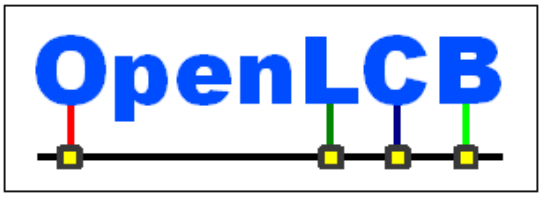
NMRA Standard	
Layout Command Control® (LCC) **Message Network**	
Feb 17, 2015	**S-9.7.3**

Adopted as a NMRA Standard

The OpenLCB Standard document appended to this cover sheet has been formally adopted as a NMRA Standard by the NMRA Board of Directors on the date shown in the *Adopted* column in the *Version History* table below.

Version History

Date	Adopted	Summary of Changes
Feb 17, 2015		Initial version



OpenLCB Standard	
Message Network	
Feb 15, 2015	Adopted

1 Introduction

This Standard contains normative information about the OpenLCB Message Network. Corresponding discussion and background can be found in the corresponding OpenLCB Message Network Technical Note.

The protocol is described via three components: the state machine within the node(s); the messages; and the basic interactions in which the nodes take part. These are separately described below in terms of the general format as well as specific message definitions.

Messages are transported across a specific data-link level implementation, for example using CAN frames, TCP/IP sockets, or other transports. The messages are described first in general terms, then mapped to specific implementations (see Sections 8 and beyond). The states and interactions are the same across all data-link implementations.

2 Intended Use

The messages and interactions described here are used by all OpenLCB nodes to connect to the OpenLCB network. **They are mandatory.**

2.1 References and Context

For background information on format and presentation, see:

- OpenLCB Common Information Technical Note

This Standard is in the context of the following OpenLCB Standard:

- The OpenLCB Unique Identifiers Standard, which specifies Unique Identifiers and how they are defined.

This Standard is in the context of the following OpenLCB CAN Standards:

- The OpenLCB CAN Frame Transfer Standard, which specifies transfer of OpenLCB messages over CAN segments. “CAN” refers to the electrical and protocol specifications as defined in ISO 11898-1:2003 and ISO 11898-2:2003 and their successors.

This Standard is in the context of the following OpenLCB-TCP/IP Standards:

- The OpenLCB-TCP/IP Segment Transfer Standard, which specifies transfer of OpenLCB messages over TCP/IP links.

Conformance with a later version of a referenced standard shall be accepted as conformance with the referenced versions.

3 Messages

3.1 Message Format

OpenLCB messages are sent using the transfer mechanism and format described in the Standard for a specific wire protocol.

All messages shall contain a source Node ID and a Message Type Indicator (MTI). The MTI defines both the general format of the message and its specific type. All messages with the same MTI are of the same type.

3.1.1 Message Type Indicators

The general Message Type Indicator (MTI) is a 16-bit quantity. The MTI values are remapped for specific wire protocols, see the appropriate sections of this document for adaptation to CAN and TCP/IP.

The current allocations are documented in a separate spreadsheet[1]. We keep them in just that one place to avoid conflicting updates. Those allocations are normative.

MTI – Message Type Indicator																
Bit(s)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved		Special	Stream or Datagram	Priority		Type within Priority					Simple Protocol	Address Present	Event Present	Modifier	

[1]See Appendix 1, and the associated TN, it provides concrete examples that may help you understand the material in this document.

MTI Bit-field Descriptions				
Field Name	**Bit Position**	**Size bits**	**Mask hex**	**Description**
Reserved	14 - 15	2	0xC000	Reserved for future use, send and check as zero.
Special	13	1	0x2000	Operationally special, 1= forward through Gateways
Stream or Datagram	12	1	0x1000	0=Regular message, 1=Stream or Datagram message
Priority	10 - 11	2	0x0C00	Gross priority of message, 0 is highest priority
Type within Priority	5 - 9	5	0x03E0	Minor priority determination
Simple Protocol	4	1	0x0010	1=This message should be handled by simple nodes
Address Present	3	1	0x0008	1=This message has a destination address-field
Event Present	2	1	0x0004	1=This message has an event-field
Modifier	0 - 1	2	0x0003	Message-specific extra information

Note that these fields inform the intent of the message, but also the overall format of the rest of the message.

