Application Note

NOVOSENSE

Protocol Layer and Physical Layer Requirements for LIN Transceivers

AN-13-0010

Author: Xiutao Lou, Lele Zhang

AN-13-00010

NOVOSENSE

Protocol Layer and Physical Layer Requirements for LIN Transceivers

ABSTRACT

As the development of intelligent vehicles accelerates, the number of electronic devices inside vehicles is gradually increasing, posing a challenge for communication among these in-vehicle electronic components. To reduce wiring harness complexity and lower communication costs in applications where performance, bandwidth, and complexity requirements are not stringent, the LIN (Local Interconnect Network) bus, as a complementary solution to the CAN bus, has been widely adopted. LIN transceiver functions as the interface between the LIN protocol controller and the LIN physical bus, and its performance determines the communication efficiency of the LIN bus, making it a critical component of LIN communication networks. This application note provides a detailed explanation of the protocol layer and physical layer requirements for LIN transceivers.

INDEX

1.Introduction

1.1.What is LIN?

The LIN (Local Interconnect Network) bus is a low-cost serial communication network defined for automotive distributed electronic systems. It serves as a complementary solution to other automotive multiplex networks like the Controller Area Network (CAN). With a maximum transmission rate of 20 kbps, LIN bus is primarily used in automotive applications where high-speed data transmission requirements and stringent safety requirements are not critical, such as lighting control, window control, seat adjustment, and windshield wiper control. The LIN bus is a low-cost serial communication protocol based on UART/SCI (Universal Asynchronous Receiver-Transmitter/Serial Communication Interface). Among automotive hierarchical networks, the LIN network has a relatively low architectural cost, and LIN standards have been well-established, which can significantly shorten development cycles. As a result, the LIN bus has experienced rapid growth and increasingly widespread adoption in the automotive industry.

1.2.History of LIN

In 1998, Audi, BMW, DaimlerChrysler, Volvo, Volkswagen, VCT, and Motorola jointly established the LIN Consortium (LIN-SUBBUS.ORG), aiming to define a unified bus communication standard. This standard was made publicly available for free use by developers. The first version of the LIN bus specification, LIN 1.0, was released by the LIN Consortium in July 1999. Subsequently, several revisions were made to the protocol (LIN 1.1 / LIN 1.2 / LIN 1.3).
In September 2003, the LIN Consortium released LIN 2.0, marking a major milestone in the evolution of the LIN protocol. This version involved two significant changes – bus diagnostics and configuration as standard features, and "designated node capability" of the LIN bus, which were designed to simplify network nodes while maintaining backward compatibility with earlier LIN versions.

LIN 2.1 was released in November 2006. It clarified certain aspects of the specification, improved the interpretation of LIN 2.0, corrected parts of the Configuration section, and described the transport layer and diagnostic layer in separate chapters. In December 2010, the LIN Consortium released LIN 2.2, which relaxed the bit sampling specifications in the physical layer. The subsequent LIN 2.2A revision corrected the wake-up signal definition in LIN 2.2, which remains the widely adopted version today.

In 2012, based on LIN 2.0, the Society of Automotive Engineers (SAE) standardized LIN as SAE J2602.

In 2016, LIN was further standardized by the International Organization for Standardization (ISO) under the designation ISO 17987:2016, ISO - Advanced search. However, following ISO's adoption, the LIN specification was no longer offered free of charge.

1.3.Characteristics of the LIN Bus

As a cost-effective serial communication bus, the LIN bus has the following characteristics:

（1）Based on universal UART/SCI hardware interface and compatible with the vast majority of microcontrollers, reducing system costs;

（2）Single-master/multi-slave architecture that eliminates the need for arbitration mechanisms, where data priority is determined by the master node's scheduling table;

（3）Slave nodes do not require quartz or ceramic oscillators, and achieve self-synchronization through the sync field in the frame header sent by the master node, saving hardware costs of slave nodes;

（4）Single-wire transmission with very low physical layer implementation cost;

（5）Maximum transmission rate up to 20kbps;

（6）Two bus level states: Dominant (“0”) and Recessive (“1”);

（7）Maximum of 16 nodes supported;

（8）Explicit signal transmission entity allows accurate calculation of signal transmission time, providing strong network predictability;

（9）LIN provides four functions: signal processing, configuration, identification and diagnosis;

（10）Low safety level: If the master node fails, the entire LIN bus will fail, making LIN unsuitable for safety-critical applications.

1.4.Network Architecture of LIN Bus

In vehicles, LIN networks are typically not used in isolation. Instead, they are often connected to upper-layer networks (e.g., CAN) to form CAN/LIN hybrid networks. This architecture leverages the respective advantages and characteristics of both protocols to minimize costs while ensuring system reliability. Figure 1. 1 provides a simplified illustration of a CAN/LIN hybrid network using headlight control as an example. For such hybrid networks involving two different bus protocols, a CAN/LIN gateway is required to perform mutual conversion of messages between the different protocols. In this example, the CAN/LIN gateway serves both as a CAN node and a LIN master node.

