Application Note

LIN Protocol and Physical Layer Requirements

Eric Hackett

ABSTRACT

The *Local Interconnect Network* (LIN), ISO17897, is a multipoint, low-cost and easily-implemented communication bus in automobiles. It works as a sub-bus for the *Controller Area Network* in most applications. This application note presents the integral parts of LIN technology, focusing on the LIN transceiver itself, information on the protocol, and the physical layer requirements for real applications.

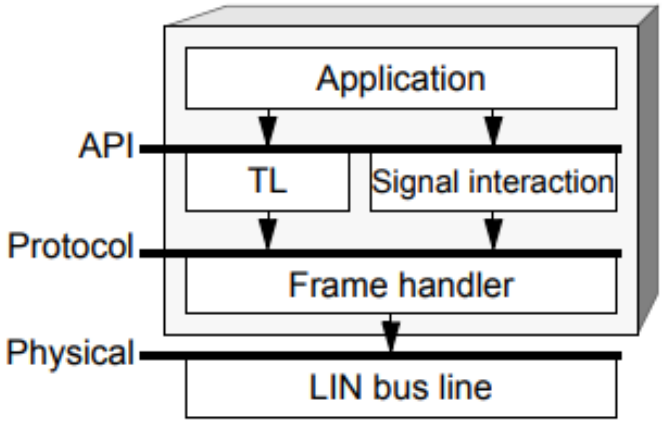


Figure 1-1. LIN Hierarchy Chart

Table of Contents

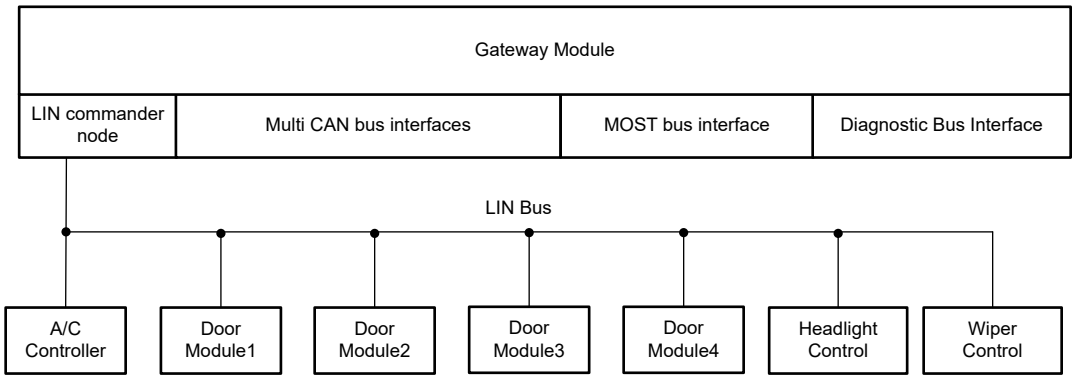
List of Figures

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

The amount of electrical systems and components continue to grow as automobiles become more intelligent, safe, and comfortable. The growth of these components and systems demand a need for communication transceivers, to facilitate their interaction in the most advantageous way possible for manufacturers. LIN was developed to manage communication between these components and systems in an efficient and straightforward fashion, where the bandwidth and versatility of CAN was not needed; though in most instances, it is a sub-bus to the CAN bus.



#The gateway includes the interfaces with all the buses which typically comprise the Vehicle network. The LIN commander nose here is configured as part of the gateway Module.

The different localized control modules connect to the LIN bus are the responder modules.

Figure 1-1. High-Level Application Diagram

1.1 LIN Specification Progression

The most up-to-date LIN standard was defined in 2010 (LIN 2.2A, the LIN Consortium). It was then transcribed to the *International Organization for Standardization* (ISO) to be accepted as ISO 17897 and officially released in 2016. Prior to 2010, LIN went through a series of revisions, being fully defined first in LIN 1.1 (1999), where the LIN Protocol Specification, LIN Configuration Language Specification, and LIN Application Interface Specification were established by a board called the LIN consortium. Each of these are necessary parts in creating the full LIN cluster in a way that is consistent across the market, allowing any car manufacturer to use the communication scheme. The LIN protocol specification describes the physical and data link layers, and the LIN Configuration Language enables the LIN cluster to be described in a file that is straightforward for any developer.

1.2 Workflow Concept

The LIN transceiver and its implementation are the focus of this application note; however, it is important to have a high-level understanding of the entire LIN network to understand the place of the transceiver in an application. As LIN became defined, it was not only specified for the actual 1's and 0's data delivery, but for a higher-level network implementation: The LIN workflow. The LIN workflow supports an easy-to-use, dependable implementation scheme for those working with the protocol. The configuration of the entire network cluster is defined and standardized, which is where the *LIN Description File* (LDF) comes in.

The LDF is what differentiates the LIN clusters from each other, defining the specific use and properties for that cluster (node amount, amount and the description of message frames, message rate, and so forth). This allows the generation of software files by developers to establish which task each node in the cluster performs. The LDF can be used to automatically generate the software involved in communication, as well as supply information for measurement and test tools involved in the LIN cluster analysis.

