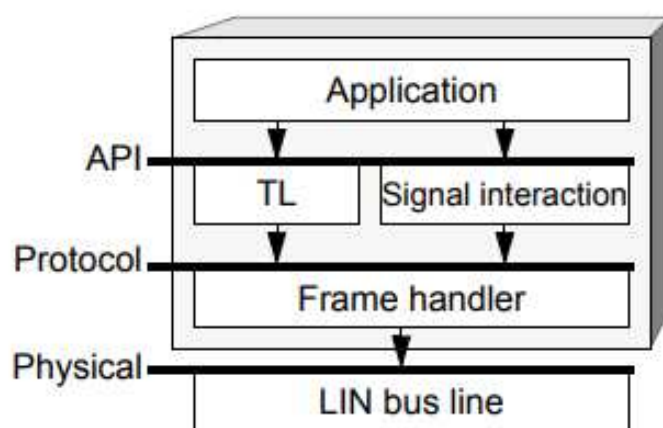


# ***LIN Protocol and Physical Layer Requirements***

Eric Hackett

## **ABSTRACT**

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



**Figure 1. LIN® Hierarchy Chart**

## **Contents**

1	Introduction .....	2
2	Network Architecture .....	3
3	Physical Layer Requirements .....	5
4	Filtering, Distance Limitations, Nodes on Bus .....	9
5	LIN® Transceiver Special Functions .....	11
6	Advantages and Disadvantages.....	13
7	Conclusion .....	14

## **List of Figures**

1	LIN® Hierarchy Chart .....	1
2	High-Level Application Diagram .....	2
3	LIN® Workflow .....	3
4	High-Level LIN® Transceiver in Network .....	3
5	LIN® Frame Header Explanation .....	5
6	LIN® Frame Response .....	5
7	Simplified LIN® Driver Schematic.....	6
8	Bus Signal Thresholds for Senders .....	7
9	Bus Signal Thresholds for Receivers .....	7
10	Bit Sampling Illustrated .....	8

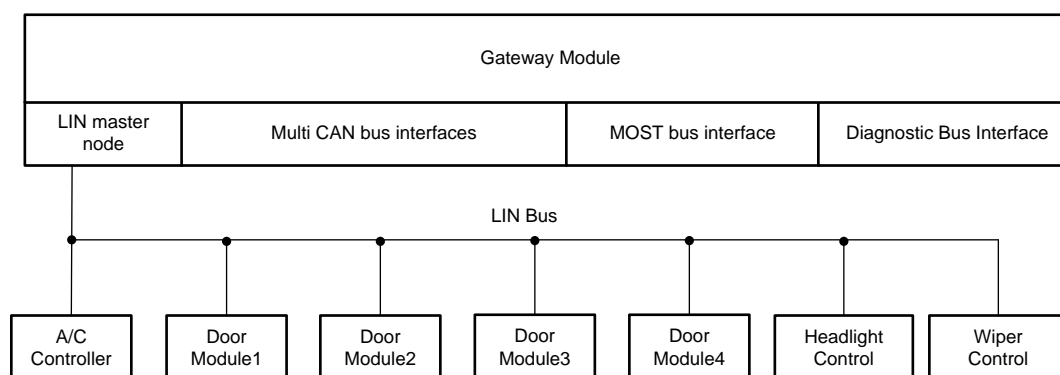
11	Bus Duty Cycle Requirement .....	9
12	LIN® Bus With 220 pF, 20 kbps Message .....	10
13	LIN® Bus With 10 nF, 20 kbps Message .....	11
14	LIN® Bus With 220 nF, 20 kbps Message .....	11
15	LIN® Wake-up Pattern .....	13

## Trademarks

LIN is a registered trademark of LIN Consortium.  
All other trademarks are the property of their respective owners.

## 1 Introduction

As automobiles continue to become more intelligent, safe, and comfortable, the amount of electrical systems and components continues to grow. With the addition of these components and systems comes a need for communication transceivers to facilitate their interaction in the most advantageous way possible for manufacturers. LIN was developed for this very reason: so car makers could manage communication between these components and systems in an efficient, 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 master nose here is configured as part of the gateway Module.

