LOCAL INTERCONNECT NETWORK - LIN -

AYOUB BOUMEZZOU

Electrical Engineering and Industrial Systems Control Engineer
Embedded Systems Development | SW/HW | R&D | MBD | ECUs | AI Researcher

boumezzouayoub@gmail.com
linkedin.com/in/Ayoub Boumezzou
github.com/Ay-Bmz

LOCAL INTERCONNECT NETWORK (LIN)

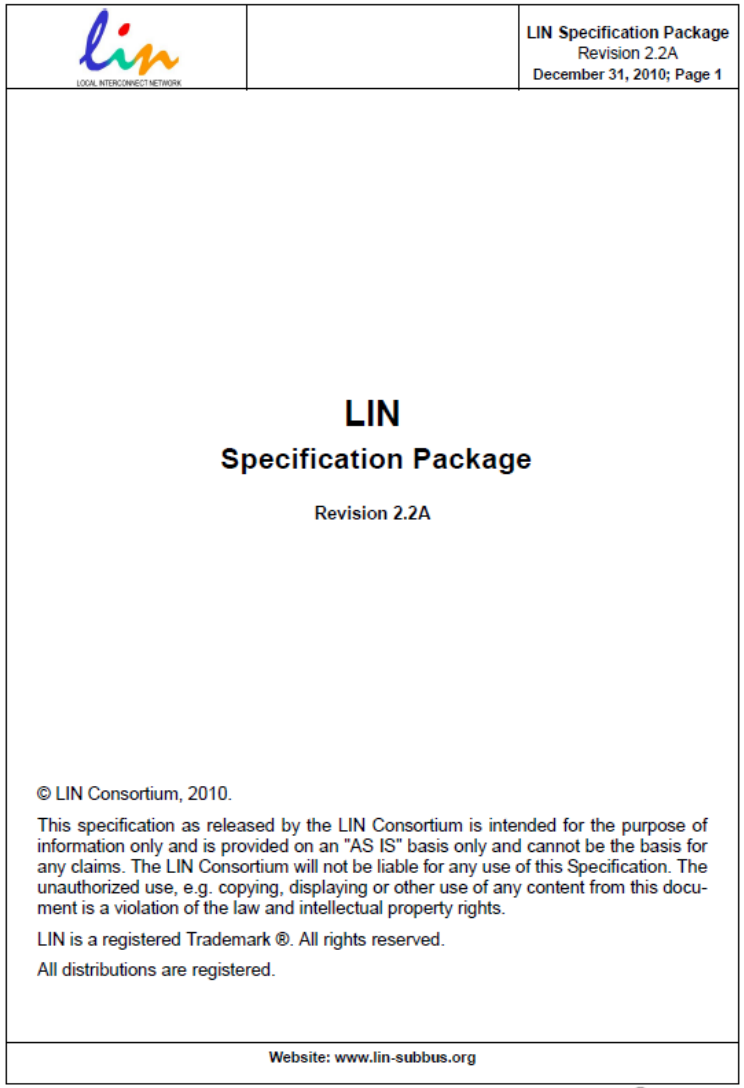
- **Cheap serial single wire system used for communication between components in vehicles.**
- **Subsystem and as complement to the existing CAN network.**
- **Network for sensors and actuators with controllers as master-slave architecture.**
- **Typically used in window lift, mirrors, wiper and rain sensors.**

LIN CONSORTIUM

- LIN revision 1.0 was released in July 1999 influenced by the VLITE bus used by some automotive companies. LIN 1.2 was released in November 2000.
- Founded in **late 1990s** by BMW, Volkswagen Group, Audi Group, Volvo Cars, Mercedes-Benz, Mentor Graphics (Formerly Volcano Automotive Group) and Freescale (Formerly Motorola, now NXP).
- First LIN specification was published as **version 1.3 in November 2002**, **version 2.0 was released in 2003 and version 2.1 in 2006**.
- In 2010 LIN consortium was closed before **LIN version 2.2A** was released.
- In parallel, the **SAE organization** developed the J2602 series as the US variant; it was published in 2004 and updated in 2012.
- In the meantime, ISO has standardized the LIN communication system and related conformance test plans in the **ISO 17987** series.
- Currently **CAN in Automation (CiA)** assigns the LIN Supplier ID and maintaining LIN website on behalf of ISO.

LIN SPECIFICATION PACKAGE REVISION 2.2A

- Specification of the transmission protocol.
- Transport Layer Specification.
- Node configuration and Identification Specification.
- Diagnostic specification.
- Physical Layer Specification.
- Application Program Interface Specification.
- Node Capability Language Specification.
- Configuration Language Specification.