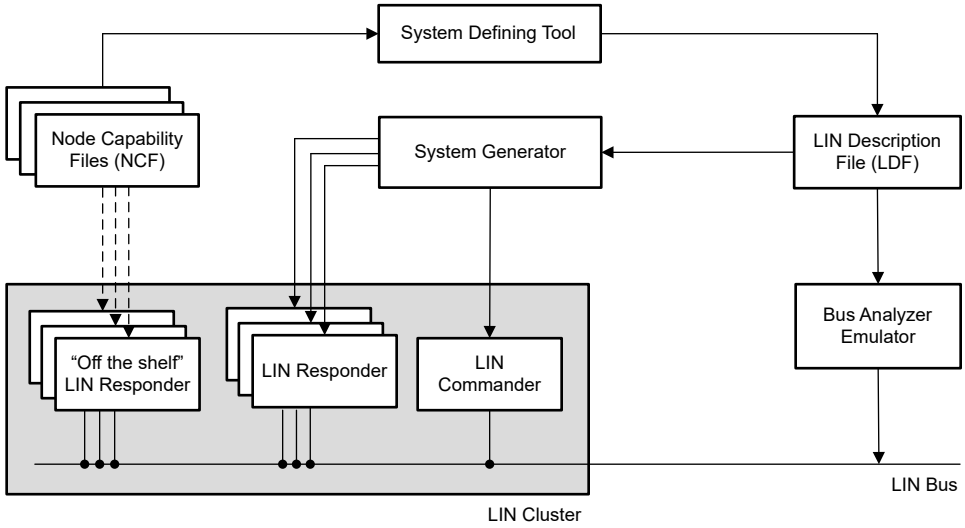


Figure 1-2. LIN Workflow

The LDF is written using syntax defined by the *LIN Configuration Language Specification*. This syntax is used in combination with the *System Defining Tool* to create the LDF, and thus define the whole network. Along with these tools, there is the *LIN Node Capability Language*. This allows the developer to define and describe the implementation of *Off-the-Shelf Nodes*, which are easily-implementable, general-purpose LIN nodes designed for typical applications that can be bought in large quantities.

2 Network Architecture

2.1 General Layout of the LIN Bus

A LIN cluster is defined as a number of LIN nodes connected through a physical cable. There are two types of nodes in every cluster: one Commander node and up to 16 subsequent Responder nodes. This commander node is what manages the communication along the bus to each responder. More detail on the *Commander-Responder* principle is discussed in Section 2.3.

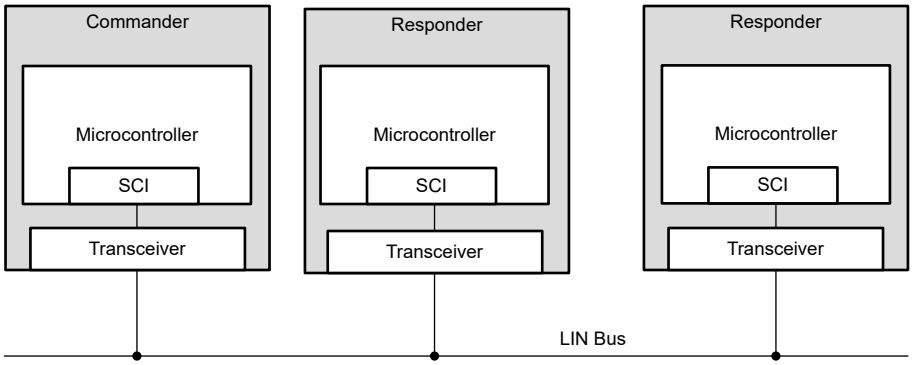


Figure 2-1. High-Level LIN Transceiver in Network

The idea of LIN is to be a simple, yet cost-effective communication interface. This is why a dedicated communication controller is not implemented. Instead, a microcontroller is programmed with the LIN protocol and used to drive the communication to the transceiver through the serial interface. This interface is named *Serial Communication Interface* (SCI) and it has replaced UART in most LIN applications. Both are typical of most microcontrollers, which allow less work on the back end to install.

The LIN bus transmission only requires one wire, and a slower communication speed is used in order to properly handle any radiated emissions issues. All nodes are passively connected to the bus, and a pullup resistor is used to ensure the bus is at the supply voltage level when the nodes are in the off-state.

2.2 Serial Communication Principles

The SCI is the dominant interface used between LIN transceivers and the microcontrollers that communicate with them. Originally UART was used, but fault-free interface through UART is known to be difficult to achieve.

The microcontroller transmits bit frames, starting with the dominant start bit. This synchronizes all receivers on the bus, followed by the least significant bit to most significant bit, then a stop bit. This constitutes one SCI frame, and a LIN message is composed of multiple SCI frames.

2.3 Commander-Responder Principle

In every cluster, there is one commander node and up to 16 responder nodes. The commander node controls all communication on the bus and contains both the commander and responder task to be delivered. The responder nodes cannot communicate with each other, contain only the responder task, and are only capable of responding to the commander if the message is directed at them. The commander sends out a request to a designated responder as a header (beginning of the frame), and the responder responds to the commander, as a response frame. There is also a case where the commander sends the responder the header and response frame, and the responder only listens but with no response. Both situations guarantee predictable yet defined bus traffic, disallowing collisions for the most part because the commander is always initiating the communication. This predictable nature allows for scheduling of messages.

If the developer of the LIN cluster does a proper job planning messages and calculating their lengths, a schedule can be developed and no collisions will occur. A schedule is the organization of messages frames into slots, and is what sets the send time of all the messages to be sent at any given time. The tokens (also referred to as requests) are sent by the commander at these given times set by the schedule. These tokens are sent to responders, and the responder can either ignore, respond, or just receive the data. The token and the data (header and response) are what make up the LIN messages, and up to 64 messages can be defined per cluster.

The problem with the commander-responder system is that the commander controls all communication, and if the commander fails the whole cluster fails. In schemes where all nodes can act as a commander and responder, this does not happen and this quality is ultimately what keeps LIN from being used in safety-related applications (that, and the slow message rate). The LIN cluster is also not inherently capable of event-driven communication, because the LIN responders can only communicate with the bus if they are requested to do so.