3.1.2 Message Content

The message content consists of:

- The MTI
- The source Node ID
- If the MTI flags it as being present, the destination Node ID
- If the MTI flags it as being present, an Event ID
- Any other content as defined for the specific message type to a maximum of 72 bytes.

The exact format and order are defined by the specific wire protocols, but in all cases the message shall be fully decodable based on the flag-bit information in the MTI.

3.2 States

The message network layer in an OpenLCB node has two states:

- Uninitialized
- Initialized

Nodes shall start in the Uninitialized state.

A node in the Uninitialized state may transmit an Initialization Complete message, but shall not transmit any other message type.

A node in the Initialized state may transmit any message type.

3.3 Definition of Specific Messages

This section defines the format of common core messages. Although there is a short description of the purpose of the message, and related interactions, this is just for identification and explanatory purposes. The meaning of the messages is defined by the interactions in which they take part. These are described in later sections.

When a Node ID is present in the data content of the message, the full 48-bit identifier shall be sent for all wire protocols, specifically including CAN, even if an alias or alternate form is available elsewhere in the message.

3.3.1 Initialization Complete

Indicates that the sending-node initialization is complete and, once the message is delivered, it is reachable on the network.

Name	Description	Simple Protocol	Dest ID	Event ID	Common MTI	Data Content
Initialization Complete	Full Protocol Required	N	N	N	0x0100	Source Node ID
	Simple-set Sufficient	N	N	N	0x0101	

This message has two MTIs, distinguished by the modifier field, to indicate whether the node requires delivery of all the messages in the full protocol, or whether delivery of the Simple Protocol subset is sufficient.

3.3.2 Verify Node ID

Issued to determine which node(s) are present and can be reached.

Name	Description	Simple Protocol	Dest ID	Event ID	Common MTI	Data Content
Verify Node ID	Addressed	-[2]	Y	N	0x0498	Optional Full Node ID
	Global	Y	N	N	0x0490	

3.3.3 Verified Node ID

Reply to the Verify Node ID message.

Name	Description	Simple Protocol	Dest ID	Event ID	Common MTI	Data Content
Verified Node ID Number	Full Protocol Required	Y	N	N	0x0170	Source Node ID
	Simple Subset Sufficient	Y	N	N	0x0171	

[2] By definition, all addressed messages are received by the addressee, so the simple bit is ignored. See TN.

This message has two MTIs, distinguished by the modifier field, to indicate whether the node requires delivery of all the messages in the full protocol, or whether delivery of the Simple Protocol subset is sufficient.

3.3.4 Optional Interaction Rejected (OIR)

This is a reply indicating failure.

Name	Simple Protocol	Dest ID	Event ID	Common MTI	Data Content
Optional Interaction Rejected	N	Y	N	0x0068	Error codes, MTI, optional info

The data contents are, in order:

- Two bytes of error code.
- Two bytes of MTI. If the frame transport only delivered part of the MTI[3], that content is returned with the rest of the MTI bits set to zero.
- Any extra bytes that the node wishes to include. There can be zero or more of these. These shall be described in the node documentation.

Nodes shall process this message even if not all of the contents are provided.

For Error Codes see section #3.5.6 Error Codes, and section #3.5 Error Handling.

3.3.5 Terminate Due to Error

This is a reply indicating failure.

Name	Simple Protocol	Dest ID	Event ID	Common MTI	Data Content
Terminate Due to Error	N	Y	N	0x00A8	Error code, MTI, optional info

The contents are, in order:

- Two bytes of error code.
- Two bytes of MTI. If the frame transport only delivered part of the MTI[4], that content is returned with the rest of the MTI bits set to zero.
- Any extra bytes that the node wishes to include. There can be zero or more of these. These shall be described in the node documentation.

Nodes shall process this message even if not all of the contents are provided.

For error codes see section #3.5.5 Error Codes and section #3.4.3.Error Handling.

3.3.6 Protocol Support Inquiry

Requests that the addressed node reply with an indication of which protocols it supports.

---

[3]For example, in this case, CAN delivers 12+1 bits of the MTI via each frame (the special bit is known to be zero).

[4]For example, in this case, CAN delivers 12+2 bits of the MTI via each frame (the special bit is known to be zero and the stream/datagram bit can be inferred).