LIN Specification Package
Revision 2.2A
December 31, 2010; Page 1

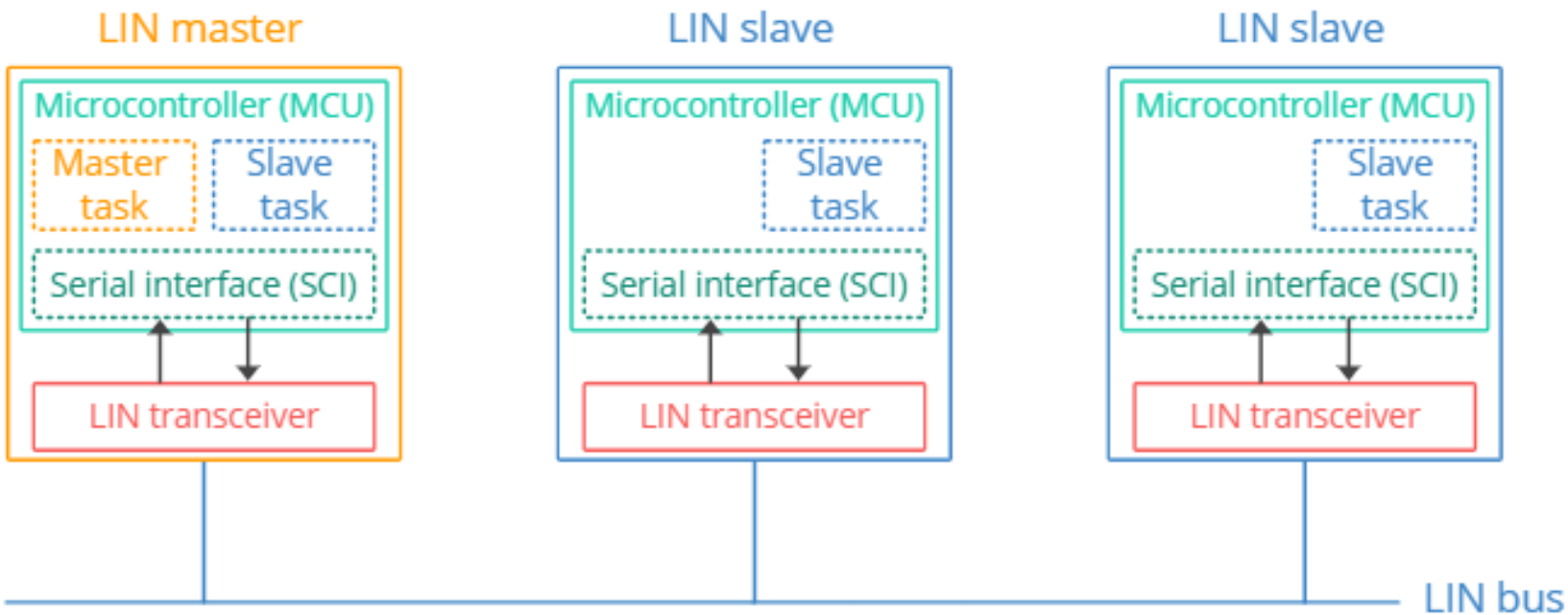
LIN

Specification Package

Revision 2.2A

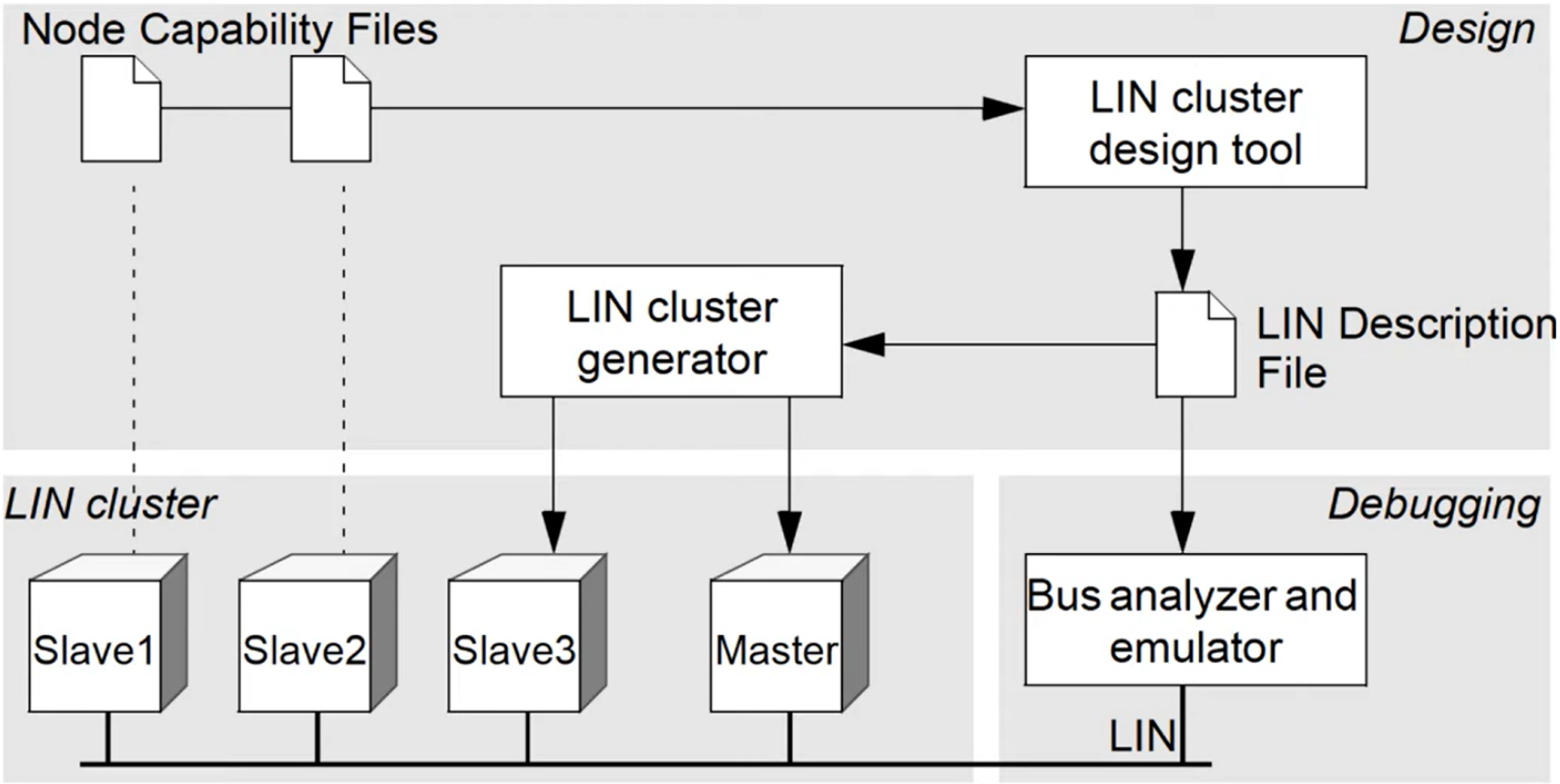
Website: www.lin-subbus.org

CONCEPT OF OPERATION



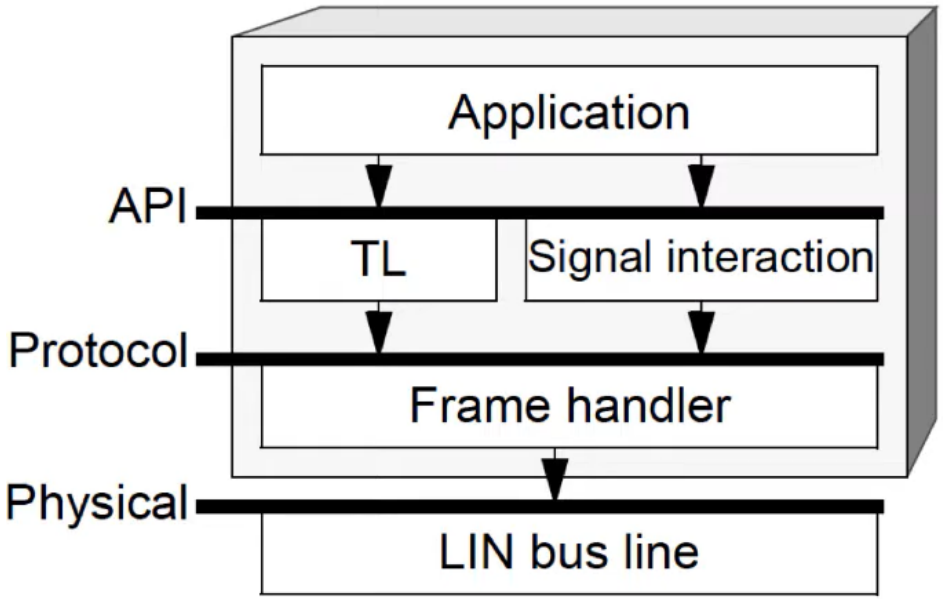
- The master task decides when and which frame shall be transferred on the bus.
- The slave tasks provide the data transported by each frame.

PLUG AND PLAY WORK FLOW CONCEPT



***Cluster** refers to a group of nodes (master/slaves) connected to the same LIN network.*

NODE CONCEPT



- A node in a cluster interfaces to the physical bus wire using a frame transceiver.
- Frames are not accessed directly by the application.
- Transport layer and signal based interaction layer exists between the application and the frame handler.

SIGNAL VS FRAME

- All information is transmitted packed as frames; a frame consists of the **header** and a **response**.
- A signal is a value (**scalar signal is between 1 and 16 bits long**) or byte array (**between one and eight bytes**) transported in the **cluster** with the LSB first and the MSB last using a **signal carrying frame**.
- **Signal carrying frame**: is a frame that carries signals shall have a **frame identifier** in the range 0 (zero) to 59 (0x3B). **Unconditional frames**, **sporadic frames**, and **event triggered frames** are signal carrying frames, unlike **diagnostic frames**.

**The diagnostic frames use frame identifiers 60 (0x3C) and 61 (0x3D), while frame identifiers 62 (0x3E) and 63 (0x3F) are reserved for a future LIN extended format.*

SIGNAL RECEPTION

- **Master node** – at next time base tick after the maximum frame length. The master node updates its received signals periodically at the time base start (i.e., at task level).
- **Slave node** – when the checksum for the received frame is validated. The slave node updates its received signals directly after the frame is finished (i.e., at interrupt level).

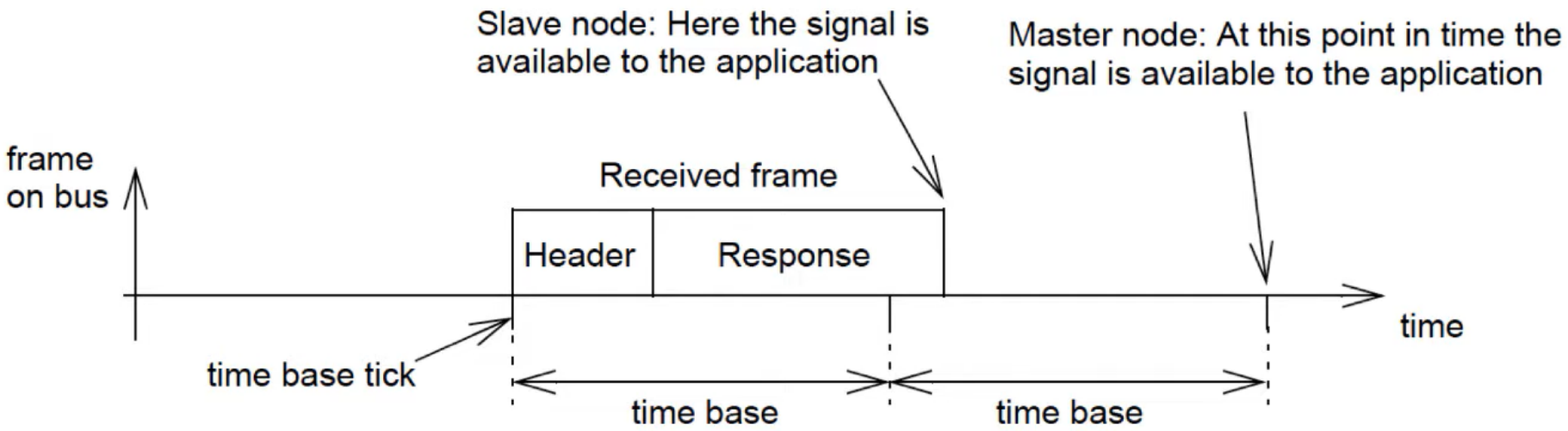


Figure - Timing of signal reception

**Master nodes use a polling mechanism, while slave nodes operate via interrupts.*

SIGNAL TRANSMISSION

- **Master node** – before the frame transmission is initiated.
- **Slave node** – when the ID for the frame is received.