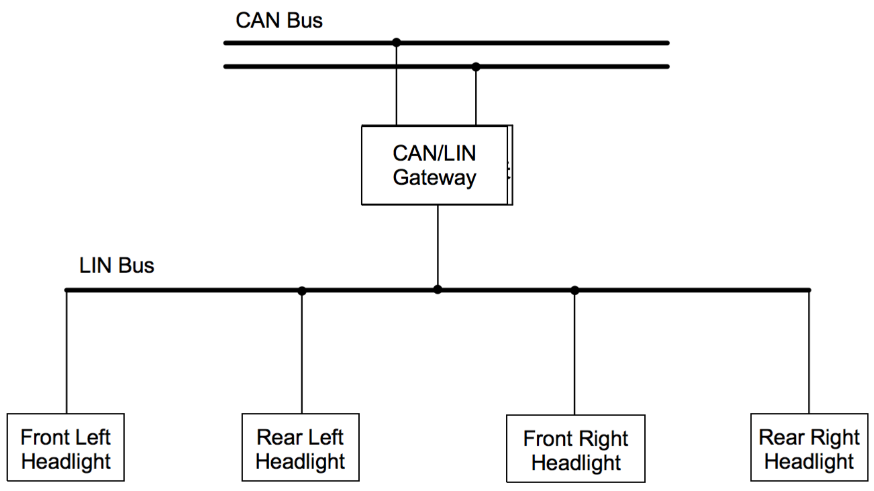


Figure 1. 1 CAN/LIN Hybrid Network

For the CAN/LIN gateway node, three primary functions are performed: receiving information from the network in front of the gateway; translating and interpreting the received information; and sending information to the network behind the gateway, as shown in Figure 1. 2. This process is reversible. The main controller is the core component of the gateway, and its performance directly determines the efficiency of the gateway.

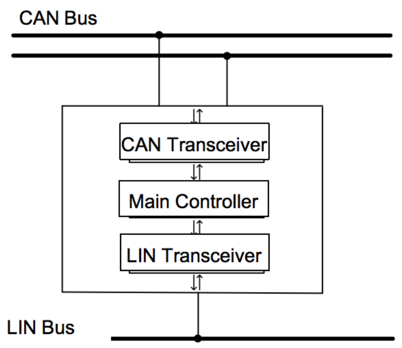


Figure 1. 2 Schematic Diagram of CAN/LIN Gateway Node

The LIN bus adopts a single-master/multi-slave network architecture, consisting of one master node and several slave nodes (theoretically no more than 16), as illustrated in Figure 1. 3. The master node on the LIN bus contains both a master task and a slave task, whereas slave nodes only contain a slave task. Communication on the bus is initiated by the master task by transmitting a frame header. This communication can occur between the master and slave nodes or among slave nodes. Slave tasks respond to the frame headers transmitted by the master task, as shown in Figure 1. 4. Therefore, the master node controls all communications in the LIN network, eliminating signal collisions and thus the need for arbitration mechanisms. The master node stores all configuration information for the entire network, allowing slave nodes to freely connect or disconnect from the LIN network without affecting other LIN nodes in the network.

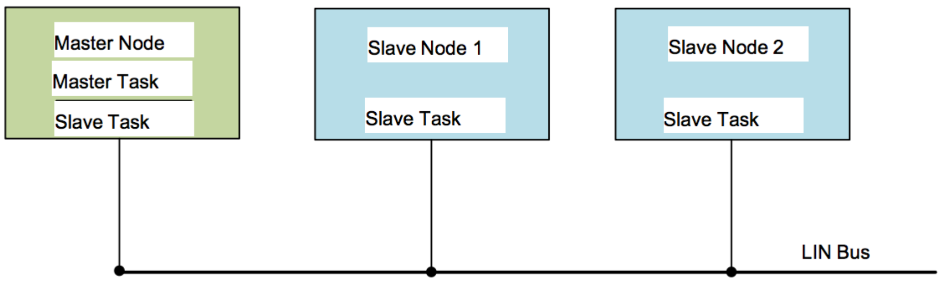


Figure 1. 3 Network Architecture of LIN Bus

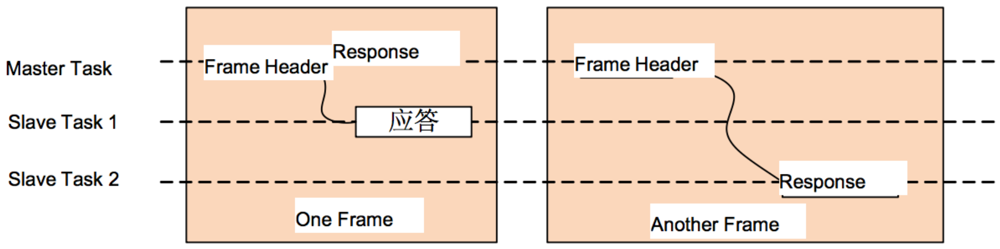


Figure 1. 4 Frame Transmission on the Bus

2.LIN Protocol Layer

2.1.Frame Structure

The entity transmitted over the LIN bus is a message frame, which consists of a frame header transmitted by the master task and a response transmitted by the slave task, as illustrated in Figure 2. 1. The frame header comprises a Sync Break Field, a Sync Field, and a Protected Identifier (PID) Field; the response consists of a Data Field and a Checksum Field. There is a Response Space between the frame header and the response, an Inter-frame Space between frames, and an Inter-byte Space within the data field to separate the data bytes. On the LIN bus, “0” represents a dominant level, while “1” indicates a recessive level. The bus operates on a “wired AND” principle: if one or more nodes transmit a dominant level, the bus will be at a dominant level; the bus will only be at a recessive level when all nodes transmit recessive levels. Therefore, the dominant level prevails on the LIN bus.

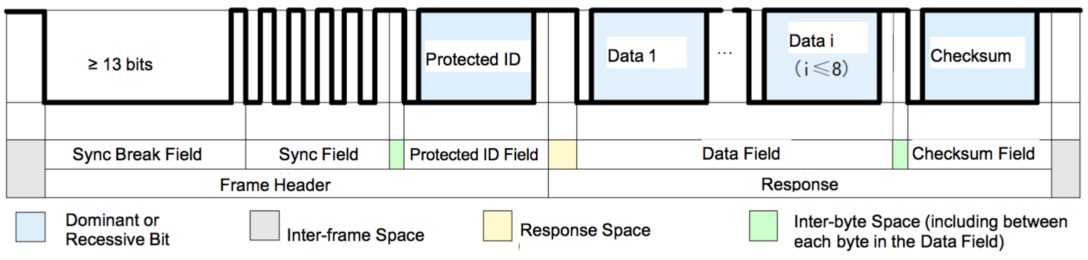


Figure 2. 1 Frame Structure of LIN

Except for the Sync Break Field, other fields in the frame structure are transmitted in a byte field format, as shown in Figure 2. 2. Data transmission starts with the Least Significant Bit (LSB) and ends with the Most Significant Bit (MSB), totaling 10 bits (including start and stop bits).

