OpenLCB Standard	
Message Network	
Feb 7, 2015	Draft

1 Introduction

This Specification 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 or TCP/IP sockets. 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 specification is in the context of the following OpenLCB Specifications:

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

This specification is in the context of the following OpenLCB-CAN Specifications:

- The OpenLCB-CAN Frame Transfer Specification, 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 specification is in the context of the following OpenLCB-TCP/IP Specifications:

- The OpenLCB-TCP/IP Segment Transfer Specification, 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 specification 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 NodeID 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	Simple Protocol	Dest ID	Event ID	Description	MTI	Data Content
Initialization Complete	N	N	N	Full Protocol Required	0x0100	Source Node ID
	Y	N	N	Simple Subset Sufficient	0x0110	

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

3.3.3 Verified Node ID

Reply to the Verify Node ID message.

Name	Simple Protocol	Dest ID	Event ID	Description	MTI	Data Content
Verified Node ID Number	N	N	N	Full Protocol Required	0x0160	Source Node ID
	Y	N	N	Simple Subset Sufficient	0x0170	

[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	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	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 #3.3.8.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	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	MTI	Data Content
Protocol Support Reply	N	Y	N	0x0668	One or more bytes identifying the supported protocols; see Table immediately below for coding.

115 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
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) NodeID, 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) NodeID, a node will reply with an unaddressed Verified Node ID message, if and only if the receiving node's NodeID 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 an non-mandatory / optional interaction, and the Node does not want to take part in that optional interaction, it silently drops 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 sends a Terminate Due to Error message to the other Node. It then resets its state so as to no longer be taking part in the addressed interaction. The message content contains 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 resets 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 Codes		
Bits	**Value**	**Description**
Error Enumeration. Exactly one of the following bits should be set:		
12-15	0x8000	Accept, no error. This value shall not be used in Reject messages.
	0x4000	Reserved. Send as 0, treat it as unspecified Permanent error.
	0x2000	Temporary error (resend OK). Re-trying the same interaction later is likely to succeed.
	0x1000	Permanent error. Re-trying the same interaction will result in the same error.
Specific Error Enumeration. Exactly one of the following bits should be set:		
8-11	Specific error enumeration. Possible values for this field are documented in particular protocol standards. Values may be re-used for different general-error-flags settings. A value of zero in this field means no further information is available.	
General error flags:		
4-7	**For Permanent error, the following flags are defined:**	
	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.
	0x1040	The transport protocol (datagram/stream) is not supported.
	0x1020	Source not permitted.
	0x1010	Unimplemented. The functionality that the message tried to invoke is not implemented in the destination node.
	For Temporary error (resend OK), the following flags are defined:	
	0x2080	Transport error.
	0x2040	Unexpected, 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.
	0x2020	Buffer unavailable.
Additional flags:		
1-3	Reserved. Send as 0, ignore on receipt.	
0	0x0001	Information logged. See the Logging protocol to retrieve the logged information. This bit may be added to any error value.

170

3.6 Routing

All messages may be, but don't have 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” concept 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, and describes the reasoning behind those choices in the current MTI definitions.

4.1.1 Messages Transmitted

Simple nodes may transmit any message, which shall be propagate like messages transmitted from any other type of node.

4.1.2 Messages Received

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

- Verify Node ID – They need to receive this so that they can reply to it.
- Verified Node ID – They need to receive this because it's the reply to their own request, which might be used to e.g. locate a node for delayed sending of status
- Protocol Support Inquiry – They need to receive this so that they can reply to it.
- Identify Consumers, Identify Producers, Identify Events – because others will ask this of them
- Learn Event – so they can be programmed
- P/C Event Report – what they do for a living

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.

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 specification is in the context of the following OpenLCB-CAN Specifications:

- The OpenLCB-CAN Frame Transfer Specification, 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

The CAN Prefix Field contains control bits specific to CAN.

7.3.1.1.1

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

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 72 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.

~~**Datagram and Stream Data messages are transmitted on CAN as one or more frames, using the frame formats as described in the specifications for those protocols, with values of 2 through 5 and 7 in the CAN Frame Type field, respectively. The 12-bit destination alias is placed in the header, along with the 12-bit source alias.**~~

~~Parts of a fragmented message may be lost during transmission. If more than 500 milliseconds elapses after the receipt of a first or middle datagram frame without receipt of a middle or end frame, the node may, but is not required to, initiate error recovery by forwarding the message, marked as received in error, for further internal processing. The time used to process higher priority messages may be subtracted from the elapsed time.~~

~~29-bit CAN Header~~								
~~Field~~	~~CAN prefix~~	~~Frame Type~~	~~Destination ID/ddd[5]~~				~~Source ID/sss~~	
~~Size & location~~	~~2 bits 0x1800,0000~~	~~3 bits 0x0700,0000~~	~~12 bits 0x00FF,F000~~				~~12 bits 0x0000,0FFF~~	
~~Value(s)~~	~~3~~	~~2,3,4,5,or 7~~	~~Destination Node Alias~~				~~Source Node Alias~~	
~~Up to 8-Byte Data Part~~								
~~Byte#~~	~~0~~	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~
~~Value~~	~~(data)~~	~~(data)~~	~~(data)~~	~~(data)~~	~~(data)~~	~~(data)~~	~~(data)~~	~~(data)~~

~~Table 4: CAN Datagram and Stream Format~~

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 (hex)	Data Content
Initialization Complete	Full	0x100	[0x19100sss]	Full Source Node ID
	Simple	0x110	[0x19110sss]	

7.3.3.2 Verify Node ID

Name	Description	CAN-MTI	CAN Header (hex)	Data Content
Verify Node ID	Addressed	0x498	[0x19498sss]	fddd[6], Optional Full Node ID
	Global	0x490	[0x19490sss]	Optional Full Node ID

[5] "ddd" and "sss" refer to the Destination Node Alias and the Source Node Alias, respectively.

