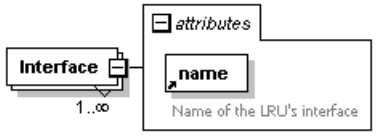
element **Arinc825System/LRUs/LRU**

diagram	
properties	minOcc 1 maxOcc unbounded content complex
children	**CANBusInterfaceList**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Name of the object in the XML file.

element **Arinc825System/LRUs/LRU/CANBusInterfaceList**

diagram	
properties	content complex
children	**Interface**
identity constraints	Name: unique; Refer: ; Selector: Interface; Field(s): @name; Annotation: LruInterfaceName_Unique
annotation	documentation List of all the LRU's interfaces

element **Arinc825System/LRUs/LRU/CANBusInterfaceList/Interface**

diagram	
properties	minOcc 1 maxOcc unbounded content complex
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Name of the LRU's interface

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **name**

type	**xs:string**
used by	elements **Arinc825System** **Arinc825System/CANBusList/CANBus** **Arinc825System/CANBusList/CANBus/CANBusStub** **Arinc825System/LRUs/LRU/CANBusInterfaceList/Interface** **Arinc825System/LRUs/LRU**
annotation	documentation Name of the object in the XML file.

XML Schema documentation generated by **XMLSpy** Schema Editor **http://www.altova.com/xmlspy**

1.2 LRU-level Schema

Schema **ARINC825-4.xsd**

schema location: **ARINC825-4.xsd**
attributeFormDefault: **unqualified**
elementFormDefault: **qualified**

Elements
A825Message_A825IdFormat
A825Message_DirectedMessage
A825Message_ElevenBitId
A825Message_OneToMany
A825Message_PeerToPeer
Arinc825Profile
CANBusInterface_CANFD
CANBusInterface_ClassicalCAN
CANBusInterface_TypeOfInterface
ContinuousSignal
DataField
DataFieldBase
EnumeratedSignal
Parameter
ParameterBase
ParameterValidityStatusSet
ParametricDataField
Signal
SignalList

Complex types
A825Message_11BitIdentifierType
A825Message_DirectedMessageType
A825Message_IdFormatType
A825Message_LCCIdFormatType
A825Message_OneToManyType
A825Message_PeerToPeerMessageType
CanBitTimingType
CANBusInterface_BaseType
CANBusInterface_CANFDType
CANBusInterface_ClassicalCANType
DataFieldBaseType
DataFieldType
MessageListType
ParameterType
ParametricDataFieldType
SignalContinuousType
SignalEnumeratedType
SignalType

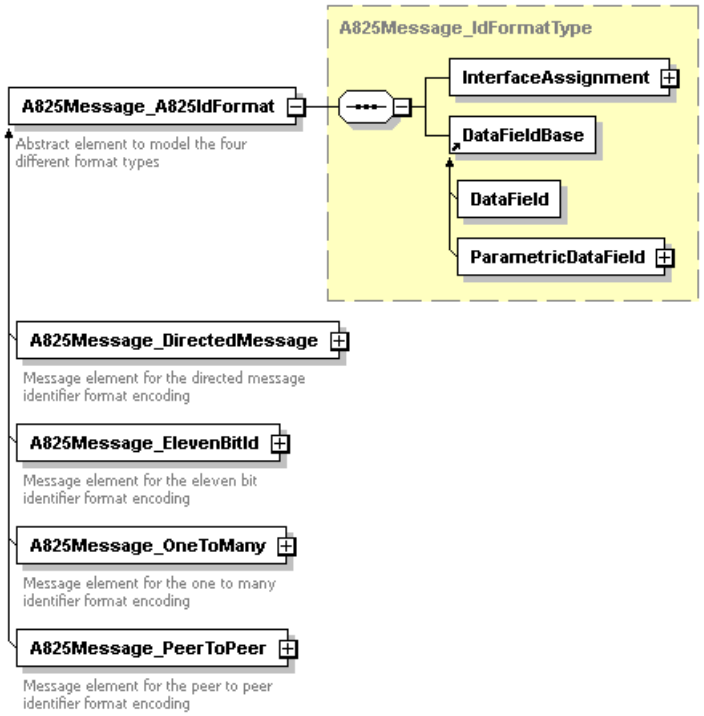
Attributes
can_fd_enabled
comment
data_bit_rate
default_rci
doc
fid
fmc_11bit_id
fsb
id
latency_max
lcc
lcl
lsb
max
min
msb
name
node_address
node_id
nominal_bit_rate
offset
pua
pvs_lsb
pvs_msb
pvt
rci_override
resolution
sid
txp
version

Attr. groups
bit_timing_attrib_grp

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **A825Message_A825IdFormat**

diagram



type: **A825Message_IdFormatType**

properties:
- content: complex
- abstract: true

children: **InterfaceAssignment** **DataFieldBase**

used by:
- complexType: **MessageListType**

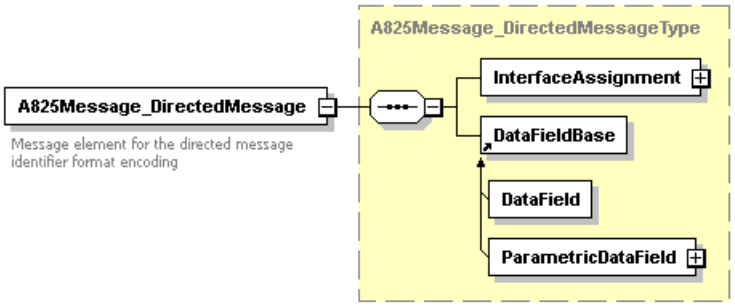
attributes:

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation: Name of the item
txp	**derived by: xs:double**	required			documentation: Transmit Period, Minimum period at which the message is sent over the bus in milliseconds as a floating-point number (shall be set to “0” for event-driven data).
comment	**xs:string**	optional			documentation: User-defined comment (not processed)

annotation:
documentation
Abstract element to model the four different format types

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **A825Message_DirectedMessage**

diagram	
type	**A825Message_DirectedMessageType**
substitution group	**A825Message_A825IdFormat**
properties	content complex
children	**InterfaceAssignment** **DataFieldBase**

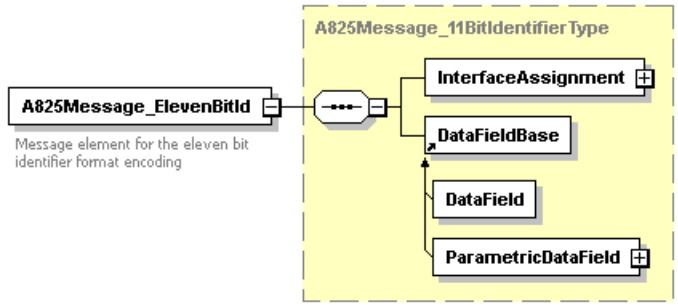
attributes

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation Name of the item
txp	**derived by: xs:double**	required			documentation Transmit Period, Minimum period at which the message is sent over the bus in milliseconds as a floating-point number (shall be set to “0” for event-driven data).
comment	**xs:string**	optional			documentation User-defined comment (not processed)
initial_role	**derived by: xs:string**	required			
source_address	**derived by: xs:unsignedByte**	optional			
source_port_id	**derived by: xs:unsignedByte**	optional			
destination_address	**derived by: xs:unsignedByte**	optional			
destination_port_id	**derived by: xs:unsignedByte**	optional			

annotation	documentation Message element for the directed message identifier format encoding

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **A825Message_ElevenBitId**

diagram	
type	**A825Message_11BitIdentifierType**
substitution group	**A825Message_A825IdFormat**
properties	content complex
children	**InterfaceAssignment** **DataFieldBase**

attributes

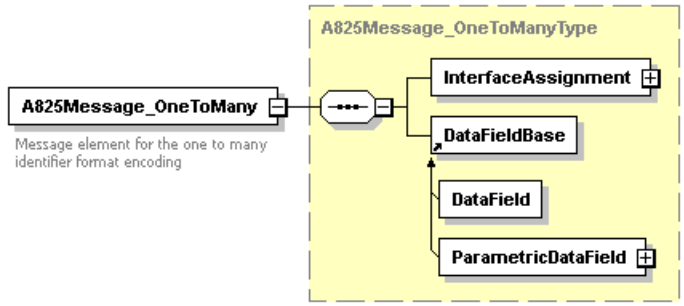
Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation Name of the item
txp	**derived by: xs:double**	required			documentation Transmit Period, Minimum period at which the message is sent over the bus in milliseconds as a floating point number (shall be set to “0” for event-driven data).
comment	**xs:string**	optional			documentation User-defined comment (not processed)
lcc	**derived by: xs:string**	required			documentation Logical Communication Channel
label	**derived by: xs:string**	required			documentation Eight-bit identifier segment of the eleven bit CAN identifier, similar to the A429 label id.

annotation	documentation Message element for the eleven-bit identifier format encoding

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **A825Message_OneToMany**

diagram



type **A825Message_OneToManyType**

substitution group **A825Message_A825IdFormat**

properties content complex

children **InterfaceAssignment** **DataFieldBase**

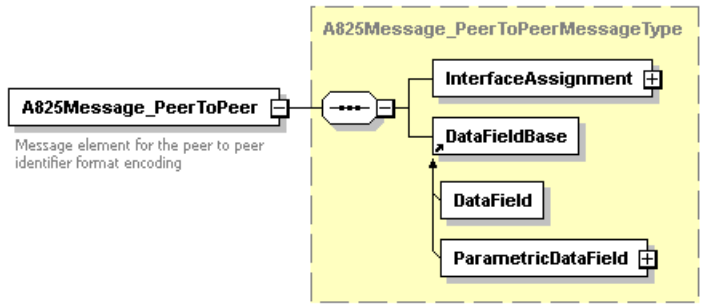
attributes

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation Name of the item
txp	**derived by: xs:double**	required			documentation Transmit Period, Minimum period at which the message is sent over the bus in milliseconds as a floating point number (shall be set to “0” for event-driven data).
comment	**xs:string**	optional			documentation User-defined comment (not processed)
lcc	**derived by: xs:string**	required			documentation Logical Communication Channel
fid	**derived by: xs:int**	required			documentation Function Identifier, 0 is multicast function id, only used with one to many message identifier type
lcl	**derived by: xs:int**	required			documentation Identifier Local Bit, only used with one to many message identifier type
pvt	**derived by: xs:int**	required			documentation Identifier Private Bit, only used with one to many message identifier type
doc	**derived by: xs:int**	required			documentation Identifier Data Object Code, only

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

	used with one to many message identifier type
annotation	documentation Message element for the one to many identifier format encoding

element **A825Message_PeerToPeer**

diagram	
type	**A825Message_PeerToPeerMessageType**
substitution group	**A825Message_A825IdFormat**
properties	content complex
children	**InterfaceAssignment** **DataFieldBase**

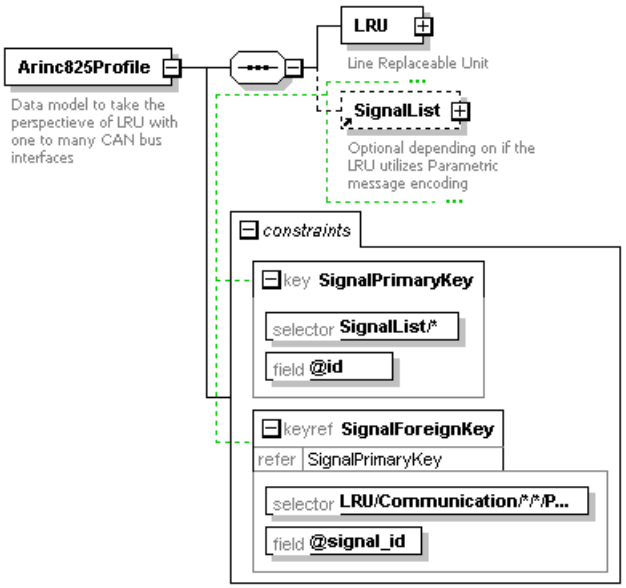
attributes

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation Name of the item
txp	**derived by: xs:double**	required			documentation Transmit Period, Minimum period at which the message is sent over the bus in milliseconds as a floating-point number (shall be set to “0” for event-driven data).
comment	**xs:string**	optional			documentation User-defined comment (not processed)
lcc	**derived by: xs:string**	required			documentation Logical Communication Channel
fid	**derived by: xs:int**	required			documentation Function Identifier, 0 is multicast function id, only used with one to many message identifier type
smt	**derived by: xs:int**	optional			
lcl	**derived by: xs:int**	optional			documentation Identifier Local Bit, only used with one to many message identifier type

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

	pvt	**derived by: xs:int**	optional	documentation Identifier Private Bit, only used with one to many message identifier type
	server_fid	**derived by: xs:unsignedInt**	optional	
	sid	**xs:int**	optional	documentation Value of the Server ID (SID) field as an integer number in the range of 0-511. This attribute is used for peer-to-peer communication parameters only.
annotation	documentation Message element for the peer to peer identifier format encoding			

element **Arinc825Profile**

diagram	
properties	content complex
children	**LRU SignalList**

attributes	Name	Type	Use	Default	Fixed	Annotation
	profile_id	**xs:string**	required			documentation ARINC 825 profile ID in the format: ID.Sub-ID
	name	**derived by: xs:string**	required			documentation Name of the item
	major	**xs:int**	optional			documentation Major Time Frame in milliseconds as an integer number in the range of 1 to 1000000

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

	minor	**xs:int**	optional			documentation Minor Time Frame in milliseconds as an integer number in the range of 1 to 1000000
	comment	**xs:string**	optional			documentation User-defined comment (not processed)
	version	**xs:double**	required		4	documentation Identification number of the ARINC-825 supplement number that this schema aligns to
identity constraints		Name	Refer	Selector	Field(s)	Annotation
	key	SignalPrimaryKey		SignalList/*	@id	
	keyref	SignalForeignKey	SignalPrimaryKey	LRU/Communication/*/*/ParametricDataField/Parameter | LRU/Communication/*/*/ParametricDataField/ParameterValidityStatusSet/Parameter	@signal_id	
annotation	documentation Data model to take the perspective of LRU with one to many CAN bus interfaces					

attribute **Arinc825Profile/@profile_id**

type	**xs:string**
properties	use required
annotation	documentation ARINC 825 profile ID in the format: ID.Sub-ID

attribute **Arinc825Profile/@major**

type	**xs:int**
properties	use optional
annotation	documentation Major Time Frame in milliseconds as an integer number in the range of 1 to 1000000

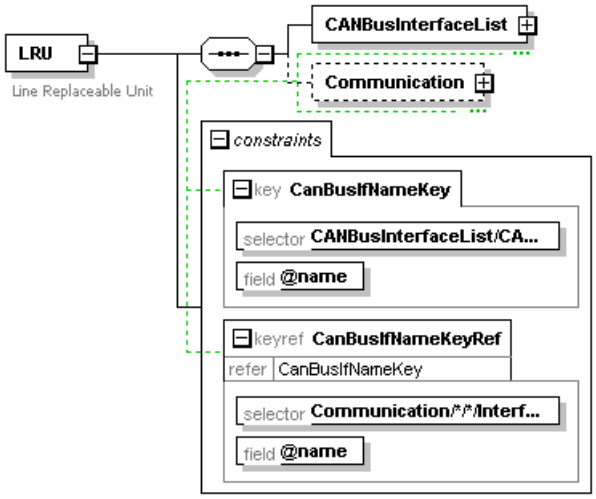
ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **Arinc825Profile/@minor**

type	**xs:int**
properties	use optional
annotation	documentation Minor Time Frame in milliseconds as an integer number in the range of 1 to 1000000

element **Arinc825Profile/LRU**

diagram



properties: content complex

children: **CANBusInterfaceList** **Communication**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation Name of the item
comment	**xs:string**	optional			documentation User-defined comment (not processed)
lru_code	**xs:unsignedShort**	required			
lru_code_meaning	**xs:string**	required			

identity constraints

	Name	Refer	Selector	Field(s)	Annotation
key	CanBusIfNameKey		CANBusInterfaceList/CANBusInterface_ClassicalCAN | CANBusInterfaceList/CANBusInterface_CANFD	@name	
keyref	CanBusIfNameKeyRef	CanBusIfNameKey	Communication/*/*/InterfaceAssignment/Interface	@name	

annotation: documentation
Line Replaceable Unit

attribute **Arinc825Profile/LRU/@lru_code**

type	**xs:unsignedShort**
properties	use required

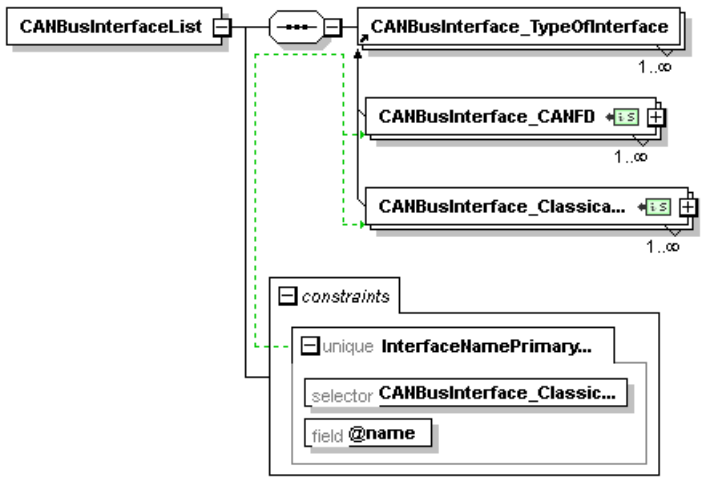
attribute **Arinc825Profile/LRU/@lru_code_meaning**

type	**xs:string**

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

properties	use required

element **Arinc825Profile/LRU/CANBusInterfaceList**

diagram	
properties	content complex
children	CANBusInterface_TypeOfInterface

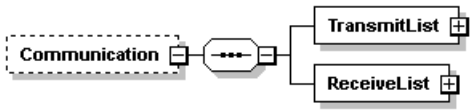
attributes

Name	Type	Use	Default	Fixed	Annotation
comment	**xs:string**	optional			documentation: User-defined comment (not processed)

identity constraints

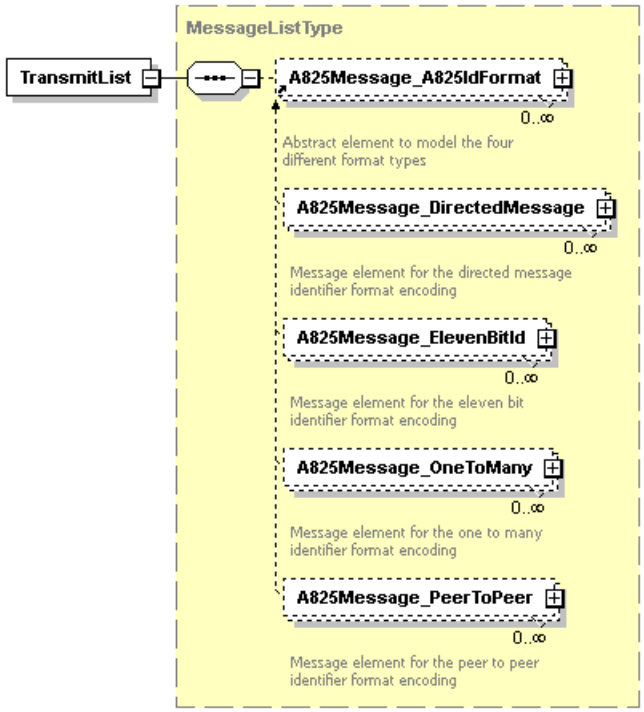
	Name	Refer	Selector	Field(s)	Annotation
unique	InterfaceNamePrimaryKey		CANBusInterface_ClassicalCAN | CANBusInterface_CANFD	@name	

element **Arinc825Profile/LRU/Communication**

diagram	
properties	minOcc 0 maxOcc 1 content complex
children	TransmitList ReceiveList

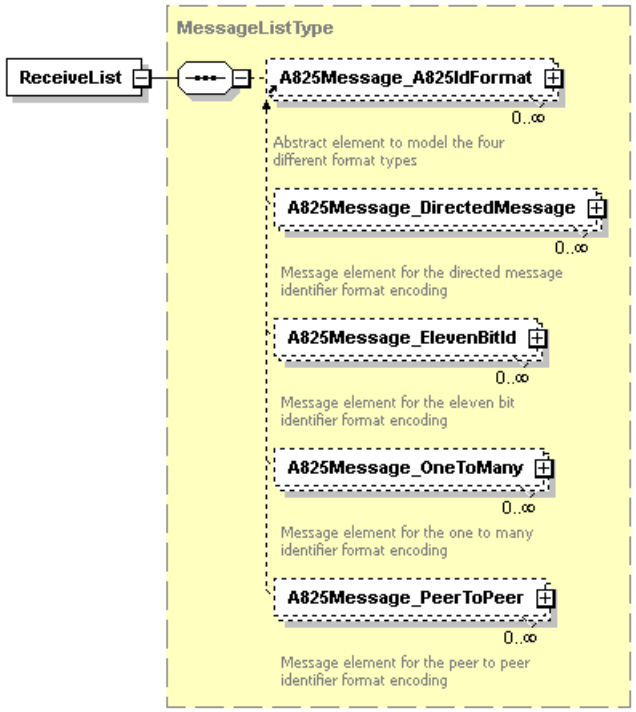
ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **Arinc825Profile/LRU/Communication/TransmitList**

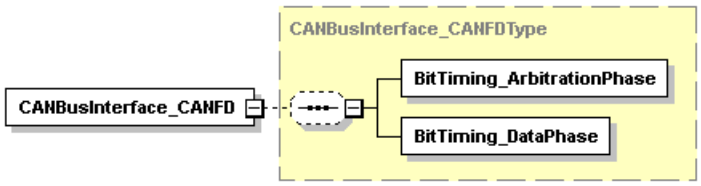
diagram	
type	MessageListType
properties	content complex
children	A825Message_A825IdFormat

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **Arinc825Profile/LRU/Communication/ReceiveList**

diagram	
type	**MessageListType**
properties	content complex
children	**A825Message_A825IdFormat**

element **CANBusInterface_CANFD**

diagram	
type	**CANBusInterface_CANFDType**
substitution group	**CANBusInterface_TypeOfInterface**
properties	content complex
children	**BitTiming_ArbitrationPhase** **BitTiming_DataPhase**

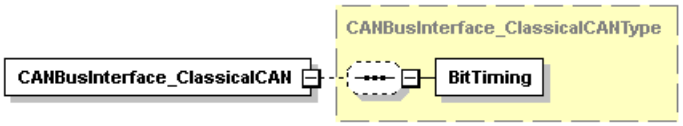
attributes

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation Unique name of the interface, primary key. Used as a foreign key for message interface associations assignments
default_rci	**derived by: xs:string**	required			documentation Identifier Redundancy

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

				Channel Identifier, this may be required on the receive interface in order to correctly calculate the Message Integrity Check (MIC) or to override the global RCI to interface assignment
	nominal_bit_rate	**xs:double**	required	documentation number of bits per time during arbitration phase, independent of the bit encoding/decoding, units are kilobits per second (kbps)
	comment	**xs:string**	optional	documentation User-defined comment (not processed)
	node_id	**xs:unsignedInt**	required	documentation This can be used as the "LRU_Code"
	node_address	**derived by: xs:byte**	required	documentation Used in the DMC identifier format to uniquely address an LRU
	data_bit_rate	**xs:double**	required	documentation number of bits per second during data phase, units are bits per second (bps)
	can_fd_enabled	**xs:boolean**	required	documentation True - The interface is able to receive and transmit messages in both CAN FD and CAN 2.0 formats. False - The interface is able to receive and transmit messages in CAN 2.0 format.
	default_brs	**derived by: xs:string**	required	