2.4 Message Frame Format

Every LIN message has a specific structure: the first part being the token and the second part data (the Header and the Response). The token is always transmitted by the commander task, and is divided up into

the sync break, sync field, and the protected identifier (PID). The sync break and sync field are used to have all the responders on the LIN bus synchronized to the commander's timing (without the need of any crystal or oscillator), and the PID is what defines which responder responds, receives, or ignores the message header being sent. The header in total consists of at least 13 bits for the SYNC break, 1 delimiter bit, 10 SYNC field bits (1 start bit, 8 bits for synchronization, and 1 stop bit), and 10 identifier bits (1 start bit, 6 bits for the identifier, 2 bits for parity, and 1 stop bit).

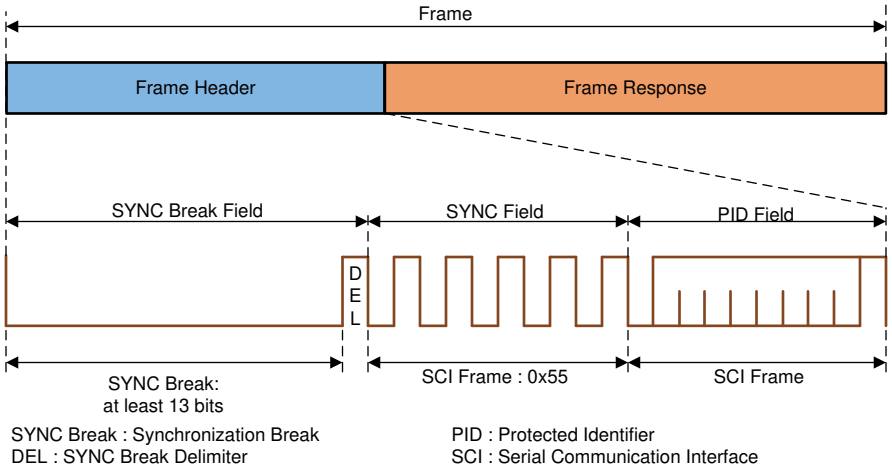


Figure 2-2. LIN Frame Header Explanation

The Data (response) portion of the message is sent by the responder task, which can be sent by the commander or the responder node depending on the PID “instruction”. The Response is broken up into data bytes (up to 8), and a checksum. The checksum is a protection scheme for the data bytes, it verifies that the message sent is what was intended, and that no errors were introduced during transmission. Any node can receive the frame response, but which node actually does use it, is dependent on the LDF. The response in total consists of 10 bits for each byte of data (1 start bit, 8 bits for data, 1 stop bit), for up to 8 data bytes, and 10 bits for checksum (1 start bit, 8 bits for checksum solution, 1 stop bit).

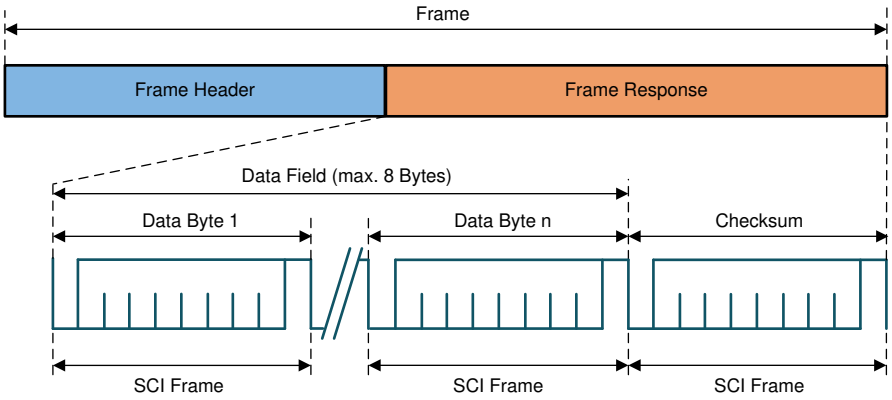


Figure 2-3. LIN Frame Response

3 Physical Layer Requirements

The LIN physical layer is based on the ISO 9141 standard, with some modifications for automotive applications; specifically EMC, ESD, Transient pulse response, and so forth. It is a bus communication interface that is bidirectional, biased to the battery voltage of the vehicle through a resistor and diode (commander node only), and is connected to the transceiver of every node in the LIN cluster.

The transceiver is what facilitates the communication between the bus and the network. The LIN transceiver converts the bit logic from the microcontroller into higher voltage levels, as a transmission along the bus and vice versa. The TXD (transmit) and RXD (receive) of the LIN transceiver, facilitate the communication to and from the bus through voltage translation that happens as the signals pass through the transceiver. The TXD is connected to the microcontroller, where the message is sent and then broadcasted on the LIN bus. The RXD monitors

the bus and converts the messages on the LIN bus to voltage levels the microcontroller can interpret, and thus respond to the communication happening on the bus. Typical voltage levels for the TXD and RXD are typical of most microcontroller levels: 3.3 V and 5 V. The LIN bus and LIN transceivers usually operate at voltages ranging from 9 V to 18 V, but some can go up to 30 V (depending on the application). A typical vehicle is a 12-V battery system, but some larger vehicles go up to 24 V.

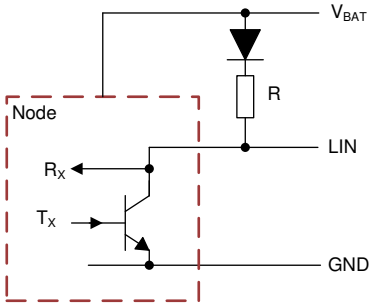


Figure 3-1. Simplified LIN Driver Schematic

3.1 Bus Signaling Fundamentals

Considering these voltage levels on the bus, there are thresholds that most transceivers abide by, with some companies deviating slightly from the norm. Because it is a single-ended communication scheme and the thresholds cannot be set by a voltage differential, these thresholds are based on a percentage of the battery voltage in the system. These thresholds determine when the bit is considered “recessive” or “dominant”. Recessive and dominant are just different ways of saying high and low (respectively) on the bus. The naming convention comes from the concept of how the bus and transceivers interact with each other and how the signals are generated inside the IC.