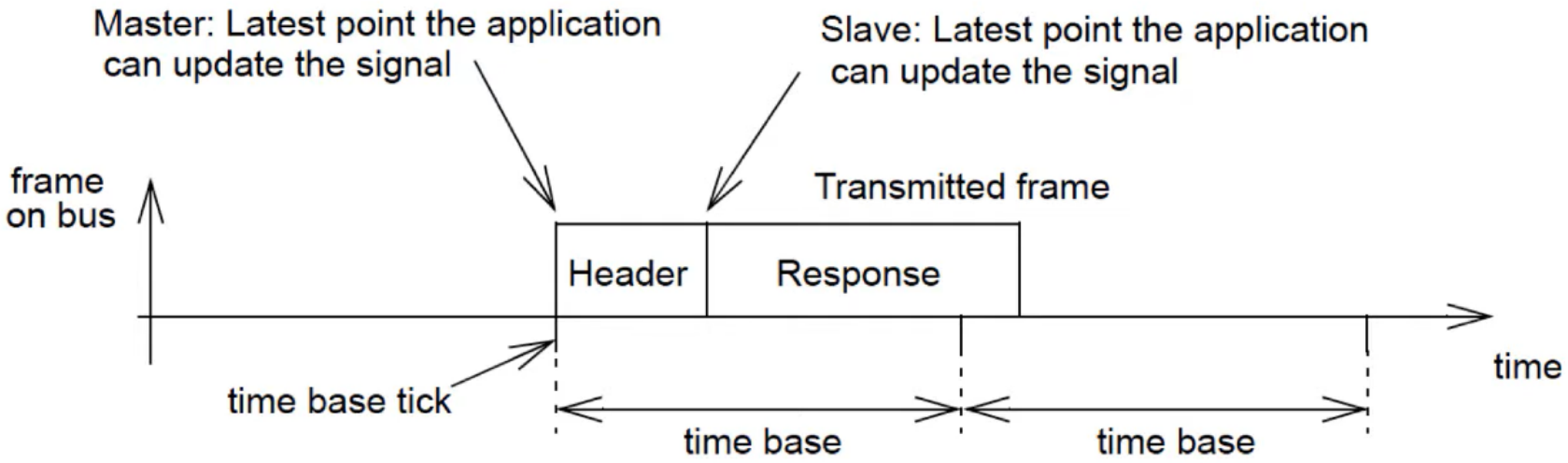


Figure - Timing of signal transmission

FRAME STRUCTURE

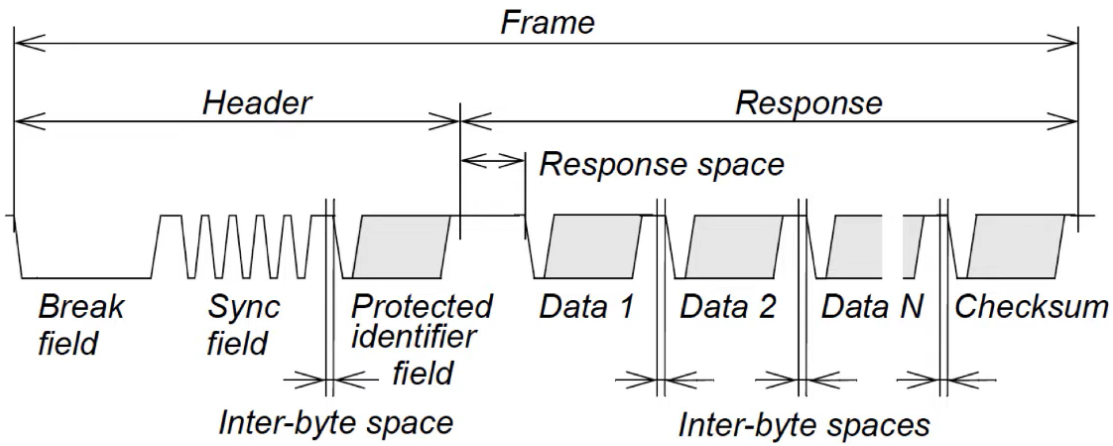


Figure - The structure of frame

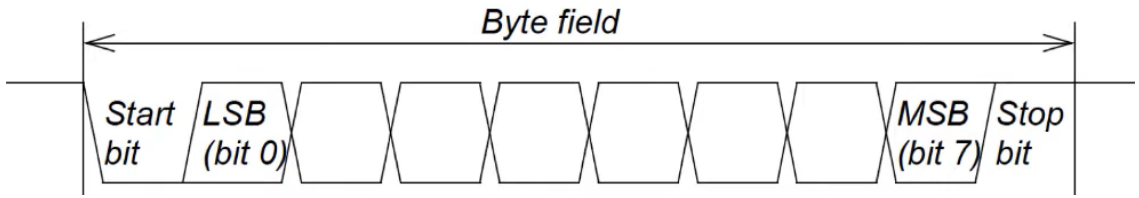


Figure - Structure of byte field

- Each byte field, except the break field, is transmitted as the byte field.
- The start bit is encoded as a bit with value zero **(dominant)** and the stop bit is encoded as a bit with value one **(recessive)**.

BREAK FIELD

- Used to signal the beginning of a new frame and always generated by the master task.
- Shall be at least **13 nominal bit times of dominant value**, followed by a **break delimiter**. The break delimiter shall be at least one nominal bit time long.

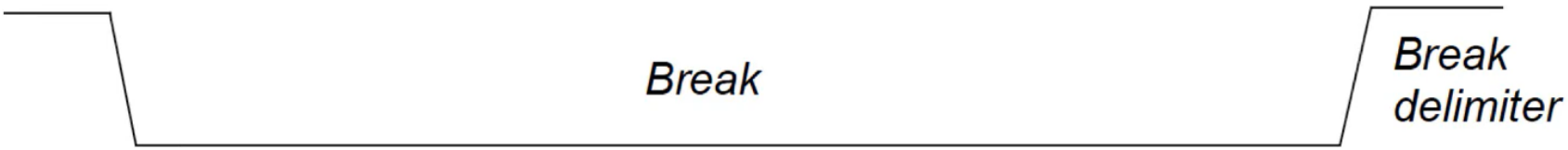


Figure - The break field

SYNC BYTE FIELD

- Sync is a byte field with the data value 0x55, helps all nodes synchronize by allowing LIN slave devices to determine the communication baud rate.

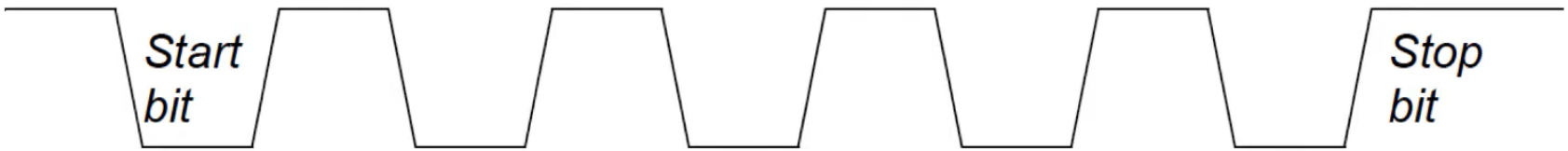


Figure - The sync byte field

PROTECTED IDENTIFIER

- Consists of two sub-fields; the **frame identifier** and the **parity**.
- **Frame identifier**
  - 0 to 59 (0x3B) are used for signal carrying frames.
  - 60 (0x3C) and 61 (0x3D) are used to carry diagnostic and configuration data.
  - 62 (0x3E) and 63 (0x3F) are reserved for future protocol enhancements.
- **Parity**
  - $\mathrm{P0} = \mathrm{ID0} \oplus \mathrm{ID1} \oplus \mathrm{ID2} \oplus \mathrm{ID4}$
  - $\mathrm{P1} =! (\mathrm{ID1} \oplus \mathrm{ID3} \oplus \mathrm{ID4} \oplus \mathrm{ID5})$

Figure - The PID field

PID EXAMPLE

ID[0..5] Dec	ID[0..5] Hex	P0 = ID0⊕ID1⊕ID2⊕ID4	P1 = ¬ ID1⊕ID3⊕ID4⊕ID5	PID-Field P1	P0	5	4	3	2	1	0	PID-Field Dec	PID-Field Hex
38	0x26	0	1	1	0	1	0	0	1	1	0	166	0xA6
39	0x27	1	1	1	1	1	0	0	1	1	1	231	0xE7

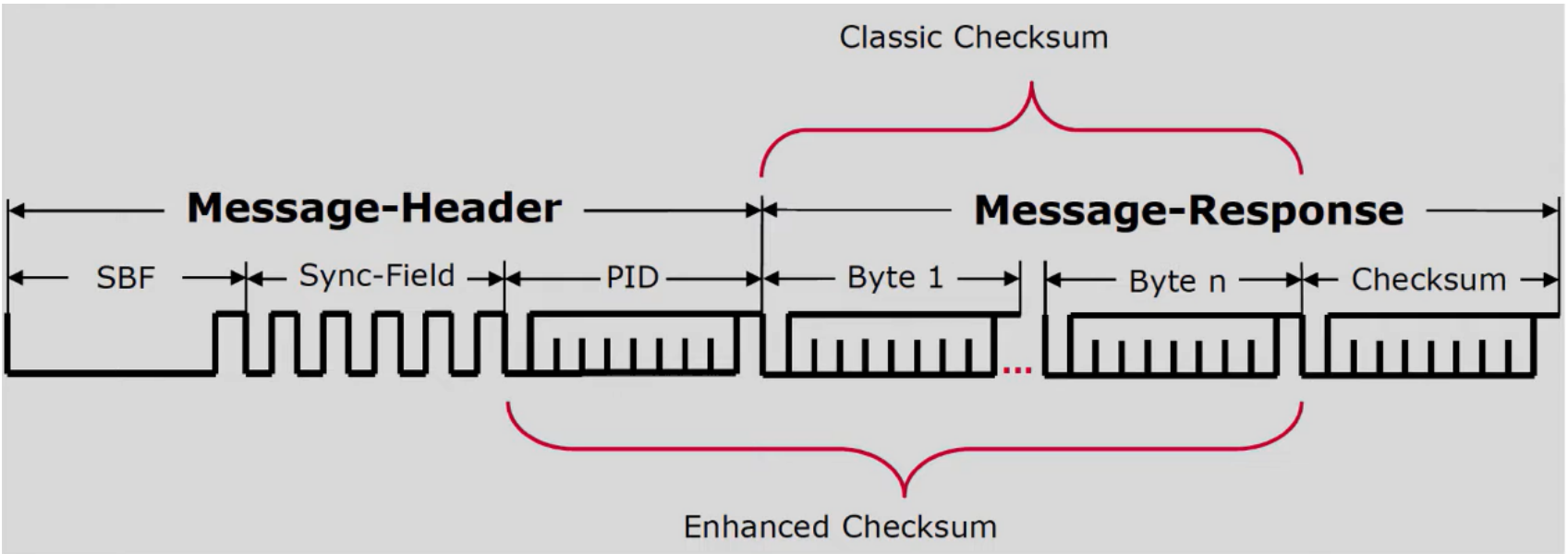
DATA

- A frame carries between one and eight bytes of data.
- The number of data bytes contained in a frame with a specific frame identifier shall be agreed upon by the publisher and all subscribers.