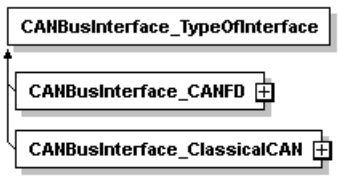
element **CANBusInterface_ClassicalCAN**

diagram	
type	**CANBusInterface_ClassicalCANType**
substitution group	**CANBusInterface_TypeOfInterface**
properties	content complex

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

children	**BitTiming**					
attributes	Name	Type	Use	Default	Fixed	Annotation
	name	**derived by: xs:string**	required			documentation: Unique name of the interface, primary key. Used as a foreign key for message interface associations assignments
	default_rci	**derived by: xs:string**	required			documentation: Identifier Redundancy Channel Identifier, this may be required on the receive interface in order to correctly calculate the Message Integrity Check (MIC) or to override the global RCI to interface assignment
	nominal_bit_rate	**xs:double**	required			documentation: number of bits per time during arbitration phase, independent of the bit encoding/decoding, units are kilobits per second (kbps)
	comment	**xs:string**	optional			documentation: User-defined comment (not processed)
	node_id	**xs:unsignedInt**	required			documentation: This can be used as the "LRU_Code"
	node_address	**derived by: xs:byte**	required			documentation: Used in the DMC identifier format to uniquely address an LRU

element **CANBusInterface_TypeOfInterface**

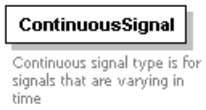
diagram						
type	**CANBusInterface_BaseType**					
properties	content: complex abstract: true					
used by	element: **Arinc825Profile/LRU/CANBusInterfaceList**					
attributes	Name	Type	Use	Default	Fixed	Annotation
	name	**derived by: xs:string**	required			documentation: Unique name of the interface, primary key. Used as a foreign key for

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

				message interface associations assignments
	default_rci	**derived by: xs:string**	required	documentation Identifier Redundancy Channel Identifier, this may be required on the receive interface in order to correctly calculate the Message Integrity Check (MIC) or to override the global RCI to interface assignment
	nominal_bit_rate	**xs:double**	required	documentation number of bits per time during arbitration phase, independent of the bit encoding/decoding, units are kilobits per second (kbps)
	comment	**xs:string**	optional	documentation User-defined comment (not processed)
	node_id	**xs:unsignedInt**	required	documentation This can be used as the "LRU_Code"
	node_address	**derived by: xs:byte**	required	documentation Used in the DMC identifier format to uniquely address an LRU

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **ContinuousSignal**

diagram	
type	**SignalContinuousType**
substitution group	**Signal**
properties	content complex

attributes

Name	Type	Use	Default	Fixed	Annotation
id	**derived by: xs:string**	required			documentation Identifier for unique elements in the xml profile
name	**derived by: xs:string**	required			documentation Name of the item
fid	**derived by: xs:int**	required			documentation Function Identifier, 0 is multicast function id, only used with one to many message identifier type
comment	**xs:string**	optional			documentation User-defined comment (not processed)
max	**derived by: xs:string**	required			documentation Maximum value of the parameter as a floating-point number. This value is a constant used for information and scaling purposes (data type dependant).
min	**derived by: xs:string**	required			documentation Minimum value of the parameter as a floating-point number. This value is a constant used for information and scaling purposes (data type dependant).
pua	**xs:string**	required			documentation Physical unit acronym

annotation	documentation Continuous signal type is for signals that are varying in time

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

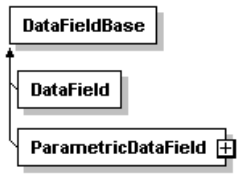
element **DataField**

diagram	
type	**DataFieldType**
substitution group	**DataFieldBase**
properties	content complex

attributes

Name	Type	Use	Default	Fixed	Annotation
byte_length	**derived by: xs:string**	required			documentation Data Field Length, specified in number of bytes (Shall be set to 0 to send a heartbeat or use as a message level functional status of No Computed Data or No Data)
encapsulation_protocol	**derived by: xs:string**	optional			documentation Optional element to indicate if the message is wrapped in the high availability or high integrity protocols, see Section 5.7
payload_protocol	**xs:string**	required			documentation User defined field (not processed)

element **DataFieldBase**

diagram	
type	**DataFieldBaseType**
properties	content complex abstract true
used by	complexType **A825Message_IdFormatType**

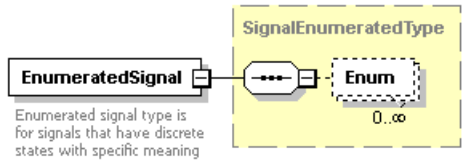
attributes

Name	Type	Use	Default	Fixed	Annotation
byte_length	**derived by: xs:string**	required			documentation Data Field Length, specified in number of bytes (Shall be set to 0 to send a heartbeat or use as a message level functional status of No Computed Data or No Data)
encapsulation_protocol	**derived by: xs:string**	optional			documentation Optional element

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

		to indicate if the message is wrapped in the high availability or high integrity protocols, see Section 5.7

element **EnumeratedSignal**

diagram	
type	**SignalEnumeratedType**
substitution group	**Signal**
properties	content complex
children	**Enum**

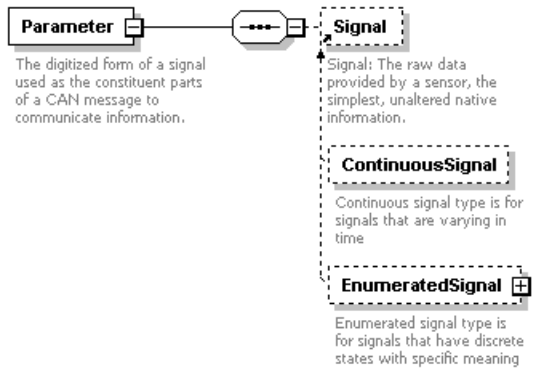
attributes

Name	Type	Use	Default	Fixed	Annotation
id	**derived by: xs:string**	required			documentation Identifier for unique elements in the xml profile
name	**derived by: xs:string**	required			documentation Name of the item
fid	**derived by: xs:int**	required			documentation Function Identifier, 0 is multicast function id, only used with one to many message identifier type
comment	**xs:string**	optional			documentation User-defined comment (not processed)
default_meaning	**xs:string**	optional			documentation Optional meaning in the case that one of the enumerated values is not set. e.g. 'INVALID', 'FAULT', 'ERROR', or 'UNKNOWN'

annotation
documentation
Enumerated signal type is for signals that have discrete states with specific meaning

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **Parameter**

diagram	
type	extension of **ParameterType**
substitution group	**ParameterBase**
properties	content complex
children	**Signal**
used by	element **ParameterValidityStatusSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
signal_id	**derived by: xs:string**	optional			
name	**derived by: xs:string**	required			documentation Name of the item
msb	**derived by: xs:string**	required			
lsb	**derived by: xs:string**	required			
dtn	**derived by: xs:string**	required			documentation data type name
resolution	**xs:double**	optional	1.0		documentation Resolution, default is 1.0
offset	**xs:double**	optional	0		documentation As in y = mx+b, this is the b
comment	**xs:string**	optional			documentation User-defined comment (not processed)

annotation
documentation
The digitized form of a signal used as the constituent parts of a CAN message to communicate information.

attribute **Parameter/@signal_id**

type	restriction of **xs:string**
properties	use optional

facets

Kind	Value	Annotation
minLength	1	
pattern	[a-zA-Z0-9_.:]+	

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **Parameter/@msb**

type	restriction of **xs:string**
properties	use required
facets	Kind Value Annotation pattern [0-9].[0-7] pattern [1-5][0-9].[0-7] pattern 6[0-3].[0-7]

attribute **Parameter/@lsb**

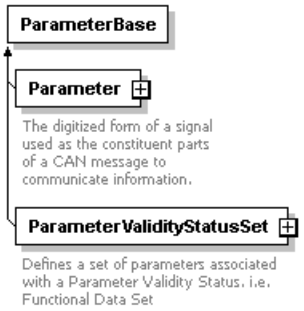
type	restriction of **xs:string**
properties	use required
facets	Kind Value Annotation pattern [0-9].[0-7] pattern [1-5][0-9].[0-7] pattern 6[0-3].[0-7]

attribute **Parameter/@dtn**

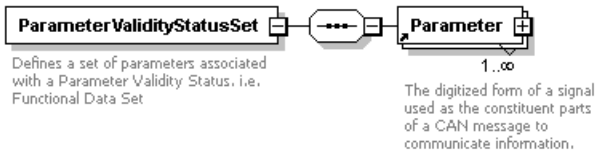
type	restriction of **xs:string**
properties	use required
facets	Kind Value Annotation enumeration NODATA enumeration ENUM enumeration CHAR enumeration UCHAR enumeration ACHAR enumeration SHORT enumeration USHORT enumeration LONG enumeration ULONG enumeration FLOAT enumeration LONG64 enumeration ULONG64 enumeration DOUBLE enumeration OPAQUE enumeration BOOL enumeration BCD
annotation	documentation data type name

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **ParameterBase**

diagram	
type	**ParameterType**
properties	content complex abstract true
used by	complexType **ParametricDataFieldType**

element **ParameterValidityStatusSet**

diagram	
type	extension of **ParameterType**
substitution group	**ParameterBase**
properties	content complex
children	**Parameter**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation Name of the item
lsb	**derived by: xs:string**	required			documentation End position of the datum within the message payload as a decimal number in the range of 0.0-7.7 (bytenumber.bitnumber). Note that bit numbers other than “0” are allowed for enumerated data types only.

annotation

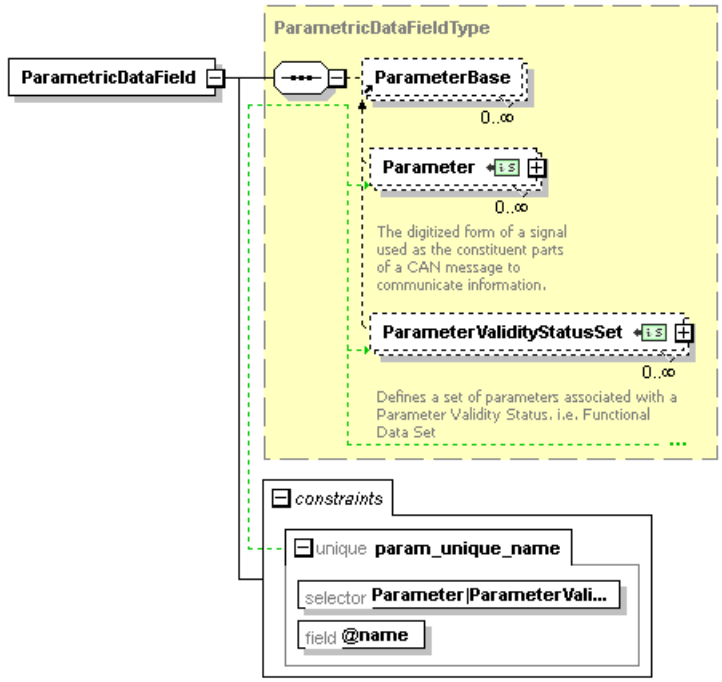
documentation

Defines a set of parameters associated with a Parameter Validity Status, i.e., Functional Data Set

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **ParametricDataField**

diagram



type **ParametricDataFieldType**

substitution group **DataFieldBase**

properties
content complex

children **ParameterBase**

attributes

Name	Type	Use	Default	Fixed	Annotation
byte_length	**derived by: xs:string**	required			documentation Data Field Length, specified in number of bytes (Shall be set to 0 to send a heartbeat or use as a message level functional status of No Computed Data or No Data)
encapsulation_protocol	**derived by: xs:string**	optional			documentation Optional element to indicate if the message is wrapped in the high availability or high integrity protocols, see Section 5.7

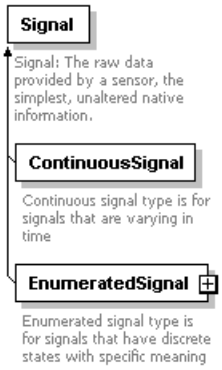
identity constraints

	Name	Refer	Selector	Field(s)	Annotation
unique	param_unique_name		Parameter|ParameterValidityStatusSet|ParameterValidityStatusSet/Parameter	@name	

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

element **Signal**

diagram



type: **SignalType**

properties:
- content: complex
- abstract: true

used by:
- elements: **Parameter** **SignalList**

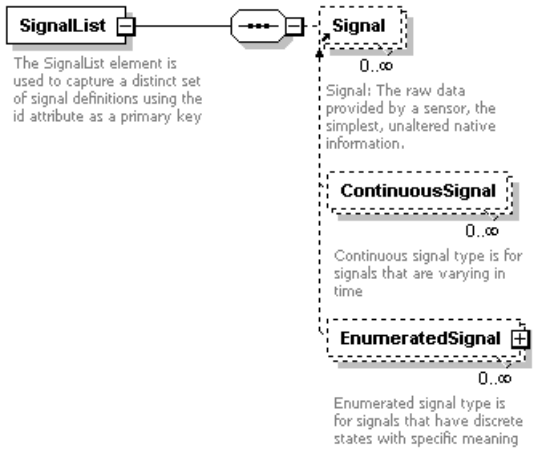
attributes

Name	Type	Use	Default	Fixed	Annotation
id	**derived by: xs:string**	required			documentation: Identifier for unique elements in the xml profile
name	**derived by: xs:string**	required			documentation: Name of the item
fid	**derived by: xs:int**	required			documentation: Function Identifier, 0 is multicast function id, only used with one to many message identifier type
comment	**xs:string**	optional			documentation: User-defined comment (not processed)

annotation
documentation
Signal: The raw data provided by a sensor, the simplest, unaltered native information.

element **SignalList**

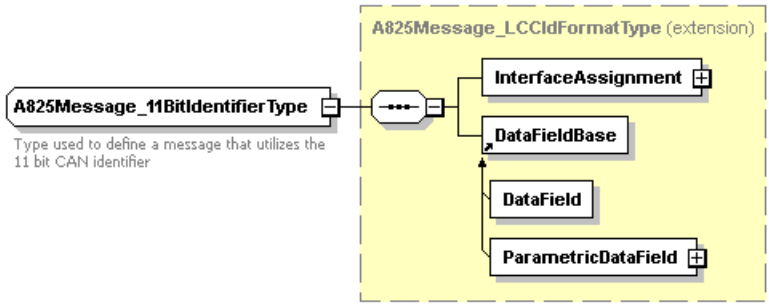
diagram



ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

properties	content complex
children	**Signal**
used by	element **Arinc825Profile**
attributes	Name: comment; Type: **xs:string**; Use: optional; Default: ; Fixed: ; Annotation: documentation: User-defined comment (not processed)
annotation	documentation: The SignalList element is used to capture a distinct set of signal definitions using the id attribute as a primary key

complexType **A825Message_11BitIdentifierType**

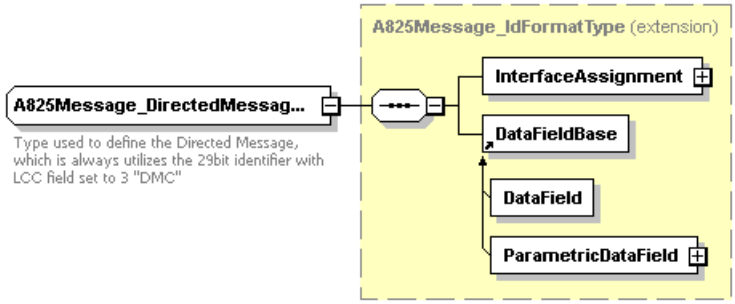
diagram	
type	extension of **A825Message_LCCIdFormatType**
properties	base A825Message_LCCIdFormatType
children	**InterfaceAssignment** **DataFieldBase**
used by	element **A825Message_ElevenBitId**
attributes	Name: name; Type: **derived by: xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Name of the item Name: txp; Type: **derived by: xs:double**; Use: required; Default: ; Fixed: ; Annotation: documentation: Transmit Period, Minimum period at which the message is sent over the bus in milliseconds as a floating-point number (shall be set to “0” for event-driven data). Name: comment; Type: **xs:string**; Use: optional; Default: ; Fixed: ; Annotation: documentation: User-defined comment (not processed) Name: lcc; Type: **derived by: xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Logical Communication Channel Name: label; Type: **derived by: xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Eight-bit identifier segment of the eleven-bit CAN identifier, similar to the A429 label ID.
annotation	documentation: Type used to define a message that utilizes the 11-bit CAN identifier

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **A825Message_11BitIdentifierType/@label**

type	restriction of **xs:string**
properties	use required
facets	Kind / Value / Annotation pattern 0x[0-9a-fA-f] pattern 0x[0-9a-fA-f][0-9a-fA-f] pattern [0-9] pattern [0-9][0-9] pattern 1[0-9][0-9] pattern 2[0-4][0-9] pattern 25[0-5]
annotation	documentation Eight-bit identifier segment of the eleven-bit CAN identifier, similar to the A429 label id.

complexType **A825Message_DirectedMessageType**

diagram	
type	extension of **A825Message_IdFormatType**
properties	base A825Message_IdFormatType
children	**InterfaceAssignment** **DataFieldBase**
used by	element **A825Message_DirectedMessage**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation Name of the item
txp	**derived by: xs:double**	required			documentation Transmit Period, Minimum period at which the message is sent over the bus in milliseconds as a floating point number (shall be set to “0” for event-driven data).
comment	**xs:string**	optional			documentation User-defined comment (not processed)
initial_role	**derived by: xs:string**	required			
source_address	**derived by: xs:unsignedByte**	optional			
source_port_id	**derived by: xs:unsignedByte**	optional			

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

	destination_address	**derived by:** **xs:unsignedByte**	optional
	destination_port_id	**derived by:** **xs:unsignedByte**	optional
annotation	documentation Type used to define the Directed Message, which is always utilizes the 29bit identifier with LCC field set to 3 "DMC"		

attribute **A825Message_DirectedMessageType/@initial_role**

type	restriction of **xs:string**		
properties	use required		
facets	Kind	Value	Annotation
	enumeration	CLIENT	
	enumeration	SERVER	

attribute **A825Message_DirectedMessageType/@source_address**

type	restriction of **xs:unsignedByte**		
properties	use optional		
facets	Kind	Value	Annotation
	minInclusive	0	
	maxInclusive	127	

attribute **A825Message_DirectedMessageType/@source_port_id**

type	restriction of **xs:unsignedByte**		
properties	use optional		
facets	Kind	Value	Annotation
	minInclusive	0	
	maxInclusive	63	

attribute **A825Message_DirectedMessageType/@destination_address**

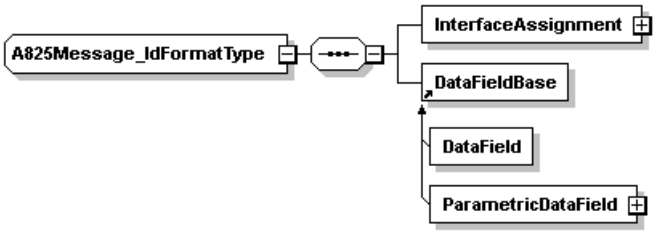
type	restriction of **xs:unsignedByte**		
properties	use optional		
facets	Kind	Value	Annotation
	minInclusive	0	
	maxInclusive	127	

attribute **A825Message_DirectedMessageType/@destination_port_id**

type	restriction of **xs:unsignedByte**		
properties	use optional		
facets	Kind	Value	Annotation
	minInclusive	0	
	maxInclusive	63	

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

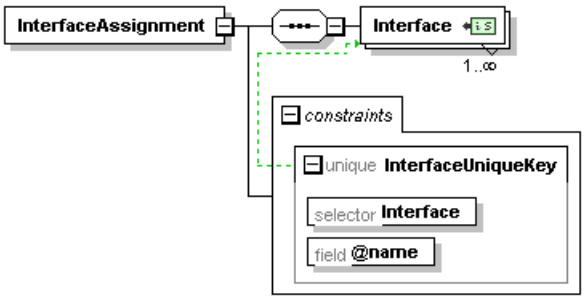
complexType **A825Message_IdFormatType**

diagram	
properties	abstract true
children	**InterfaceAssignment** **DataFieldBase**
used by	element: **A825Message_A825IdFormat** complexTypes: **A825Message_DirectedMessageType**, **A825Message_LCCIdFormatType**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation: Name of the item
txp	**derived by: xs:double**	required			documentation: Transmit Period, Minimum period at which the message is sent over the bus in milliseconds as a floating-point number (shall be set to “0” for event-driven data).
comment	**xs:string**	optional			documentation: User-defined comment (not processed)

element **A825Message_IdFormatType/InterfaceAssignment**

diagram	
properties	content complex
children	**Interface**

identity constraints

	Name	Refer	Selector	Field(s)	Annotation
unique	InterfaceUniqueKey		Interface	@name	

element **A825Message_IdFormatType/InterfaceAssignment/Interface**

diagram	

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

properties	minOcc 1 maxOcc unbounded content complex

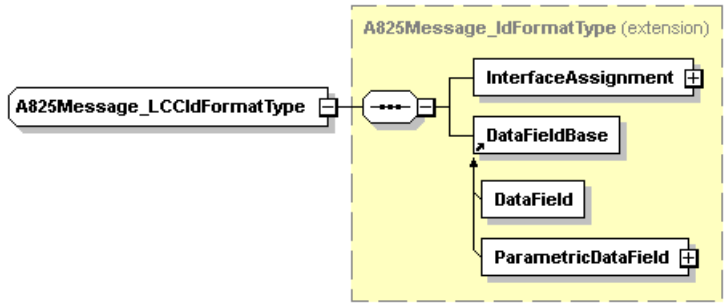
attributes	Name	Type	Use	Default	Fixed	Annotation
	name	**derived by: xs:string**	required			documentation Name of the item
	rci_override	**derived by: xs:string**	optional			documentation Identifier Redundancy Channel Identifier, this may be required on the receive interface in order to correctly calculate the Message Integrity Check (MIC) or to override the global RCI to interface assignment
	brs_override	**derived by: xs:string**	optional			

attribute **A825Message_IdFormatType/InterfaceAssignment/Interface/@brs_override**

type	restriction of **xs:string**
properties	use optional

facets	Kind	Value	Annotation
	enumeration	ENABLED	
	enumeration	DISABLED	

complexType **A825Message_LCCIdFormatType**

diagram	

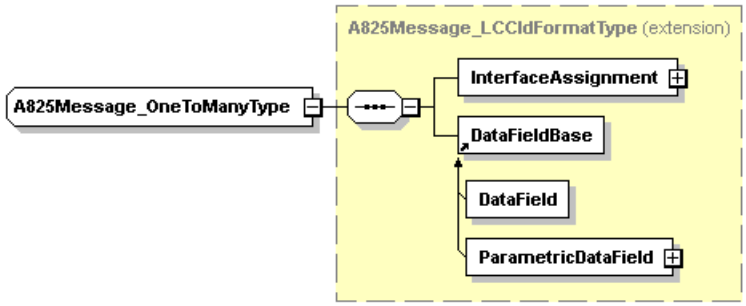
type	extension of **A825Message_IdFormatType**
properties	base A825Message_IdFormatType abstract true
children	**InterfaceAssignment** **DataFieldBase**
used by	complexTypes **A825Message_11BitIdentifierType** **A825Message_OneToManyType** **A825Message_PeerToPeerMessageType**

attributes	Name	Type	Use	Default	Fixed	Annotation
	name	**derived by: xs:string**	required			documentation Name of the item
	txp	**derived by: xs:double**	required			documentation Transmit Period, Minimum period at which the message is sent