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

**Figure 2. 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 paper; however, it is important to have a high-level understanding of the whole 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 what 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.

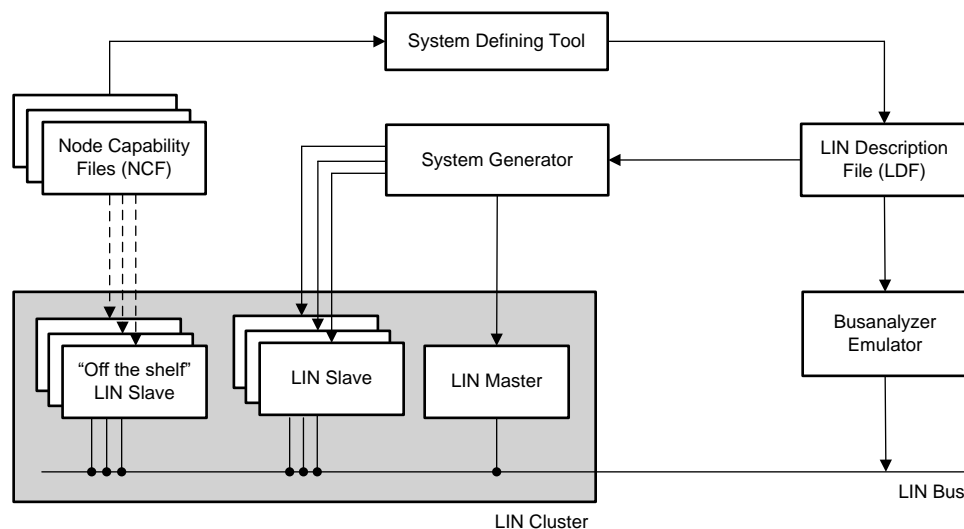


Figure 3. LIN® Workflow

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*, which 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 Master node and up to 16 subsequent Slave nodes. This master node, as can be assumed by the name, is what manages the communication along the bus to each slave. More detail on the *Master-Slave* principle is discussed in [Section 2.3](#).

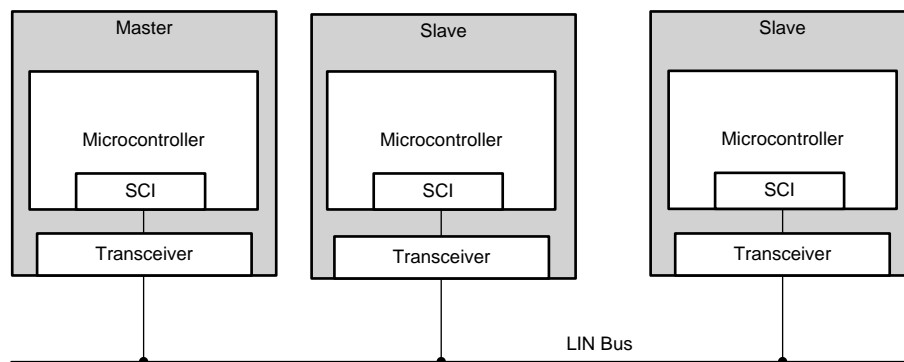


Figure 4. High-Level LIN® Transceiver in Network

Because the whole idea of LIN is to be a simple, cost-effective communication interface, a dedicated communication controller is not implemented. Rather a microcontroller is programmed with the LIN protocol, and used to drive the communication to the transceiver via the serial interface. This interface is called *Serial Communication Interface* (SCI) and took the place of UART in most LIN applications. Both are typical of most microcontrollers, which allows for 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 previously-mentioned SCI is the dominant interface used between LIN transceivers and the microcontrollers that communicate with them. UART was used originally, but fault-free interface via 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 Master-Slave Principle