Figure - Numbering of the data bytes in a frame with eight data bytes

CHECKSUM

- Checksum is the last field of a frame.
- Contains the inverted eight-bit sum with carry over all data bytes (**Classic Checksum** – LIN 1.1, LIN 1.2, and LIN 1.3) or all data bytes and the protected identifier (**Enhanced Checksum** – LIN 2.0, LIN 2.1, and LIN 2.2).
- Diagnostic Frames 60 (0x3C) to 61 (0x3D) shall always use classic checksum.



CHECKSUM CALCULATION EXAMPLE

- If the frame has four data bytes or the protected identifier and three data bytes, the calculation is the same.
- Data = 0x4A, 0x55, 0x93, 0xE5

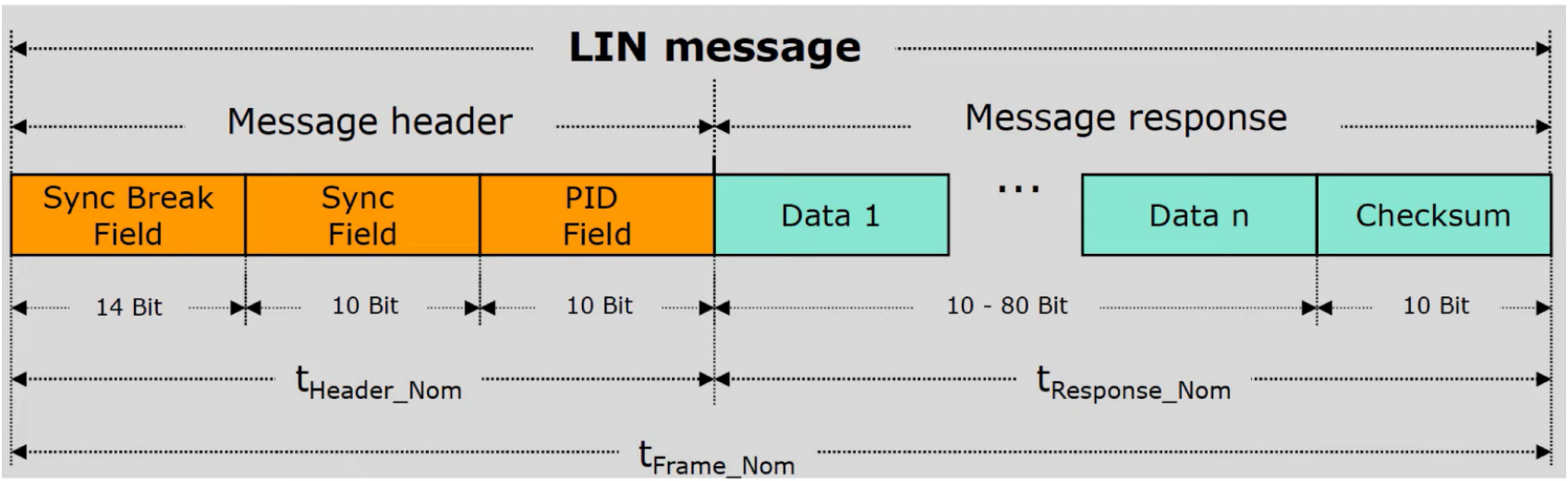
Action	hex	Carry	D7	D6	D5	D4	D3	D2	D1	D0
0x4A	0x4A		0	1	0	0	1	0	1	0
+0x55 =	0x9F	0	1	0	0	1	1	1	1	1
(Add Carry)	0x9F		1	0	0	1	1	1	1	1
+0x93 =	0x132	1	0	0	1	1	0	0	1	0
Add Carry	0x33		0	0	1	1	0	0	1	1
+0xE5 =	0x118	1	0	0	0	1	1	0	0	0
Add Carry	0x19		0	0	0	1	1	0	0	1
Invert	0xE6		1	1	1	0	0	1	1	0
0x19+0xE6 =	0xFF		1	1	1	1	1	1	1	1

FRAME LENGTH

$$T_{Frame\_Nominal} = T_{Header\_Nominal} + T_{Response\_Nominal} = (n * 10 + 44) * T_{Bit}$$

- Maximum space between the bytes is additional 40% duration compared to the nominal transmission time.

$$T_{Frame\_Maximum} = 1.4 * T_{Frame\_Nominal}$$



*T_{bit} time is defined by the baud rate.

FRAME TYPES – UNCONDITIONAL FRAME

- Carry signals and their frame identifiers are in the range 0 (zero) to 59 (0x3B).

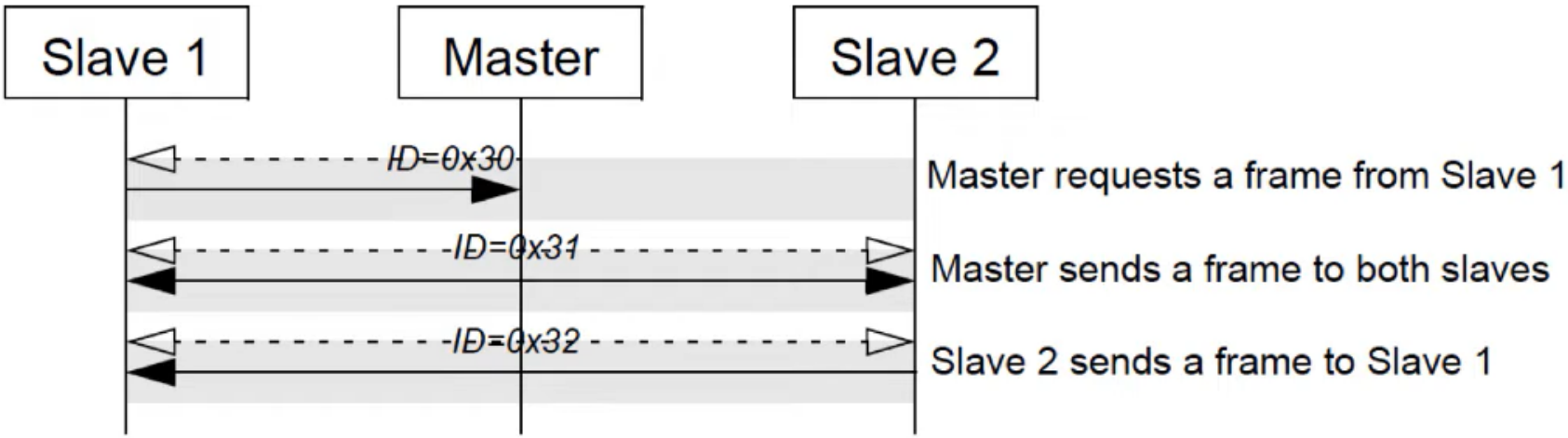


Figure - Three unconditional frame transfers

EVENT TRIGGERED FRAME

- To increase the responsiveness of the LIN cluster without assigning too much of the bus bandwidth to the polling of multiple slave nodes with seldom occurring events.
- Event triggered frames carry the response of one or more unconditional frames.
- If more than one slave node responds to the header in the same frame slot, a collision will occur.
- The master node has to resolve the collision in a collision resolving schedule table.

EVENT TRIGGERED FRAME – EXAMPLE

- Event-triggered frame (ID=0x10) associated with two unconditional frames from slave 1 (ID=0x11) and slave 2 (ID=0x12).