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

						over the bus in milliseconds as a floating-point number (shall be set to “0” for event-driven data).
	comment	**xs:string**	optional			documentation User-defined comment (not processed)
	lcc	**derived by: xs:string**	required			documentation Logical Communication Channel

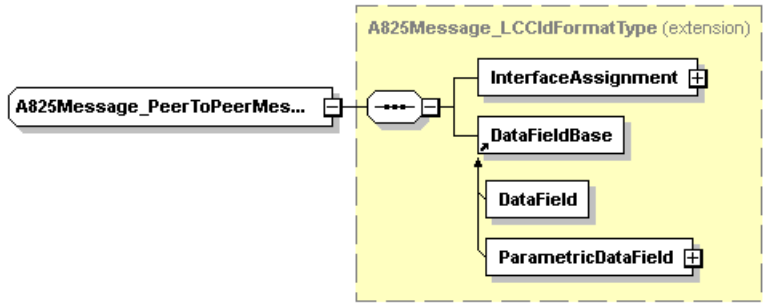
complexType **A825Message_OneToManyType**

diagram						
type	extension of **A825Message_LCCIdFormatType**					
properties	base A825Message_LCCIdFormatType					
children	**InterfaceAssignment** **DataFieldBase**					
used by	element **A825Message_OneToMany**					
attributes	Name	Type	Use	Default	Fixed	Annotation
	name	**derived by: xs:string**	required			documentation Name of the item
	txp	**derived by: xs:double**	required			documentation Transmit Period, Minimum period at which the message is sent over the bus in milliseconds as a floating point number (shall be set to “0” for event-driven data).
	comment	**xs:string**	optional			documentation User-defined comment (not processed)
	lcc	**derived by: xs:string**	required			documentation Logical Communication Channel
	fid	**derived by: xs:int**	required			documentation Function Identifier, 0 is multicast function id, only used with one to many message identifier type

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

	lcl	**derived by: xs:int**	required	documentation Identifier Local Bit, only used with one to many message identifier type
	pvt	**derived by: xs:int**	required	documentation Identifier Private Bit, only used with one to many message identifier type
	doc	**derived by: xs:int**	required	documentation Identifier Data Object Code, only used with one to many message identifier type

complexType **A825Message_PeerToPeerMessageType**

diagram	
type	extension of **A825Message_LCCIdFormatType**
properties	base A825Message_LCCIdFormatType
children	**InterfaceAssignment** **DataFieldBase**
used by	element **A825Message_PeerToPeer**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation Name of the item
txp	**derived by: xs:double**	required			documentation Transmit Period, Minimum period at which the message is sent over the bus in milliseconds as a floating-point number (shall be set to “0” for event-driven data).
comment	**xs:string**	optional			documentation User-defined comment (not processed)
lcc	**derived by: xs:string**	required			documentation Logical Communication Channel
fid	**derived by: xs:int**	required			documentation Function Identifier, 0 is multicast function id, only

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

				used with one to many message identifier type
	smt	**derived by: xs:int**	optional	
	lcl	**derived by: xs:int**	optional	documentation Identifier Local Bit, only used with one to many message identifier type
	pvt	**derived by: xs:int**	optional	documentation Identifier Private Bit, only used with one to many message identifier type
	server_fid	**derived by: xs:unsignedInt**	optional	
	sid	**xs:int**	optional	documentation Value of the Server ID (SID) field as an integer number in the range of 0-511. This attribute is used for peer-to-peer communication parameters only.

attribute **A825Message_PeerToPeerMessageType/@smt**

type	restriction of **xs:int**		
properties	use optional		
facets	Kind	Value	Annotation
	minInclusive	0	
	maxInclusive	1	

attribute **A825Message_PeerToPeerMessageType/@server_fid**

type	restriction of **xs:unsignedInt**		
properties	use optional		
facets	Kind	Value	Annotation
	minInclusive	0	
	maxInclusive	127	

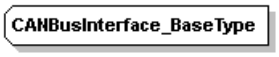
complexType **CanBitTimingType**

diagram						
used by	elements	**CANBusInterface_ClassicalCANType/BitTiming** **CANBusInterface_CANFDType/BitTiming_ArbitrationPhase** **CANBusInterface_CANFDType/BitTiming_DataPhase**				
attributes	Name	Type	Use	Default	Fixed	Annotation
	sample_point	**derived by: xs:double**	required			documentation Defined as percent of the bit period, used to calculate tseg1 and tseg2
	resync_jump_width	**derived by: xs:double**	required			documentation Defined as percent of the bit

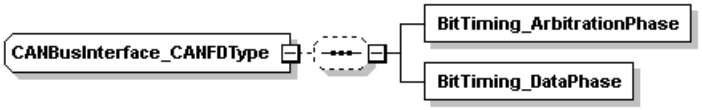
ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

	propagation_delay	**derived by: xs:double**	optional		period documentation Defined as percent of the bit period
	sample_mode	**derived by: xs:string**	optional	SINGLE	documentation Single or Triple

complexType **CANBusInterface_BaseType**

diagram						
properties	abstract true					
used by	element CANBusInterface_TypeOfInterface complexTypes CANBusInterface_CANFDType CANBusInterface_ClassicalCANType					
attributes	Name	Type	Use	Default	Fixed	Annotation
	name	**derived by: xs:string**	required			documentation Unique name of the interface, primary key. Used as a foreign key for message interface associations assignments
	default_rci	**derived by: xs:string**	required			documentation Identifier Redundancy Channel Identifier, this may be required on the receive interface in order to correctly calculate the Message Integrity Check (MIC) or to override the global RCI to interface assignment
	nominal_bit_rate	**xs:double**	required			documentation number of bits per time during arbitration phase, independent of the bit encoding/decoding, units are kilobits per second (kbps)
	comment	**xs:string**	optional			documentation User-defined comment (not processed)
	node_id	**xs:unsignedInt**	required			documentation This can be used as the "LRU_Code"
	node_address	**derived by: xs:byte**	required			documentation Used in the DMC identifier format to uniquely address an LRU

complexType **CANBusInterface_CANFDType**

diagram	
type	extension of CANBusInterface_BaseType
properties	base CANBusInterface_BaseType

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

children	**BitTiming_ArbitrationPhase** **BitTiming_DataPhase**					
used by	element **CANBusInterface_CANFD**					
attributes	Name	Type	Use	Default	Fixed	Annotation
	name	**derived by: xs:string**	required			documentation Unique name of the interface, primary key. Used as a foreign key for message interface associations assignments
	default_rci	**derived by: xs:string**	required			documentation Identifier Redundancy Channel Identifier, this may be required on the receive interface in order to correctly calculate the Message Integrity Check (MIC) or to override the global RCI to interface assignment
	nominal_bit_rate	**xs:double**	required			documentation number of bits per time during arbitration phase, independent of the bit encoding/decoding, units are kilobits per second (kbps)
	comment	**xs:string**	optional			documentation User-defined comment (not processed)
	node_id	**xs:unsignedInt**	required			documentation This can be used as the "LRU_Code"
	node_address	**derived by: xs:byte**	required			documentation Used in the DMC identifier format to uniquely address an LRU
	data_bit_rate	**xs:double**	required			documentation number of bits per second during data phase, units are bits per second (bps)
	can_fd_enabled	**xs:boolean**	required			documentation True - The interface is able to receive and transmit messages in both CAN FD and CAN 2.0 formats. False - The interface is able to receive and transmit messages in CAN 2.0 format.
	default_brs	**derived by: xs:string**	required			

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **CANBusInterface_CANFDType/@default_brs**

type	restriction of **xs:string**
properties	use required
facets	Kind: enumeration, Value: ENABLED Kind: enumeration, Value: DISABLED

element **CANBusInterface_CANFDType/BitTiming_ArbitrationPhase**

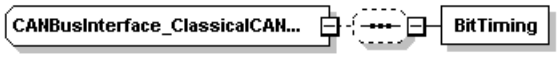
diagram	BitTiming_ArbitrationPhase
type	**CanBitTimingType**
properties	content complex
attributes	Name: sample_point, Type: **derived by: xs:double**, Use: required, Default: , Fixed: , Annotation: documentation: Defined as percent of the bit period, used to calculate tseg1 and tseg2 Name: resync_jump_width, Type: **derived by: xs:double**, Use: required, Default: , Fixed: , Annotation: documentation: Defined as percent of the bit period Name: propagation_delay, Type: **derived by: xs:double**, Use: optional, Default: , Fixed: , Annotation: documentation: Defined as percent of the bit period Name: sample_mode, Type: **derived by: xs:string**, Use: optional, Default: SINGLE, Fixed: , Annotation: documentation: Single or Triple

element **CANBusInterface_CANFDType/BitTiming_DataPhase**

diagram	BitTiming_DataPhase
type	**CanBitTimingType**
properties	content complex
attributes	Name: sample_point, Type: **derived by: xs:double**, Use: required, Default: , Fixed: , Annotation: documentation: Defined as percent of the bit period, used to calculate tseg1 and tseg2 Name: resync_jump_width, Type: **derived by: xs:double**, Use: required, Default: , Fixed: , Annotation: documentation: Defined as percent of the bit period Name: propagation_delay, Type: **derived by: xs:double**, Use: optional, Default: , Fixed: , Annotation: documentation: Defined as percent of the bit period Name: sample_mode, Type: **derived by: xs:string**, Use: optional, Default: SINGLE, Fixed: , Annotation: documentation: Single or Triple

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

complexType **CANBusInterface_ClassicalCANType**

diagram	
type	extension of **CANBusInterface_BaseType**
properties	base CANBusInterface_BaseType
children	**BitTiming**
used by	element **CANBusInterface_ClassicalCAN**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**derived by: xs:string**	required			documentation: Unique name of the interface, primary key. Used as a foreign key for message interface associations assignments
default_rci	**derived by: xs:string**	required			documentation: Identifier Redundancy Channel Identifier, this may be required on the receive interface in order to correctly calculate the Message Integrity Check (MIC) or to override the global RCI to interface assignment
nominal_bit_rate	**xs:double**	required			documentation: number of bits per time during arbitration phase, independent of the bit encoding/decoding, units are kilobits per second (kbps)
comment	**xs:string**	optional			documentation: User-defined comment (not processed)
node_id	**xs:unsignedInt**	required			documentation: This can be used as the "LRU_Code"
node_address	**derived by: xs:byte**	required			documentation: Used in the DMC identifier format to uniquely address an LRU

element **CANBusInterface_ClassicalCANType/BitTiming**

diagram	
type	**CanBitTimingType**
properties	content complex

attributes

Name	Type	Use	Default	Fixed	Annotation
sample_point	**derived by: xs:double**	required			documentation: Defined as percent of the bit period, used to calculate tseg1

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

	resync_jump_width	**derived by: xs:double**	required		and tseg2 documentation Defined as percent of the bit period
	propagation_delay	**derived by: xs:double**	optional		documentation Defined as percent of the bit period
	sample_mode	**derived by: xs:string**	optional	SINGLE	documentation Single or Triple

complexType **DataFieldBaseType**

diagram	
properties	abstract true
used by	element **DataFieldBase** complexTypes **DataFieldType** **ParametricDataFieldType**

attributes

Name	Type	Use	Default	Fixed	Annotation
byte_length	**derived by: xs:string**	required			documentation Data Field Length, specified in number of bytes (Shall be set to 0 to send a heartbeat or use as a message level functional status of No Computed Data or No Data)
encapsulation_protocol	**derived by: xs:string**	optional			documentation Optional element to indicate if the message is wrapped in the high availability or high integrity protocols, see Section 5.7

attribute **DataFieldBaseType/@byte_length**

type	restriction of **xs:string**
properties	use required

facets

Kind	Value	Annotation
enumeration	DYNAMIC	documentation The data field length can vary between 0 to 64 bytes
enumeration	0	documentation DLC 0
enumeration	1	documentation DLC 1
enumeration	2	documentation DLC 2
enumeration	3	documentation DLC 3
enumeration	4	documentation DLC 4
enumeration	5	documentation DLC 5

	enumeration	6	documentation DLC 6
	enumeration	7	documentation DLC 7
	enumeration	8	documentation DLC 8
	enumeration	12	documentation DLC 9
	enumeration	16	documentation DLC 10
	enumeration	20	documentation DLC 11
	enumeration	24	documentation DLC 12
	enumeration	32	documentation DLC 13
	enumeration	48	documentation DLC 14
	enumeration	64	documentation DLC 15
annotation	documentation Data Field Length, specified in number of bytes (Shall be set to 0 to send a heartbeat or use as a message level functional status of No Computed Data or No Data)		

attribute **DataFieldBaseType/@encapsulation_protocol**

type	restriction of **xs:string**		
properties	use optional		
facets	Kind	Value	Annotation
	enumeration	HIGH_AVAILABILITY	
	enumeration	HIGH_INTEGRITY	
annotation	documentation Optional element to indicate if the message is wrapped in the high availability or high integrity protocols, see Section 5.7		

complexType **DataFieldType**

diagram						
type	extension of **DataFieldBaseType**					
properties	base DataFieldBaseType					
used by	element **DataField**					
attributes	Name	Type	Use	Default	Fixed	Annotation
	byte_length	**derived by: xs:string**	required			documentation Data Field Length, specified in number of bytes (Shall be set to 0 to send a heartbeat or use as a message level functional status of No Computed Data or No Data)
	encapsulation_protocol	**derived by: xs:string**	optional			documentation Optional element to indicate if the message is wrapped in the high availability or high integrity protocols, see

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

	payload_protocol	**xs:string**	required	Section 5.7 documentation User defined field (not processed)

attribute **DataFieldType/@payload_protocol**

type	**xs:string**
properties	use required
annotation	documentation User defined field (not processed)

complexType **MessageListType**

diagram	
properties	abstract false
children	A825Message_A825IdFormat
used by	elements Arinc825Profile/LRU/Communication/ReceiveList Arinc825Profile/LRU/Communication/TransmitList

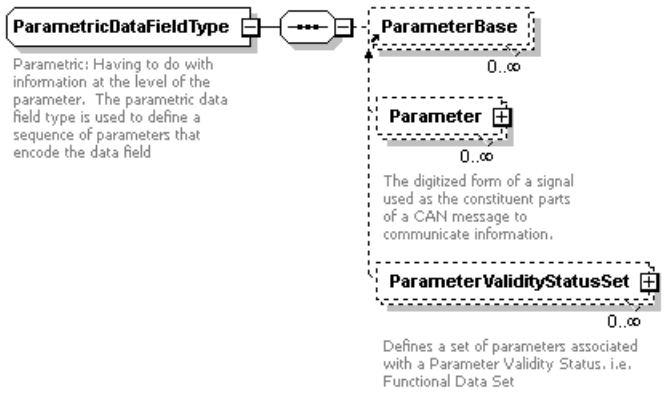
complexType **ParameterType**

diagram	
properties	abstract true
used by	elements Parameter ParameterBase ParameterValidityStatusSet

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

complexType **ParametricDataFieldType**

diagram



type: extension of **DataFieldBaseType**

properties: base DataFieldBaseType

children: **ParameterBase**

used by: element **ParametricDataField**

attributes

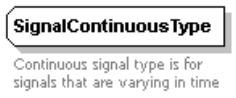
Name	Type	Use	Default	Fixed	Annotation
byte_length	**derived by: xs:string**	required			documentation: Data Field Length, specified in number of bytes (Shall be set to 0 to send a heartbeat or use as a message level functional status of No Computed Data or No Data)
encapsulation_protocol	**derived by: xs:string**	optional			documentation: Optional element to indicate if the message is wrapped in the high availability or high integrity protocols, see Section 5.7

annotation

documentation
Parametric: Having to do with information at the level of the parameter. The parametric data field type is used to define a sequence of parameters that encode the data field

complexType **SignalContinuousType**

diagram



type: extension of **SignalType**

properties: base SignalType

used by: element **ContinuousSignal**

attributes

Name	Type	Use	Default	Fixed	Annotation
id	**derived by: xs:string**	required			documentation: Identifier for unique elements in the xml profile

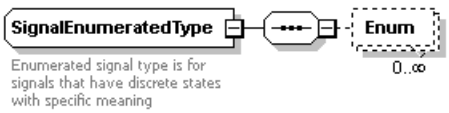
ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

	name	**derived by: xs:string**	required	documentation Name of the item
	fid	**derived by: xs:int**	required	documentation Function Identifier, 0 is multicast function id, only used with one to many message identifier type
	comment	**xs:string**	optional	documentation User-defined comment (not processed)
	max	**derived by: xs:string**	required	documentation Maximum value of the parameter as a floating-point number. This value is a constant used for information and scaling purposes (data type dependent).
	min	**derived by: xs:string**	required	documentation Minimum value of the parameter as a floating-point number. This value is a constant used for information and scaling purposes (data type dependent).
	pua	**xs:string**	required	documentation Physical unit acronym
annotation	documentation Continuous signal type is for signals that are varying in time			

attribute **SignalContinuousType/@pua**

type	**xs:string**
properties	use required
annotation	documentation Physical unit acronym

complexType **SignalEnumeratedType**

diagram	
type	extension of **SignalType**
properties	base SignalType
children	**Enum**
used by	element **EnumeratedSignal**

attributes	Name	Type	Use	Default	Fixed	Annotation
	id	**derived by: xs:string**	required			documentation Identifier for

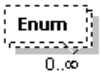
ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

				unique elements in the xml profile
	name	**derived by: xs:string**	required	documentation Name of the item
	fid	**derived by: xs:int**	required	documentation Function Identifier, 0 is multicast function id, only used with one to many message identifier type
	comment	**xs:string**	optional	documentation User-defined comment (not processed)
	default_meaning	**xs:string**	optional	documentation Optional meaning in the case that one of the enumerated values is not set, e.g., 'INVALID', 'FAULT', 'ERROR', or 'UNKNOWN'
annotation	documentation Enumerated signal type is for signals that have discrete states with specific meaning			

attribute **SignalEnumeratedType/@default_meaning**

type	**xs:string**
properties	use optional
annotation	documentation Optional meaning in the case that one of the enmerated values is not set, e.g., 'INVALID', 'FAULT', 'ERROR', or 'UNKNOWN'

element **SignalEnumeratedType/Enum**

diagram						
properties	minOcc 0 maxOcc unbounded content complex					
attributes	Name	Type	Use	Default	Fixed	Annotation
	value	**derived by: xs:string**	required			
	meaning	**xs:string**	required			
	comment	**xs:string**	optional			documentation User-defined comment (not processed)

attribute **SignalEnumeratedType/Enum/@value**

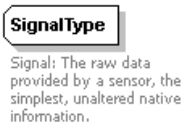
type	restriction of **xs:string**		
properties	use required		
facets	Kind	Value	Annotation
	pattern	0x[0-9A-Fa-f]+|[0-9]+	

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **SignalEnumeratedType/Enum/@meaning**

type	**xs:string**
properties	use required

complexType **SignalType**

diagram	
used by	element **Signal** complexTypes **SignalContinuousType** **SignalEnumeratedType**

attributes

Name	Type	Use	Default	Fixed	Annotation
id	**derived by: xs:string**	required			documentation Identifier for unique elements in the xml profile
name	**derived by: xs:string**	required			documentation Name of the item
fid	**derived by: xs:int**	required			documentation Function Identifier, 0 is multicast function id, only used with one to many message identifier type
comment	**xs:string**	optional			documentation User-defined comment (not processed)

annotation
documentation
Signal: The raw data provided by a sensor, the simplest, unaltered native information.

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **can_fd_enabled**

type	**xs:boolean**
used by	complexType **CANBusInterface_CANFDType**
annotation	documentation True - The interface is able to receive and transmit messages in both CAN FD and CAN 2.0 formats. False - The interface is able to receive and transmit messages in CAN 2.0 format.

attribute **comment**

type	**xs:string**
used by	elements **Arinc825Profile** **Arinc825Profile/LRU/CANBusInterfaceList** **SignalEnumeratedType/Enum** **Arinc825Profile/LRU** **Parameter** **SignalList** complexTypes **A825Message_IdFormatType** **CANBusInterface_BaseType** **SignalType**
annotation	documentation User-defined comment (not processed)

attribute **data_bit_rate**

type	**xs:double**
used by	complexType **CANBusInterface_CANFDType**
annotation	documentation number of bits per second during data phase, units are bits per second (bps)

attribute **default_rci**

type	restriction of **xs:string**
used by	complexType **CANBusInterface_BaseType**
facets	Kind / Value / Annotation enumeration A enumeration B enumeration C enumeration D
annotation	documentation Identifier Redundancy Channel Identifier, this may be required on the receive interface in order to correctly calculate the Message Integrity Check (MIC) or to override the global RCI to interface assignment

attribute **doc**

type	restriction of **xs:int**
used by	complexType **A825Message_OneToManyType**
facets	Kind / Value / Annotation minInclusive 0 maxInclusive 16383
annotation	documentation Identifier Data Object Code, only used with one to many message identifier type

attribute **fid**

type	restriction of **xs:int**
used by	complexTypes **A825Message_OneToManyType** **A825Message_PeerToPeerMessageType** **SignalType**
facets	Kind / Value / Annotation minInclusive 0 maxInclusive 127
annotation	documentation Function Identifier, 0 is multicast function id, only used with one to many message identifier type

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **fmc_11bit_id**

type	**xs:byte**
annotation	documentation Use to define the message identifier value only with an 11-bit identifier type Frame Migration Channel message. Since first three bits of the identifier are LCC=FMC only eight bits are available for definition

attribute **fsb**

type	restriction of **xs:string**
properties	default 0
facets	Kind / Value / Annotation enumeration 0 enumeration 1
annotation	documentation Identifier Functional Status Bit, only used with one to many message identifier type

attribute **id**

type	restriction of **xs:string**
used by	complexType **SignalType**
facets	Kind / Value / Annotation minLength 1 maxLength 39
annotation	documentation Identifier for unique elements in the xml profile

attribute **latency_max**

type	restriction of **xs:double**
facets	Kind / Value / Annotation minInclusive -1.0
annotation	documentation Maximum amount of time, in milliseconds as a floating-point number, that a message is allowed to be delayed before it becomes stale. -1 indicates there is no latency associated

attribute **lcc**

type	restriction of **xs:string**
used by	complexType **A825Message_LCCIdFormatType**
facets	Kind / Value / Annotation enumeration ECC enumeration NOC enumeration DMC enumeration NSC enumeration UDC enumeration TMC enumeration FMC
annotation	documentation Logical Communication Channel

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **lcl**

type	restriction of **xs:int**
used by	complexTypes **A825Message_OneToManyType** **A825Message_PeerToPeerMessageType**
facets	Kind / Value / Annotation minInclusive 0 maxInclusive 1
annotation	documentation Identifier Local Bit, only used with one to many message identifier type

attribute **lsb**

type	restriction of **xs:string**
used by	element **ParameterValidityStatusSet**
facets	Kind / Value / Annotation pattern [0-9]+.[0-7]
annotation	documentation End position of the datum within the message payload as a decimal number in the range of 0.0-7.7 (bytenumber.bitnumber). Note that bit numbers other than “0” are allowed for enumerated data types only.

attribute **max**

type	restriction of **xs:string**
used by	complexType **SignalContinuousType**
facets	Kind / Value / Annotation pattern [0-9]+.[0-9]+ pattern INF pattern [0-9]+ pattern 0x[0-9A-Fa-f]+ pattern [+-]?[0-9].[0-9][eE][+-]?[0-9]+
annotation	documentation Maximum value of the parameter as a floating-point number. This value is a constant used for information and scaling purposes (data type dependent).

attribute **min**

type	restriction of **xs:string**
used by	complexType **SignalContinuousType**
facets	Kind / Value / Annotation pattern [+-]?[0-9]+.[0-9]+ pattern -INF pattern [0-9]+ pattern 0x[0-9A-Fa-f]+ pattern [+-]?[0-9].[0-9][eE][+-]?[0-9]+
annotation	documentation Minimum value of the parameter as a floating-point number. This value is a constant used for information and scaling purposes (data type dependent).