From a high level, a cluster is equivalent to an open-drain circuit, meaning the bus requires a pullup resistor and all the nodes are passively connected to the bus through transceivers. When prompted, the transceivers control the voltage levels on the bus. The pullup resistor ensures the voltage levels on said bus meet or are close to the battery voltage level when the TXD control of the transceiver is in the off state. Once the transceiver becomes active and the TXD control transistor turns on, the bus is driven low, to nearly ground level and the high state is overwritten. Therefore, the bus is pulled up high when the transceiver is off or in a passive state, hence “recessive”. When the transceiver TXD begins conducting and becomes active, the bus is driven low, hence “dominant”.

3.2 Pullup Values

The pullup resistor values are different for the commander and responder nodes. The commander node requires an external pullup resistor and diode according to the LIN specification. The typical value seen is 1 kΩ (other typical values are 600 Ω and 500 Ω) in series with a diode – for reverse-polarity protection – connected to the battery voltage. The typical pullup value of the LIN responder is 30 kΩ, and in all modern LIN transceivers this is integrated within the IC, so no external pullup is necessary in the responder configuration.

3.3 Threshold Values

The sender and receiver have different levels that meet these recessive and dominant voltage level requirements. For dominant pulses (low), the sender must drive the voltage level down to 20% of the battery voltage level, while the receiver will interpret a dominant bit when the voltage level reaches 40% on their end. For recessive pulses (high), the sender must drive the voltage to 80% of the battery voltage, while the receiver interprets a recessive bit when the voltage level reaches 60% on the bus. The difference in levels between the receiver and sender is due to differences in the external supply voltage and the actual LIN bus voltage. Drops in voltage that may happen in the cabling, ground shifts, or just changes caused by filtering components along the bus are the main causes for the deviation of the external supply versus the bus level.

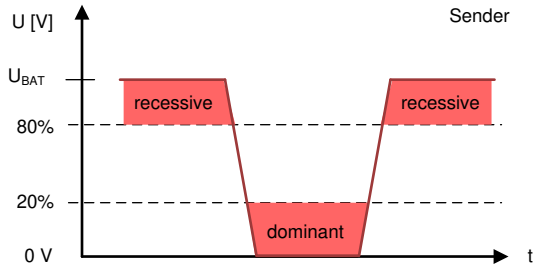


Figure 3-2. Bus Signal Thresholds for Senders

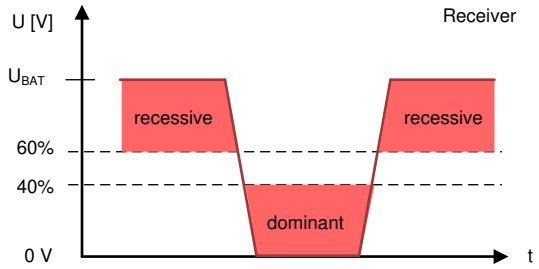


Figure 3-3. Bus Signal Thresholds for Receivers

3.4 Bit-Rate Tolerance and Timing Requirements

The bit rate for LIN ranges from 1 to 20 kilobits per second, with a bit rate tolerance ±14%. This 14% value comes from the fact that low-cost, on-chip oscillators are used and with internal calibration, better than ±14% can be achieved. This accuracy allows detection of a break in the message stream, and with timing calibration using the synch field, the reception and transmission of messages are ensured by the message frames themselves. Temperature changes and voltage drift can cause changes in the bit rate, and the on-chip oscillator accounts for these variations when measuring the bit rate and generating the rest of the message frame (after the synch field).

3.5 Synchronization and Bit Sampling

Barring a special use case, all bit times use the bit timing of the commander node as the reference. The synch byte consists of ‘0x55’ (8 bits of alternating 1’s and 0’s, starting with 0) which is essentially a clock signal at a given frequency. The falling edges of the pattern are used for synchronizations that are combined with the start and stop bit (10 bits total), which allows 4 total falling edges for synchronization of the responder nodes. This also allows for accurate bit width (T_{BIT}) measurement. However, because of different methods of synchronization on the market being used in terms of bit sampling (not necessarily on the falling edge of the start bit), the LIN 2.2 specification removed the specification of start-bit sampling. This allows all methods for start-bit sampling that meet the timing requirements of the byte field synchronization (t_{BFS}, which is defined as 1/16 of T_{BIT} typical, 2/16 T_{BIT} maximum).