Name	Simple Protocol	Dest ID	Event ID	Common MTI	Data Content
Protocol Support Inquiry	-	Y	N	0x0828	(none)

3.3.7 Protocol Support Reply

Replying indicating the protocols that the node supports.

Name	Simple Protocol	Dest ID	Event ID	Common MTI	Data Content
Protocol Support Reply	N	Y	N	0x0668	One or more bytes identifying the supported protocols; see Table immediately below for coding.

A 1 in a bit position of the data indicates that the corresponding protocol is supported by the transmitting node. A 0 in the bit position indicates that the corresponding protocol is not supported by the transmitting node.

If a node transmits less than the full length of the currently-defined reply data, any missing bits shall be interpreted as zero.

Protocol	Protocol Flags
Simple Protocol subset	0x80 00 00
Datagram Protocol	0x40 00 00
Stream Protocol	0x20 00 00
Memory Configuration Protocol	0x10 00 00
Reservation Protocol	0x08 00 00
Event Exchange (Producer/Consumer) Protocol	0x04 00 00
Identification Protocol	0x02 00 00
Teaching/Learning Configuration Protocol	0x01 00 00
Remote Button Protocol	0x00 80 00
Abbreviated Default CDI Protocol	0x00 40 00
Display Protocol	0x00 20 00
Simple Node Information Protocol	0x00 10 00
Configuration Description Information (CDI)	0x00 08 00
Traction Control Protocol (Train Protocol)	0x00 04 00
Function Description Information (FDI)	0x00 02 00
DCC Command Station Protocol	0x00 01 00
Simple Train Node Information Protocol	0x00 00 80
Function Configuration	0x00 00 40
Reserved for future protocol bits. Shall be sent as 0 and ignored upon receipt. Trailing 0-bytes do not need to be sent.	All others

3.4 Interactions

All nodes shall be able to take part in all standard interactions, as defined below.

3.4.1 Node Initialization

Newly functional nodes, once their start-up is complete and they are fully operational, shall send an Initialization Complete message and enter the Initialized state. Nodes shall not emit any other OpenLCB message before the Initialization Complete message.

3.4.2 Node ID Detection

Upon receipt of an directed (addressed) Verify Node ID message addressed to it, a node shall reply with an unaddressed Verified Node ID message.

Upon receipt of a global (unaddressed) Verify Node ID message that does not contain an (optional) Node ID, a node shall reply with an unaddressed Verified Node ID message.

Upon receipt of a global (unaddressed) Verify Node ID message that contains an (optional) Node ID, a node will reply with an unaddressed Verified Node ID message, if and only if the receiving node's Node ID matches the one received.

If a node receives multiple Verify Node ID messages before it replies to one or more, it may, but is not required to, combine multiple responses into one.

3.4.3 Protocol Support Inquiry and Response

On receipt of a Protocol Support Inquiry message, a node will reply with a Protocol Support Response with bit values corresponding to the protocols that the node implements.

3.5 Error Handling

There are multiple error-handling scenarios defined.

3.5.1 Reject Addressed Optional Interaction

If a Node receives an addressed message with an MTI that is not part of the mandatory set, and it does not want to take part in that interaction, it shall send an Optional-Interaction Rejected (OIR) message addressed to the originating node, with the original MTI in the message content to indicate that the MTI is not recognized or not implemented by this node. The OIR message content may also contain a reason code and a data value, as defined by the protocol, as listed below in section 3.5.6.

3.5.2 Reject Unaddressed Optional-Interaction

If a Node receives an unaddressed message with an MTI that indicates the start of a non-mandatory / optional interaction, and the Node does nottake part in that optional interaction, it may silently drop the message without reply.

3.5.3 Reject Addressed Standard Interaction Due to Error

If a Node is taking part in an addressed interaction with another node, where either node may be able to send the next message, and some error condition prevents this Node from continuing the interaction, to terminate the interaction, this Node shall send a Terminate Due to Error message to the other Node. It shall reset its state so as to no longer be taking part in the addressed interaction. The message content shall contain the most recent MTI received in this interaction, a mandatory reason code and an optional data value. Although the use of these fields is to be defined, space must reserved for messages, so we specified the number of bytes.

In addition, upon receipt of the Terminate Due to Error message, the other Node also shall reset its state so as to no longer be taking part in the addressed interaction.