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **msb**

type	restriction of **xs:string**
facets	Kind: pattern; Value: [0-9]+.[0-7]; Annotation:
annotation	documentation: Start position of the datum within the message payload as a decimal number in the range of 0.0-7.7 (bytenumber.bitnumber). Note that bit numbers other than “0” are allowed for enumerated data types only.

attribute **name**

type	restriction of **xs:string**
used by	elements: **Arinc825Profile** **A825Message_IdFormatType/InterfaceAssignment/Interface** **Arinc825Profile/LRU** **Parameter** **ParameterValidityStatusSet**; complexTypes: **A825Message_IdFormatType** **CANBusInterface_BaseType** **SignalType**
facets	Kind: pattern; Value: [a-zA-Z0-9_]+; Annotation:
annotation	documentation: Name of the item

attribute **node_address**

type	restriction of **xs:byte**
used by	complexType: **CANBusInterface_BaseType**
facets	Kind: minInclusive; Value: 1; Annotation: Kind: maxInclusive; Value: 127; Annotation:
annotation	documentation: Used in the DMC identifier format to uniquely address an LRU

attribute **node_id**

type	**xs:unsignedInt**
used by	complexType: **CANBusInterface_BaseType**
annotation	documentation: This can be used as the "LRU_Code"

attribute **nominal_bit_rate**

type	**xs:double**
used by	complexType: **CANBusInterface_BaseType**
annotation	documentation: number of bits per second during arbitration phase, units are bits per second (bps)

attribute **offset**

type	**xs:double**
properties	default: 0
used by	element: **Parameter**
annotation	documentation: As in y = mx+b, this is the b

attribute **pua**

type	**xs:string**
annotation	documentation: Physical unit acronym

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **pvs_lsb**

type	restriction of **xs:string**
facets	Kind Value Annotation pattern [0-7].[0-7]
annotation	documentation Parameter Validity Status End Position is a two-bit enumeration, 00-Failure Warning, 01-No Computed Data, 10-Functional Test, 11-Normal Operation

attribute **pvs_msb**

type	restriction of **xs:string**
facets	Kind Value Annotation pattern [0-7].[0-7]
annotation	documentation Parameter Validity Status Start Position is a two-bit enumeration, 00-Failure Warning, 01-No Computed Data, 10-Functional Test, 11-Normal Operation

attribute **pvt**

type	restriction of **xs:int**
used by	complexTypes **A825Message_OneToManyType** **A825Message_PeerToPeerMessageType**
facets	Kind Value Annotation minInclusive 0 maxInclusive 1
annotation	documentation Identifier Private Bit, only used with one to many message identifier type

attribute **rci_override**

type	restriction of **xs:string**
used by	element **A825Message_IdFormatType/InterfaceAssignment/Interface**
facets	Kind Value Annotation enumeration A enumeration B enumeration C enumeration D
annotation	documentation Identifier Redundancy Channel Identifier, this may be required on the receive interface in order to correctly calculate the Message Integrity Check (MIC) or to override the global RCI to interface assignment

attribute **resolution**

type	**xs:double**
properties	default 1.0
used by	element **Parameter**
annotation	documentation Resolution, default is 1.0

attribute **sid**

type	**xs:int**
used by	complexType **A825Message_PeerToPeerMessageType**
annotation	documentation Value of the Server ID (SID) field as an integer number in the range of 0-511. This attribute is used for peer-to-peer communication parameters only.

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **txp**

type	restriction of **xs:double**
used by	complexType **A825Message_IdFormatType**
facets	Kind: minInclusive; Value: 0.0; Annotation:
annotation	documentation: Transmit Period, Minimum period at which the parameter is sent over the bus in milliseconds as a floating-point number (shall be set to “0” for event-driven data).

attribute **version**

type	**xs:double**
properties	fixed 4
used by	element **Arinc825Profile**
annotation	documentation: Identification number of the ARINC-825 supplement number that this schema aligns to

attributeGroup **bit_timing_attrib_grp**

used by	complexType **CanBitTimingType**					
attributes	Name	Type	Use	Default	Fixed	Annotation
	sample_point	**derived by: xs:double**	required			documentation: Defined as percent of the bit period, used to calculate tseg1 and tseg2
	resync_jump_width	**derived by: xs:double**	required			documentation: Defined as percent of the bit period
	propagation_delay	**derived by: xs:double**	optional			documentation: Defined as percent of the bit period
	sample_mode	**derived by: xs:string**	optional	SINGLE		documentation: Single or Triple

attribute **bit_timing_attrib_grp/@sample_point**

type	restriction of **xs:double**
properties	use required
facets	Kind: minInclusive; Value: 0.0; Annotation: Kind: maxInclusive; Value: 100.0; Annotation:
annotation	documentation: Defined as percent of the bit period, used to calculate tseg1 and tseg2

attribute **bit_timing_attrib_grp/@resync_jump_width**

type	restriction of **xs:double**
properties	use required
facets	Kind: minInclusive; Value: 0.0; Annotation: Kind: maxInclusive; Value: 100.0; Annotation:
annotation	documentation: Defined as percent of the bit period

ATTACHMENT 1
COMMUNICATION PROFILE DATABASE

attribute **bit_timing_attrib_grp/@propagation_delay**

type	restriction of **xs:double**
properties	use optional
facets	Kind / Value / Annotation minInclusive 0.0 maxInclusive 100.0
annotation	documentation Defined as percent of the bit period

attribute **bit_timing_attrib_grp/@sample_mode**

type	restriction of **xs:string**
properties	use optional default SINGLE
facets	Kind / Value / Annotation enumeration SINGLE enumeration TRIPPLE
annotation	documentation Single or Triple

XML Schema documentation generated by **XMLSpy** Schema Editor **http://www.altova.com/xmlspy**

ATTACHMENT 2 GLOSSARY

Acknowledgement Error

An error generated by a transmitting CAN interface when a message has been transmitted on the bus and the recessive bit in the ACK Slot has not been driven dominant by another CAN interface.

ACK Field

The portion of the CAN Header that contains the acknowledgement slot (ACL Slot) and acknowledgement delimiter (ACK delimiter) of a transmitted message. Both bits are transmitted recessive; receiving nodes will drive the ACK Slot bit to dominant.

Aperiodic

Appearing or occurring at irregular intervals. Event based data.

Arbitration Field

The portion of a CAN message that is monitored during transmission on the bus and compared to what was published. Detection of a dominant bit in an expected recessive bit position results in a loss of arbitration.

ASIC

Application Specific Integrated Circuit: A microchip designed for a special application, such as a particular kind of transmission protocol.

Attenuation

The gradual loss in signal strength through the bus medium (i.e., copper, fiber, etc.).

Bandwidth

The rate of data transfer, bit rate, or throughput, measured in bits per second (bps).

Base Frame

The 11-bit CAN identifier type. ARINC 825 CAN nodes do not use this type.

Bit-Stuffing

Bit(s) added to the CAN Message of the opposite value after six consecutive bits of one value are transmitted on the bus.

Bit-Stuffing Error

An error generated when six or more consecutive bits of the same logic value are detected in a message transmitted on the bus.

Block Data

CAN data that exceeds the 8 byte size of a CAN message. This data is presented as a block of data that must be subdivided into at most 8 byte-sized messages subsequently transmitted across the bus until the entire block of data has been transmitted.

ATTACHMENT 2
GLOSSARY

Bus-Off

The state of a node when the TEC exceeds 255 and has logically disconnected from the network. In this state the node neither transmits nor receives CAN messages and does not send error frames.

Capacitance

The ratio of the change in an electric charge in a system to the corresponding change in its electric potential.

CAN

Controller Area Network: A network standard designed to allow microcontrollers and devices to communicate with each other within a system.

CAN Base Frame Migration Channel

The last defined Logical Communication Channel with a value of 7 (0b111). This lowest priority channel is provided to account for applications utilizing CAN base frames with 11-bit identifiers. Data on this channel need only comply with the LCC assignment to ensure interoperability with potential extended frame CAN messages on the network.

Client

The initiator of a node service. A Client can request data from the Server and operates on the received data.

Common Mode Rejection

The measure of the differential gain over the common-mode gain. Typically presented as a ratio.

Control Field

That portion of the CAN header that communicates message specific information about the length of the message (Data Length Code). Contains reserved bits for future use.

CRC Error

An error generated when a transmitted CRC does not match the CRC value calculated by a receiving node.

Cyclic-Redundancy Check (CRC) field

The portion of the CAN Header that contains the CRC sequence for the message. Includes a CRC delimiter bit.

Data Frame/Data Field/Data Set

The portion of a CAN message that contains the parametric information to be communicated across the network. Also referred to as payload.

Data Length Code (DLC)

Located in the Control Field of the CAN Header the portion of the CAN message that communicates the Data Frame size.

ATTACHMENT 2
GLOSSARY

Data Link

An electronic connection for the exchange of information.

Data Link Layer (DLL)

The protocol layer of the OSI model (Layer 2) that transfers data between adjacent network nodes in a wide area network or between nodes on the same local area network segment.

Data Object Code (DOC)

Bits 15 to 2 of the extended CAN message identifier used to provide specific information on the content of the message and/or additional destination information.

Data Rate

The maximum speed at which data is transferred across the bus between devices.

Destination

The target node of a particular CAN message; in a broadcast message, the destination will be multiple nodes.

Dominant

The most important, powerful, or influential. For CAN, this translates to a bit value of zero (0).

EMI

Electromagnetic Interference. The disruption of operation of an electronic device when it is in the vicinity of an electromagnetic field (EM field) in the Radio Frequency (RF) spectrum that is caused by another electronic source (e.g., Lightning indirect effects, surge, HIRF, etc.).

End of Frame (EOF)

Seven recessive bits located at the end of a CAN message transmitted after the ACK Delimiter.

Error Active Mode

The normal operational state of a CAN interface when TEC/REC are below 128.

Error Frames

The area of a disrupted CAN message where an error in the message is being identified. Contains two parts: an Error Flag and an Error Delimiter.

Error Passive Mode

The state of the CAN interface when either TEC or REC equal or have exceeded 128. In this mode, normal operation continues but error frames are sent in recessive states.

ATTACHMENT 2
GLOSSARY

Exception Event Channel

The highest priority Logical Communication Channel, value of 0 (0b000). Data on this channel should be exclusively high priority (emergency event) data and will typically require some form of immediate action. Data on this channel is exclusively one-to-many (broadcast) data.

Extended Frame Format

The form of CAN that utilizes a 29-bit message identifier.

Frame

A frame is comprised of a start of frame bit, arbitration Field, Control Field, Data Field, CRC field, ACK Field, and End of Frame Field.

Form error

An error generated when one or more pre-defined bits in the CAN header is read as an unexpected/undefined value.

Function Code Identifier (FID)

7 bits of the extended CAN message identifier (starting at bit 25 and extended to include bit 19) after the LCC used to specify a system and/or subsystem the CAN message originated from.

Functional Status Bit

Bit 18 of the One to Many identifier format that is used in conjunction with the Data Length code of a CAN message to communicate functional status of a CAN message.

Half-Duplex

In network communications: allowing the transmission of signals in both directions but not simultaneously.

High Integrity Protocol

ARINC Specification 825 defined protocol to provide additional integrity to transmitted CAN messages. Introduces Sequence Number (SNo) and additional CRC (Message Integrity Check – MIC) that encompasses the CAN message ID and payload, including the SNo, and stored in the final two bytes of the payload.

Identifier Extension Flag (IDE)

Single bit located in the Arbitration field used to distinguish between CAN base format and the extended format.

Inter-Frame Space (IFS)

The separation (minimum three bits) between the last bit of a CAN Frame and the first bit of the next CAN Frame transmitted.

IPT

Information Processing Time: The time segment starting at the sample point for the calculation of bit level.

ATTACHMENT 2
GLOSSARY

Impedance

The effective resistance of an electric circuit or component to alternating current, arising from the combined effects of ohmic resistance and reactance.

Interoperability

The ability to make systems work together.

Latency

The time difference between transmission of a message and its reception by the receiver.

Local (LCL)

Bit 17 of the extended CAN message identifier intended to identify a CAN message that will remain on the local network, i.e., will not be processed by a gateway node.

Linear

Arranged in or extending along a straight or nearly straight line.

Logical Communication Channel (LCC)

The first three bits (bits 28 – 26) of the CAN message ID defined by ARINC 825 in a hierarchal order for the purpose of creating independent network layers.

Logical Link Control (LLC)

The higher of the two data link layer sublayers defined by the IEEE. The LLC sublayer handles error control, flow control, framing, and MAC-sublayer addressing.

Medium Access Layer (MAC)

A sub-layer of the data link layer in the OSI model (layer 2) that provides addressing and channel access control mechanisms that make it possible for several network nodes to communicate within a multiple access network that incorporates a shared medium.

Message

See Frame.

Message Integrity Check (MIC)

Additional 16-bit CRC added to High Integrity CAN Messages that includes the CAN Message ID and payload. Added to the last two bytes of the High Integrity Message.

Node

The connection of a CAN controller to the network. There may be multiple Nodes contained in one LRU.

Node Service

The defined command/response peer-to-peer process that allows for node to node dialogs. May be “connectionless.”

ATTACHMENT 2
GLOSSARY

Node Service Channel

The third defined Logical Communication Channel with a value of 4 (0b100). Data on this channel is provided to isolate the Node Services from the typical CAN communication and assigned a lower priority to ensure non-interference with basic network communication. Data on this channel is exclusively peer-to-peer communication.

Normal Operation Channel

The second defined Logical Communication Channel with a value of 2 (0b010). Data on this channel is intended to be the basic communication between nodes on the network and is broadcast in nature (See one-to-many).

One-to-Many

Broadcast communication between nodes on a bus: one transmitter with one or more receivers.

Oscillator Tolerance

Deviation in the oscillator from its stated value that can change slightly over time. If two oscillators were to deviate from the norm value by a significant amount and in opposite directions, there is the possibility that they would be unable to read each other's messages.

Overload Frame

An atypical CAN transmission sent by one node to other nodes on the network that indicates an inability to currently receive CAN messages, delaying the transmission of further CAN messages on the network. Overload frames have two bit fields, an Overload Flag and an Overload Delimiter. Use of Overload Frames on an ARINC 825 network is not allowed.

Parameter

The digitized form of a signal used as the constituent parts of a CAN message to communicate information.

Parametric

Having to do with information at the level of the parameter.

Payload

See Data Frame/Data Set.

Peer-to-Peer

Directed communication between two defined nodes on the network

Periodic

Appearing or occurring at defined intervals. Scheduled data.

Periodic Health Status Message (PHSM)

An ARINC 825 defined message of low periodicity and priority that communicates basic node health information.

ATTACHMENT 2
GLOSSARY

Point-to-Point

A simple network, or logical connection, consisting of two nodes.

Private (PVT)

Bit 16 of the extended CAN message identifier used to identify messages which have no meaning to nodes other than those which are specifically programmed to use them. Messages with this bit set may have no published description and are for private use only.

Profile ID

Profile ID is a string in the format ID.Sub-ID, where ID is a number (USHORT) and Sub-ID is a number (USHORT).

Propagation Delay

The time it takes for the first bit of a signal to travel from the sender to the receiver.

Receive Error Count (REC)

A count by a CAN interface of each detected corrupted message received.

Recessive

The least important or influential. For CAN, this translates to a bit value of one (1).

Redundancy Channel Identifier (RCI)

The last two bits of the extended CAN message identifier used to identify up to four redundant messages and/or channels.

Remote Frame

An atypical CAN message from a receiving node initiating transmission of source data. Consists of six defined bit fields: Start of Frame, Arbitration Field, Control Field, CRC Field, ACK Field, and End of Frame. Use of a remote frame in an ARINC 825 network is strongly discouraged.

Remote Transmission Request (RTR)

Located in the arbitration field and used only for extended format CAN message to indicate a remote frame is being requested. For ARINC 825 CAN Messages, this bit will always be set to dominant as Remote Frames are highly discouraged.

Sample Point

The point of time within a CAN bit reception at which the bus level is read and interpreted as a 0 or a 1.

Sequence Number (SNo)

One byte field in a High Integrity CAN message used to identify the correct order of messages and used to ensure the messages are received in the correct order.

Serial Bus

A shared channel that transmits data one bit after the other on a common wire.

ATTACHMENT 2
GLOSSARY

Service Function Code

The first two bytes of a CAN Node Service Message that uniquely identify the node service.

Server

The target of a node service. The Server provides data to the Client.

Signal

The raw data provided by a sensor, the simplest, unaltered native information.

Source

The originator of a particular CAN message.

Stub

The cable length between a node's CAN interface and the connection to the bus.

Substitute Remote Request (SRR)

Used in the CAN header for an extended format identifier message to ensure base frames prevail in a collision with extended frame format. Located in the Arbitration field, this bit will always be recessive for ARINC 825 CAN Messages as base frames are not utilized in ARINC 825 networks.

Subsystem (or sub-system)

A self-contained system within a larger system.

Synchronization Jump Width

Defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

System

A defined set of connected nodes forming a complex network.

Transmit Error Count (TEC)

A count by a transmitting CAN interface of each transmitted message detected to have been transmitted corrupted.

Test and Maintenance Channel

The fifth defined Logical Communication Channel with a value of 6 (0b110). Data transmitted in this channel will be relegated to test data and/or maintenance data.

Time Quantum

A defined measure relative to the internal clock of the node.

Topology

A schematic description of the arrangement of a network.

ATTACHMENT 2
GLOSSARY

User Defined Channel

The fourth defined Logical Communication Channel with a value of 5 (0b101). Data on this channel will be defined by the system designer for CAN messages that do not conform to ARINC Specification 825 in its entirety.

APPENDIX A
COMMUNICATION PROFILE DATABASE

APPENDIX A DELETED BY SUPPLEMENT 1

APPENDIX B COMMUNICATION PROFILE EXAMPLE

Attachment 1 and Appendix B were combined and support files were added. The XSD files can be found at [Add new link to support files].

APPENDIX C ACRONYMS

ACK	Acknowledge
ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
BCS	BIT Control Service
BER	Bit Error Rate
BIT	Built-In Test
CAN	Controller Area Network
CMS	Central Maintenance System
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
DAL	Design Assurance Level
DDS	Data Download Service
DLC	Data Length Code
DLL	Data Link Layer
DOC	Data Object Code
DMC	Directed Message Channel
DTC	Data Type Code
DUS	Data Upload Service
EEC	Exception Event Channel
EM	Elecromagnetic
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Immunity
EOF	End of Frame
FH	Flight Hour
FHA	Functional Hazard Analysis
FID	Function Code Identifier
FIFO	First In, First Out
FLS	Field Loadable Software
FMEA	Failure Modes and Effects Analysis
FMC	Frame Migration Channel
FSB	Functional Status Bit
FSS	Functional Status Set
FT	Functional Test
FW	Fail/Warn
GA	General Aviation
HIRF	High Intensity Radiated Fields
ICD	Interface Control Document

APPENDIX C
ACRONYMS

ID	Identifier
IDE	Identifier Extension
IDS	Node Identification Service
IFS	Inter-Frame Space
IMA	Integrated Modular Avionics
IP	Internet Protocol
IPT	Information Processing Time
ISO	International Standardization Organization
LCC	Logical Communication Channel
LCL	“Local” Bit
LLC	Logical Link Control
LRU	Line Replaceable Unit
MAC	Media Access Control
MIC	Message Integrity Check
MSB	Most Significant Bit
MTU	Minimum Transmit Unit
NCD	No Computed Data
ND	No Data
NID	Node ID
NIS	Node ID Setting Service
NO	Normal Operation
NOC	Normal Operation Channel
NSC	Node Service Channel
NSS	Node Synchronization Service
NVS	Non-Volatile Storage Service
OMS	Onboard Maintenance System
OSI	Open Systems Interconnect
PHSM	Periodic Health Status Message
PVS	Parameter Validity Status
PVT	“Private” Bit
RCI	Redundancy Channel Identifier
REC	Receive Error Counter
RF	Radio Frequency
RSD	“Reserved” Bit
RTR	Remote Transmission Request
SCS	Service Control Service
SDL	Software Data Loading
SEU	Single Event Upset
SFC	Service Function Code
SID	Server ID
SJW	Synchronization Jump Width

APPENDIX C
ACRONYMS

SMT	Service Message Type
SNo	Sequence Number
SOF	Start of Frame
SRR	Substitute Remote Request
TCP	Transmission Control Protocol
TEC	Transmit Error Counter
TMC	Test and Maintenance Channel
TQ	Time Quanta
TVS	Transient Voltage Suppression
UDC	User-Defined Channel
UDP	User Datagram Protocol
WIP	Wire Integration Panel
WKP	Well Known Port

APPENDIX D SAMPLE DATA OBJECT CODE (DOC) ASSIGNMENT LIST

The following sample DOC assignment list covers a range of parameters common to aircraft in general and defines the DOC as well as the unit for each of them. For the purpose of this example, each parameter receives a unique DOC (one parameter per ARINC Specification 825 message only):

Function	Function Code ID (FID)	Data Object Code (DOC)	Parameter Name	Unit
Flight State	4	64	Body Longitudinal Acceleration	m/s^2
Flight State	4	72	Body Lateral Acceleration	m/s^2
Flight State	4	80	Body Normal Acceleration	m/s^2
Flight State	4	88	Body Pitch Rate	rad/s
Flight State	4	96	Body Roll Rate	rad/s
Flight State	4	104	Body Yaw Rate	rad/s
Flight State	4	112	Body Pitch Angle	rad
Flight State	4	120	Body Roll Angle	rad
Flight State	4	128	Body Heading Angle	rad
Flight State	4	136	Body Normal Velocity	rad
Flight State	4	144	Body Longitudinal Velocity	rad
Flight State	4	152	Body Lateral Velocity	rad
Flight State	4	160	Body Sideslip	rad
Flight State	4	168	Turn Coordination Rate	rad/s
Flight Controls	10	64	Pitch Control Position Pilot	mm
Flight Controls	10	72	Roll Control Position Pilot	mm
Flight Controls	10	80	Yaw Control Position Pilot	mm
Flight Controls	10	88	Elevator Position Angle	rad
Flight Controls	10	96	Port Aileron Position Angle	rad
Flight Controls	10	104	Starboard Aileron Position Angle	rad
Flight Controls	10	112	Rudder Position Angle	rad
Flight Controls	10	120	Flaps Position Angle	rad
Flight Controls	10	128	Slats Position Angle	rad
Flight Controls	10	136	Speedbrake Position	mm
Flight Controls	10	144	Pitch Trim Position Command	mm
Flight Controls	10	152	Roll Trim Position Command	mm
Flight Controls	10	160	Yaw Trim Position Command	mm
Flight Controls	10	168	Pitch Trim Speed	rad/s
Flight Controls	10	176	Roll Trim Speed	rad/s
Flight Controls	10	184	Yaw Trim Speed	rad/s
Electrical Systems	13	64	AC System Voltage	V
Electrical Systems	13	72	AC System Current	A
Electrical Systems	13	80	DC System Voltage	V
Electrical Systems	13	88	DC System Current	A
Fuel	18	64	Fuel Pressure	Pa
Fuel	18	72	Fuel Temperature	K
Fuel	18	80	Fuel Quantity	l
Propulsion	22	64	Engine Crankshaft RPM (N1)	1/s
Propulsion	22	72	Engine Propeller RPM (N2)	1/s
Propulsion	22	80	Engine Torque	Nm