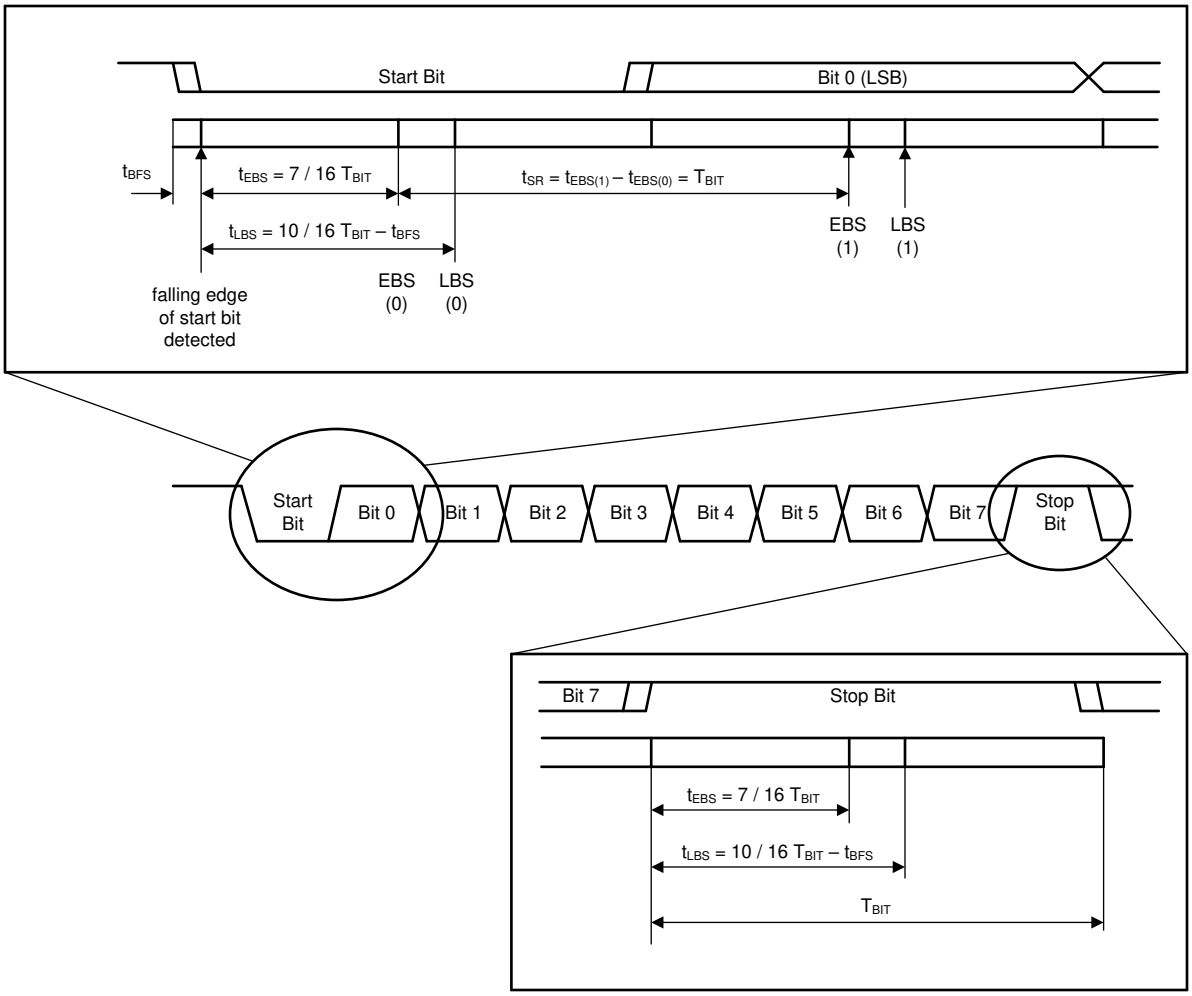


Figure 3-4. Bit Sampling Illustrated

After synchronization of the responder nodes occurs, each bit must be sampled accurately to ensure proper interpretation of the message by the LIN cluster. Each bit shall be sampled between the earliest bit sample (t_{EBS}) and the latest bit sample (t_{LBS}); t_{LBS} is dependent on t_{BFS} via Equation 1:

$$t_{LBS} = 10/16\ T_{BIT} - t_{BFS} \quad (1)$$

T_{EBS} is defined as minimum 7/16 T_{BIT}. The rest of the bits after the first bit are then sampled with a sample rate (t_{SR}). These are based on the earliest bit sample of the former bit (n – 1) and the earliest bit sample of the current bit (n), and given by the following equation:

$$t_{SR} = t_{EBS(n)} - t_{EBS(n-1)} = T_{BIT} \quad (2)$$

3.6 Duty Cycle

To ensure that messages sent are interpreted correctly, the LIN bus must meet the correct voltage levels based on the battery supply and these voltages have to be met within the correct bit sampling time of the receiver.