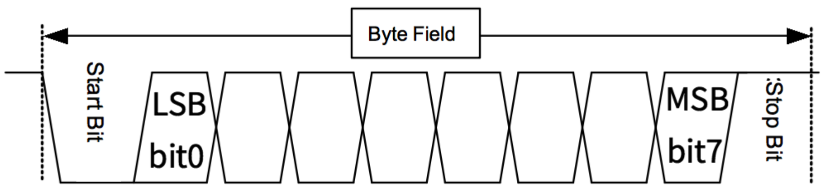


Figure 2. 2 Byte Field Structure

2.1.1. Sync Break Field

The Sync Break Field signifies the start of a new message frame transmission and consists of two components: the Sync Break and the Sync Break Delimiter, as illustrated in Figure 2. 3. The Synch Break requires at least 13 dominant bits, while the Sync Break Delimiter needs at least one recessive bit.

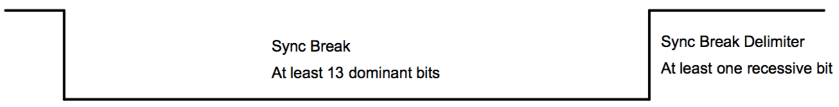


Figure 2. 3 Sync Break Field

2.1.2. Sync Field

In LIN networks, the master node utilizes a high-precision clock (such as quartz or ceramic oscillators), whereas slave nodes use lower-precision and lower-cost clocks. To achieve synchronization of bit rate between the slave and master nodes, the Sync Field is used. This field employs the byte value 0x55 (binary 01010101b), alternating between 1s and 0s, as shown in Figure 2. 4. LIN synchronization is determined by the falling edge, and slave nodes calculate the master node's clock frequency using the following formula and adjust their own clocks accordingly:

$$\text{Bit Time} = \frac{\text{Time of Falling Edge on Bit 7} - \text{Time of Falling Edge on Start Bit}}{8}$$

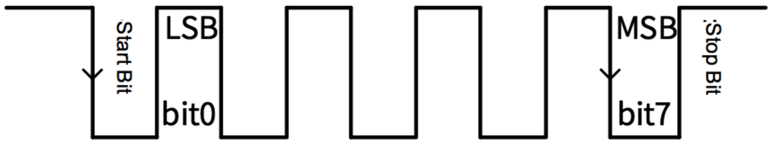


Figure 2. 4 Sync Field

2.1.3. Protected ID Field

The Protected ID Field, also known as PID Field, includes a start bit, a 6-bit Frame ID, 2 parity bits, and a stop bit, as depicted in Figure 2. 5.

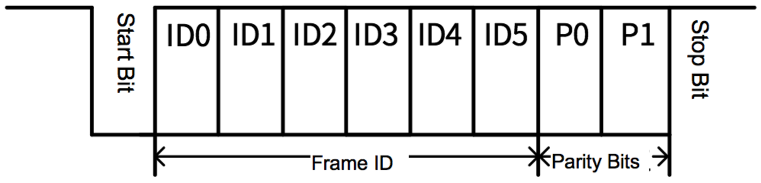


Figure 2. 5 Protected ID Field

The Frame ID (ID0-ID5) ranges from 0x00 to 0x3F (0 to 63), totaling 64 IDs. The frame type is directly related to the Frame ID, and there are three frame types, as shown in Table 2. 1. Additionally, the data length in the Data Field is determined by ID4/ID5 of the Frame ID, which can be 2/4/8 bytes, as illustrated in Table 2. 2.

Frame Type		Frame ID
Signal-Carrying Frame	Unconditional Frame	0x00 ~ 0x3B
	Event-Triggered Frame	
	Sporadic Frame	
Diagnostic Frame	Master Request Frame	0x3C
	Slave Response Frame	0x3D
Reserved Frame		0x3E, 0x3F

Table 2. 1 Relationship Between Frame Types and Frame IDs

ID4/ID5	Data Field Byte Length	Frame ID Range
0,0	2	0x00 ~ 0x1F
1,0	2	
0,1	4	0x20 ~ 0x2F
1,1	8	0x30 ~ 0x3F

Table 2. 2 Relationship Between Data Field Length and Frame ID

The parity checksum is calculated based on the Frame ID using the following algorithm:

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4$$

$$P1 = \neg (ID1 \oplus ID3 \oplus ID4 \oplus ID5)$$

Where:

“ $\oplus$ ” represents the XOR (exclusive OR) operation; “¬” represents the NOT (bitwise negation) operation. Therefore, under normal communication conditions, the PID Field should not contain the values 0x00 or 0xFF. If these values do appear, it indicates a transmission error.

2.1.4. Data Field

The Data Field contains up to 8 bytes of data, with the lowest-numbered byte transmitted first, as shown in Figure 2. 6. Each individual byte consists of 10 bits, including a start bit, 8 data bits, and a stop bit. There are two data types in the Data Field – signals and diagnostic messages. Signals are transmitted by signal-carrying frames and diagnostic messages are transmitted by diagnostic frames.