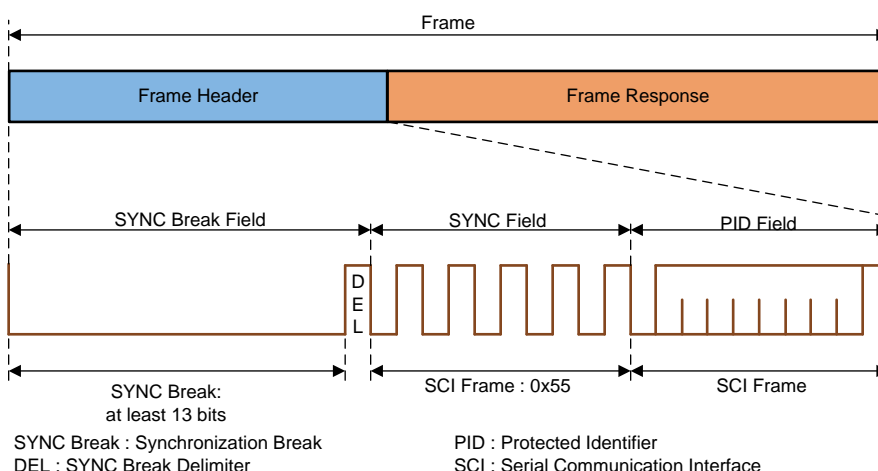
In every cluster, there is one master node, and up to 16 slave nodes. The master node controls all communication on the bus, and contains the master task and the slave task to be delivered. The slave nodes are unable to communicate with each other, contain only the slave task, and are only capable of responding to the master if the message is directed at them. The master sends out a request to a designated slave as a header (beginning of the frame), and the slave responds to the master as a response frame. There is also a case where the master sends the slave the header and response frame, and the slave only listens with no response. Both situations guarantee predictable, defined bus traffic, disallowing collisions almost always because the master 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 master at these given times set by the schedule. These tokens are sent to slaves, and the slave can 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 master-slave system is because the master controls all communication, if the master fails, the whole cluster fails. In schemes where all nodes can act as a master and slave, 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 slaves can only communicate with the bus if they are requested to do so (speak when spoken to!).

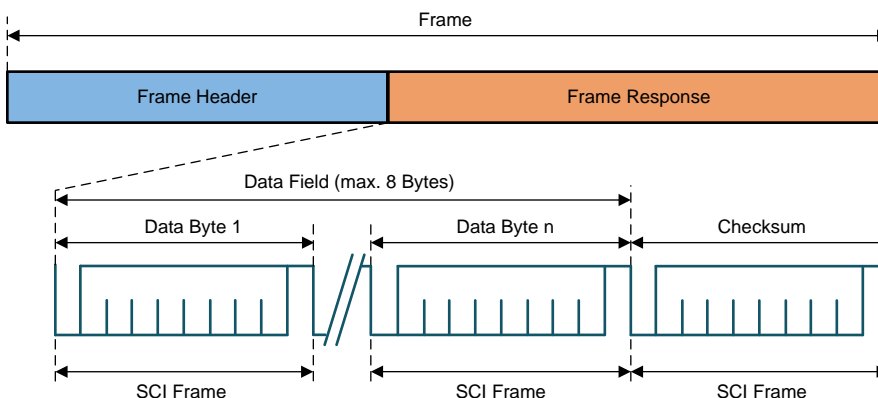
## 2.4 Message Frame Format

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 master task, and is divided up into the **sync break**, the **sync field**, and the **protected identifier** (PID). The sync break and sync field are used to have all the slaves on the LIN bus synchronized to the master timing (without the need of any crystal or oscillator), and the PID is what defines which slaves respond, receive, or ignore 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 5. LIN® Frame Header Explanation**

The Data (response) portion of the message is sent by the slave task, which can be sent by the master or the slave 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, to verify that the message sent is what was intended, and no errors were introduced during transmission. Any node can receive the frame response, 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).