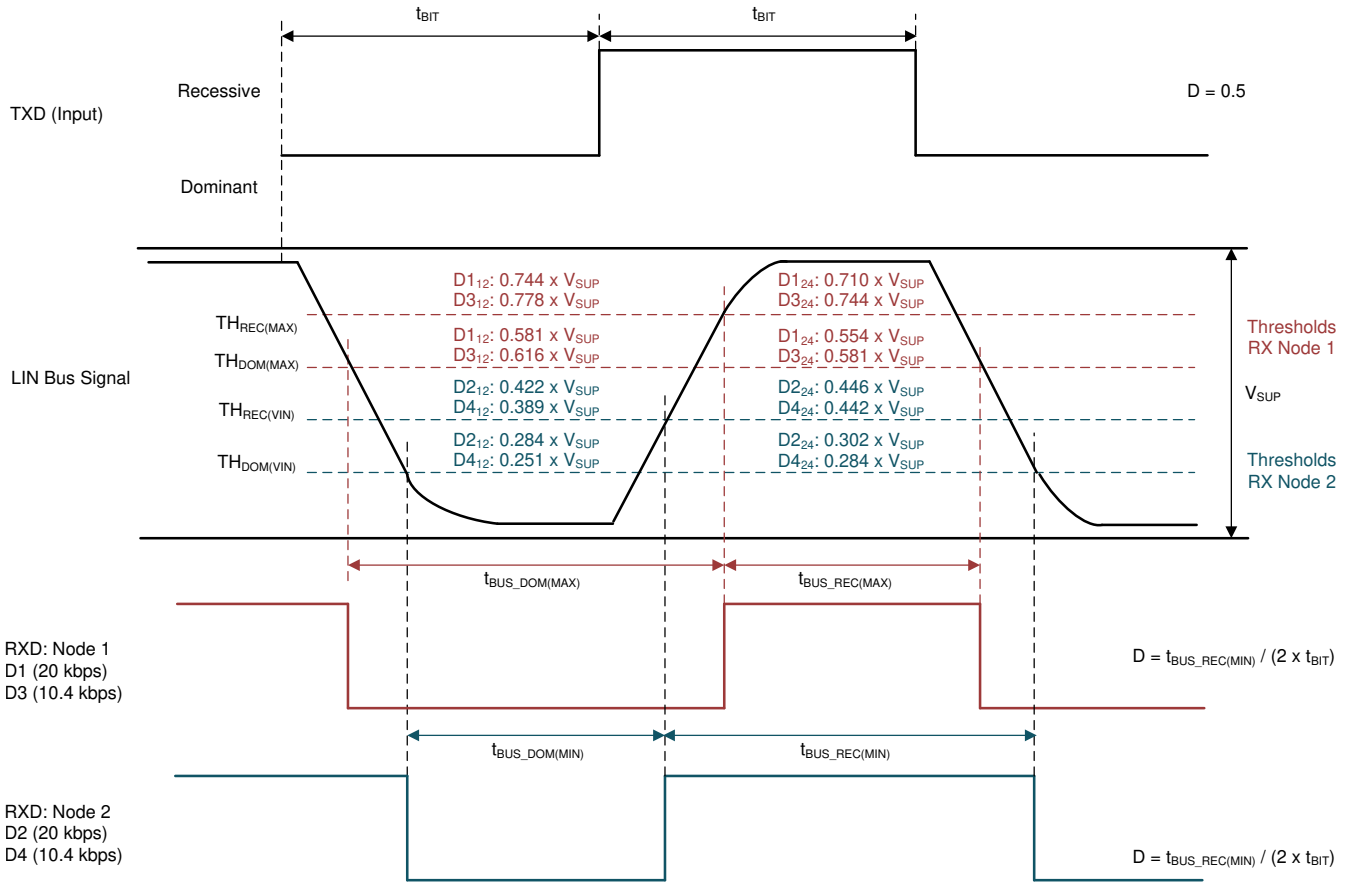


Figure 3-5. Bus Duty Cycle Requirement

Figure 3-5 is from *TLIN1029-Q1 Local Interconnect Network (LIN) Transceiver with Dominant State Timeout*, and is referenced from the LIN specification. It defines the bus timing as a requirement for the proper sampling of each bit. This definition is defined so that when transceivers are designed, the duty cycle is not distorted when propagating from TXD to LIN and from LIN to RXD. Because there is no clock signal sent with the messages and the synchronization is based on the synch field, if there is too much duty cycle variation the commander clock or responder clock can also vary. This affects timing for the remainder of that power cycle.

4 Filtering, Distance Limitations, Nodes on Bus

4.1 EMI and Signal Conditioning

EMI filtering for the LIN bus is highly recommended and should be done to mitigate any EMI issues (from or into the transceiver) as well as help with transient pulses and ESD strikes. In addition to the LIN messages themselves radiating noise with rising and falling edges as well as asymmetrical waveforms, noise from the rest of the vehicle can penetrate the LIN bus. This can be done through cabling, the GND, or the battery line itself. The battery line is an especially nasty culprit of noise, since it is connected to every other system in the vehicle, though isolation and heavy filtering from other ECUs help with this.

The bare minimum bus filtering consists of a shunt capacitor at the commander and each responder node. Careful consideration must be taken as to not overload the capacitance on the bus, as this slows down the edges too much and corrupts the interpretation of bits on the bus. Typical values for bus capacitance are 220 pF at the responder nodes, and up to ten times that value at the commander node. Other methods used are inductors in line with the bus, ferrite beads, and inductor-capacitor-inductor T-filters. Ferrite beads and T-filters tend to be a little more expensive, so inductors are more standard practice. This creates an LC filter along the bus, which is a suitable low-cost suppression technique.

4.2 ESD and Transients

ESD strike and *Transient* pulse suppression are also important for the LIN bus. In any application, ESD strikes and transient pulses are present, but specifically in automotive environments, because there are so many ECUs

in close proximity of each other. It is especially important for the systems to be immune to any destruction or interruption in function, due to these high-voltage phenomena. The capacitor on the bus helps with ESD strikes by slowing down the pulse edge, but this is not enough to stop it from driving current into the device and potentially disrupting communication. Specially-made ESD protection and TVS diodes are designed to help with these conditions. LIN transceivers are tested to verify they can withstand these strikes. Typical passing level values of these tests are ±6 kV for ESD direct contact, and ±100 V for transient pulses.

4.3 Distance and Node Limitations

The LIN specification defines the maximum number of nodes that can be connected to a LIN bus: 1 commander node and 16 responder nodes and the max cable harness length: 40m. Because of the definitions, there is no concern about having too many nodes on the bus nor the cable being too long, opposed to other communication interfaces. Capacitance on the bus still needs to be within a reasonable range for proper communication, and this can be affected by cable length, nodes, or bus filtering.

Since the cable length and node amount are limited by definition, the only parameter to keep in mind is any added capacitance; a good guideline is to stay under or close to 10-nF total bus capacitance at 20 kbps communication speed. Figure 4-1, Figure 4-2, and Figure 4-3 show the effect of having the nominal amount of capacitance, and what having too much capacitance looks like. In the case with too much capacitance, the rising edges are not able to reach the full voltage level in time for the next bit, and thus the bits are not interpreted correctly, as the RXD waveform illustrates.