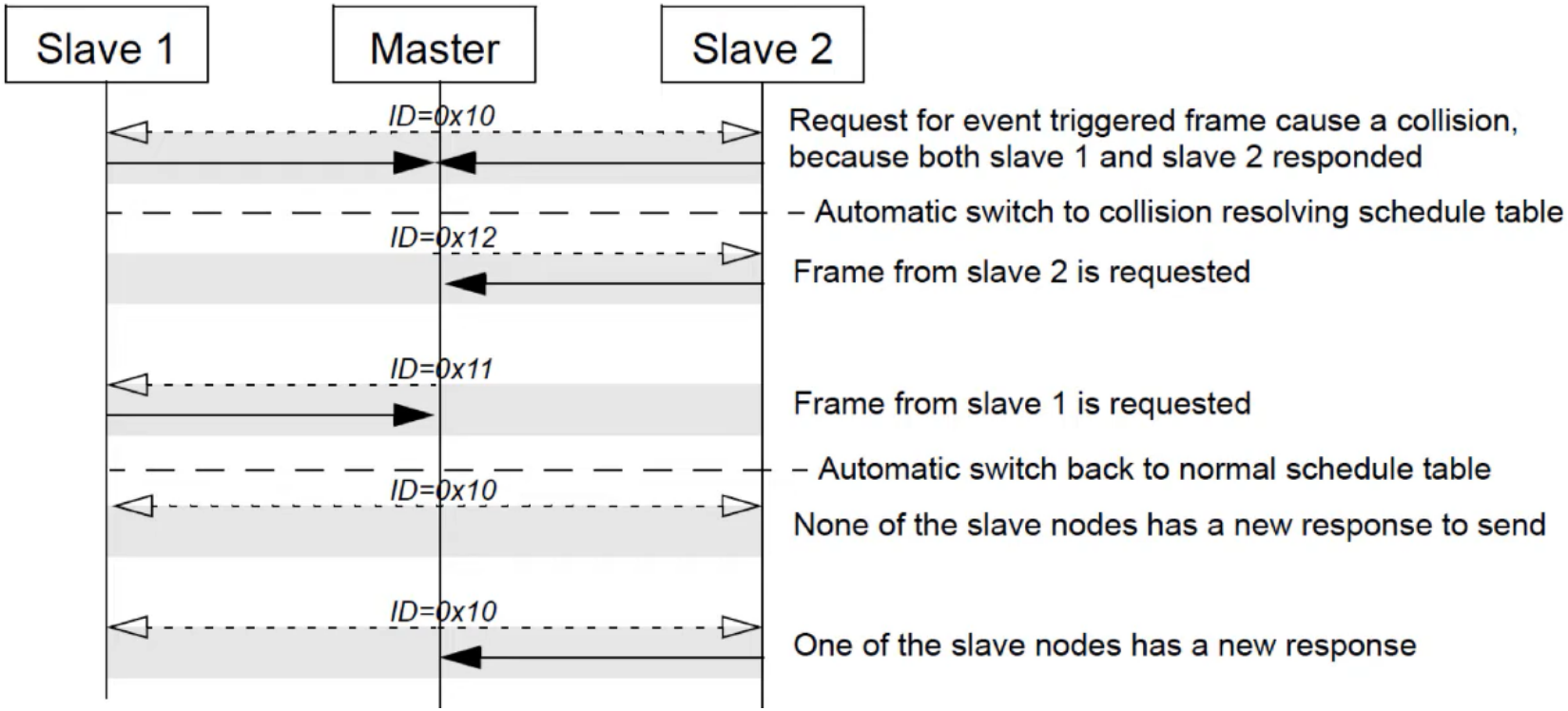


Figure - Event-triggered frame example

SPORADIC FRAME

- A group of unconditional frames that share the same frame slot.
- When the sporadic frame is due for transmission, the unconditional frames are checked to see if they have any updated signals.
- If no signals are updated, no frame will be transmitted, and the frame slot will be empty.
- If one signal has been updated, the corresponding frame will be transmitted.

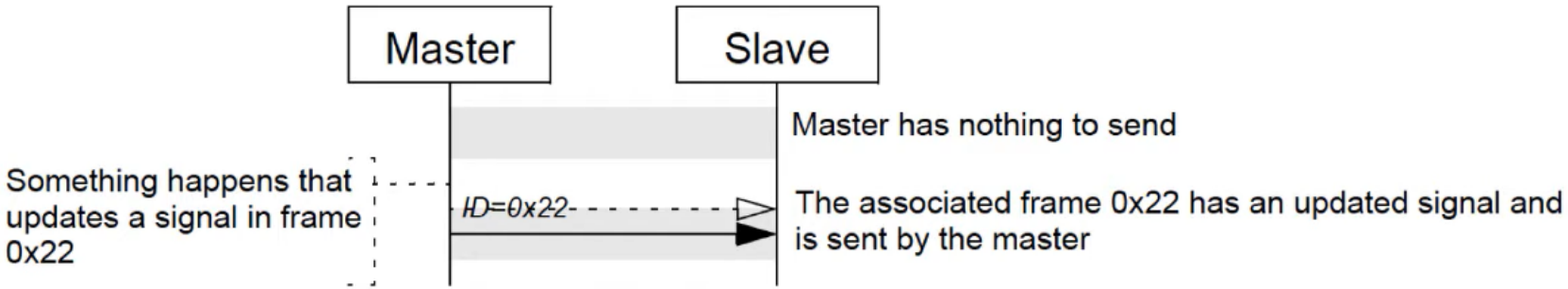
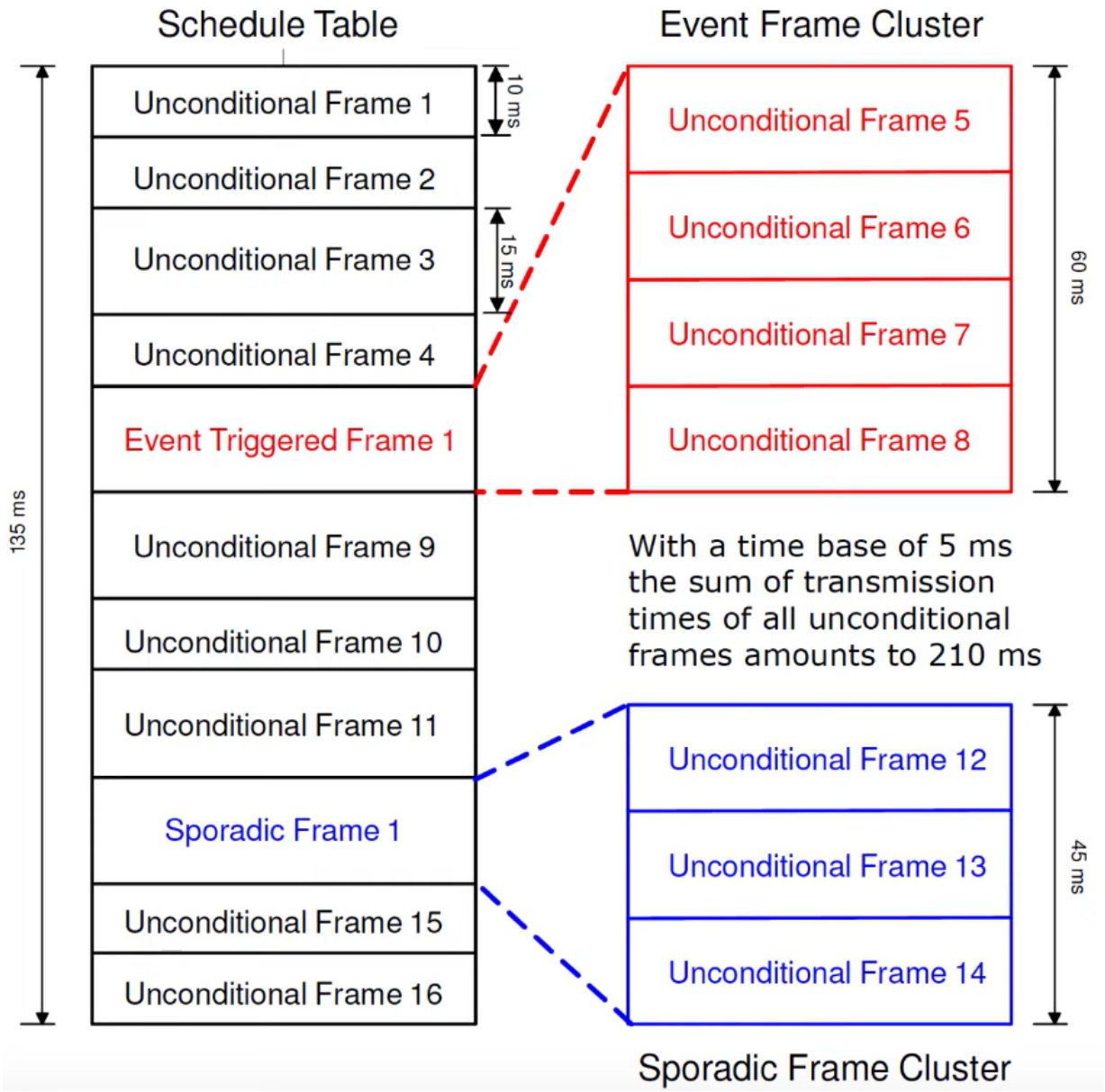


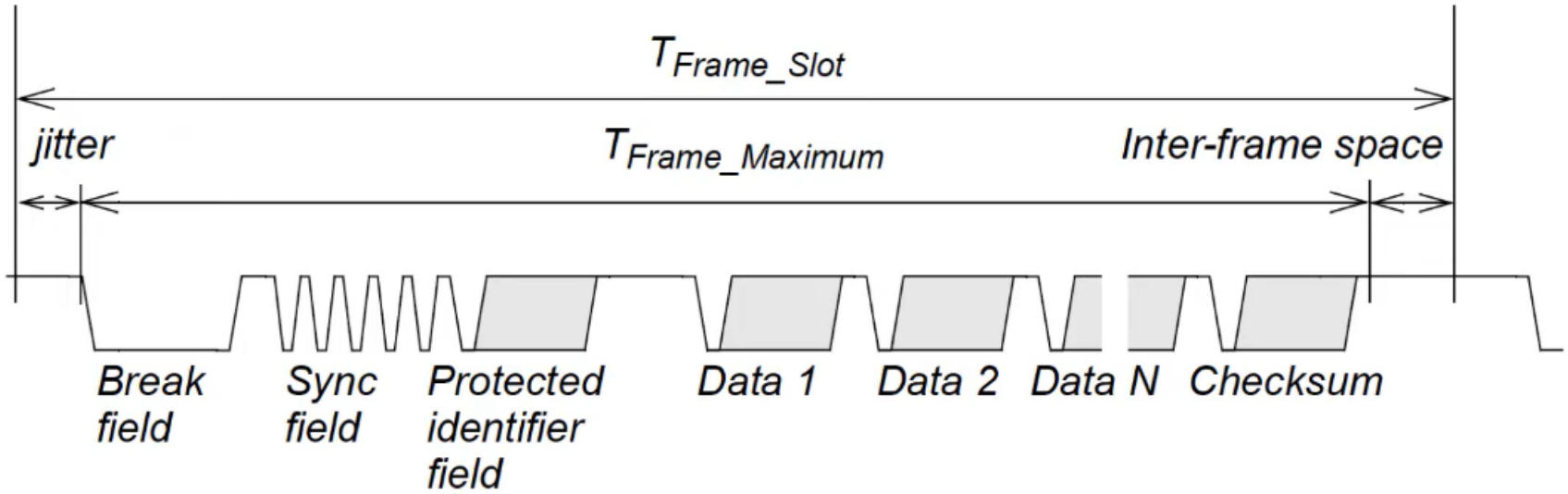
Figure - Sporadic frame example

FRAME TYPES


Schedule Table
Event Frame Cluster
Unconditional Frame 1
Unconditional Frame 2
Unconditional Frame 3
Unconditional Frame 4
Event Triggered Frame 1
Unconditional Frame 9
Unconditional Frame 10
Unconditional Frame 11
Sporadic Frame 1
Unconditional Frame 15
Unconditional Frame 16
10 ms
15 ms
135 ms
Unconditional Frame 5
Unconditional Frame 6
Unconditional Frame 7
Unconditional Frame 8
60 ms
With a time base of 5 ms the sum of transmission times of all unconditional frames amounts to 210 ms
Unconditional Frame 12
Unconditional Frame 13
Unconditional Frame 14
45 ms
Sporadic Frame Cluster