3.5.4 Duplicate Node ID Discovery

OpenLCB nodes shall indicate an error when they detect an incoming message with a Source Node ID equal to their own using whatever indication technology is available. See the General Event documentation for one method of indication that uses a well-known Global-Event.

3.5.5 Error codes

Numerous messages are defined to carry status information and error codes. An OpenLCB Error code is a 2-byte value of the following format:

Error Code Format	
Bits	**Description**
12-15	Error type flags. Zero or one bit shall be selected.
8-11	Reserved. Send as zero, ignored on receipt.
4-7	General Error Enumeration 0-15. Default as zero, and ignored on receipt.
0-3	Reserved for use of specific protocols for error reporting. Interpretation of these bits will be defined in the specific protocol documentation. A value of zero in this field means no further information is available. Default as 0, and ignored on receipt.

The following General Error Codes are defined, and may be used by all Protocols:

General Error Codes	
Value	**Description**
Permanent Error Enumeration. Re-trying the same interaction will result in the same error.	
0x1000	Permanent error, not further specified.
0x1010	Reserved.
0x1020	Source not permitted.
0x1030	Reserved.
0x1040	Not implemented, not further specified.
0x1080	Invalid arguments. Some of the values sent in the message fall outside of the expected range, or do not match the expectations of the receiving node.
0x1090-0x10F0	Reserved.
Temporary Error (resent OK). Re-trying the same interaction later is likely to succeed.	
0x2000	Temporary error, not further not specified.
0x2010	Timeout, the expected message or message-part did not arrive in time.
0x2020	Buffer unavailable or destination node busy.
0x2030	Reserved.
0x2040	Not expected, Out of order. An inconsistency was found in the message or frame sequence received, the arrived message is unexpected or does not match the state of the receiving node.
0x2050-0x2070	Reserved.
0x2080	Transfer error. The message or received message was ill-formed, failed checksum, or is otherwise uninterpretable. On CAN, this is handled by the hardware.
0x2090-0x20F0	Reserved.
Reserved	
0x4000	Reserved. Write as zero, ignored on read.
Accept Flag	
0x8000	Accept, no error. This value shall not be used in reject messages.

NB: Nodes may return 0x0000 as an error-code.

The following Error Codes may be specifically used by the Message Network Protocol:

Message Network Specific Error Codes	
Value	**Description**
Permanent Error Enumeration. Re-trying the same interaction will result in the same error.	
0x1041	Not implemented, subcommand is unknown.
0x1042	Not implemented, Datagram-type, Stream-type, or command is unknown.
0x1043	Not implemented, unknown MTI, or Transport protocol (datagrams/streams) is not supported.
Temporary Error (resent OK). Re-trying the same interaction later is likely to succeed.	
0x2011	Time-out, waiting for End-frame.
0x2041	Out of Order, Middle- or End-frame without a Start-frame.
0x2042	Out of Order, Start-frame before finishing previous message.

3.6 Routing

All messages may be, but are not required to be, presented to every node for processing.

Addressed messages shall to be routed to the addressed node. The node transmitting an addressed message addressed to itself shall take part in any interactions required by the message.

Unless otherwise specified in a protocol document, global messages shall be forwarded to all nodes. The node transmitting a global message shall take part in any interactions required by the message.

3.7 Delays and Timeouts

Nodes shall send messages required by OpenLCB protocols within 750 milliseconds, unless otherwise indicated in the documentation for the specific protocol interaction.

Nodes may, but are not required to, use a timeout mechanism to protect against messages lost due to malfunctions. Such a timeout shall not be shorter than 3 seconds.

4 Simple Node Protocol Subset

OpenLCB uses the Simple Node Protocol to distinguish a subset of global message types that are never needed by certain “simple” nodes. They can then be rapidly ignored by those nodes, and gateways can filter them from segments that contain only simple nodes, etc. It is not normative.

4.1 Protocol Description

Operationally, the simple node protocol is defined by the MTIs that carry a set Simple Node bit, plus all addressed messages. This section summarize received transmitted messages.

4.1.1 Messages Transmitted

Simple nodes may transmit any message, which shall be propagated normally.

4.1.2 Messages Received

Simple nodes shall receive any message specifically addressed to them, plus the following unaddressed global messages:

- Verify Node ID ;
- Verified Node ID;
- Protocol Support Inquiry;
- Identify Consumers;
- Identify Producers;
- Identify Events;
- Learn Event;
- P/C Event Report;

In the future, additional MTIs will be defined. If simple nodes need to received them, the MTI will indicate that via the Simple bit – see previous section.