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

7.3.3.3 Verified Node ID.

Name	Description	CAN-MTI	CAN Header (hex)	Data Content
Verified Node ID	Full	0x160	[0x19160sss]	Full Source Node ID
	Simple	0x170	[0x19170sss]	

7.3.3.4 Optional Interaction Rejected

Name	CAN-MTI	CAN Header (hex)	Data Content
Optional Interaction Rejected	0x068	[0x19068sss]	fddd, error, optional info

7.3.3.5 Terminate Due to Error

Name	CAN-MTI	CAN Header (hex)	Data Content
Terminate Due to Error	0x0A8	[0x190A8sss]	fddd, error, optional info

7.3.3.6 Protocol Support Inquiry

Name	CAN-MTI	CAN Header (hex)	Data Content
Protocol Support Inquiry	0x828	[0x19828sss]	fddd

7.3.3.7 Protocol Support Reply

Name	CAN-MTI	CAN Header (hex)	Data Content
Protocol Support Reply	0x668	[0x19668sss]	fddd, Protocol flags

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

7.3.3.8 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.

7.3.4 Interactions

No special provisions for CAN transport layer.

7.3.5 Error Handling

No special provisions for CAN transport layer.

7.3.6 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.

310

Appendix 1

The following table is for reference only. In the case of errors, the values in the text of the body of the document shall take precedence.

2015-01-05 06:49

Full MTI Format

		0	1	2	3	4 5	6 7 8 9 10	11	12	13	14 15		
		15	14	13	12	11 10	9 8 7 6 5	4	3	2	1 0		
Category	**Description**	**Reserved**	**Reserved**	**Special**	**Stream or Datagram**	**Priority**	**Type within priority (decimal)**	**Simple Protocol**	**Address present**	**Event# present**	**Modifier within Priority / Type**	**Full MTI (Hex)**	**Data Content**
	Number of bits in field:	1	1	1	1	2	5	1	1	1	2	16	n
Basic	Initialization Complete	0	0	0	0	0	8	0	0	0		0100	Source Node ID
	Initialization Complete (simple)	0	0	0	0	0	8	1	0	0		0110	
	Verify Node ID Number Addressed	0	0	0	0	1	4	1	1	0		0498	Optional Node ID
	Verify Node ID Number Global	0	0	0	0	1	4	1	0	0		0490	
	Verified Node ID Number	0	0	0	0	0	11	0	0	0		0160	Source Node ID
	Verified Node ID Number (Simple)	0	0	0	0	0	11	1	0	0		0170	
	Optional Interaction Rejected	0	0	0	0	0	3	0	1	0		0068	Error codes, MTI, optional info
	Terminate Due to Error	0	0	0	0	0	5	0	1	0		00A8	Error codes, MTI, optional info
Protocol Support	Protocol Support Inquiry	0	0	0	0	2	1	0	1	0		0828	(none)
	Protocol Support Reply	0	0	0	0	1	19	0	1	0		0668	Supported protocols

CAN

	28	27	26 25 24	23 22 21 20 19 18 17 16 15	11 10 9 8 7 6 5 4 3 2 1 0		
	0	1	2 3 4	5 6 7 8 9 10 11 12 13	17 18 19 20 21 22 23 24 25 26 27 28		
Description	**Priority**	**Frame-type**	**CAN Frame Type**	**CAN-MTI or Destination-alias** ("Variable Part") **(hex)**	**Source-alias**	**CAN Header (hex)**	**Data-Part { } = optional**
Number of bits in field:	1	1	3	12	12	29	64
Initialization Complete	1	1	1	100	sss	[19100sss]	Full Source Node ID
Initialization Complete (simple)	1	1	1	110	sss	[19110sss]	Full Source Node ID
Verify Node ID Number Addressed	1	1	1	498	sss	[19498sss]	fddd, {Full Node ID}
Verify Node ID Number Global	1	1	1	490	sss	[19490sss]	{Full Node ID}
Verified Node ID Number	1	1	1	160	sss	[19160sss]	Full Source Node ID
Verified Node ID Number (Simple)	1	1	1	170	sss	[19170sss]	Full Source Node ID
Optional Interaction Rejected	1	1	1	068	sss	[19068sss]	fddd, error, optional info
Terminate Due to Error	1	1	1	0A8	sss	[190A8sss]	fddd, error, optional info
Protocol Support Inquiry	1	1	1	828	sss	[19828sss]	fddd
Protocol Support Reply	1	1	1	668	sss	[19668sss]	fddd, Protocol flags

Illustration 1: Summary of General Message protocol messages

Table of Contents

315