SCHEDULE TABLES

- Assure that the bus will never be overloaded, also the key component to guarantee the periodicity of signals.
- The minimum time unit used in a LIN cluster is the **time base** (T_{base}). Usually, a time base is 5 or 10 ms.
- T_{Frame_Slot} is the time that controls the schedule table timing.
  - $T_{Frame\_Slot} = T_{base} \times n$
  - $T_{Frame\_Slot} > \text{jitter} + T_{Frame\_Maximum}$



DIAGNOSTIC FRAMES

- Always carry transport layer data (Diagnostic) and they always contain eight data bytes.
- The frame identifier is either 60 (0x3C), **called master request frame**, or 61 (0x3D), called **slave response frame**.

RESERVED FRAMES

- Reserved frames shall not be used in a LIN 2.x cluster.
- Their frame identifiers are 62 (0x3E) and 63 (0x3F).

MASTER TASK STATE MACHINE

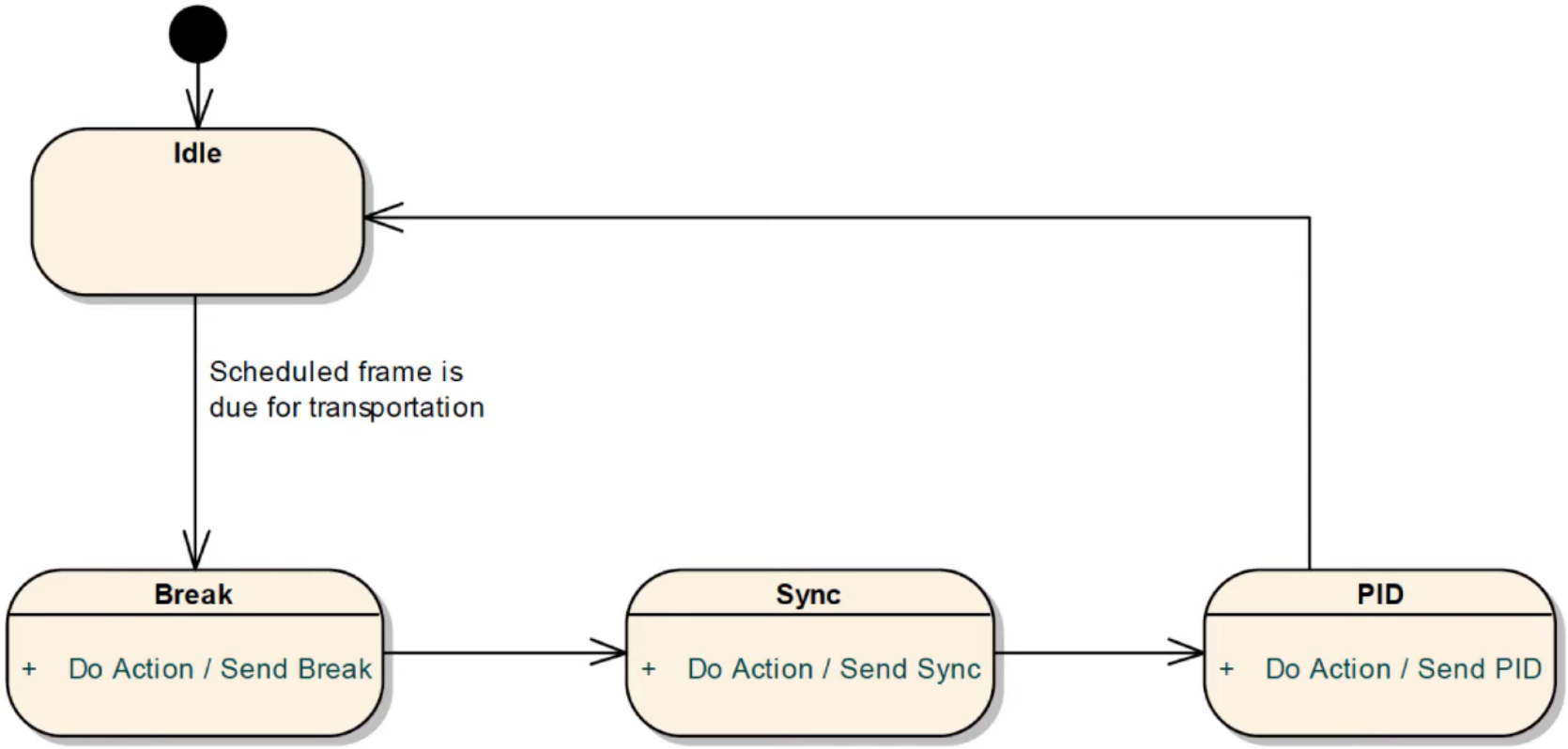


Figure - Complete state machine for the master task

SLAVE TASK STATE MACHINE

- Responsible for transmitting the frame response when it is the publisher and for receiving the frame response when it is a subscriber. The slave task is modeled with two state machines :
  - **Break/Sync field sequence detector**
  - **Frame processor**