4.1.3 Messages Not Received

Messages not listed in the section above do not need to be received by simple nodes.

4.1.4 Messages Directed at Gateways

Gateways need to know which nodes consider themselves to be Simple. These nodes may use the variants of the Initialization Complete and Verified Node ID messages that indicate Simple Set Sufficient (0x0101 and 0x0171, respectively) to indicate to Gateways that they are simple nodes.

5 Gateway Processing

No Standard content, see TN for discussion.

6 Expansion

No Standard content, see TN for discussion.

7 CAN Adaptations

7.1 Introduction

This section specifies how the Common Protocols are mapped to the CAN. While CAN is relatively inexpensive and robust, it is limited by its bandwidth and its total frame size, which is about 12-bytes, and by its speed (125 kbps). Each CAN frame includes a header (29-bits) and a data-part (8-bytes). In addition, the CAN-header has special properties which both enhance and limit its use. The Common Messages are adapted to CAN by:

- The Common-MTI is shortened to 12 bits, and carried in the header.
- Node IDs are shortened to 12 bit CAN-Aliases.
- Longer messages are fragmented into one or more CAN-frames. Special frames formats are used to implement Datagrams and Streams.

These mappings result in a set of modified formats as documented in this section. This section's numbering parallels that of the Common sections.

Common Message	***CAN Mapping***	***Comment***
Most messages	*Usually single frame*	*Multiple frames are used for longer messages.*
Datagram	*First-, Middle-,..., Last- frames*	*Small Datagrams may be carried by a single Only-frames.*
Stream	*Stream-Data-frames*	*A common Stream-message will be mapped to one or more Stream-Data-frames.*

7.2 Intended Use

CAN is intended a local transport for smaller layouts, or for local segments for larger layouts. Accommodation and specialization is necessary to implement the protocols onto CAN because CAN frames are limited to about 12 bytes of information. The main mappings include a shortened CAN-MTI (12-bits) and the use of 12 bit aliases for Node Ids. These have implications for Gateways which are specified below.

7.2.1 References and Context

This Standard is in the context of the following OpenLCB CAN Standards:

- The OpenLCB CAN Frame Transfer Standard, which specifies transfer of OpenLCB messages over CAN segments. “CAN” refers to the electrical and protocol specifications as defined in ISO 11898-1:2003 and ISO 11898-2:2003 and their successors.

7.3 Messages

7.3.1 Message Format

The CAN mapping uses several formats. The general form of the CAN header is:

29-bit CAN Header								
Field	**CAN prefix**	**Frame Type**	**Variable Field**				**Source ID**	
Size & location	2 bits 0x1800,0000	3 bits 0x0700,0000	12 bits 0x00FF,F000				12 bits 0x0000,0FFF	
Value(s)	3	1,2,3,4,5,or 7	CAN-MTI or Destination Node Alias				Source Node Alias	
Up to 8 Byte Data-Part								
Byte#	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Values	variable	variable	(Data)	(Data)	(Data)	(Data)	(Data)	(Data)

Table 1: CAN Frame Format

CAN frames also contain other bit fields not detailed here, including a DLC or length field.

7.3.1.1 CAN Prefix Field

250 The CAN Prefix Field contains control bits specific to CAN.

7.3.1.2 CAN MTI Mapping

The Common MTI is mapped to one of eight frame types:

Frame Type	Meaning
0	(Reserved)
1	Global & Addressed MTI
2	Datagram complete in frame
3	Datagram first frame
4	Datagram middle frame
5	Datagram final frame
6	(Reserved)
7	Stream Data

Table 2: CAN Frame Type Values

7.3.1.3 Global and Addressed Messages, CAN Frame Type 1

255 Global and Addressed messages are those Common MTIs with the “stream and datagram” bits unset, and are mapped to and from CAN MTI type 1.

The CAN MTI is carried in the CAN header. It consists of the lowest-order 12 bits of the Common MTI, and, by implication has the “stream or datagram” and “special” bits as zeros.