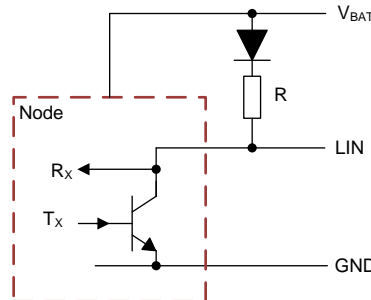
**Figure 6. LIN® Frame Response**

### 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 (master 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, typically operate at voltages ranging from 9 V to 18 V, but some 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 7. Simplified LIN® Driver Schematic**

### 3.1 Bus Signaling Fundamentals

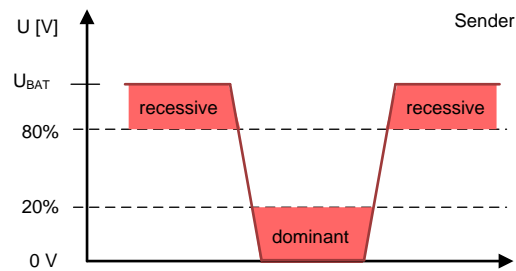
Considering these voltage levels on the bus, there are thresholds that most transceivers abide by, with some companies deviating slightly from the norm. These thresholds are based on a percentage of the battery voltage in the system, since it is a single-ended communication scheme and the thresholds cannot be set by a voltage differential. These thresholds determine when the bit is considered “recessive” and when it is considered “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 via 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”, and when the transceiver TXD begins conducting and becomes active, the bus is driven low, hence “dominant”.

### 3.2 Pullup Values

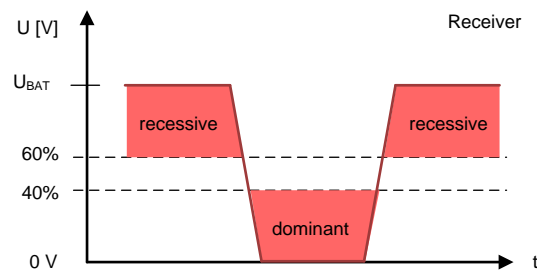
The pullup resistor values are different for the master and slave nodes. The master node requires an external pullup resistor and diode according to the LIN specification. The typical value seen is **1 kΩ** (other typical values are 600 Ω and 500 Ω) in series with a **diode** – for reverse-polarity protection – connected to the battery voltage. The typical pullup value of the LIN slave is 30 kΩ, and in all modern LIN transceivers, this is integrated within the IC, so no external pullup is necessary in the slave 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 8. Bus Signal Thresholds for Senders**