SLAVE TASK STATE MACHINE

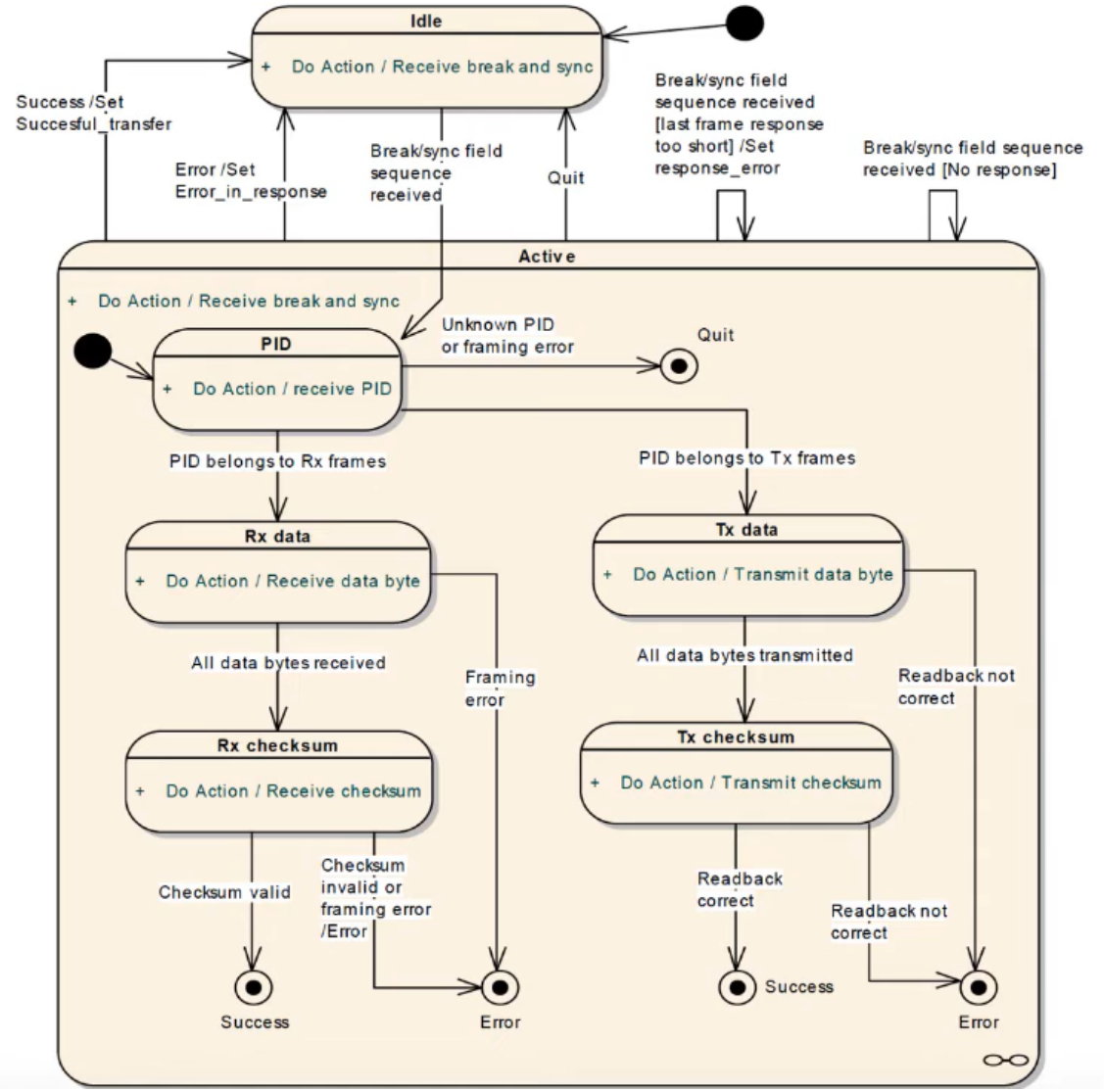


Figure - Complete state machine for the slave task

SLAVE COMMUNICATION STATE DIAGRAM

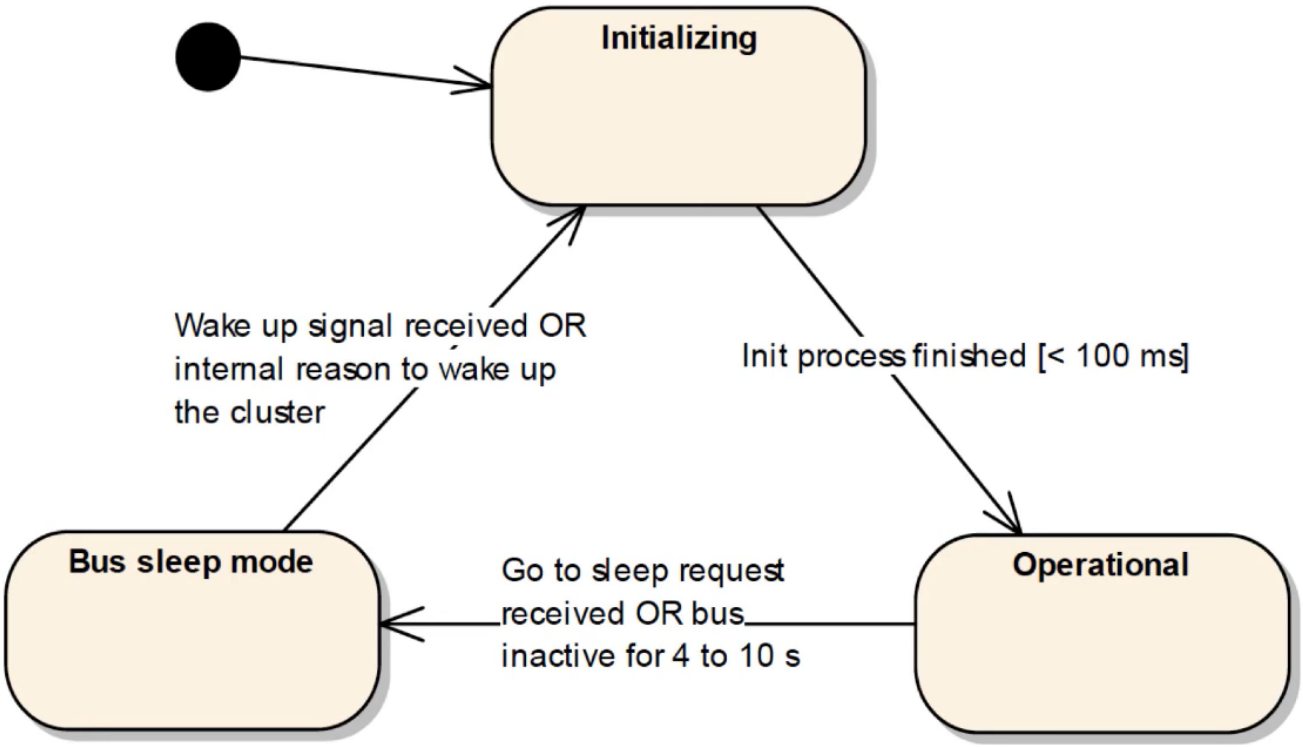


Figure - Slave node communication state diagram

WAKE UP SIGNAL

- Forcing the bus to the dominant state for 250 µs to 5 ms and is valid with the return of the bus signal to the recessive state.
- The master node may issue a break field since the break will act as a wake-up signal.
- Every slave node should detect the wake-up signal. (dominant pulse longer than 150 µs followed by a rising edge of the bus signal and be ready to listen to bus commands within 100 ms).

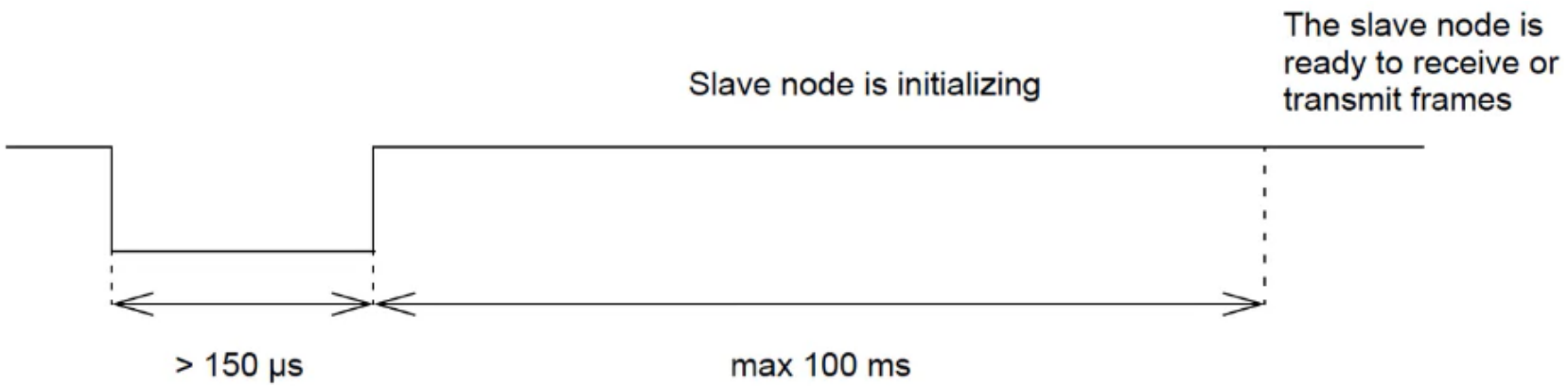


Figure - Wake up signal reception in slave nodes

WAKE UP SIGNAL

- If the node issuing the wake-up signal does not receive a wake-up signal (from another node) or if the master node does not transmit a break field within 150 ms to 250 ms from the wake-up signal, the node issuing the wake-up shall transmit a new wake-up signal.
- After three (failing) requests, the node shall wait a minimum of 1.5 seconds before issuing a fourth wake-up signal.

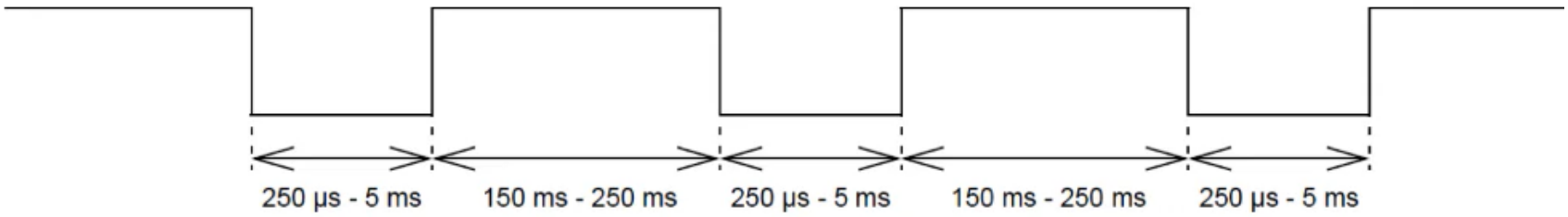


Figure - One block of wake up signals

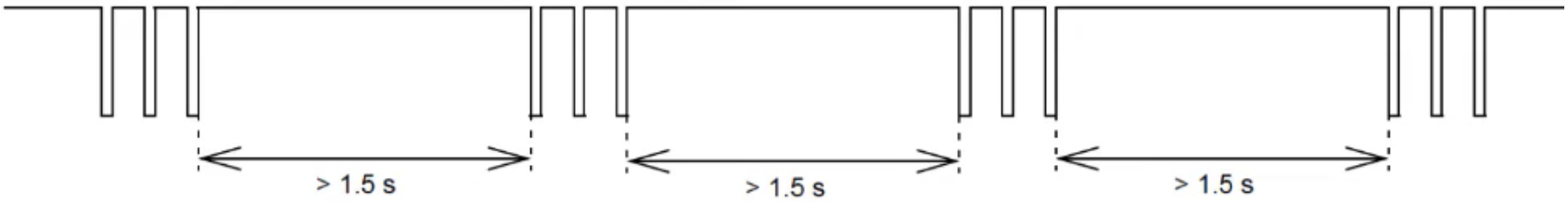


Figure - Wake up signals over long time

GO TO SLEEP

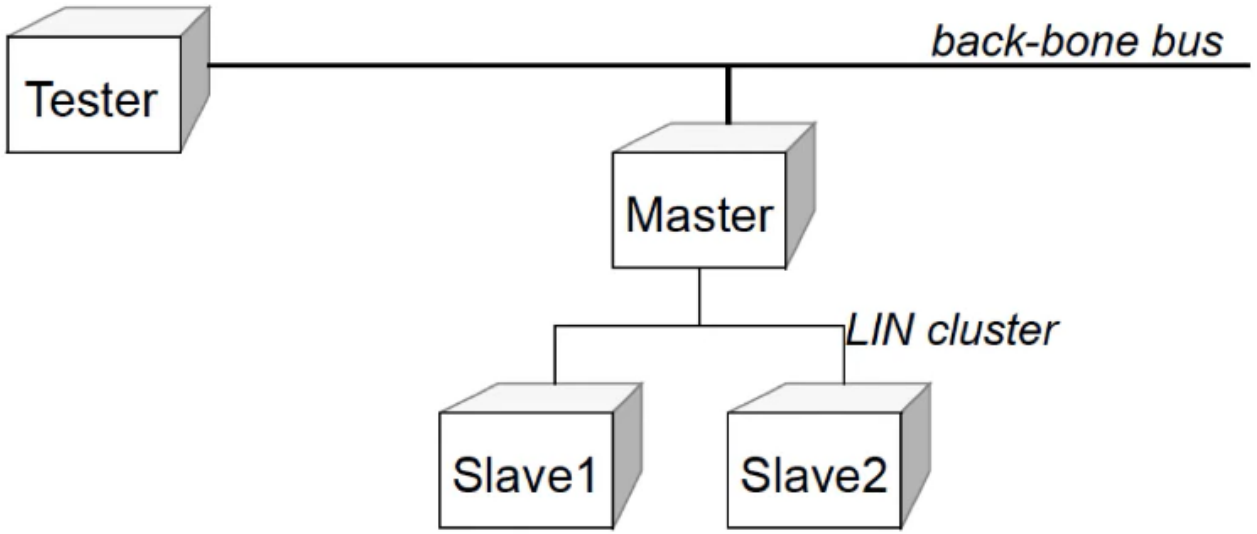
- Go to sleep command is a master request frame with the first data field set to **0x0** and the rest set to **0xFF**.