APPENDIX D
SAMPLE DATA OBJECT CODE ASSIGNMENT LIST

Function	Function Code ID (FID)	Data Object Code (DOC)	Parameter Name	Unit
Propulsion	22	88	Engine Turbine Inlet Temperature	K
Propulsion	22	96	Engine Inter-turbine Temperature	K
Propulsion	22	104	Engine Turbine Outlet Temperature	K
Propulsion	22	112	Engine Fuel Flow Rate	l/s
Propulsion	22	120	Engine Manifold Pressure	Pa
Propulsion	22	128	Engine Oil Pressure	Pa
Propulsion	22	136	Engine Oil Temperature	K
Propulsion	22	144	Engine Cylinder Head Temperature	K
Propulsion	22	152	Engine Oil Quantity	l
Propulsion	22	160	Engine Coolant Temperature	K
Hydraulic	48	64	Hydraulic System Pressure	Pa
Hydraulic	48	72	Hydraulic System Fluid Temperature	K
Hydraulic	48	80	Hydraulic System Fluid Quantity	l
Communication	51	64	VHF COM 1 Frequency	Hz
Communication	51	72	VHF COM 2 Frequency	Hz
Air Data	52	80	Indicated Airspeed	m/s
Air Data	52	88	Calibrated Airspeed	m/s
Air Data	52	96	True Airspeed	m/s
Air Data	52	104	Mach Number	Mach
Air Data	52	112	Total Pressure	Pa
Air Data	52	120	Static Pressure	Pa
Air Data	52	128	Differential Pressure	Pa
Air Data	52	136	Standard Altitude	m
Air Data	52	144	Baro Corrected Altitude	m
Air Data	52	152	True Altitude	m
Air Data	52	160	Density Altitude	m
Air Data	52	168	Altitude Rate	m/s
Air Data	52	176	Baro Correction	Pa
Air Data	52	184	Angle of Attack	rad
Air Data	52	192	Angle of Sideslip	rad
Air Data	52	200	Outside Air Temperature	K
Air Data	52	208	Static Air Temperature	K
Air Data	52	216	Total Air Temperature	K
Air Data	52	224	Wind Speed	m/s
Air Data	52	232	Wind Direction	rad
Navigation	52	240	GPS Latitude	rad
Navigation	52	248	GPS Longitude	rad
Navigation	52	256	GPS Height above Ellipsoid	m
Navigation	52	264	GPS Ground Speed	m/s
Navigation	52	272	True Track	rad
Navigation	52	280	Magnetic Track	rad
Navigation	52	288	Cross Track Error	rad
Navigation	52	296	Track Error Angle	rad
Navigation	52	304	Glideslope Deviation	rad
Navigation	52	312	Localizer Deviation	rad
Navigation	52	320	Magnetic Heading	rad

APPENDIX D
SAMPLE DATA OBJECT CODE ASSIGNMENT LIST

Function	Function Code ID (FID)	Data Object Code (DOC)	Parameter Name	Unit
Navigation	52	328	Radio Height	m
Navigation	52	336	DME Distance	m
Navigation	52	344	DME Time-to-Go	s
Navigation	52	352	DME Ground Speed	m/s
Navigation	52	360	True East Velocity	m/s
Navigation	52	368	True North Velocity	m/s
Navigation	52	376	True Vertical Velocity	m/s
Cabin Pressure Control	53	64	Cabin Altitude	m
Cabin Air Conditioning Control	53	72	Cabin Temperature	K

APPENDIX E
DESIGN CHECKLIST

APPENDIX E ARINC 825 PROCESS CHECKLIST

The design checklist is presented in a typical chronological order beginning with selection of the network through final certification. Selection of a network should include comparison of the application requirements against the network capabilities. Once the network is selected, the designer needs to incorporate the CAN fundamentals as put forth in this standard.

APPENDIX F CAN SPECIFICATION TEMPLATE

F-1 PURPOSE

This document describes the overall purpose of the bus specification.

This specification describes the operation of the CAN bus for this application and details the system configurable parameters. The same specification may be applicable to multiple buses on the aircraft. Where operating environment or bus characteristics require differences between the buses, these may be identified in tables where applicable.

F-1.1 Scope

The specification covers general operation of the bus (for this application) and the requirements for the nodes sharing the common bus.

F-1.2 Applicable Documents

The latest version of the following documents apply. This list may not be complete and is provided as a representational.

ARINC 664	Aircraft Data Network, Part 7 – Avionics Full Duplex Switched Ethernet Network
ARINC 825	General Standardization of CAN (Controller Area Network) Bus Protocol For Airborne Use
ISO 11898	Road vehicles — Controller Area Network (CAN) — Interchange of digital information – Controller area network (CAN) for high-speed communication
ISO 11898-1	Road vehicles — Controller Area Network (CAN) — Part 1: Data link layer and physical signaling
ISO 11898-2	Road vehicles — Controller area network (CAN) — Part 2: High-speed medium access unit
ISO 16845	Qualification Test Specification
RTCA DO-160E	Environmental Conditions And Test Procedures For Airborne Equipment
	CAN Specification 2.0 (Bosch)
	CANaerospace Specification (Stock Flight Systems)

F-1.3 Acronyms

CAN	Controller Area Network
TSP	Twisted Shielded Pair

F-1.4 Definitions

Upload:	The sending of blocks of data shall be called “uploading” when the sender initiates the transfer.
Download:	Requesting blocks of data from another node shall be called “downloading” when the receiver initiates the transfer.
Stub	A short connection (less than 1 meter) of TSP 120 ohm wire from the main CAN bus to a node.

APPENDIX F
CAN SPECIFICATION TEMPLATE

F-2 FUNCTIONAL REQUIREMENTS

This section is intended to capture the CAN general operating requirements and behavior of the nodes connected to the bus. The bus should consist of a pair of twisted shielded wire, with a characteristic impedance of 120 ohms. The bus may be connected daisy chain fashion or node may be connected via stubs.

F-2.1 Node Basics (Theory of Operation) [ref. 7.5.7]

CAN is used to allow multiple nodes to share a common interconnect. Data transmitted by one node on the bus may be received by any other node on the bus. The CAN protocol has an arbitration scheme that allows only the highest priority message waiting to be transmitted to access the bus. ARINC Specification 825 defines how the message identifiers are used and assigned to categories of equipment. The message identifiers are used in the CAN protocol to determine the message's priority. For deterministic operation, all nodes on a bus must always allow enough time for all the messages to be transmitted at their required transmission rates.

F-2.2 Data Loading [ref. 7.5.3]

Nodes may be loadable over CAN. This section describes a method to accomplish this.

F-2.3 Node Services [ref. 5.5]

Each node on the bus shall implement the following node services:

Table F-1 – Node Services

Code	Service Name	Response
0x00	Node Identification Service	**8 byte message**
0x0z	**Next Node Service**	**xxxx**

F-2.3.1 Node Identification Service

Table F-2 – Node Identification Service

Msg ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

F-2.3.2 Next Node Service

Table F-3 – Next Node Service

Msg ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

F-2.3.3 CAN Message ID Format

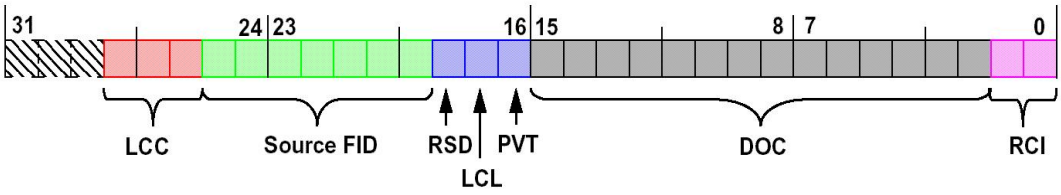


Figure F-1 – CAN Message ID Format

F-2.4 Data Formats [ref. 5.3.1]

All the nodes shall use the data formats defined in ARINC Specification 825. If a node is required to transmit data that is not defined by ARINC Specification 825, it may use a private data format with the permission of the bus owner.

F-2.5 Software Driver [ref. 7.5.5.5]

F-2.6 Redundancy Concept [ref. 7.6]

State the redundancy concept for the bus and how the nodes are expected to handle the redundant data.

F-2.7 Node Identifier [ref. 2.2]

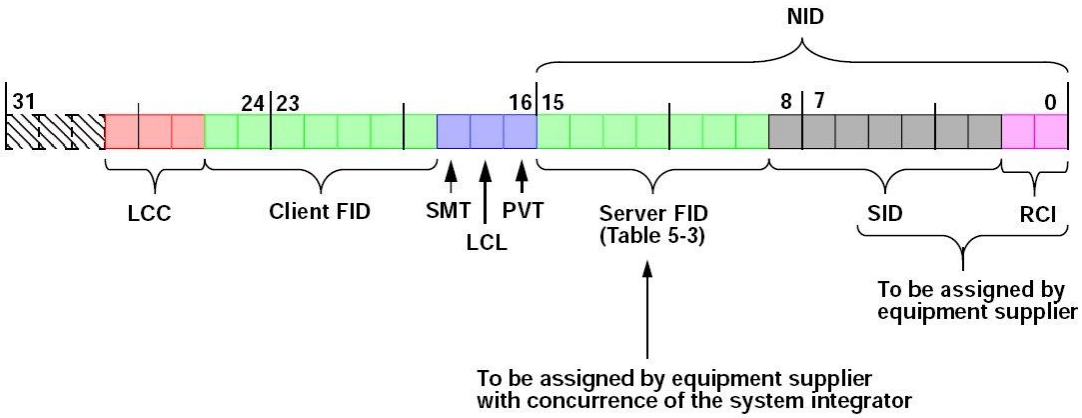


Figure F-2 – Node Identifier Format

F-2.8 Error and Fault Behavior [ref. 5.4]

F-2.9 Maintainability and Self-Test Provisions

Node may perform self-test by transmitting unique message data patterns not normally transmitted to test for faults in the CAN controller [Note: This would require a dual controller design; typical CAN controllers monitor all their transmitted data, but this mechanism may not detect some ‘stuck at’ bit faults within the controller].

APPENDIX F
CAN SPECIFICATION TEMPLATE

F-3 BUS INTERFACE PERFORMANCE

F-3.1 Startup Behavior [ref. 7.5.11]

Type of startup must be designated, ‘Passive’ or ‘Active’, where ‘Active’ is defined as any node may send an initial message when the node is ready to send and receive and ‘Passive’ is defined as when a node powers up it stays silent and waits to receive a message from a master or another node on the bus. State what the bus startup time should be and the normal startup period for a node.

F-3.2 Bus Error Recovery (Bus Off) [ref. 4.6]

F-3.3 Physical Layer [ref. 3.0]

F-3.3.1 Physical Media [ref. 3.0]

Per ARINC Specification 825, the bus shall use twisted shielded pair wire with a characteristic impedance of 120 ohms.

F-3.3.2 Bus Data Rate [ref. 3.1.1.3.1]

This refers to the basic CAN bit transmission rate, i.e., 83.333 kbit/s, 125 kbit/s, 250 kbit/s, 500 kbit/s or 1000 kbit/s. Higher data rates may transfer more data but the nodes on the bus must be capable of processing messages at a faster rate. In contrast, low data rate buses transfer less data per time interval but may be physically longer and nodes may have less processing capabilities than those on high-speed bus.

F-3.3.3 Termination Resistors [ref. 7.3.6.4]

Each bus should have two and only two termination resistors. The termination resistor should be located at the physical ends of the bus, those points that by wiring lengths are the farthest apart. Star topologies should be avoided to ensure the bus is able to be properly terminated. Typically, the termination resistor should not be inside nodes on the bus unless the layout and routing of the wiring between nodes is known in advance [Note: if units contain the termination resistors there must only be two on the bus and these units must be located at the physical ends of the bus].

F-3.3.4 Controller (Restrictions) [ref. 7.4.2]

F-3.3.5 Electronics Stabilization of Each Node

Same as active start-up sequence.

F-3.3.6 Node is Ready to Process Message

Same as active start-up sequence.

F-3.3.7 Master Node Communicates

At power-up, the nodes completing the two steps above will wait to receive a frame from the master. In the event the master is the first node to complete the steps above, the master will communicate alone on the bus sending its first frame. If the master is first to transmit, it will follow the same sequence defined for active start-up described previously.

APPENDIX F
CAN SPECIFICATION TEMPLATE

F-3.3.8 Start-up Sequence is Completed

F-3.4 Data Traffic Shaping

F-3.5 EMI and Lightning [ref. 7.3.4, 7.3.4.3]

F-4 REDUNDANCY MANAGEMENT [ref. 7.6]

F-5 GATEWAY CHARACTERISTICS [ref. 6]

F-6 SAFETY AND INTEGRITY [ref. 7.7]

F-7 ARINC 825 COMPATIBILITY MATRIX

F-8 BUS ICD

APPENDIX G FAA AC 20-156 CROSS REFERENCE

AC 20-156 Section	ARINC 825 Section	Summarized AC 20-156 Item Description
3		**SAFETY REQUIREMENTS.**
3a	7.9.5	The data bus architecture and implementation
3b	–	Data bus availability and reliability requirements
3c	–	Partitioning and protection requirements
3d	–	*Failure detection, reporting, and management*
3e	–	Fault containment, fault tolerance, and monitoring
3f	–	Common cause
3g	–	Methods or ways to reconfigure node(s) and the network
4		**DATA INTEGRITY REQUIREMENTS**
4a	7.9.1.4	The maximum error rate per byte
4b	4.2.2 4.5.1	Data bus-provided means to detect and recover from failures and errors
4c	5.6	A data bus loading analysis
4d	4.2.4 4.7 7.9.2.1.1 7.9.2.1.2	The buffer overflow and underflow limits
4e	3.1.1.2 7.9.2	Issues related to the data bus integrity
4f	–	The ability of the data bus to reconfigure the node
4g	4.5 5.4	Bi-directional error detection implementations
4h	–	The switch saturation limits
5		**DATA BUS PERFORMANCE REQUIREMENTS**
5a	3.1.1.3.1 5.6	Data bus operating speed and scheduling of messages
5b	7.9.2.2	Loss of the data bus through shorting or opening of the data bus connections
5c	2.9, 2.10, 3.0, 5.3	System interoperability to include the data bus or network topology
5d	3.1.1.3.1 7.3.5	Data bus length, stub length, and cable coupling
5e	5.4, 5.6, 7.9 .2.2	Degraded data bus operation and performance abilities
5f	4.3, 4.5	Retry algorithms
5g	5.6	Bandwidth capability of the data bus
5h	3.1.1.3.1, 5.6, 7.6.1	Actual specification of the data bus capacity
5i	7.6.1, 7.6.3, 7.6.5.3, 7.6.5.4	Data latency and efficiency
5j	7.6.1, 7.8.2	Per-transmission overhead

APPENDIX G
FAA AC 20-156 CROSS REFERENCE

5k	5.4, 7.5.4, 7.9.1	System's failure management
5l	4.5.3.2, 4.6, 7.4.3	System start-up configuration and reintegration
6		**SOFTWARE AND HARDWARE ASSURANCE REQUIREMENTS**
6a	7.9.5	Complex electronic device must meet the appropriate hardware design assurance requirements
6b	7.9.5	Develop the software to the appropriate software design assurance level
6c	–	Tools to develop or verify the data bus software or hardware
7		**ELECTROMAGNETIC COMPATIBILITY REQUIREMENTS**
7a	3.1.1.1, 3.1.1.2, 3.1.1.3	The effects of electromagnetic emissions and susceptibility
7b	3.1.1.1, 3.2	Consider the entire data bus system, including the terminals
7c	7.3.4	Electromagnetic emissions
7d	3.1.1.1, 3.2, 7.3.4	Address the electromagnetic compatibility guidelines
7d1	3.1.1.1, 7.3.4.2	Data bus data rate
7d2	3.1.1.1, 3.2, 7.3.4	Radio frequency emissions and susceptibility
7d3	3.1.1.1, 3.2, 7.3.4, 7.3.5.4	Electromagnetic compatibility of data bus
7d4	3.1.1.1, 7.3.4.3, 7.3.5.2	Lightning and high-intensity radiated field (HIRF) immunity commensurate
8		**VERIFICATION AND VALIDATION REQUIREMENTS.**
8a	–	Validate the data bus requirements for the aircraft
8b	–	Evaluate whether the data bus meets RTCA/DO-160E environmental standards
8c	–	Perform appropriate verification of the data bus
8d	–	Perform a functional test of the integrated data bus
8e	–	Verify and validate the data bus operation, architecture, and performance claims
8f	–	Test the data bus failure and recovery procedures
8g	–	Verify all built-in-tests
8h	–	Test the data bus failure management features
8i	–	Test the data bus in a degraded mode
8j	–	When you use test benches to verify the performance of the data bus
9		**CONFIGURATION MANAGEMENT REQUIREMENTS**
9a	–	Integrate the data bus into the aircraft or aircraft engine design, consider the total system
9b	–	Establish and maintain configuration control of the data bus during certification
9c	–	You need to document physical and logical rules used by the data bus
9d	–	Documentation to support development and operation of the data bus

APPENDIX G
FAA AC 20-156 CROSS REFERENCE

9e	–	Provide documentation to support and to configure layers
9f	–	Document data bus wiring and installation procedures
9g	–	Strategy, plan, or process for future bus expansion
10		**SECURITY ASSURANCE REQUIREMENTS**
10a	7.9.4	Access Security
10b	5.5.2.3, 5.5.2.4, 5.7	Information and Data Protection

APPENDIX H ARINC 825 COMPLIANCE CHECKLIST

§	Description	Summary	Comply (Y/N)	Notes
3.1.1	Node Characteristics	ISO 11898-2, EME, Dominant Timeout, Impedance, disturbances, bus speed, bit timing		
3.2	Design Considerations	Pinout		
3.2.1	Cabling Characteristics	ISO 11898-2 Cabling		
3.2.3	Installation and Grounding Characteristics	Grounding, Topology, termination, ground offset		
4.5	Error Handling	Monitor TEC/REC, Bus Off rejoin rules		
4.6	Node State Machine	State machine and transitions		
4.7	Performance and Robustness Considerations	100% bus loading		
5.2.1	Logical Communication Channels	Honor LCCs		
5.2.2.1	One-to-Many	Broadcast, sampling messages		
5.2.2.2	Directed Message	Targeted messaging		
5.2.2.3	Peer-to-Peer	A825 P2P		
	Peer-to-Peer for PHSM	Only P2P message is PHSM (should really be one-to-many…)		
5.3	Interoperability and 5.3.1	Endianness, data types,		
5.3.2	Profiles	Include comm profile with node (needs to point to schema), FIDS		
5.3.3	Message Level Functional Status	One-to-Many functional status bit		
5.4	Periodic Health Status	PHSM and field definition		
5.5	Node Service Interface	Node Services		
5.7	High Integrity Protocol	SeqNo, CRC		
5.7.6	High Availability Message	SeqNo only		
Appendix J	MIBs	MIB counters and rules		

APPENDIX I CONFIGURATION OF BIT TIMING

I-1 Summary

One of the most critical aspects of CAN bus interoperability is the configuration of the bit timing within CAN controllers. This becomes more complex and nuanced with the introduction of CAN FD. This white paper proposes new bit timing requirements for ARINC 825 that encompass CAN FD bit timing, update Classical CAN bit timing, and provides the equations and design guidance for using the requirements. The result is a method that is more like that used in the automotive industry, starting with interoperability in the time domain and deriving the microcontroller settings from those requirements.

I-2 CAN Bit Time

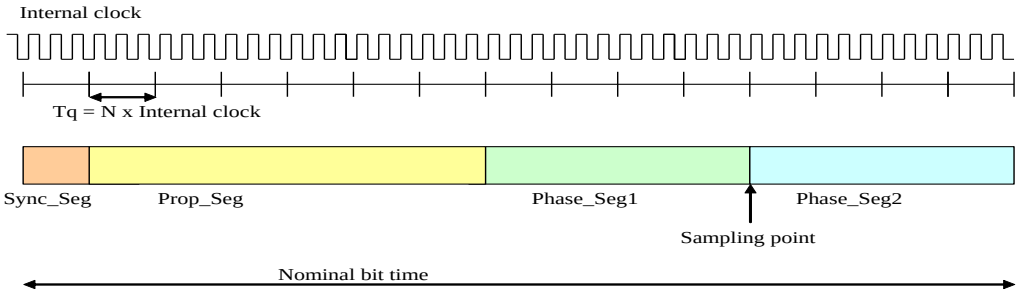


Figure I-1 - Bit Timing (from ARINC 825)

The following table is a summary of the design considerations relevant to bit timing that the LRU designer has control over:

Interface impedance	The number of nodes and the transient voltage protection mechanisms often have a significant impact on the waveform, rounding off the rising and falling edges and/or causing reflections and ringing.
Oscillator Tolerance	The relative error between the oscillators of different CAN nodes can cause their bit times to diverge. Nodes correct for this error by resynchronizing on rising edges up to the amount of the Synchronization Jump Width. The CAN protocol adds stuff bits to place an upper bound on the time between edges.
Bit Tolerance	Depending on the clock speed selected and oscillator aging and temperature characteristics, some CAN nodes may not be able to achieve the exact baud rate specified. For example, the bus may be specified as 125kbps but with oscillator tolerances and CAN controller constraints, a node may be operating at 124kbps.
Time Quanta	A TQ is measured in seconds and is used by the CAN controller to translate the analog signal into the digital domain. One can think of it like the time resolution of the CAN controller. Within a CAN controller, the CAN bit is divided into a number of TQ based on the internal clock speed and prescaler. The bit segments are specified in terms of TQ. Note that the Synchronization Segment is always 1TQ.

APPENDIX I
CONFIGURATION OF BIT TIMING

Sample Point	The sample point is the location within a bit that the node samples the state of the bus. Some CAN controllers allow for 3 sample points to try to filter out disturbances. However, this is rarely used as it reduces determinism (some controllers implement the extra samples before the usual sample point, some after, and some on both sides) and it does not provide much improvement (the bit is supposed to stabilize prior to the sample point).
Synchronization Jump Width	This parameter, specified within a CAN controller in TQ, is the maximum amount of a bit time that the CAN controller can add or subtract to resynchronize to a rising edge. The resolution of the SJW compensation is one TQ.
Propagation Segment	This bit segment compensates for the signal propagation delay, the transceiver loop delay, and the impact of the physical layer on the bit rise/fall and stabilization time. The purpose is to allow the signal to stabilize at all nodes before they sample the bit.
Phase Segment 1 and Phase Segment 2	These segments surround the sample point and are lengthened or shortened by SJW to maintain synchronization between nodes, accommodating the tolerance error buildup between nodes. Phase Segment 1 is often combined with the Propagation Segment.
Transmitter Delay Compensation	This feature, introduced for CAN FD, allows a transmitting CAN controller to delay its samples of the bus during the CAN FD data phase to compensate for the transceiver loopback delay. Otherwise, the bits may transmit so fast that the controller samples before its own signal propagates back through the transceiver.