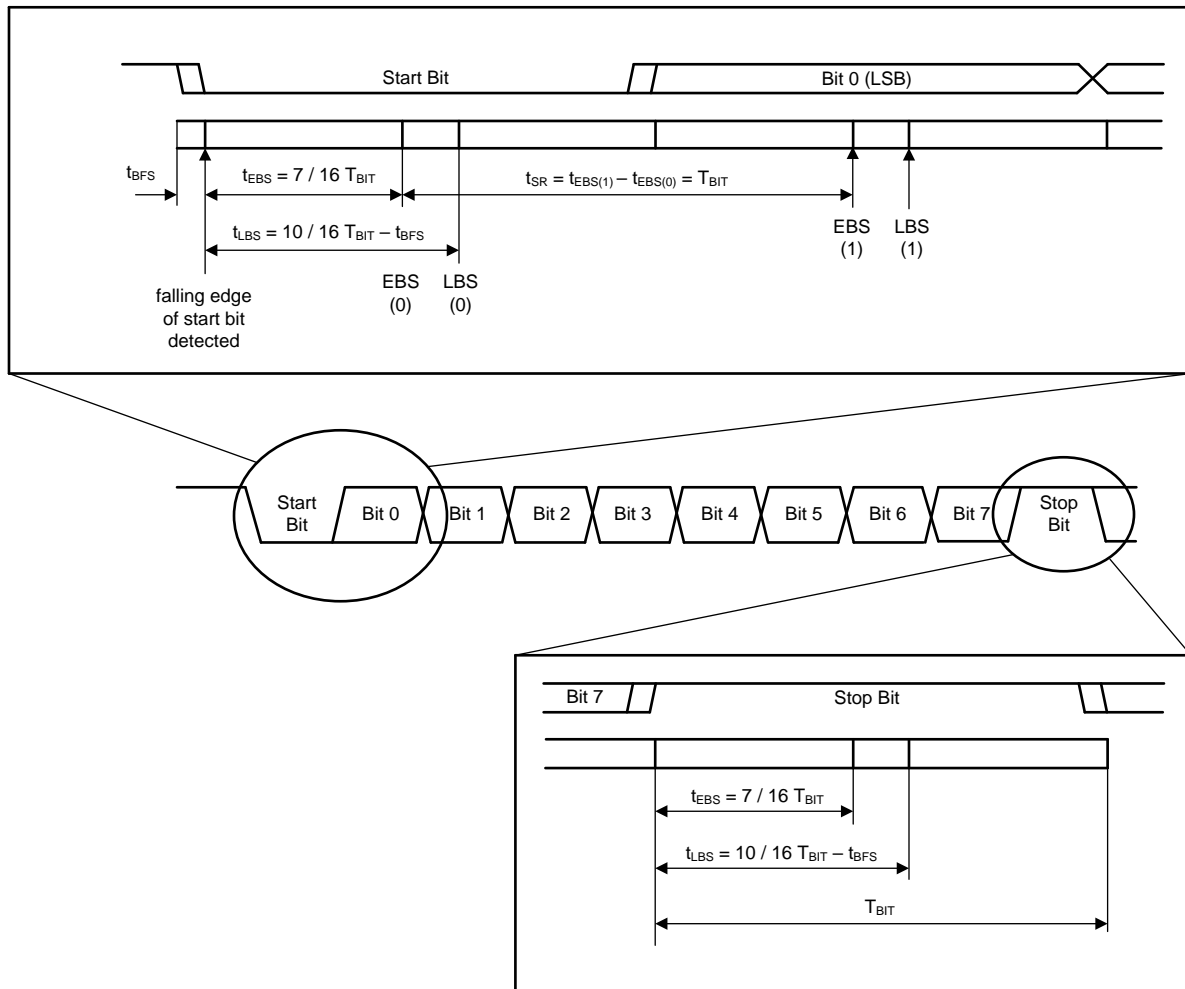
**Figure 9. Bus Signal Thresholds for Receivers**

### 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  $\pm 14\%$ . This 14% value comes from the fact that low-cost on-chip oscillators are used, and with internal calibration, better than  $\pm 14\%$  can be achieved. This accuracy allows for the 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 master 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, and combined with the start and stop bit (10 bits total), this allows for 4 total falling edges for synchronization of the slave 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 10. Bit Sampling Illustrated**

After synchronization of the slave 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](#):