data1	data2	data3	data4	data5	data6	data7	data8
0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

Table - Go to sleep command

TRANSPORT LAYER SPECIFICATION

- Transport layer messages are transported by diagnostic frames as specified in the Protocol Specification.
- Transport layer is targeting systems where diagnostics are performed on the back-bone bus (e.g. CAN) and where the system builder wants to use the same diagnostic capabilities on the LIN sub-bus clusters.
- Messages are in fact identical to the ISO 15765-2 diagnostics on CAN transport layer.



PDU (PACKET DATA UNIT)

- Units that are **transported in a transport layer frame**.
- Can be a **complete message or a part of a message** (multiple concatenated PDUs form the complete message).
- Messages issued by the client (tester, master node) are called **requests**, and messages issued by the server (master node, slave node) are called **responses**.
- The first byte in the payload is used as a node address (**NAD**).
- Frames have fixed frame IDs, since the **diagnostic frames 60, 61** are used.

PDU (PACKET DATA UNIT)

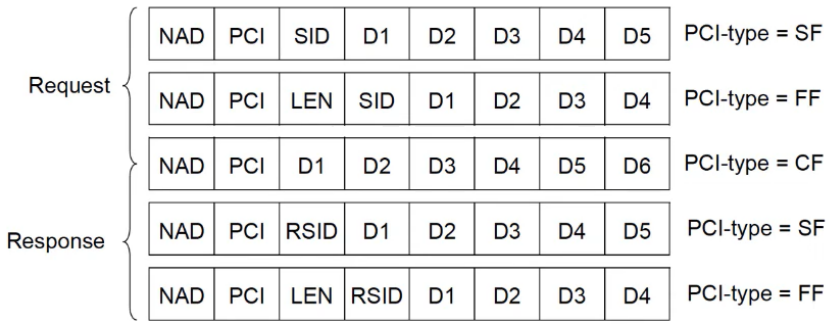


Figure - PDUs supported by the LIN transport layer

PCI (Protocol Control Information)

Type	PCI type				Additional information			
	B7	B6	B5	B4	B3	B2	B1	B0
SF	0	0	0	0	Length			
FF	0	0	0	1	Length/256			
CF	0	0	1	0	Frame counter			

Table - Structure of PCI byte

Property	Wildcard value
NAD	0x7F
Supplier ID	0x7FFF
Function ID	0xFFFF

Table - Wildcards usable in all requests

- **Single Frame (SF)**: Indicates that the transported message fits into a single PDU (max 5 bytes). The length shall then be set to the number of used data bytes plus one (for the SID or RSID).
- **First Frame (FF)**: Used to indicate the start of a multi-PDU message. The following frames are of CF type. The total number of data bytes in the message plus one (for the SID or RSID) shall be transmitted as Length. 4 bits in PCI and 8 bits in LEN, so the maximum is 0xFFF.
- **Consecutive Frames (CF)**: Used for multi-PDU messages.
- **Service Identifier (SID)**: Specifies the request that shall be performed by the slave node addressed. 0 to 0xAF and 0xB8 to 0xFE are used for diagnostics, while 0xB0 to 0xB7 are used for node configuration.

**The Response Identifier (RSID) is derived by adding a hexadecimal offset of 40 to the Frame Identifier (SID).*

NODE CONFIGURATION AND IDENTIFICATION SPECIFICATION

- Define how a slave node is configured and identifying a slave node using the identification service.
- Node configuration and identification services are transported by the transport layer, only the single frame (SF) is used.

D1	D2	D3	D4	D5
Supplier ID LSB	Supplier ID MSB	Function ID LSB	Function ID MSB	Variant ID

Table - LIN product identification

D1	D2	D3	D4
LSB	...	...	MSB

Table - Serial number

NAD

NAD value	Description
0	Reserved for go to sleep command, see Section 2.6.3
1 - 125 (0x7D)	Slave node addresses (NAD)
126 (0x7E)	Functional node address (functional NAD), only used for diagnostics (using the transport layer)
127 (0x7F)	Slave node address broadcast (broadcast NAD)
128 (0x80) - 255 (0xFF)	Free usage. Diagnostic frames with the first byte in the range 128 (0x80) to 255 (0xFF) are allocated for free usage since the LIN 1.2 standard. See user defined diagnostics Section 5.2.6.

Table - NAD values

SERVICE IDENTIFIER

SID	Service	type
0 - 0xAF	reserved	reserved
0xB0	Assign NAD	Optional
0xB1	Assign frame identifier	Obsolete
0xB2	Read by Identifier	Mandatory
0xB3	Conditional Change NAD	Optional
0xB4	Data Dump	Optional
0xB5	Reserved	Reserved
0xB6	Save Configuration	Optional
0xB7	Assign frame identifier range	Mandatory
0xB8 - 0xFF	reserved	reserved

- Response Service Identifier (RSID) for a positive response is always **SID + 0x40**.

ASSIGN NAD

NAD	PCI	SID	D1	D2	D3	D4	D5
Initial NAD	0x06	0xB0	Supplier ID LSB	Supplier ID MSB	Function ID LSB	Function ID MSB	New NAD

Table - Assign NAD request

NAD	PCI	RSID	Unused				
Initial NAD	0x01	0xF0	0xFF	0xFF	0xFF	0xFF	0xFF

Table - Positive assign NAD response

READ BY IDENTIFIER

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	0x06	0xB2	Identifier	Supplier ID LSB	Supplier ID MSB	Function ID LSB	Function ID MSB

Table - Read by identifier request

Identifier	Interpretation	Length of response
0	LIN Product Identification	5 + RSID
1	Serial number	4 + RSID
2 - 31	Reserved	-
32 - 63	User defined	User defined
64 - 255	Reserved	-

Table - Supported identifiers using read by identifier request

Id	NAD	PCI	RSID	D1	D2	D3	D4	D5
0	NAD	0x06	0xF2	Supplier ID LSB	Supplier ID MSB	Function ID LSB	Function ID MSB	Variant
1	NAD	0x05	0xF2	Serial 0, LSB	Serial 1	Serial 2	Serial 3, MSB	0xFF
32-63	NAD	0x02 - 0x06	0xF2	user defined	user defined	user defined	user defined	user defined

Table - Possible positive read by identifier response

NAD	PCI	RSID	D1	D2	Unused		
NAD	0x03	0x7F	Requested SID (= 0xB2)	Error code (= 0x12)	0xFF	0xFF	0xFF

Table - Negative response

DIAGNOSTIC SPECIFICATION

- Node configuration can be performed by the master node independently while diagnostic services are always routed on request from an external or internal test tool.
- Three different classes of diagnostic nodes are defined:
  - **Class I**: Uses normal signaling.
  - **Class II** and **Class III**: Use the transport layer.
- **For Diagnostic Class I**: Smart and simple devices like intelligent sensors and actuators, requiring none or very low amount of diagnostic functionality.
- **Class III** slave nodes optionally support flash/eeprom reprogramming. This requires an implementation of a boot-loader and necessary diagnostic services to unlock the device, initiate downloads, and transfer data, etc.

Slave Diagnostic Class	1	2	3	UDS - SID
Diagnostic Transport Protocol Requirements				
Single frame transport only	+			
Full transport protocol (multi-segment)		+	+	
Required Configuration Services				
Assign frame identifier range	+	+	+	0xB7
Read by identifier (0 = product id)	+	+	+	0xB2 0x00
Read by identifier (all others)	optional	optional	+	0xB2 0xXX
Assign NAD	optional	optional	optional	0xB0
Conditional change NAD	optional	optional	optional	0xB3
Pos. response on supported config. services	+	+	+	SID + 0x40
Required UDS Services				
Read data by identifier:				0x22
– hardware and software version		+	+	0x22
– hardware part number (OEM specific)		+	+	0x22
– diagnostic version		+	+	0x22
Read by identifier (parameters)		+	+	0x22
Write by identifier (parameters)		if applicable	if applicable	0x2E
Session control			+	0x10
Read by identifier (sensor and actuator data)			+	0x22
I/O control by identifier			+	0x2F
Read and clear DTC (fault memory)			+	0x19, 0x14
Routine control			if applicable	0x31
Other diagnostic services			if applicable	...
Flash Reprogramming Services				
Flash programming services			optional	0xXX

+ = it is mandatory for the class

<Thank> You </Thank>