For CAN bit timing, the bit time accuracy and clock tolerances influence the SJW. The physical layer and bit time configuration determine the location of the sample point. As it turns out, once these two parameters are specified, the remainder of the bit timing can be derived and applied to each specific CAN controller. The following sections present the constraints for setting the SJW and sample point location.

I-3 Synchronization Jump Width

Bit timing tolerance is the range over which nodes can have fundamental differences in their base bit timing. As a result, they accumulate skew over time. The longer it is left uncorrected, the larger the accumulated skew can become. The Synchronization Jump Width (sometimes called Resynchronization Jump Width) compensates for the drift between bit times and the readjustment of nodes to the winner of arbitration when the bus is in normal operation. The equations below, in ISO-11898-1 and explained by Dr. Arthur Mutter of Bosch, were derived from the CAN timing parameters and provide the bounding conditions for the SJW in terms of the clock speeds and error. They factor in the bit stuffing algorithm of the CAN protocol. Conditions 4 and 5 assess the bit rate switching between the arbitration and data phases for CAN FD.

Table I-1 – Conditions for Arbitration Phase [1]

Condition 1	Resynchronization	$df < \frac{sjw_A}{2 \cdot 10 \cdot bt_A}$

APPENDIX I
CONFIGURATION OF BIT TIMING

Condition 2	Sampling of Bit after Error Flag	$df < \frac{\min(ps1_A, ps2_A)}{2 \cdot [13 \cdot bt_A - ps2_A]}$

Table I-2 – Conditions for CAN FD [1]

Condition 3	Resynchronization	$df < \frac{sjw_D}{2 \cdot 10 \cdot bt_D}$
Condition 4	Sampling of Bit after Error Flag	$df < \frac{\min(ps1_A, ps2_A)}{2 \cdot \left[(6 \cdot bt_D - ps2_D) \cdot \frac{BRP_D}{BRP_A} + 7 \cdot bt_A\right]}$
Condition 5	Switching from Arbitration Phase to Data Phase	$df < \frac{sjw_D - \max\left(0; \frac{BRP_A}{BRP_D} - 1\right)}{2 \cdot \left[(2 \cdot bt_A - ps2_A)\frac{BRP_A}{BRP_D} + ps2_D + 4 \cdot bt_D\right]}$

These equations can be used to produce a bound on the bit timing tolerance. Note that the bit time tolerance is a combination of the oscillator tolerance/drift and the inability of certain CAN controllers to configure the exact bit time, e.g., 83.3 kbps versus 83.33 kbps. These errors have the same impact on the bit and together they add to a timing error, called *df,* in the equations above and throughout this paper. Figure I-2 illustrates the bit tolerance error. The plot in Figure I-3, from Dr. Mutter's paper, demonstrates these bounds for an example CAN controller; refer to Dr. Mutter's paper for more description. These equations are the drivers for determining the bounds on oscillator and bit time tolerance.

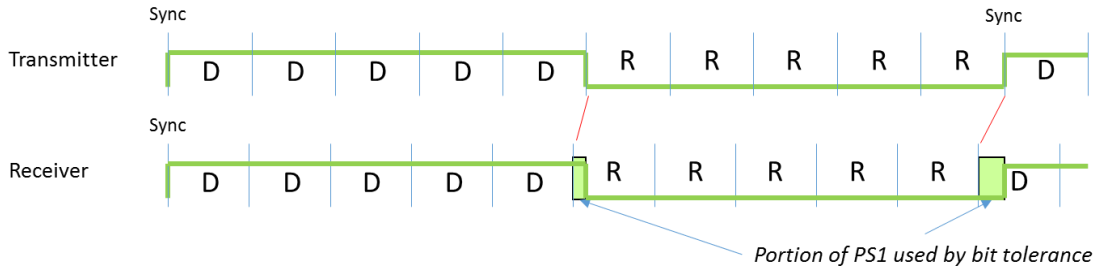


Figure I-2 – Fast Receiver, Slow Transmitter Impacting Receiver's PS1

A key assumption behind these equations is that the Sample Point location for all nodes is the same for both Arbitration and Data phases. This is the going forward position for the automotive industry. As will be shown later, there is a significant impact if the sample points vary.

APPENDIX I
CONFIGURATION OF BIT TIMING

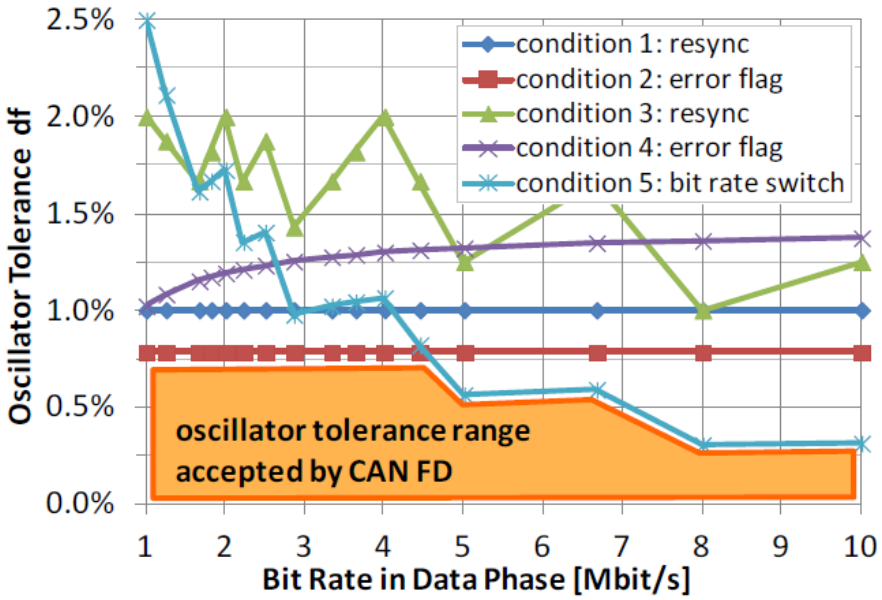


Figure I-3 – Bound on CAN Bit Time Tolerance [1]

A second effect of the SJW is to allow nodes to synchronize during arbitration.[2] When nodes begin arbitration, all nodes attempting to transmit start sending recessive and dominant bits. When there are multiple nodes competing, nodes that send a bit slightly later than others (due to oscillator drift between CAN messages) adjust their bit times to synchronize to the node that transmitted the earliest. If the earliest node loses arbitration, the remaining node(s) adjust to the next earliest transmitter (from their viewpoint). This sequence is demonstrated in the figure below.

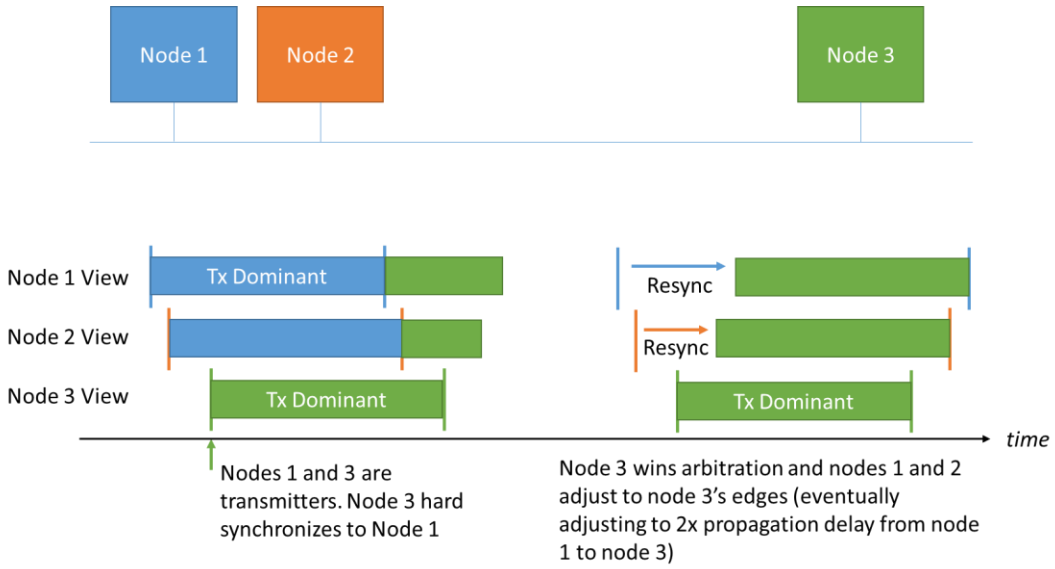


Figure I-4 – Resynchronization to Arbitration Winner

The effect above requires that the SJW be greater than the bit tolerance so nodes can adjust to the transmitter even in the presence of bit skew.

Bit timing tolerance and oscillator tolerance result in a combined effect: skewing the relative edges of bits between multiple nodes. The SJW counters this effect but it must be set to provide enough margin. ARINC 825 is strict, compared to the automotive industry, in the allowable tolerances for oscillator and bit timing. While the automotive industry allowed 1% oscillator tolerances or more for Classical CAN and about 0.5% tolerance for CAN FD, ARINC 825 currently requires a 0.01% oscillator tolerance and 0.15% bit time tolerance over the life of the device. These combine to a 0.16% tolerance on the bit timing that can accumulate between rising edges over the different conditions defined above.

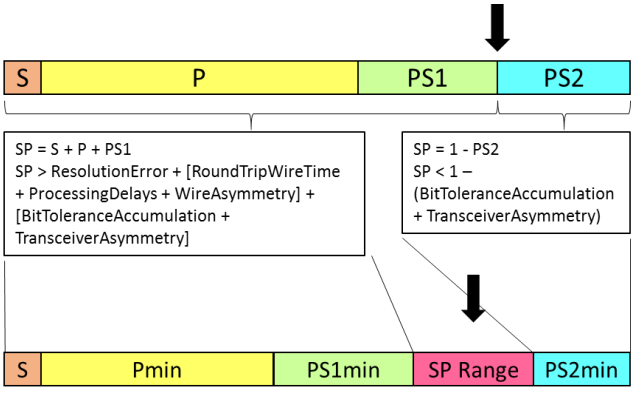
The SJW, at a minimum, must accommodate this skew and, as can be seen in Figure I-3, could accommodate higher bit time tolerances than allowed in ARINC 825 even for very high data rates.

Given the analysis above and the data rate limitations imposed by CAN implementations in avionics, it is proposed that an oscillator tolerance of 0.05% is well within acceptable bounds and enables cost reduction for LRU. The SJW required for this worst-case tolerance over the life of the unit (0.2% of a bit time total, combining 0.15% and 0.05% allowables) is approximately 1/16 of a bit time.

I-4 Arbitration Phase Bit Timing

The arbitration phase is unchanged between Classic CAN and CAN FD relative to bit timing. All nodes can be transmitting in the arbitration phase and bits between nodes must be aligned in order for them to properly determine priority. Figure I-5 illustrates the bit time and the physical effects that must be accounted for. This section will show the impact of those effects and how they combine to form the equations governing bit timing.

For the equations in all of the following sections, bit timing segments are given in units of percentage of bit time (e.g., PS2 may be 15% of bit time, or 0.15). To convert into units of time, they can be multiplied by the bit time (bt_A or bt_D).



APPENDIX I
CONFIGURATION OF BIT TIMING

Figure I-5 – Arbitration Bit Time Constraints

I-4.1 Resolution Error

The Synchronization Segment is the first part of the bit and allows nodes to resynchronize their clocks to rising edges. It is always one TQ in length (note that the size of TQ can vary between nodes). Whenever an edge lies outside of this segment, the CAN controller adjusts the bit time, up to SJW, to align to the edge. In this way, differences that have exceeded 1 TQ can be corrected. This segment, however, also results in resolution error on the bit. This error applies to both sides of the sample point, as is shown below.

The node will not take action if the edge is at the very beginning of the Synchronization Segment or at the very end. When the edge is at the end of the sync segment, the node effectively samples 1 TQ too early in the bit time. When one considers that the bit needs to have stabilized by the Sample Point location, this resolution error of 1 TQ must be added to the stabilization time. Figure I-6 illustrates the resolution error and how it subtracts from the time the signal has to stabilize.

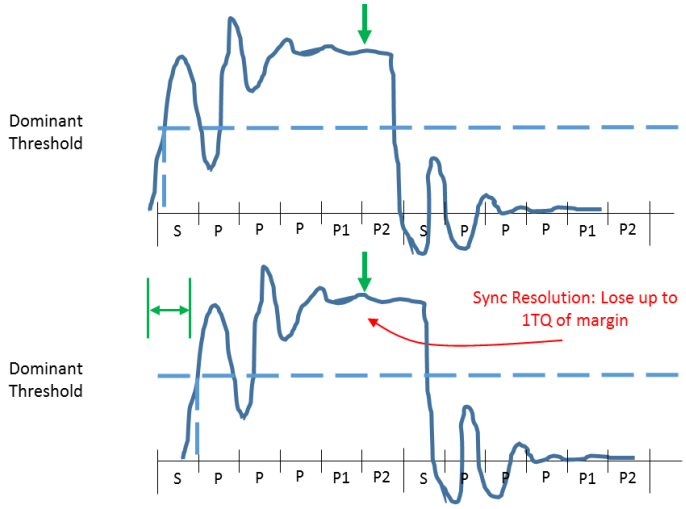


Figure I-6 – TQ Resolution Impact on Propagation Margin

I-4.2 Round Trip Wire Delay

There are several papers and proofs on how CAN arbitration is one of the drivers of the sample point. It can also be seen in Figure I-4. CAN bits have to travel between the farthest nodes and back within the propagation time segment in order to successfully perform bitwise arbitration. This drives the propagation segment to be at least twice the wire delay (T_{Wire}). This constraint does not impact PS2 as it is accounted for by the propagation segment.

I-4.3 Processing Delays

In addition to round-trip wire times, the propagation time also needs to factor in the processing times of nodes so they can coherently assess and respond to a bit. This processing time, T_{Node}, must be added to the

wire delay. This constraint does not impact PS2 as it is accounted for by the propagation segment.

I-4.4 Wire Asymmetry

What is not captured in the wire delay or processing delays is the impact of other physical layer components; they are sometimes included in the wire time but this does not clearly capture the effect.

Aviation LRU have very strict requirements for transient voltage suppression, notably from lightning and the impact of high frequency power and communication equipment. The components used in the interface circuits, such as transorbs, ferrites, and chokes, have a significant impact on the CAN signal. Figure I-7 shows an example.

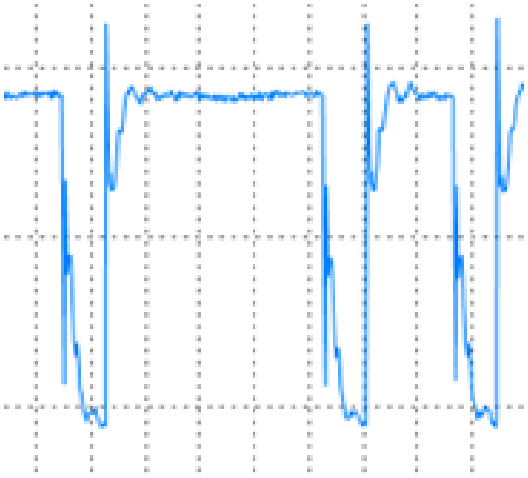


Figure I-7 – Impact of Aviation Interface Requirements on CAN Waveforms

These protections lead to reflections, ringing, and delayed rising or falling edges. To highlight the assessment of aerospace CAN bus design, the impact of these effects can be explicitly added to the previous equation. What matters for CAN, whether during arbitration or during the data field, is that nodes detect the edge and that the signal has stabilized by the sample point of all nodes. For the arbitration field, the time needed for propagation is the sum of wire and node effects combined with the edge and reflection effects, called T_{Rise} or T_{Fall} in the equations. Figure I-8 below illustrates the impact of the rise time on the “effective propagation time.”

APPENDIX I
CONFIGURATION OF BIT TIMING

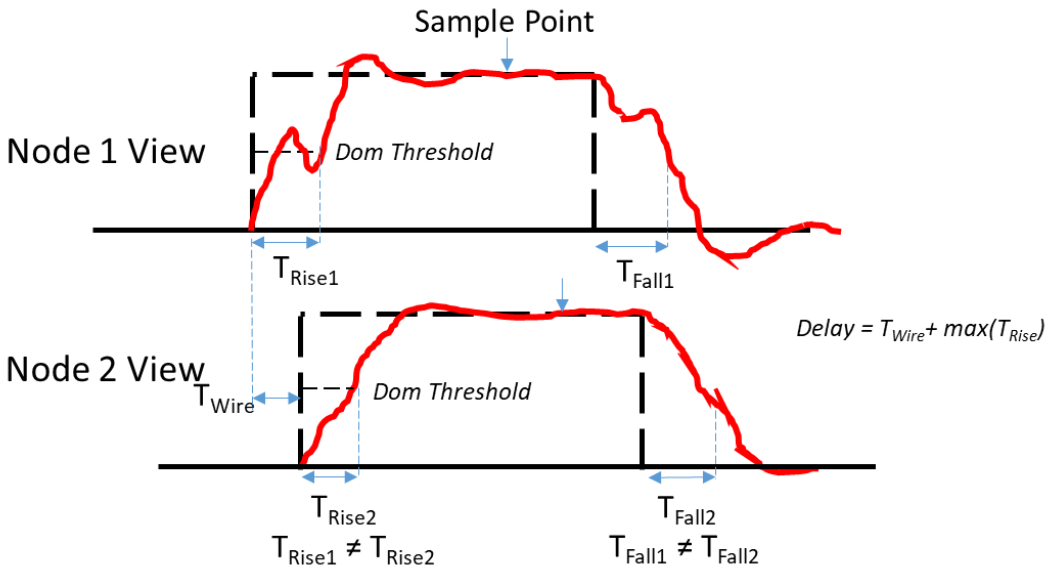


Figure I-8 – Propagation Delay and Rise Time Effects

While the SJW aligns the bits to the edge, the combined impact of the rise time and wire delay can impact the detected edge, delaying the sample point placement. Furthermore, there are two rise times to take into account (the round trip of arbitration). The worst-case round-trip rise and fall times must now be added to the propagation: 2*max (T_{Rise}, T_{Fall}). This is twice the worst-case rise or fall time for any transmitter-receiver pair on the bus. This will most often be the fall time due to the transient voltage suppression placed on nodes and the nature of the CAN bus physical layer. This constraint does not impact PS2 as it is accounted for by the propagation.

I-4.5 Bit Tolerance Accumulation

Because bit timing tolerance (oscillator and bit setting error) is allowed between nodes, they can accumulate skew as time progress. While this is corrected for by resynchronization, there can be many bit-times of accumulated error before a rising edge is seen. ISO-11898-1 provides equations for arbitration bit tolerance accumulations. There are three scenarios for bit tolerance accumulation. The worst case should be used when assessing a bus.

1. Accumulation of error between a rising edge and the next falling edge. Note that Bit Tolerance error accumulation is larger for cases (2) and (3). Condition (1), however, becomes relevant when considering transceiver asymmetry, as will be shown later.
2. Accumulation of error between two resynchronizations. This can cause a receiver to be too fast or too slow when compared to a transmitter.

APPENDIX I
CONFIGURATION OF BIT TIMING

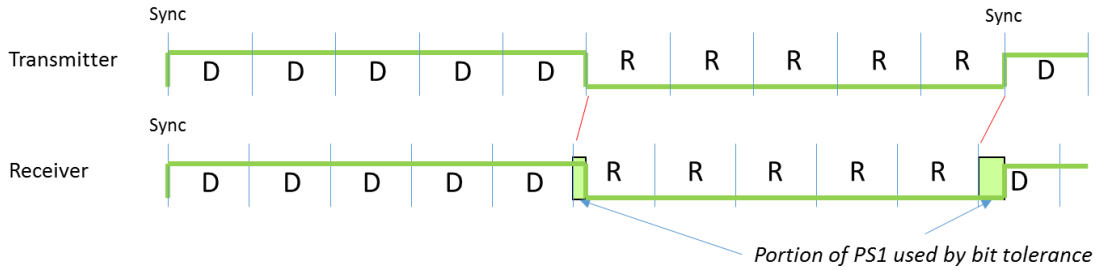


Figure I-9 – Fast Receiver, Slow Transmitter Impacting Receiver's PS1

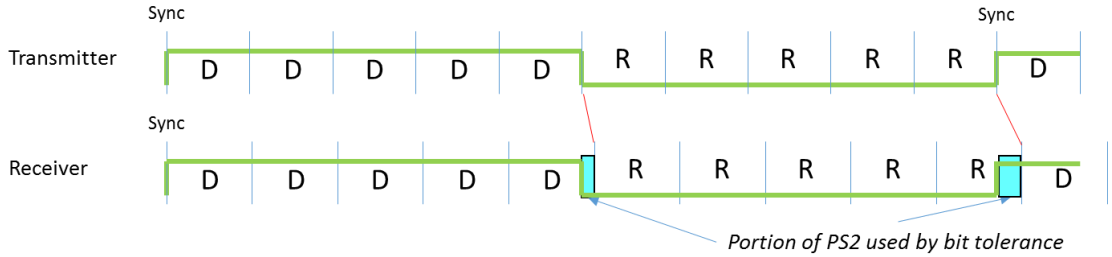


Figure I-10 – Slow Receiver, Fast Transmitter Impacting Receiver's PS2

The accumulated error from this scenario is:

$$BitToleranceAccumulation \leq 20dfbt_A$$

3. Accumulation of error between a resynchronization and a node sampling its own bit after an error flag. The accumulated error can be up to:

$$BitToleranceAccumulation \leq 2dfbt_A(12 + SP_A)$$

The error accumulated from scenario (3) is always worse than (2).

I-4.6 Transceiver Asymmetry

As Dr. Mutter has noted in his recent publications, transceivers are not ideal and do not have the crisp edges on bits and can have bit lengths that differ from the nominal. For example, a transceiver may have tolerance such that the average dominant bit is longer than the average recessive bit while still maintaining overall bit time.

When a bit is too long, it eats into the subsequent bit. This requires moving the sample point further to the right (later in the bit time) for the shortened bit. When a received bit appears too short, the other nodes may sample after the bit has already started to transition. This is compensated for by ensuring PS2 can accommodate this type of error. Note that the rising edge of transceivers is general accurate and is ignored when compared to the falling edge behavior.

There are thus two types of transceiver asymmetry: one that makes the dominant bit longer and one that makes the dominant bit shorter. For the latency calculations and simplification of bus design, the worst-case

asymmetries should be used. This information is found in the datasheet for the transceiver but
ISO-11898-2:2016 also provides limits on the skew, as shown in Table I-3 and Table I-4. These limits will be used as all transceivers should at least be complaint to this standard.