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

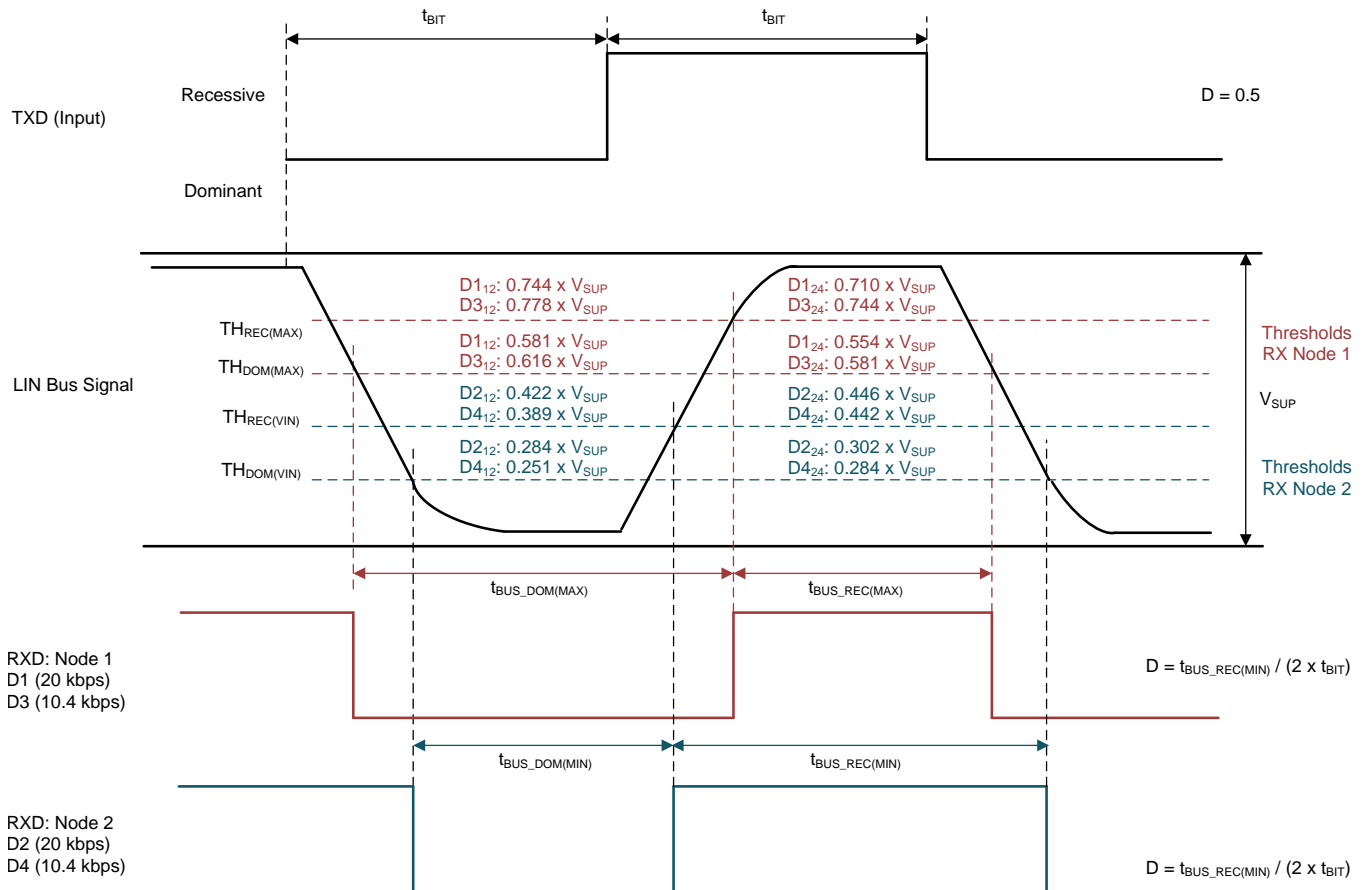
$T_{EBS}$  is defined as minimum  $7/16 T_{BIT}$ . The rest of the bits after the first bit are then sampled with a sample rate ( $t_{SR}$ ) based on the earliest bit sample of the former bit ( $n - 1$ ) and the earliest bit sample of the current bit ( $n$ ), and given by the following equation:

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

### 3.6 Duty Cycle

In order to send messages that are interpreted correctly, not only does the LIN bus have to meet the correct voltage levels based on the battery supply, these voltages have to be met within the correct bit sampling time of the receiver.





**Figure 11. Bus Duty Cycle Requirement**

Figure 11 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, this can also vary the master clock or slave clock, and thus affect 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 in order to mitigate any EMI issues (from or into the transceiver), as well as help with transient pulses and ESD strikes. Along with the LIN messages themselves radiating noise with rising and falling edges, and asymmetrical waveforms, noise from the rest of the vehicle can penetrate the LIN bus 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.