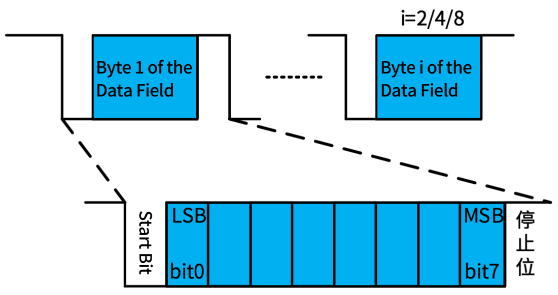
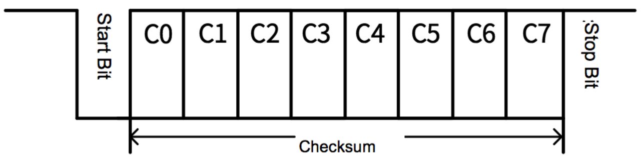


Figure 2. 6 Data Field

2.1.5. Checksum Field

The checksum field is used to verify the correctness of the transmitted frame content, as illustrated in Figure 2. 7. The checksums are divided into two types: standard checksum and enhanced checksum, managed by the master node and determined by the frame ID. Diagnostic frames (0x3C and 0x3D) use the standard checksum, while other frame IDs use the enhanced checksum. The standard checksum verifies all bytes in the data field, whereas the enhanced checksum verifies all bytes in both the data field and the PID field.



Checksum Type	Verification Objects	Applicable Frame Types
Standard Checksum	All bytes in the data field	Diagnostic frames (0x3C and 0x3D)
Enhanced Checksum	All bytes in the data field and PID field	All frames except diagnostic ones

Figure 2. 7 Checksum Field

The checksum calculation method is as follows: The sender performs a binary addition with carry on all corresponding bytes of the verification object (subtracting 255 if the result is greater than or equal to 256). The sender then performs a bitwise NOT operation on the final sum to obtain the checksum value. The receiver, based on the checksum type and the received data, performs the same binary addition with carry but without inverting the result. The resulting value is added to the received checksum. If the sum equals 0xFF, the checksum is correct. The LIN protocol ensures data transmission accuracy using this method.

2.2.Frame Transmission Time

Figure 2.1 provides the schematic diagram of a single frame. Without considering the inter-frame space, response space, and inter-byte space, the minimum frame transmission time is illustrated in Figure 2. 8, and the calculation formulas are as follows:

$$T_{FHmin} = 34bit,\ T_{response\ min} = (i+1)*10bit$$

Typically, there is a 40% margin for both the frame header time and response time, so the maximum transmission time is calculated as follows:

$$T_{FH\ max} = 1.4*\ T_{FH\ min},\ T_{response\ max} = 1.4*\ T_{response\ min}$$

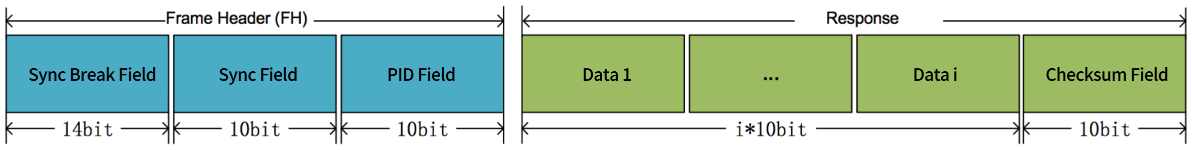


Figure 2. 8 Minimum Frame Length

2.3.Frame Types

As mentioned in Table 2. 1 in Section 2, different frame IDs correspond to different frame types. The frame type is a prerequisite for efficient frame transmission. In a network however, it is not necessary to use all frame types.

2.3.1. Unconditional Frames

Unconditional frames have frame IDs ranging from 0x00 to 0x3B (0 to 59). These frames are transmitted in the frame slots allocated by the master node, with the frame header always being sent by the master node. Once a frame header is transmitted on the bus, a response from a slave task must follow. This response can be communication between the master node and a slave node or between slave nodes. Three typical examples of unconditional frames are shown in Figure 2. 9:

（1）Frame ID = 0x30: After the master node sends the frame header, Slave Node 1 responds to the master node, meaning that Slave Node 1 acts as the publishing node, and the master node is the receiving node.
（2）Frame ID = 0x31: After the master node sends the frame header, the master node responds to both Slave Node 1 and Slave Node 2, meaning that the master node acts as the publishing node, and both Slave Node 1 and Slave Node 2 are receiving nodes.
（3）Frame ID = 0x32: After the master node sends the frame header, Slave Node 2 responds to Slave Node 1, meaning that Slave Node 2 acts as the publishing node, and Slave Node 1 is the receiving node.

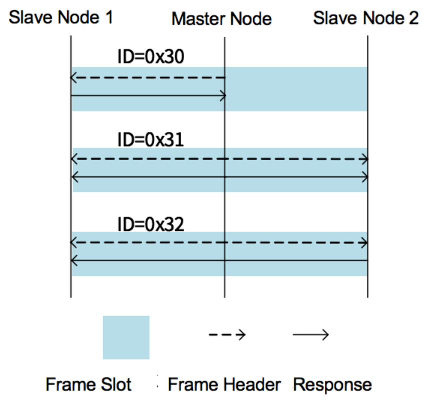


Figure 2. 9 Unconditional Frames

2.3.2. Event-Triggered Frame

If the master node needs to continuously poll signals from all slave nodes, but the signal update rate of slave nodes is low, using unconditional frames would occupy excessive bandwidth. To address this, the LIN protocol introduces event-triggered frames. An event-triggered frame is used by the master node within a predefined frame slot to query whether any slave node's signal has changed during that frame time.

An event-triggered frame is associated with one or more unconditional frames, and only sends data when the signal of associated unconditional frames is updated. Therefore, it allows no response from the slave nodes. If multiple slave nodes respond simultaneously, the master mode uses a predefined schedule table during the development stage to resolve conflicts.