Table I-3 – ISO-11898-2:2016 2mbps Transceiver Asymmetry Requirements

Parameter	Notation	Min (ns)	Max (ns)
Transmitted recessive bit width at 2 Mbit/s	$t_{Bit(Bus)}$	435	530
Received recessive bit width at 2 Mbit/s	$t_{Bit(RXD)}$	400	550
Receiver timing symmetry at 2 Mbit/s	Δt_{Rec}[a]	-65	+40
[a] $\Delta t_{Rec} = t_{Bit(RXD)} - t_{Bit(Bus)}$ *All requirements in this table apply concurrently. Therefore, not all combinations of $t_{Bit(Bus)}$ and Δt_{Rec} are compliant with $t_{Bit(RXD)}$.*			

Table I-4 – ISO-11898-2:2016 5mbps Transceiver Asymmetry Requirements

Parameter	Notation	Min (ns)	Max (ns)
Transmitted recessive bit width at 5 Mbit/s	$t_{Bit(Bus)}$	155	210
Received recessive bit width at 5 Mbit/s	$t_{Bit(RXD)}$	120	220
Receiver timing symmetry at 5 Mbit/s	Δt_{Rec}[a]	-45	+15
[a] $\Delta t_{Rec} = t_{Bit(RXD)} - t_{Bit(Bus)}$ *All requirements in this table apply concurrently. Therefore, not all combinations of $t_{Bit(Bus)}$ and Δt_{Rec} are compliant with $t_{Bit(RXD)}$.*			

I-4.7 Combining Bit Timing Tolerance and Transceiver Asymmetry

Bit timing tolerance and transceiver asymmetry conditions are additive for certain bit scenarios. Finding the worst case of the combinations is required before setting the bit time.

Since asymmetry is most significant at a falling edge, there are bit timing settings that drive Bit Tolerance condition (1) + asymmetry to be worse than bit tolerance condition (2) and (3). This is illustrated in Figure I-10 and Figure I-11.

APPENDIX I
CONFIGURATION OF BIT TIMING

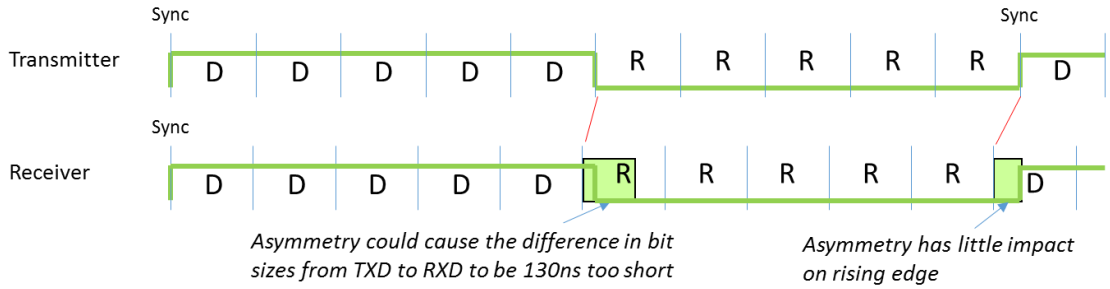


Figure I-11 - Fast Receiver, Slow Transmitter and Asymmetry Impacts Receiver's PS1

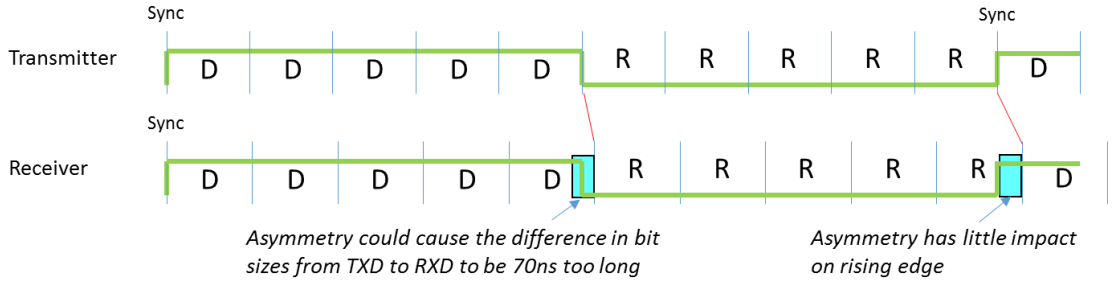


Figure I-12 – Slow Receiver, Fast Transmitter and Asymmetry Impacts Receiver's PS2

The bit tolerance and asymmetry error impacting Phase Segment 1 is governed by:

$$\max[20dfbt_A, 2dfbt_A(12 + SP_A), 10dfbt_A + Asym_1]$$

$$Asym_1 = bt_{Reference} - \min(t_{bit(Bus)}) - \min(\Delta t_{Rec})$$

The bit tolerance and asymmetry error impacting Phase Segment 2 is governed by:

$$\max[20dfbt_A, 2dfbt_A(12 + SP_A), 10dfbt_A + Asym_2]$$

$$Asym_2 = \max(t_{bit(Bus)}) - bt_{Reference} + \max(\Delta t_{Rec})$$

I-5 Synchronization Jump Width

As noted previously, the SJW must be larger than the bit error accumulation such that skew does not build up over time. For arbitration, the result is that

$$bt_A * SJW_A = bt_A * \min(PS1_A, PS2_A) > Bit\ Tolerance\ Error$$

$$bt_A * SJW_A > df * (2 * bt_A[13 - PS2_A])$$

Since 1-SP = PS2, this can also be written as:

$$SJW_A > 2df(12 + SP_A)$$

APPENDIX I
CONFIGURATION OF BIT TIMING

I-6 Arbitration Phase Equation

With the equations in Figure I-4, the definitions above, and the CAN bus design parameters, a range can be placed on the arbitration sample point location. Graphically, this is illustrated in Figure I-9.

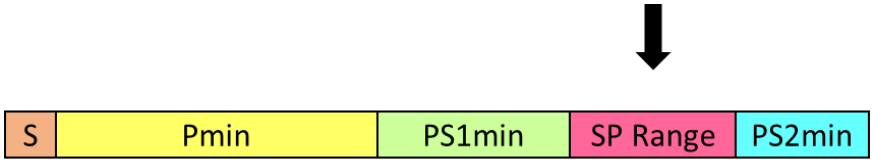


Figure I-13 – Sample Point Range

Earliest Sample Point Location:

$$SP > ResultionError + [RoundTripWireTime + ProcessingDelays + WireAsymmetry] + [BitToleranceAccumulation + TransceiverAsymmetry]$$

$$bt_A SP_A > bt_A R_A + 2[T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})] + \max[20dfbt_A, 2dfbt_A(12 + SP_A), 10dfbt_A + Asym_1]$$

$$Asym_1 = bt_{Reference} - \min(t_{bit(Bus)}) - \min(\Delta t_{Rec})$$

Latest Sample Point Location:

$$1 - SP > [BitToleranceAccumulation + TransceiverAsymmetry]$$

$$bt_A(1 - SP_A) > \max[20dfbt_A, 2dfbt_A(12 + SP_A), 10dfbt_A + Asym_2]$$

$$Asym_2 = \max(t_{bit(Bus)}) - bt_{Reference} + \max(\Delta t_{Rec})$$

These equations can be manipulated to solve for the maximum bit rates allowed, as is shown in the appendix. The results, in terms of bit time, will be used for the proposed requirements.

Using ARINC 825-3 bit timing settings and commonly observed wire quality parameters and transceiver impedance, the equations produce results similar to the allowable from ARINC 825, as seen in Figure I-14. Note that the wire and node assumptions are conservative in short-bus regimes.

APPENDIX I
CONFIGURATION OF BIT TIMING

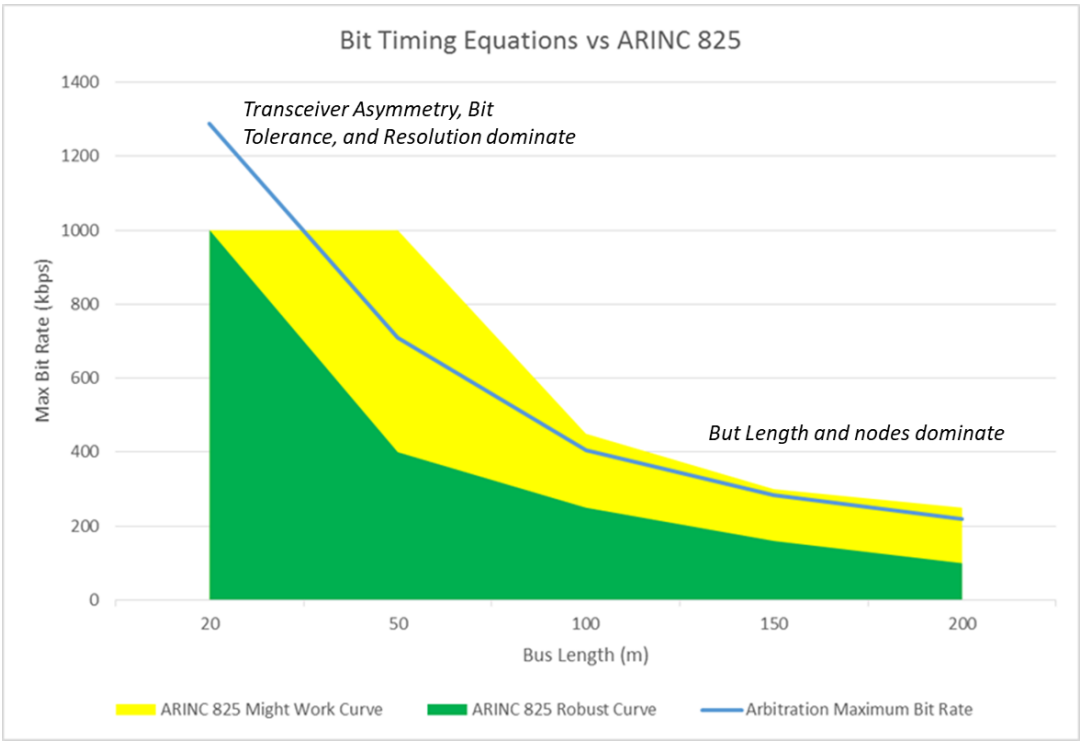


Figure I-14 – Arbitration Equations versus ARINC 825

I-7 Data Phase Bit Timing

There are two distinct conditions that constrain the data phase bit time: the nominal data phase stuff bit and the bit rate switch. For the most part, the data phase is simpler to assess than arbitration. However, the bit rate switch introduces additional complexity.

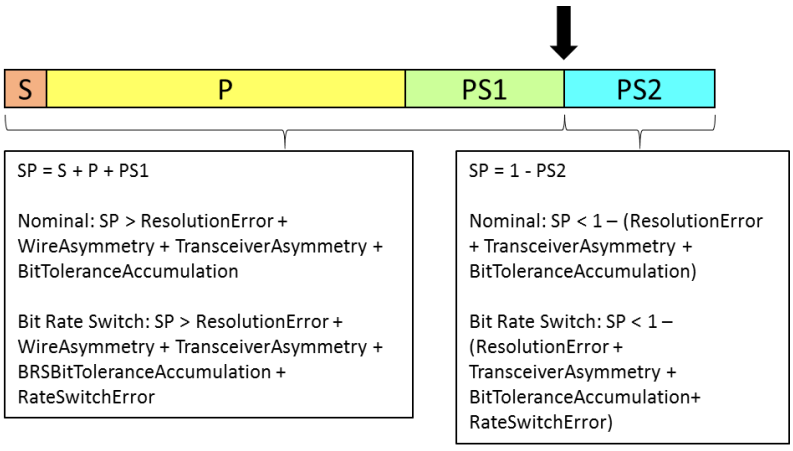


Figure I-15 – Data Phase Bit Time Constraints

I-7.1 Nominal Data Phase Constraints

During the data phase, the equations for sample point are very similar to those for the arbitration phase. The elements from the arbitration

phase equation are listed below with their applicability to the data phase.

- Resolution Error: Still applies as in the arbitration phase. It can cause jitter to the bit of 1 TQ.
- Round Trip Wire Time: Not relevant as data flow is unidirectional.
- Processing Delays: Not relevant as data flow is unidirectional with no arbitration processing required.
- Wire Asymmetry: Continues to round off the bit but the factor of 2 is removed since flow is unidirectional. The bit must still stabilize above the dominant threshold or below the recessive threshold prior to the sample point.
- Transceiver Asymmetry: applies as in arbitration phase. This is a major limitation on the data phase since it, like the wire effects, is not proportional to the bit rate but rather is an absolute time.
- Bit Tolerance Accumulation: same conditions as in arbitration phase. This is proportional to the bit rate so has the same relative impact as in arbitration.

Earliest Sample Point Location:

$$bt_D * SP_D > [ResolutionError] + [WireAsymmetry] + [TransceiverAsymmetry] + [BitToleranceAccumulation]$$

$$bt_D SP_D > bt_D R_D + [\max(T_{Rise}, T_{Fall})] + \max[20dfbt_D, 10dfbt_D + Asym_1]$$

$$Asym_1 = bt_{Reference} - \min\left(t_{bit(Bus)}\right) - \min(\Delta t_{Rec})$$

Latest Sample Point Location:

$$bt_D * SP_D < 1 - ([TransceiverAsymmetry] + [BitToleranceAccumulation])$$

$$bt_D(1 - SP) > \max[20dfbt_D, 10dfbt_D + Asym_2]$$

$$Asym_2 = \max\left(t_{bit(Bus)}\right) - bt_{Reference} + \max(\Delta t_{Rec})$$

APPENDIX I
CONFIGURATION OF BIT TIMING

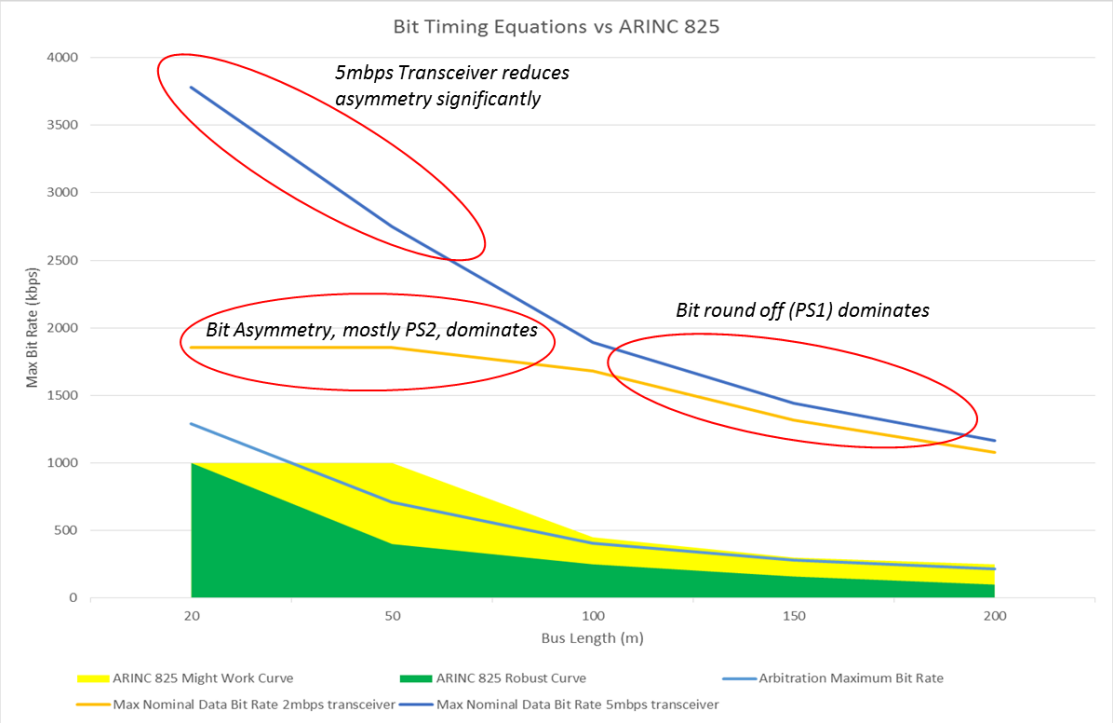


Figure I-16 – Nominal Data Rate Curves

I-8 Bit Rate Switch Data Phase Constraints

The Bit Rate Switch constraints have a different formula for the Bit Tolerance Accumulation and the significant impact of varying sample point locations between nodes takes effect.

I-8.1 Bit Rate Switch Bit Tolerance Accumulation

The bit rate switch region of a CAN message has a specific bit pattern prior to the data field. This pattern, combined with the worst-case pattern of data bits, produces the sequence shown in Figure I-17. PS2 of the BRS bit is switched to the data rate so BRS can be shorter than a normal arbitration bit. Since the bit rate switch ends with a rising edge, bit asymmetry has no impact on the calculations.

APPENDIX I
CONFIGURATION OF BIT TIMING

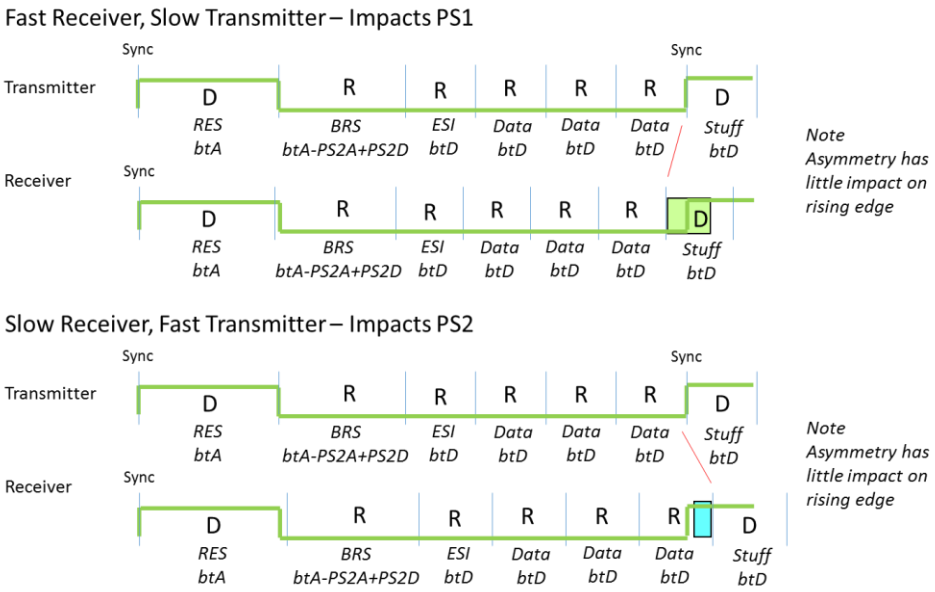


Figure I-17 – BRS Bit Tolerance Accumulation

The accumulated bit tolerance error can be:

$$2df(2bt_A - bt_APS2_A + bt_DPS2_D + 4bt_D)$$

I-8.2 Bit Rate Switch Sample Point Difference

A significant skew is introduced by a CAN implementation where nodes have different sample point locations. This is not recommended but derived here to show the impact and allow bus designers to study the impact of varying the sample point.

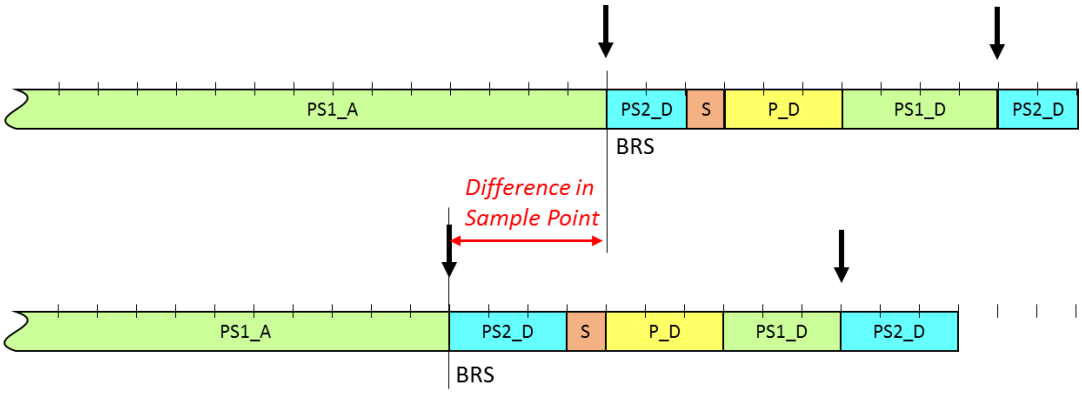


Figure I-18 – Bit Rate Switch and Sample Point Difference Impacting PS1

APPENDIX I
CONFIGURATION OF BIT TIMING

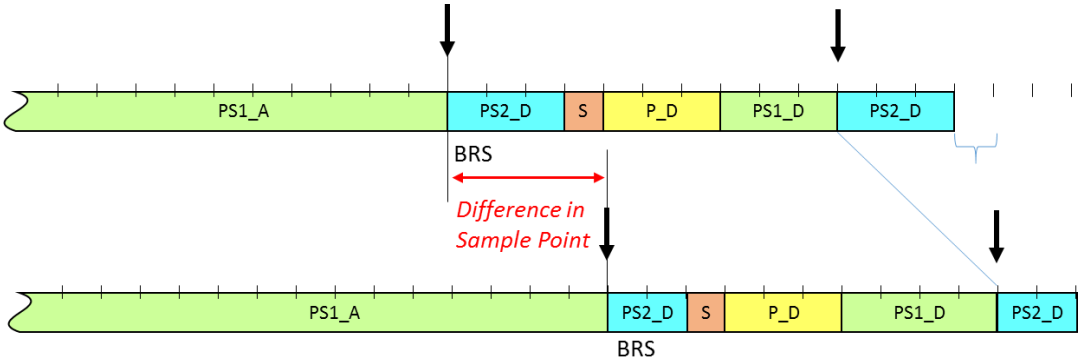


Figure I-19 – Bit Rate Switch and Sample Point Difference Impacting PS2

When sample points differ between nodes, skew equal to the time difference of the sample points is introduced into the data phase sampling, as shown in Figure I-16 and Figure I-17. To prevent this from causing a mis-sample, there must be enough margin left in Phase Segments 1 and 2 to accommodate this skew.

$$Rate\ Switch\ Error = bt_A abs(SP_{A1} - SP_{A2}) = bt_A * dSP_A$$

I-8.3 Bit Rate Switch Equations

Pulling these effects together, one can derive the bounds imposed by the bit rate switch on the data phase sample point location.

Earliest Sample Point Location:

$$bt_D * SP_D > [ResolutionError] + [WireAsymmetry] + [TransceiverAsymmetry] + [BitToleranceAccumulation] + [RateSwitchError]$$

$$bt_D SP_D > bt_D R_D + [\max(T_{Rise}, T_{Fall})] + [2df(2bt_A - bt_A PS2_A + bt_D PS2_D + 4bt_D)] + bt_A * dSP_A$$

Latest Sample Point Location:

$$bt_D * (1 - SP_D) > BRSBitToleranceAccumulation + RateSwitchError$$

$$bt_D(1 - SP_D) > [2df(2bt_A - bt_A PS2_A + bt_D PS2_D + 4bt_D)] + bt_A * dSP_A$$

When arbitration sample points can vary, the bit rate switch has a major impact on the allowable data phase speeds. As the data phase moves to higher speeds (X = bit time A / bit time D), the limitation becomes more constraining, as seen by the gray line in Figure I-20 and Figure I-21 for a 10% sample point variance.