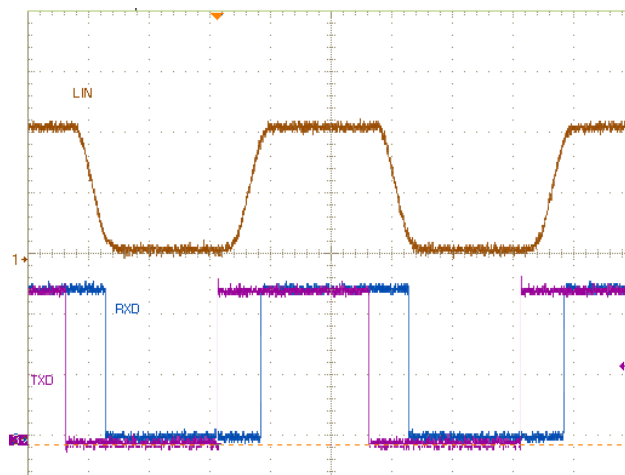
The bare minimum bus filtering consists of a shunt capacitor at the master and each slave node. Careful consideration has to be taken as to not overload the capacitance on the bus, as this slows down the edges too much and corrupts the interpretation of bits on the bus. Typical values for bus capacitance are 220 pF at the slave nodes, and up to ten times that value at the master 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

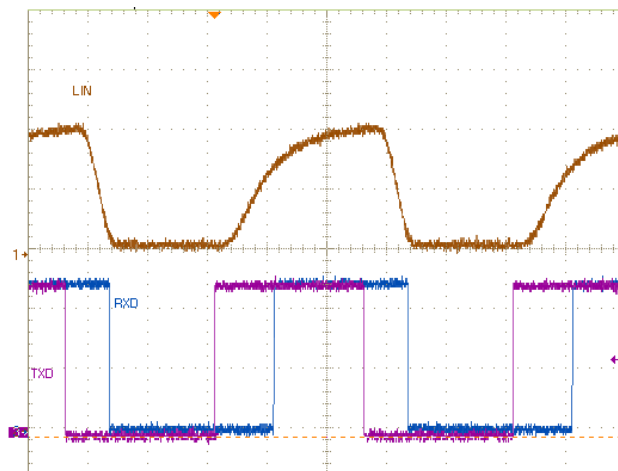
ESD strike and *Transient* pulse suppression are also important for the LIN bus. In any application, ESD strikes and transient pulses are present, but specifically in automotive environments, because there are so many ECUs in close proximity of each other. It is especially important for the systems to 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  $\pm 6$  kV for ESD direct contact, and  $\pm 100$  V for transient pulses.

## 4.3 Distance and Node Limitations

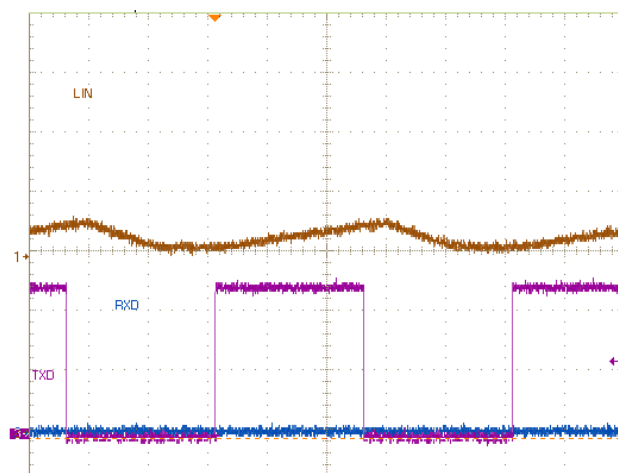
The LIN specification defines the maximum number of nodes that can be connected to a LIN bus: 1 master node and 16 slave nodes. Along with the definition of the max cable harness length: 40 m. Because of the definitions, there is not a need to worry about having too many nodes on the bus, or the cable being too long, as there is with 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, and 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 12](#), [Figure 13](#), and [Figure 14](#) 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 12. LIN® Bus With 220 pF, 20 kbps Message**



**Figure 13. LIN® Bus With 10 nF, 20 kbps Message**



**Figure 14. 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. How these are described 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 could be different 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