Each event-triggered frame is equipped with a collision resolution schedule table, which is automatically switched by the master node's driver. The collision resolution schedule table becomes active in the next frame slot following the collision. Once the conflict is resolved, the master node switches back to the previous schedule table, and continues execution from the point immediately after the entry that caused the conflict.
The unconditional frames associated with an event-triggered frame must meet the following five conditions:
（1）Same number of bytes in the Data Field;
（2）Same checksum type used;
（3）The first byte of the Data Field must be the PID of the corresponding unconditional frame, allowing easy identification of the unconditional frame that has provided the response;
（4）Transmitted by different slave nodes;
（5）Must not be in the same schedule table as the event-triggered frame.

Figure 2. 10 illustrates three typical examples of event-triggered frames. One schedule table contains only one event-triggered frame with Frame ID = 0x10. This frame is associated with two unconditional frames having Frame IDs = 0x11 (Slave Node 1) and 0x12 (Slave Node 2). These unconditional frames are also included in the collision resolution schedule table.

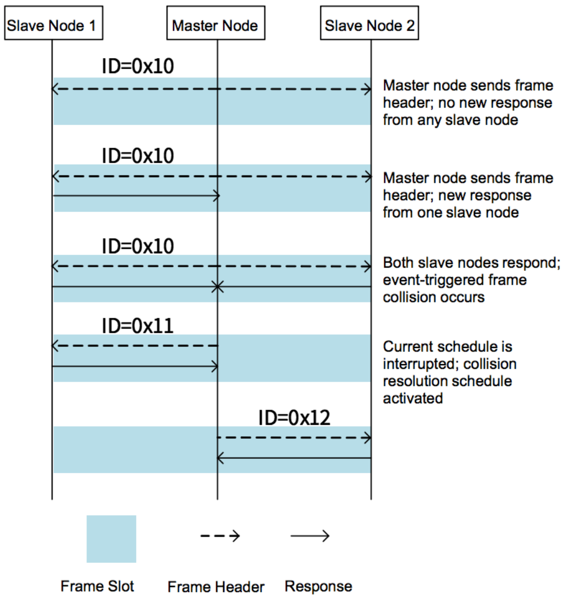

Slave Node 1
Master Node
Slave Node 2
ID=0x10
Master node sends frame header; no new response from any slave node
ID=0x10
Master node sends frame header; new response from one slave node
ID=0x10
Both slave nodes respond; event-triggered frame collision occurs
ID=0x11
Current schedule is interrupted; collision resolution schedule activated
ID=0x12
Frame Slot
Frame Header
Response

Figure 2. 10 Event-Triggered Frame

2.3.3. Sporadic Frame

A sporadic frame is sent by the master node in a specific frame slot when its own signal changes. In this frame type, only the master node can act as the publisher. A sporadic frame is also associated with a set of unconditional frames, and since the master node knows the priority of each unconditional frame in advance, no transmission conflicts will occur.

（1）When no signal change occurs in associated unconditional frames, the frame slot remains silent, and the master node does not even send a frame header, as shown in Figure 2. 11.

（2）When one of the associated unconditional frames experiences a signal change, the master node acts as the publisher and sends the updated signal to the corresponding slave node(s), as illustrated in Figure 2. 11.

（3）When two or more associated unconditional frames contain changed signals, arbitration is performed based on predefined priorities. The frame with the highest priority responds first, while the lower-priority frame responses wait until the sporadic frame header appears.

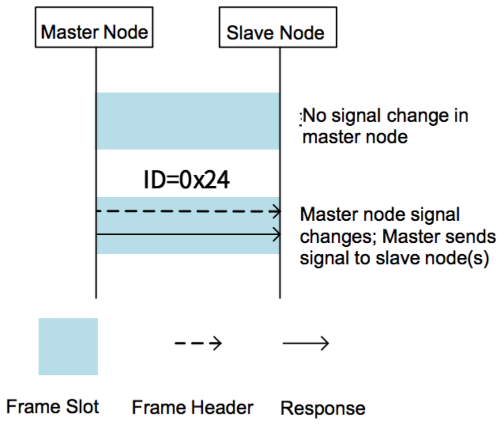

Master Node
Slave Node
No signal change in master node
ID=0x24
Master node signal changes; Master sends signal to slave node(s)
Frame Slot
Frame Header
Response

Figure 2. 11 Sporadic Frame

2.3.4. Diagnostic Frame

There are two diagnostic frame types – master request frame (Frame ID = 0x3C) and slave response frame (Frame ID = 0x3D). Both frame types contain a fixed 8-byte data field, and always use the standard checksum. These frames are used for configuration, identification, and diagnosis.

2.3.5. Reserved Frame

Reserved frames have frame IDs of 0x3E and 0x3F. These are reserved for future expansion of the LIN protocol.

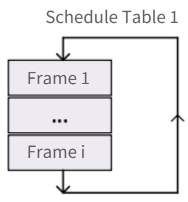
2.4.Schedule Table

A key feature of the LIN protocol is the use of schedule tables, which define the transmission order and timing of frames on the LIN bus. Schedule tables ensure that the bus is never overloaded, and are crucial for guaranteeing signal periodicity. Schedule tables reside in the master node, which is responsible for initiating them. A LIN network can have multiple schedule tables.

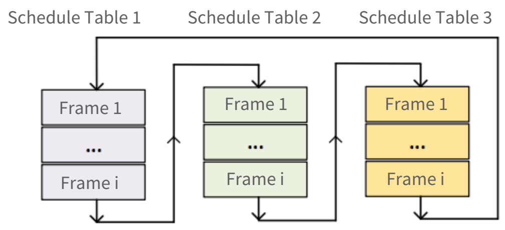
（1） For single schedule table in loop execution, if no new schedule table is triggered during execution, the current schedule table runs sequentially from the first frame to the last, then loops back to the first frame and repeats continuously, as illustrated in Figure 2. 12 (a).