APPENDIX I
CONFIGURATION OF BIT TIMING

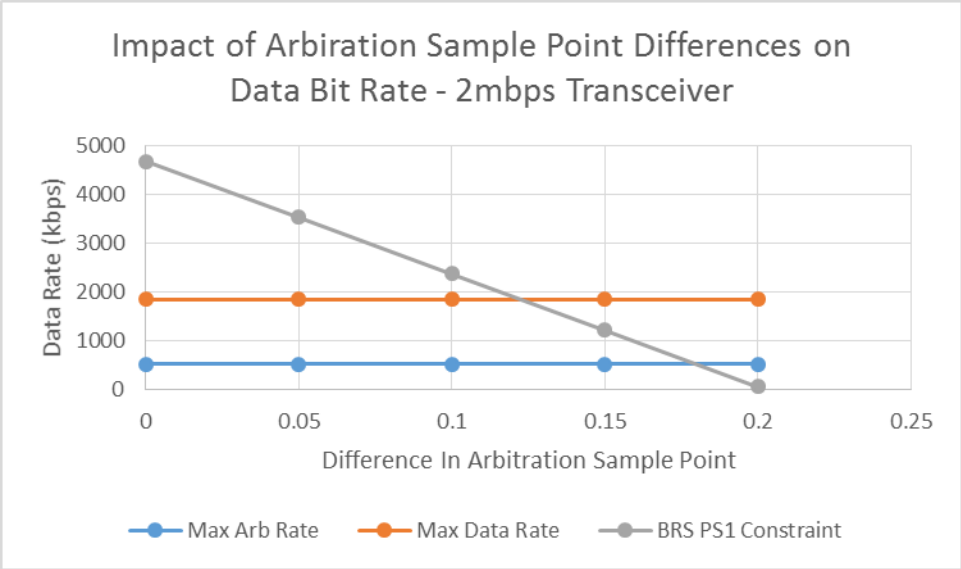


Figure I-20 – Bit Rate Switch Impacts Example Bit Timing – 2mbps Transceiver

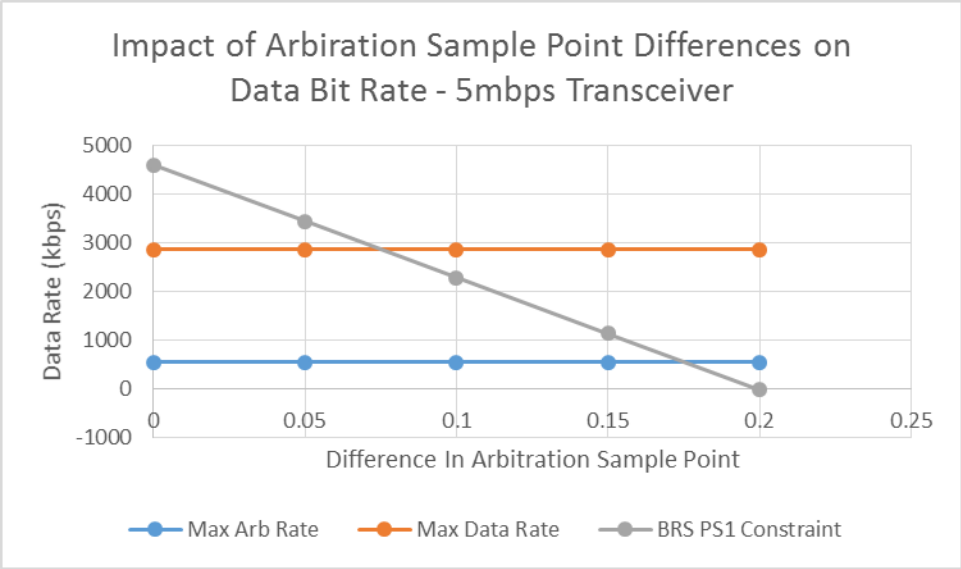


Figure I-21 – Bit Rate Switch Impacts Example Bit Timing – 5mbps Transceiver

The figures above show the phase segment 1 constraint on the bit time (gray line) versus the max rates when ignoring BRS switch error (blue and orange lines). In many cases, the phase segment 2 constraint is worse; this is not shown here but keep in mind that many of the points in the curves above violate this constraint. This condition imposes a significant constraint on the bit time. For example, if the sample point is allowed to vary between 75% to 85% of the bit time, the maximum allowed data rate increase is only 1.2. Further constraining the bit variance to be between 75% to 82.5% of the bit time allows for data rates up to 2 times the arbitration rate.

APPENDIX I
CONFIGURATION OF BIT TIMING

I-9 References

1. Dr. Arthur Mutter; Robustness of CAN FD Bus System – About Oscillator Tolerance and Edge Deviations; Proceedings of the 14th international CAN Conference, Paris, France, 2013.
2. Florian Hartwich; The Configuration of the CAN Bit Timing; Proceedings of the 6th international CAN Conference, Turin, Italy, 1999.

I-10 Bit Timing Derivations

df	Bit Tolerance – combination of skew between nodes' oscillators and bit configuration error.
T_{Wire}	Wave travel time on the wire between the furthest apart nodes – unidirectional travel time.
T_{Rise}	Rise time of the waveform at a given node (recessive to dominant edge)
T_{Fall}	Fall time of the waveform at a given node (dominant to recessive edge)
T_{Node}	Transceiver delay of the node
R_A	The Resolution Error of a given node during the arbitration phase in percentage of bit time. This is equivalent to 1 TQ for the node.
R_D	The Resolution Error of a given node during the data phase in percentage of bit time. This is equivalent to 1 TQ for the node.
B_A	Nominal Bit Rate of the arbitration phase
TQ_A	The length of a Time Quanta of a given node during the arbitration phase. TQ per bit is given by bt_A / TQ_A.
bt_A	Duration of a bit during arbitration
SP_A	Sample Point location in arbitration, in percentage of bit time
dSP_A	Tolerance of location of Sample Point location between two nodes, in percentage of bit time. If SP is specified as 80% ± 5%, dSP_A = 5%.
S_A	Sync Segment during arbitration, in percentage of bit time. This segment is equivalent to 1 TQ for the node.
P_A	Propagation Segment during the arbitration phase for a given node, in percentage of bit time
$PS1_A$	Phase Segment 1 during the arbitration phase for a given node, in percentage of bit time
$PS2_A$	Phase Segment 2 during the arbitration phase for a given node, in percentage of bit time
X	Multiplier between arbitration and data phase rates. This is equivalent to B_D/B_A or bt_A/bt_D
B_D	Nominal Bit Rate of the data phase
TQ_D	The length of a Time Quanta of a given node during the data phase. TQ per bit is given by bt_D / TQ_D.
bt_D	Duration of a bit during data
SP_D	Sample Point location in data, in percentage of bit time
dSP_D	Difference in sample point location between two nodes, in percentage of bit time
S_D	Sync Segment during data in percentage of bit time. This is equivalent to 1 TQ for the node.
P_D	Propagation Segment during the data phase for a given node, in percentage of bit time

APPENDIX I
CONFIGURATION OF BIT TIMING

$PS1_D$	Phase Segment 1 during the data phase for a given node, in percentage of bit time
$PS2_D$	Phase Segment 2 during the data phase for a given node, in percentage of bit time

I-10.1 Arbitration

I-10.1.1 Arbitration Phase Segment 1 (Min Sample Point)

$$SP > ResultionError + [RoundTripWireTime + ProcessingDelays + WireAsymmetry] + [BitToleranceAccumulation + TransceiverAsymmetry]$$

$$Asym_1 = bt_{Reference} - \min(t_{bit(Bus)}) - \min(\Delta t_{Rec})$$

$$2dfbt_A(13 - PS2_A) = 2dfbt_A(12 + SP_A)$$

$$bt_A SP_A > bt_A R_A + 2[T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})] + \max[20dfbt_A, 2dfbt_A(12 + SP_A), 10dfbt_A + Asym_1]$$

Note that the following is always true since the sample point cannot be larger than a bit time:

$$2dfbt_A(12 + SP_A) > 20dfbt_A$$

SJW also must always be greater than the accumulating error.

$$bt_A SJW \geq 2dfbt_A(12 + SP_A)$$

Concerning the maximum of the bit tolerance and transceiver asymmetry stack up, if bit time is 250kbps or slower, generally the error flag condition is the worst. If faster, the transceiver asymmetry case is worst. For example, at 500kbps 10 bits times yields 102ns of skew but the 5-bit times plus asymmetry case yields 170ns of skew. These calculations assume 0.2% oscillator tolerance and 2mpbs capable transceivers.

I-10.1.1.1 Slow Bit Time Case

This case is used when the following condition holds:

$$2dfbt_A(12 + SP_A) > 10dfbt_A + Asym_1$$

Our primary equation becomes:

$$bt_A SP_A > bt_A R_A + 2[T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})] + 2dfbt_A(12 + SP_A)$$

Solving for sample point restriction:

$$bt_A SP_A > \frac{bt_A R_A + 2[T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})] + 24dfbt_A}{1 - 2df}$$

Solving for bit time limit:

$$bt_A(SP_A(1 - 2df) - R_A - 24df) > 2[T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})]$$

$$bt_A > \frac{2[T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})]}{SP_A(1 - 2df) - R_A - 24df}$$

I-10.1.1.2 Fast Bit Time Case

This case is used when the following condition holds:

$$2dfbt_A(12 + SP_A) < 10dfbt_A + Asym_1$$

Our primary equation becomes:

$$bt_A SP_A > bt_A R_A + 2[T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})] + 10dfbt_A + Asym_1$$

$$bt_A(SP_A - R_A - 10df) > 2[T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})] + Asym_1$$

Solving for bit time limit:

$$bt_A > \frac{2[T_{wire} + T_{Node} + \max(T_{Rise}, T_{Fall})] + Asym_1}{(SP_A - R_A - 10df)}$$

I-10.1.2 Arbitration Phase Segment 2 (Max Sample Point)

$$1 - SP > [BitToleranceAccumulation + TransceiverAsymmetry]$$

$$Asym_2 = \max\left(t_{bit(Bus)}\right) - bt_{Reference} + \max(\Delta t_{Rec})$$

$$bt_A(1 - SP_A) > \max[2dfbt_A(12 + SP_A), 10dfbt_A + Asym_2]$$

I-10.1.2.1 Slow Bit Time Case

This case is used when the following condition holds:

$$2dfbt_A(12 + SP_A) > 10dfbt_A + Asym_2$$

Our primary equation becomes:

$$(1 - SP_A) > 2df(12 + SP_A)$$

For this case there is no bit time dependency; the equations provide a restriction on bit proportions to ensure that Phase Segment 2 is large enough.

$$SP_A < \frac{1 - 24df}{1 + 2df}$$

I-10.1.2.2 Fast Bit Time Case

This case is used when the following condition holds:

$$2dfbt_A(12 + SP_A) < 10dfbt_A + Asym_2$$

Our primary equation becomes:

$$bt_A(1 - SP_A) > 10dfbt_A + Asym_2$$

$$bt_A(1 - SP_A - 10df) > Asym_2$$

Solving for bit time limit:

$$bt_A > \frac{Asym_2}{(1 - SP_A - 10df)}$$

I-10.2 Data

I-10.2.1 Data Phase Segment 1 – Nominal (Min Sample Point)

$$SP > ResultionError + [WireAsymmetry] + [BitToleranceAccumulation + TransceiverAsymmetry]$$

$$Asym_1 = bt_{Reference} - \min(t_{bit(Bus)}) - \min(\Delta t_{Rec})$$

$$bt_D SP_D > bt_D R_D + [\max(T_{Rise}, T_{Fall})] + \max[20dfbt_D, 10dfbt_D + Asym_1]$$

A node sampling its own error flag when switching from data phase to arbitration phase is left out of the bit tolerance accumulation as that only impacts arbitration bit timing and, even then, is less constraining than the conditions captured for arbitration above.

I-10.2.1.1 Slow Bit Time Case

This case is used when the following condition holds:

$$20dfbt_D > 10dfbt_D + Asym_1$$

Our primary equation becomes:

$$bt_D SP_D > bt_D R_D + [\max(T_{Rise}, T_{Fall})] + 20dfbt_D$$

$$bt_D(SP_D - R_D - 20df) > [\max(T_{Rise}, T_{Fall})]$$

Solving for bit time limit:

$$bt_D > \frac{[\max(T_{Rise}, T_{Fall})]}{(SP_D - R_D - 20df)}$$

I-10.2.1.2 Fast Bit Time Case

This case is used when the following condition holds:

$$20dfbt_D < 10dfbt_D + Asym_1$$

Our primary equation becomes:

$$bt_D SP_D > bt_D R_D + [\max(T_{Rise}, T_{Fall})] + 10dfbt_D + Asym_1$$

$$bt_D(SP_D - R_D - 10df) > [\max(T_{Rise}, T_{Fall})] + Asym_1$$

Solving for bit time limit:

$$bt_D > \frac{[\max(T_{Rise}, T_{Fall})] + Asym_1}{(SP_D - R_D - 10df)}$$

I-10.2.2 Data Phase Segment 2 – Nominal (Max Sample Point)

$$1 - SP > [BitToleranceAccumulation + TransceiverAsymmetry]$$

$$bt_D(1 - SP) > \max[20dfbt_D, 10dfbt_D + Asym_2]$$

APPENDIX I
CONFIGURATION OF BIT TIMING

$$Asym_2 = \max\left(t_{bit(Bus)}\right) - bt_{Reference} + \max(\Delta t_{Rec})$$

I-10.2.2.1 Slow Bit Time Case

This case is used when the following condition holds:

$$20dfbt_D > 10dfbt_D + Asym_1$$

Our primary equation becomes:

$$1 - SP_D > 20df$$

For this case there is no bit time dependency; the equations provide a restriction on bit time to ensure that Phase Segment 2 is large enough.

$$SP_D < 1 - 20df$$

I-10.2.2.2 Fast Bit Time Case

This case is used when the following condition holds:

$$20dfbt_D < 10dfbt_D + Asym_1$$

Our primary equation becomes:

$$bt_D(1 - SP) > 10dfbt_D + Asym_2$$

$$bt_D(1 - SP_D - 10df) > Asym_2$$

Solving for bit time limit:

$$bt_D < \frac{Asym_2}{(1 - SP_D - 10df)}$$

I-10.2.3 Data Phase Segment 1 – Bit Rate Switch (Min Sample Point)

There is no transceiver asymmetry impact in this case since bit sequence ends with rising edge.

$$SP > ResultionError + [WireAsymmetry] + [BRSBitToleranceAccumulation] + RateSwitchError$$

$$bt_DSP_D > bt_DR_D + [\max(T_{Rise}, T_{Fall})] + [2df(2bt_A - bt_APS2_A + bt_DPS2_D + 4bt_D)] + bt_A * dSP_A$$

$$Rate\ Switch\ Error = bt_Aabs(SP_{A1} - SP_{A2}) = bt_A * dSP_A$$

$$1 - SP = PS2$$

$$X = bt_A/bt_D$$

$$bt_DSP_D > bt_DR_D + [\max(T_{Rise}, T_{Fall})] + 2dfbt_D[\mathrm{X}(1 + SP_A) + (5 - SP_D)] + Xbt_D * dSP_A$$

$$\frac{bt_DSP_D}{1 + 2df} > \frac{bt_DR_D + [\max(T_{Rise}, T_{Fall})] + 2dfbt_D[\mathrm{X}(1 + SP_A) + 5] + Xbt_D * dSP_A}{1 + 2df}$$

$$bt_D(SP_D - R_D - 2df[\mathrm{X}(1 + SP_A) + (5 - SP_D)] - Xbt_D * dSP_A) > [\max(T_{Rise}, T_{Fall})]$$

APPENDIX I
CONFIGURATION OF BIT TIMING

$$bt_D > \frac{[\max(T_{Rise}, T_{Fall})]}{(SP_D - R_D - 2df[\mathrm{X}(1 + SP_A) + (5 - SP_D)] - X * dSP_A)}$$

I-10.2.4 Data Phase Segment 2 – Bit Rate Switch (Max Sample Point)

There is no transceiver asymmetry impact in this case since the bit sequence ends with rising edge.

$$1 - SP > [BRSBitToleranceAccumulation] + RateSwitchError$$

$$bt_D(1 - SP_D) > [2df(2bt_A - bt_A PS2_A + bt_D PS2_D + 4bt_D)] + bt_A * dSP_A$$

$$Rate\ Switch\ Error = bt_A(SP_{A1} - SP_{A2}) = bt_A * dSP_A$$

$$1 - SP = PS2$$

$$X = bt_A / bt_D$$

$$bt_D(1 - SP_D) > 2df bt_D[\mathrm{X}(1 + SP_A) + (5 - SP_D)] + Xbt_D * dSP_A$$

For this case there is no bit time dependency; the equations provide a restriction on bit time to ensure that Phase Segment 2 is large enough.

$$SP_D < \frac{1 - 2df[X(1 + SP_A) + 5] - XdSP_A}{1 - 2df}$$

APPENDIX J MANAGEMENT INFORMATION BASE (MIB) COUNTERS

J-1 Management Information Base

Many of the network technologies used in industry have a common set of statistics that are tracked on each network interface for use by any network management entity to troubleshoot and debug. In Ethernet networks, this is referred to as the Management Information Base (MIB) and accessible by a network manager via the Simple Network Management Protocol (SNMP). The automotive industry also keeps a standard set of information for all OBD-II complaint CAN nodes.

In order to foster CAN LRU interoperability and consistent design patterns to enable higher level system and tool designs, a set of key MIBs for CAN interfaces is proposed below, based off of network management experience for CAN bus. The counters proposed were chosen because they categorize the different events and actions that can be taken on topics specified in ARINC 825. Whenever a node takes an action for something that is in the scope of ARINC 825 (OSI layers 1, 2, and some of 3 through the High Integrity and High Availability messages), there should be a counter to track it.

J-2 Use of the Counters

Each MIB counter is set to zero on LRU initialization or LRU reset.

Each MIB counter rolls over to zero when the maximum size is exceeded.

Each MIB counter is incremented when the associated event occurs, as defined in the MIB description.

Designs may keep track of additional MIBs as deemed necessary.

When MIB objects are not tracked, the value is set to all 1's (0xFF) similar to the PHSM.

If a request for an unsupported MIB is received, nodes either ignore the request or set the response value to all 1’s (0xF).

The choice of method for extracting the MIB counters is up to the higher-level protocol(s). For example, additional messages can be defined to transmit the data, as shown in the example Figure J-1 below, or a protocol or JTAG can be used to extract the data.

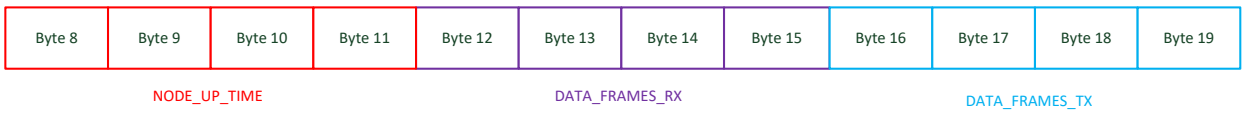


Figure J-1 – Example MIB Data Message

APPENDIX J
MANAGEMENT INFORMATION BASE (MIB) COUNTERS

Table J-1 – Suggested ARINC 825 MIB Counters

OBJECT-TYPE	DATA-TYPE	SIZE-IN-BITS	UNITS	Cardinality	DESCRIPTION
EQUIPMENT_UP_TIME	UNSIGNED INTEGER	32	centiseconds	(1 … 1)	The number of centiseconds since the LRU was initialized or initialized after a reset. Defined as a unit of time equal to 0.01 seconds
STATE_TRNS_BO_TO_EA	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one every time the associated CAN controller transitions from Bus Off state to Error Active State or from INIT State to Error Active state (i.e., counter will be 1 when the node first joins the bus)
STATE_TRNS_EA_TO_EP	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one every time the associated CAN controller transitions from Error Active State to Error Passive State
STATE_TRNS_EP_TO_BO	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one every time the associated CAN controller transitions from Error Passive state to Bus Off state. A subset of this counter is used in the PHSM.
STATE_TRNS_EP_TO_EA	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one every time the associated CAN controller transitions from Error Passive state to Error Active State
NB_ERR_BIT	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each Bit Error that is detected by the CAN Controller
NB_ERR_STUFF	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each Stuff Error that is detected by the CAN Controller
NB_ERR_CRC	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each CAN protocol CRC Error that is detected by the CAN Controller
NB_ERR_FORM	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each Form Error that is detected by the CAN Controller
NB_ERR_ACK	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each ACK Error that is detected by the CAN Controller. A subset of this counter is used in the PHSM.
NB_ERR_ RX	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each Error that occurs while the CAN Controller is a Receiver. A subset of this counter is used in the PHSM.
NB_ERR_TX	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each Error, other than an Acknowledgment Error, that occurs while the CAN Controller is a Transmitter. A subset of this counter is used in the PHSM.

APPENDIX J
MANAGEMENT INFORMATION BASE (MIB) COUNTERS

DATA_FRAME_OCTETS_RX	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by a value ranging from 6 to 72 depending on the id type and data field size for each data frame received by CAN Controller. This is a layer two calculation, neglecting stuff bits, where the result is a sum of the frame overhead (6 bytes standard id or 8 bytes extended id) plus the number of octets of the data field (0 – 64 bytes)
DATA_FRAME_OCTETS_TX	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by a value ranging from 6 to 72 depending on the id type and data field size for each data frame transmitted by CAN Controller. This is a layer two calculation, neglecting stuff bits, where the result is a sum of the frame overhead (6 bytes standard id or 8 bytes extended id) plus the number of octets of the data field (0 – 64 bytes)
DATA_FRAMES_RX	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each successfully received Data Frame
DATA_FRAMES_TX	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each successfully transmitted Data Frame
REMOTE_FRAME_RX	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each received Remote Frame
REMOTE_FRAME_TX	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each transmitted Remote Frame
OVERLOAD_FRAME_RX	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each received Overload Frame
OVERLOAD_FRAME_TX	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each transmitted Overload Frame
TOTAL_EXT_ID_FRAME	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each Data Frame received utilizing the extended (29 bit) ID format
TOTAL_STD_ID_FRAME	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each Data Frame received utilizing the standard (11 bit) ID format
ABORTED_TX_COUNTER	UNSGINED INTEGER	32	scalar	(1 … *)	Incremented by one for each Data frame that was loaded into the CAN controller for transmission but aborted prior to successful transmission
TX_QUEUE_OVERFLOW_COUNTER	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one whenever a message, either queued or buffered for transmission or about to be added to the queue or buffer, is dropped during an enqueue process
TX_QUEUE_REMOVED_COUNTER	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one whenever a message in a transmit queue or buffer is removed out of the queue or buffer for reasons other than an overflow (the enqueue of a new message failing or bumping an old one)

APPENDIX J
MANAGEMENT INFORMATION BASE (MIB) COUNTERS

RX_QUEUE_OVERFLOW_COUNTER	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one whenever the CAN interface's mailboxes or receive queues overflow, resulting in a discarded message
CONFIG_REG_MISMATCH	UNSGINED INTEGER	32	scalar	(1 … *)	Incremented by one for each detected mismatch between the expected configuration value and the current value of the CAN controller configuration registers
CAN_CONTROLLER_REC	UNSIGNED INTEGER	16	scalar	(1 … *)	The CAN controller's Receive Error Counter (REC)
CAN_CONTROLLER_TEC	UNSIGNED INTEGER	16	scalar	(1 … *)	The CAN controller's Transmit Error Counter (TEC)
UNEXPECTED_DATA_LENGTH	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one when the payload length of a received message does not match the preconfigured length, if applicable
HIGH_INTEGIRTY_MIC_ERRORS*	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each received High Integrity Message that is rejected due to a payload CRC failure
HIGH_INTEGRITY_SNO_ERRORS*	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each received High Integrity Message that is rejected due to its sequence number
AVAILABILITY_SNO_ERRORS	UNSIGNED INTEGER	32	scalar	(1 … *)	Incremented by one for each received High Availability Message that is rejected due to its sequence number