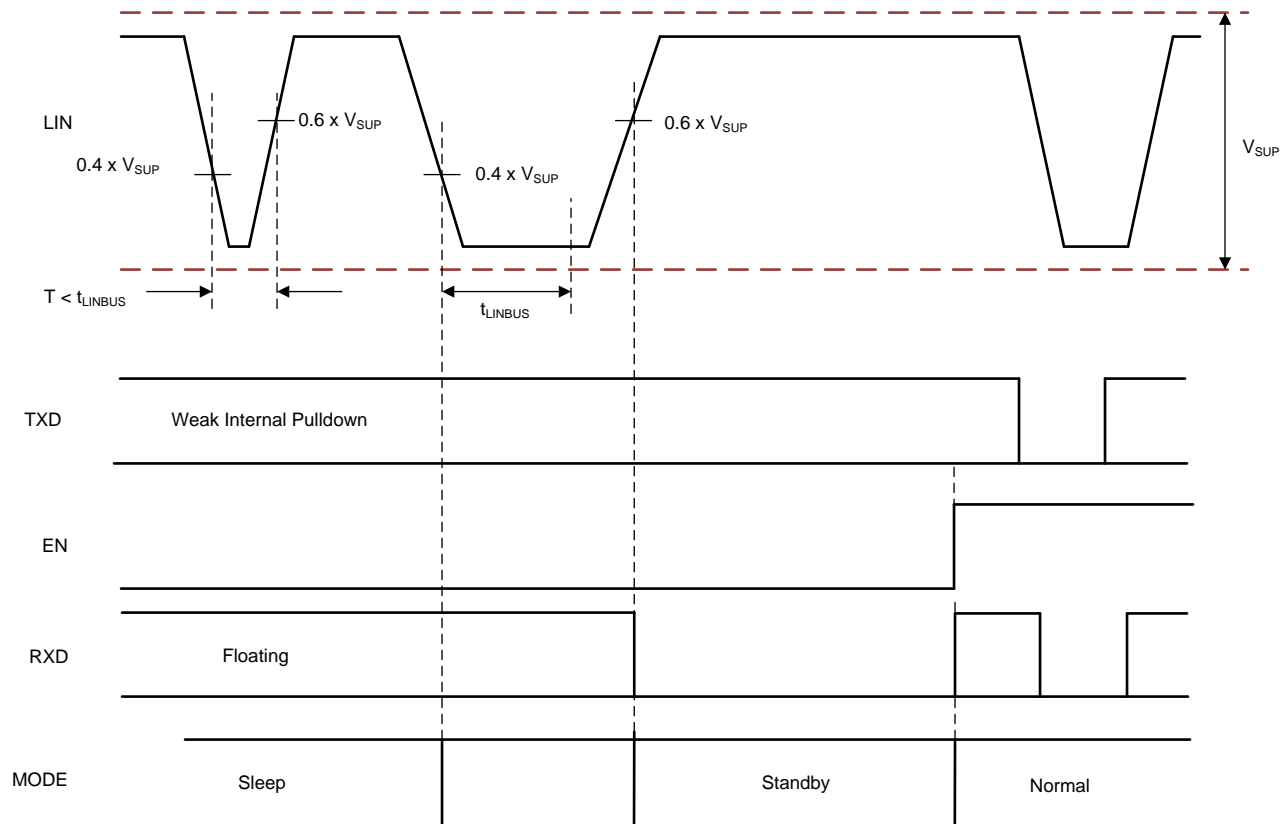
### 5.2.1 Pin Wakeup

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 15](#) shows a timing diagram of the procedure.



**Figure 15. 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, meaning 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  $\mu$ s, 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 and low-cost to implement and 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 for predictable messages frames, and introduces no collisions if the system is developed well.

Some disadvantages are related to the speed and *Master-Slave* 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. And because the master controls all communication on the bus, there is not the possibility for event-driven communication. The other big problem with the *Master-Slave* scheme is that if the master is lost, the whole cluster becomes useless, as 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, and user-experience.

## IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ("TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products (<http://www.ti.com/sc/docs/stdterms.htm>), [evaluation modules](#), and [samples](http://www.ti.com/sc/docs/sampterm.htm) (<http://www.ti.com/sc/docs/sampterm.htm>).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2018, Texas Instruments Incorporated