（2） Multiple schedule tables can be executed in a predefined order and cyclically, as shown in Figure 2. 12 (b).

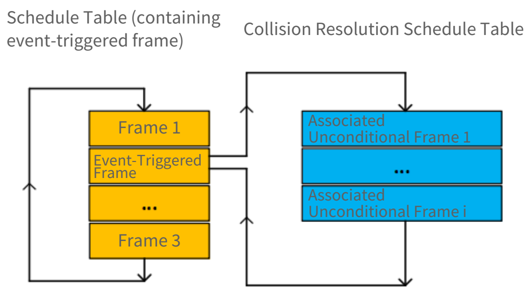
（3） If an interrupt occurs during the execution of one schedule table, another schedule table can be temporarily executed, as depicted in Figure 2. 12 (c).



（a） Cyclic Execution of Single Schedule Table



（b） Sequential Cyclic Execution of Three Schedule Tables


Schedule Table (containing event-triggered frame)
Collision Resolution Schedule Table
Frame 1
Event-Triggered Frame
...
Frame 3
Associated Unconditional Frame 1
...
Associated Unconditional Frame i

（C） Collision Resolution

Figure 2. 12 Illustration of Schedule Tables

2.5.State Machine

2.5.1. Master Task State Machine

When the master node publishes a frame header onto the bus, the master task sequentially sends the Sync Break Field, the Sync Field, and the PID Field, as shown in the figure below.

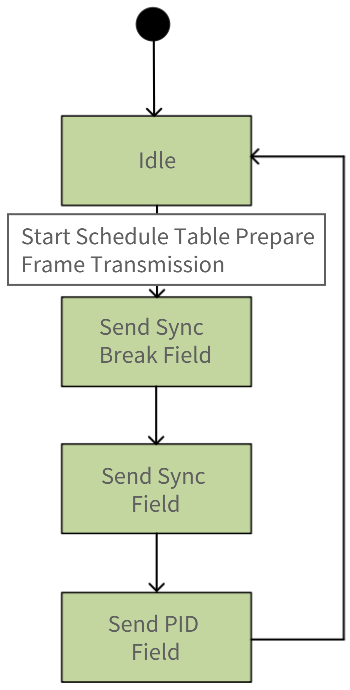


Figure 2. 13 Master Task State Machine

2.5.2. Slave Task State Machine

When acting as a publisher, the slave task is responsible for sending frame's responses; when acting as a receiver, it is responsible for receiving response frames. Two state machine models are used:

（1） Sync Break Field & Sync Field Sequence Detector

Any node should recognize these fields at any state. The start of a new frame can be detected through the Sync Break Field. The Sync Field helps keep the slave node synchronized with the master node's baud rate.

（2） Frame Processor

The frame processor has two main states: Idle State and Active State. The Active State is further divided into five sub-states: Receive and Analyze PID, Receive Data, Receive Checksum, Send Data, and Send Checksum. Upon receiving the sync break and sync fields (regardless of the current state or sub-state), the system transitions into the PID sub-state under the Active State.