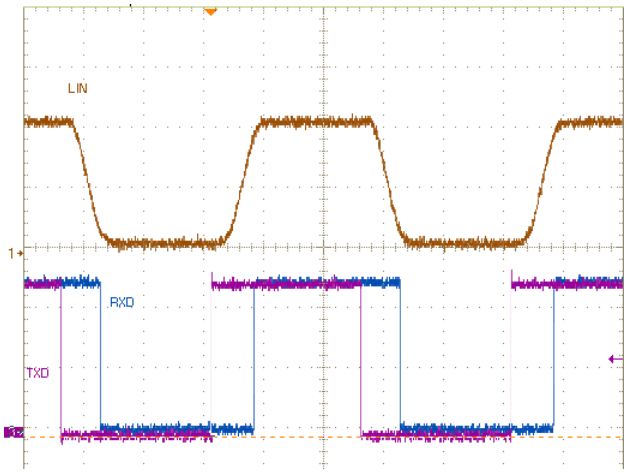


Figure 4-1. LIN Bus With 220 pF, 20 kbps Message

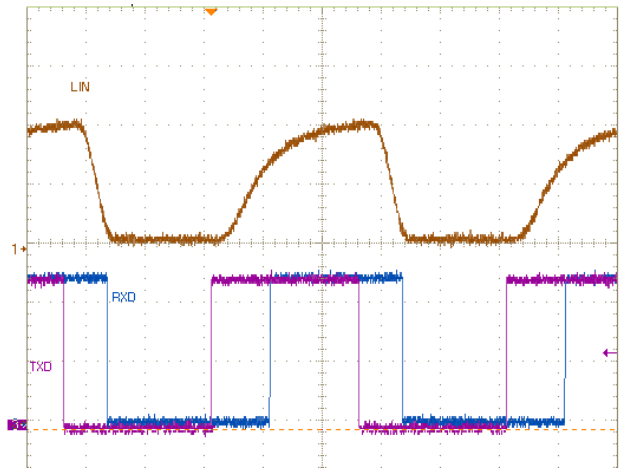


Figure 4-2. LIN Bus With 10 nF, 20 kbps Message

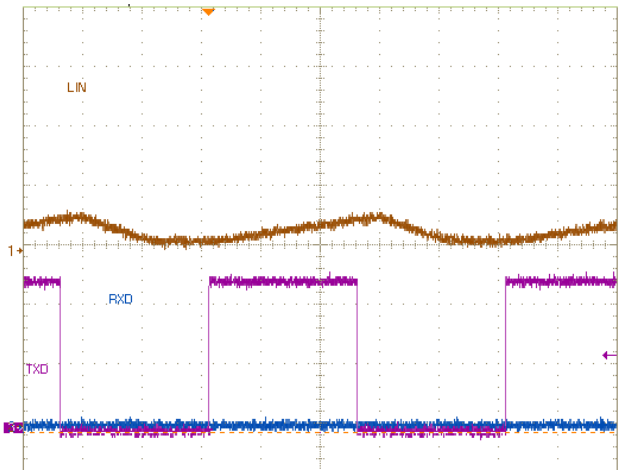


Figure 4-3. LIN Bus With 220 nF, 20 kbps Message

5 LIN Transceiver Special Functions

With most modern LIN transceivers, there are special functions that help with specific application needs. Most of the features listed are for systems that focus on low power consumption. The descriptions of these features are specific to TI devices, but can be generalized for all transceivers across the industry. Naming conventions of the modes and specific internal function of the transceiver in the modes and features, can differ for devices from other companies.

5.1 Low-Power Modes

5.1.1 Sleep Mode

Sleep Mode is the low-power mode of LIN transceivers. This mode is used to save power when the LIN transceiver is not needed in any part of the system. The mode is typically entered by putting a logic low on the Enable pin (if there is one) of the device. The name *Sleep Mode* implies the device is in a less functional state, but is still able to monitor the LIN bus for any wakeup signals (explained further in Section 5.2).

In Sleep mode, the LIN driver is disabled and the internal LIN bus termination is switched off to minimize current draw if the LIN bus is shorted to ground for any reason. A low-power receiver is enabled and the normal receiver function is disabled, and EN input is still active.

5.1.2 Standby Mode

Standby Mode is also a low-power mode, and is the mode the transceiver transitions to when a wakeup request is sent, but the EN pin is still in a low state. The main difference between Standby and Sleep mode is that in Standby mode, the RXD output is low, while in Sleep mode the RXD output is floating. This signals to the controller that the device is in Standby mode after a wakeup request, and can be transitioned into *Normal Mode* through the control of the EN pin. Without the EN pin held high at power up, the LIN transceiver turns on in Standby mode by default.

5.2 Wakeup

5.2.1 Pin Wakeup

All LIN transceivers have pins specific to waking the device from Sleep mode (if they have sleep mode), and these can be used instead of the LIN bus wakeup request. The WAKE pin on LIN transceivers is typically a high-voltage pin, and responds to either a negative transition (high-to-low voltage level), positive transition (low-to-high voltage level), or both.

The EN pin is an IO level pin and can also be used to transition in and out of Sleep mode, though the transition polarity matters. A negative transition places the device into Sleep mode, while a positive transition puts the device back into Normal mode.

5.2.2 LIN Wakeup

LIN wakeup is a request made on the LIN bus while the transceiver is in Sleep mode. This request is a specific pattern that the transceiver detects while monitoring the bus in the low-power mode. The pattern is a recessive-to-dominant transition where the dominant position is held for the specified amount of time. When the dominant pulse is held for the correct amount of time, the LIN transceiver transitions into Standby mode, and RXD is held low. Figure 5-1 shows a timing diagram of the procedure.