29-bit CAN Header									
Field	**CAN prefix**	**Frame Type**	**Static Priority**	**Type within Priority**	**Simple Node flag**	**Address Present**	**Event ID present**	**Modifier Bits**	**Source ID**
Size & location	2 bits 0x1800, 0000	3 bits 0x0700, 0000	2 bits 0x00C0, 0000	5 bits 0x003E, 0000	1 bit 0x0001, 0000	1 bit 0x0000, 8000	1 bit 0x0000, 4000	2 bits 0x0000, 3000	12 bits 0x0000, 0FFF
Value(s)	3	**1**	**CAN-MTI**						Source Node Alias

Up to 8-Byte Data-Part								
Byte#	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Value	Optional Flags/Destn Alias or (Data)		(Data)	(Data)	(Data)	(Data)	(Data)	(Data)

Table 3: CAN Frame Type 1 Format

If the "addressed" bit is set to 1, then the destination address is placed in the 1st two bytes of the data part of the CAN frame: the top nibble of the 1st byte contains flags (see below); the lower nibble of the 1st byte and the entire 2nd byte contain the 12-bit destination alias.
The format of this in binary is: **0brrff dddd, dddd dddd** (or in hex: **0xfddd**)
where:

- The two rr bits are reserved, and read and send as zeros.
- The two ff bits can be used for packing and unpacking large messages to a sequence of CAN frames, see below. The coding is:
  - 00 Only frame
  - 01 First frame of more than one
  - 10 Last frame of more than one
  - 11 Middle frame of more than 2.
- You can think of these as active-zero start and end bits, respectively.
- Messages are limited to 254 bytes of data.

CAN frames marked as First or Middle frame shall carry eight total data bytes. CAN frames marked as Last or Only frame shall have from two through eight total data bytes.

7.3.1.4 Datagram and Stream Messages, CAN Frame Types 2-5 and 7

Frame Types 2-5 and 7 are used specifically for Datagram and Stream messages.

7.3.2 States

No special provisions for CAN transport layer.

7.3.3 Definition of Specific Messages

CAN messages definitions parallel the common messages in section #3.3.Definition of Specific Messages.

7.3.3.1 *Initialization Complete*

Name	Description	CAN-MTI	CAN Header	Data Content
Initialization Complete	Full protocol	0x100	[0x1910,0sss]	Full Source Node ID
	Simple-set sufficient	0x101	[0x1910,1sss]	

7.3.3.2 *Verify Node ID*

Name	Description	CAN-MTI	CAN Header	Data Content
Verify Node ID	Addressed	0x498	[0x1949,8sss]	fddd[5], Optional Full Node ID
	Global	0x490	[0x1949,0sss]	Optional Full Node ID

7.3.3.3 *Verified Node ID.*

Name	Description	CAN-MTI	CAN Header	Data Content
Verified Node ID	Full protocol	0x170	[0x1917,0sss]	Full Source Node ID
	Simple-set sufficient	0x171	[0x1917,1sss]	

7.3.3.4 *Optional Interaction Rejected*

Name	CAN-MTI	CAN Header	Data Content
Optional Interaction Rejected	0x068	[0x1906,8sss]	fddd, error, optional info

7.3.3.5 *Terminate Due to Error*

Name	CAN-MTI	CAN Header	Data Content
Terminate Due to Error	0x0A8	[0x190A,8sss]	fddd, error, optional info

7.3.3.6 *Protocol Support Inquiry*

Name	CAN-MTI	CAN Header	Data Content
Protocol Support Inquiry	0x828	[0x1982.8sss]	fddd

[5] "fddd" refers to a flag-nibble "f", containing multipart flags, and the 12-bit Destination Node Alias "ddd". See TN.

7.3.3.7 Protocol Support Reply

Name	CAN-MTI	CAN Header	Data Content
Protocol Support Reply	0x668	[0x1966,8sss]	fddd, Protocol flags

This shall be a multi-part message when there are more than 48 protocol-bits.

7.3.4 Extensibility

Since the earlier nodes may reply as soon as they have processed only the data of which they are aware, their replies may be earlier than the last message fragment is received. Sending-nodes shall be able to receive and process these replies as they are received.

7.3.5 Interactions

No special provisions for CAN transport layer.

7.3.6 Error Handling

No special provisions for CAN transport layer.

7.3.7 Routing

CAN implementations shall send the frames of a message together to reduce buffering. Higher-priority messages may be sent in the middle of a lower-priority message.

305

Table of Contents