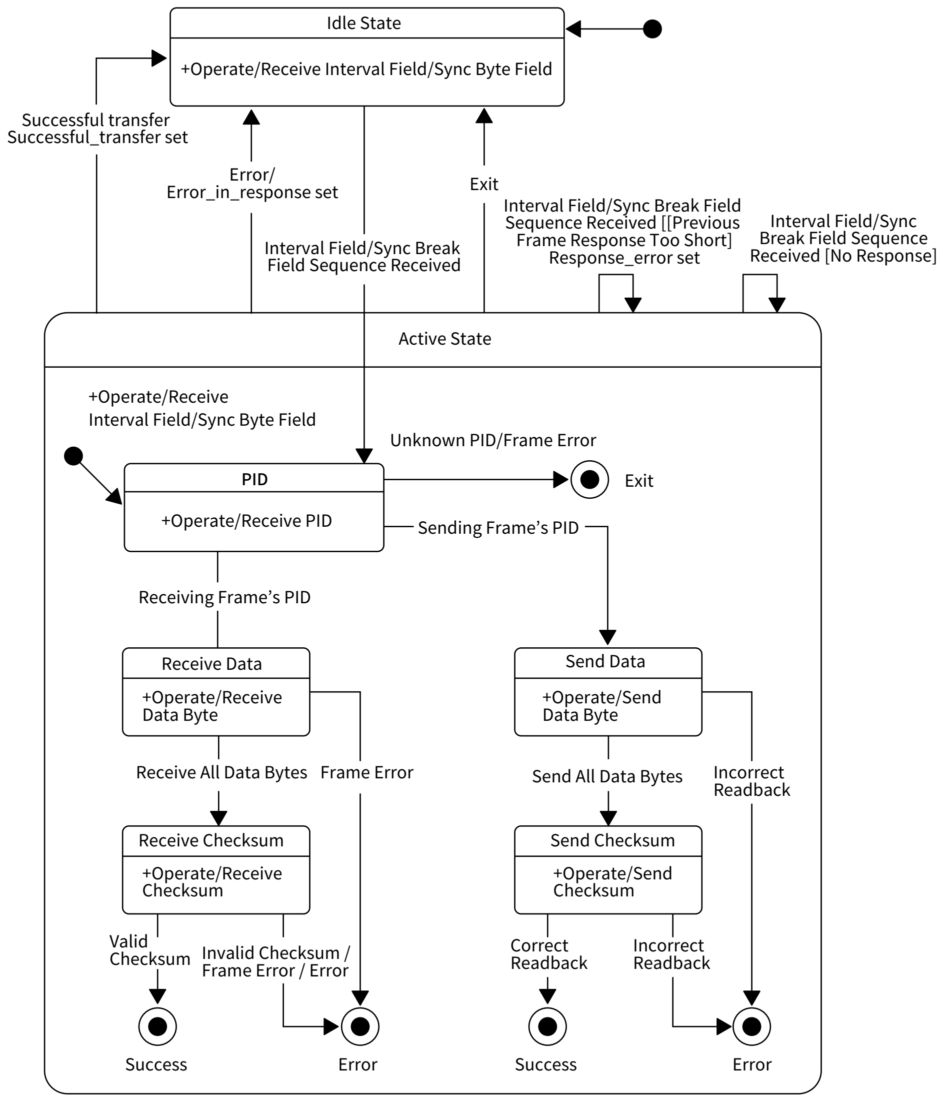


Figure 2. 14 Slave Task Frame Processor State Machine

2.6.Error Types

While the LIN standard does not fully specify all possible errors, common error types are listed below:

（1）Bit Error: It occurs when the publisher node detects a mismatch between the data on the bus and the transmitted data.

（2）Bus Error: It occurs when the bus is shorted to power source or ground, disrupting normal communication.

（3）Sync Field Error: It occurs when a slave node, based on the sync field, detects that the bit rate deviation is beyond tolerance.

（4）Parity Bit Error: It occurs when the parity calculated by the receiving node based on the received PID does not match the received value.

（5）Checksum Field Error: The receiver performs a binary addition with carry based on the checksum type and the received data, without inverting the result. The checksum field error occurs when the sum of the resulting value and the received checksum is not equal to 0xFF.

（6）No Response Error: It occurs when no response is detected on the bus following the transmission of a frame header (except event-triggered frames) by the master node.

（7）Frame Error: It occurs when a stop bit in a byte field shows a dominant level.

（8）Incomplete Response Error: It occurs when the data field received by the receiving node is incomplete data or has missing checksum field.

2.7.Sleep and Wake-Up

2.7.1.Sleep

In a LIN network, there are two methods for entering sleep mode:

（1）Sending a Host Request Frame (0x3C) with the first byte of the Data Field set to 0x00 and the remaining seven bytes as 0xFF.

Data Field Byte 1	Data Field Byte 2	Data Field Byte 3	Data Field Byte 4	Data Field Byte 5	Data Field Byte 6	Data Field Byte 7	Data Field Byte 8
0x00	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

（2）A LIN node automatically enters sleep mode if the bus remains inactive (no switching between dominant and recessive levels) for 4s-10s.

2.7.2. Wake-Up

When a LIN network enters sleep mode, both the master node and slave nodes can send a wake-up signal on the LIN bus to wake up the LIN network. The dominant duration of the wake-up signal must be between 250 µs and 5 ms, as shown in Figure 2. 15. If no frame header is detected on the bus within 150-250 ms after sending the wake-up signal, the node will retransmit the wake-up signal. It is import to note that the wake-up signal can be sent up to three times. If no frame header is detected after three attempts, the node must wait at least 1.5s before attempting to send another wake-up signal, as illustrated in Figure 2. 16.

After the wake-up signal is transmitted, the dominant threshold for the remaining nodes is 150 µs. In addition, both the master node and slave nodes must be ready to communicate within 100 ms after the rising edge of the wake-up signal.

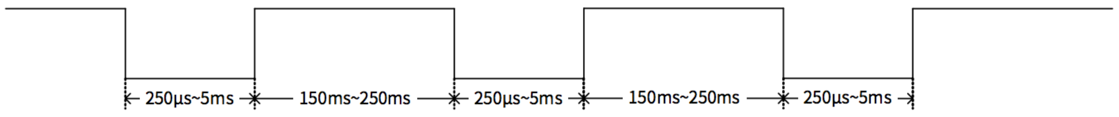


Figure 2. 15 Wakeup Signal Waveform

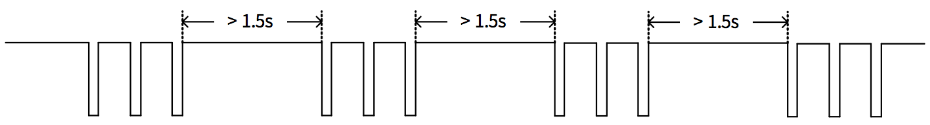


Figure 2. 16 Wakeup Re-attempt Intervals

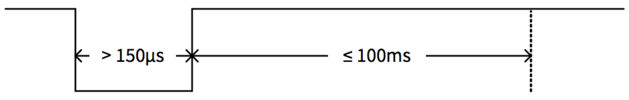


Figure 2. 17 Wakeup Signal Detection and Initialization

3.LIN Physical Layer

3.1.Physical Layer Hardware

Figure 3. 1 illustrates the physical connection of a node on the LIN bus, which consists of three main components: the protocol controller, the LIN bus transceiver, and the LIN bus itself.

（1）Protocol Controller

The protocol controller is typically a microcontroller with SCI/UART, operating in half-duplex mode. The protocol controller should perform both "Transmit" and "Receive" functions: for the "Transmit" function, it converts parallel binary data into serial high/low-level signals for the bus transceiver; for the "Receive" function, it receives serial high/low-level signals from the bus transceiver and converts them back into parallel binary data.

（2）LIN Bus Transceiver

The LIN bus transceiver serves as the interface between the protocol controller and the physical bus, with maximum data transmission rate of up to 20 kbps. TXD pin receives digital high/low-level signals from the protocol controller and converts them into dominant/recessive bus levels. RXD Pin converts the dominant/recessive levels on the LIN bus into high/low-level signals for the protocol controller.

（3）LIN Bus

The LIN bus is the physical communication medium connecting all LIN nodes in the network. It is typically implemented using a copper wire.