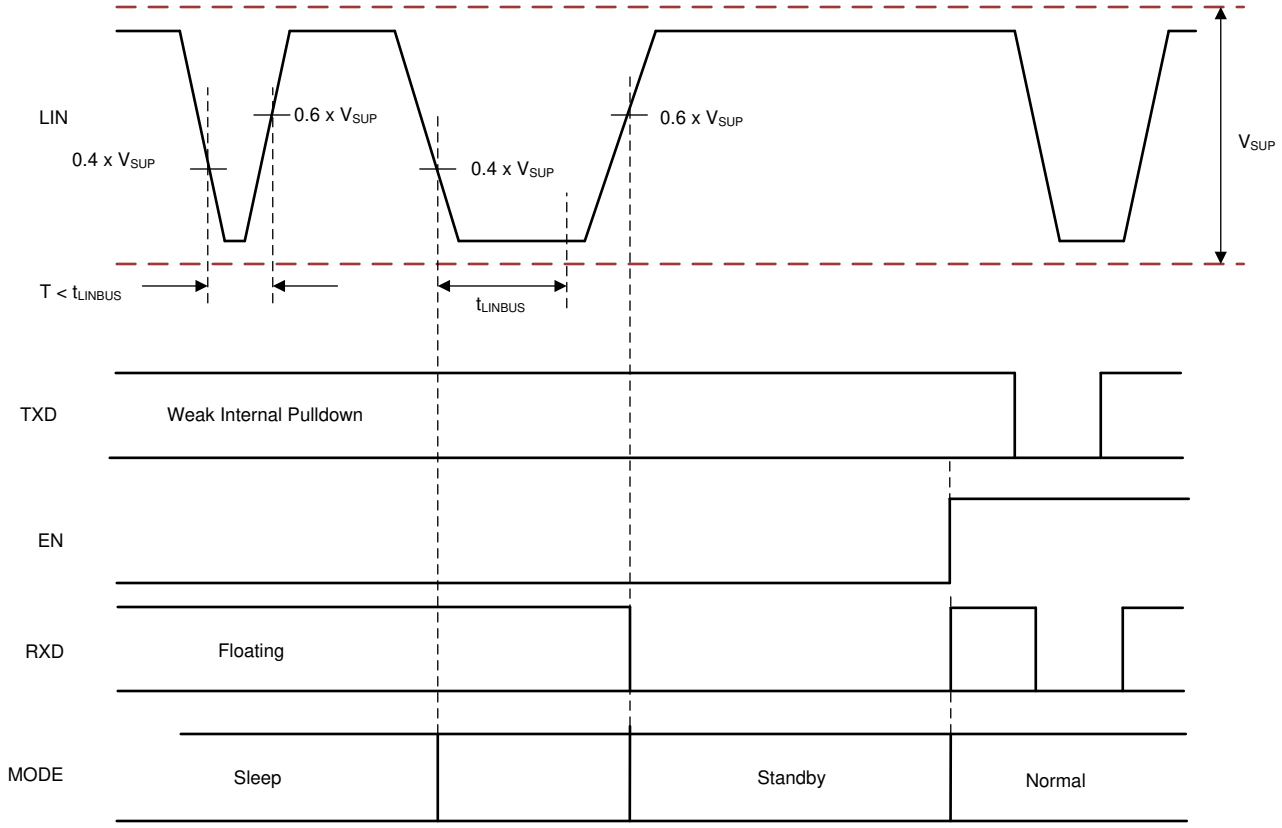


Figure 5-1. LIN Wake-up Pattern

5.2.3 Dominant Timeout

Dominant timeout is a condition that happens as a failsafe for the LIN bus, but only in Normal mode. If TXD is driven low (dominant) unintentionally for an extended period of time, the LIN bus will timeout. This meaning that the transmitter is disabled and the bus is pulled up to a recessive state. The extended period of time for TI devices is typically 20 µs, but can vary from part to part depending on design intent. The protection is cleared and the timer is reset once a rising edge is detected on TXD. During this condition, the transmitter is disabled, the device stays in Normal mode, and RXD follows the LIN bus. This protection is in place to make sure excessive power is not consumed in case there is a short-to-battery while the LIN bus is held dominant.

6 Advantages and Disadvantages

The main advantages of using LIN over a more robust communication interface are based in simplicity versus complexity. The LIN interface is simple, low-cost to implement and has use relative to CAN (and other differential interfaces) with a small, readily-available component requirement. The single-wire implementation contributes to the low-cost and ease of implementation (less harness cabling), as well as the self-synchronization (no need for external oscillators). The low speeds help mitigate EMI and the deterministic nature of the communication scheme allows predictable messages frames. It also does not introduce any collisions in a well developed system.

Some disadvantages are related to the speed and *Commander-Responder* concept. Because the speed is slow, it is not ideal for any safety or other important systems inside the vehicle; the low bandwidth does not help with this either. Also, because the commander controls all communication on the bus, there is not a possibility for event-driven communication. Another serious problem with the *Commander-Responder* scheme, is that if the commander is lost, the entire cluster becomes useless because there is nothing there to drive the communication on the bus.

7 Conclusion

The *Local Interconnect Network* is in almost all automobiles today. It is a tremendous complement to CAN and has lowered the price, complexity, and space-constraint of the comfort systems inside cars. This allows for even more interesting and useful technologies to be added to the vehicle without needing to sacrifice price point, fuel efficiency, or user-experience.

8 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (February 2018) to Revision A (August 2022)	Page
• Changed all instances of legacy terminology to commander and responder where mentioned	1

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265