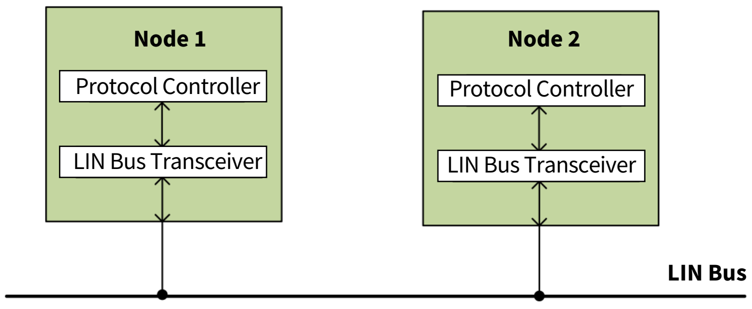


Figure 3. 1 LIN Node Hardware Diagram

3.2.LIN Bus Transceiver

In the bus transceiver design, the TXD and RXD pins are directly connected with the protocol controller, usually operating at logic levels of 3.3V/5V. The power supply for the transceiver, referred to as V_{BAT}, can be 12V and 24V, corresponding to the vehicle battery voltage.

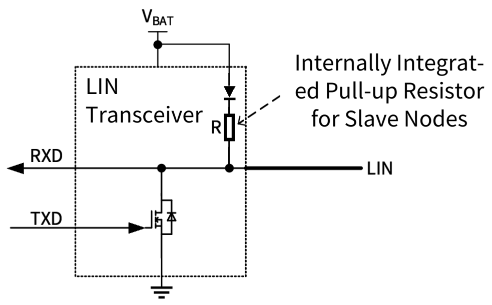


Figure 3. 2 Schematic Diagram of LIN Transceiver Model

3.2.1. Pull-Up Resistor

In a LIN system, the bus transceiver of the master node must connect its LIN port to VBAT or INH through a diode and a 1kΩ resistor in series. The pull-up resistor for slave nodes has a resistance value of 30 kΩ, and most LIN bus transceivers currently available integrate a 30 kΩ pull-up resistor required by slave nodes, as shown in Figure 3. 3.

The application considerations regarding pull-up resistors are described below:

（1）The diode is mandatory, and it prevents the LIN bus from back-feeding power to V_{BAT} when the battery is disconnected.

（2）The 1 kΩ external pull-up resistor for the master node is required. It enhances the bus drive capability, reduces signal delay, and improves signal stability.

（3）The 1 kΩ external pull-up resistor for the master node can be connected to V_{BAT} or INH, but connection to INH is recommended. This allows the LIN transceiver to enter sleep mode when a short circuit occurs on the LIN bus. In that case, the INH pin becomes floating, preventing continuous consumption of power from V_{BAT}.

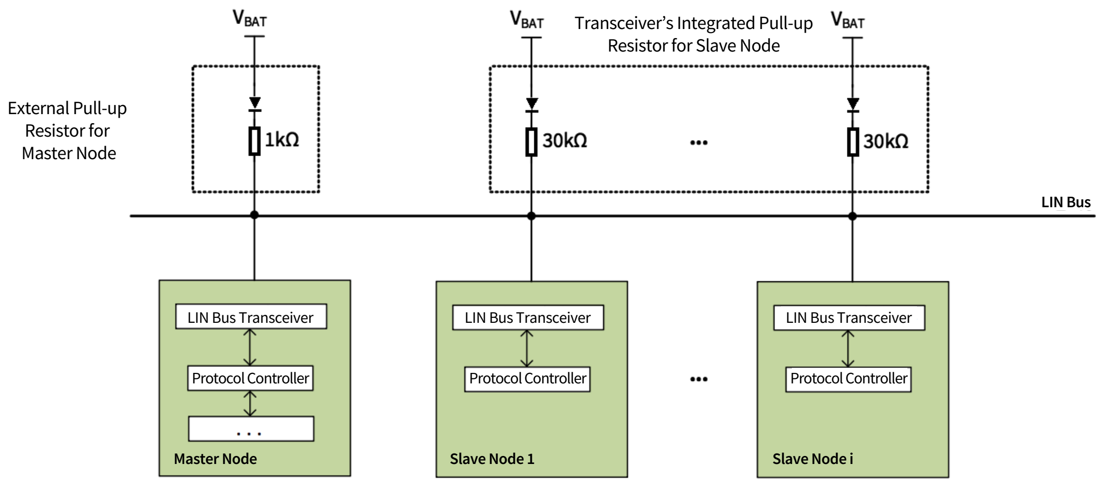


Figure 3. 3 Schematic Diagram of Pull-up Resistors

Here are the typical values and ranges for pull-up resistors for both master and slave nodes:

Node Type	Min.	Typical	Max.	Unit
Master Node	900	1000	1100	Ω
Slave Node	20	30	60	kΩ

3.2.2. LIN Bus Levels

During transmission, when TXD is at low level, the corresponding switch transistor turns OFF, allowing the LIN bus to be pulled up by the pull-up resistor to a level close to VBAT (taking into account the diode drop); when TXD is at high level, the corresponding switch transistor turns ON, pulling the LIN bus down to a level near GND (also considering voltage drop).

3.2.3. Threshold Levels

For LIN transceiver, the transmitter and receiver have different threshold level requirements due to factors such as voltage drops over communication lines and ground offsets, as shown in Figure 3. 4.

For transmitters, a signal must be driven below 20% * V_{BAT} to exhibit a dominant level, and must be driven above 80% * V_{BAT} to represent a recessive level.

For receivers, if the voltage on the bus is below 40% * V_{BAT}, it is considered a dominant level; if the voltage on the bus is above 60% * V_{BAT}, it is considered a recessive level.

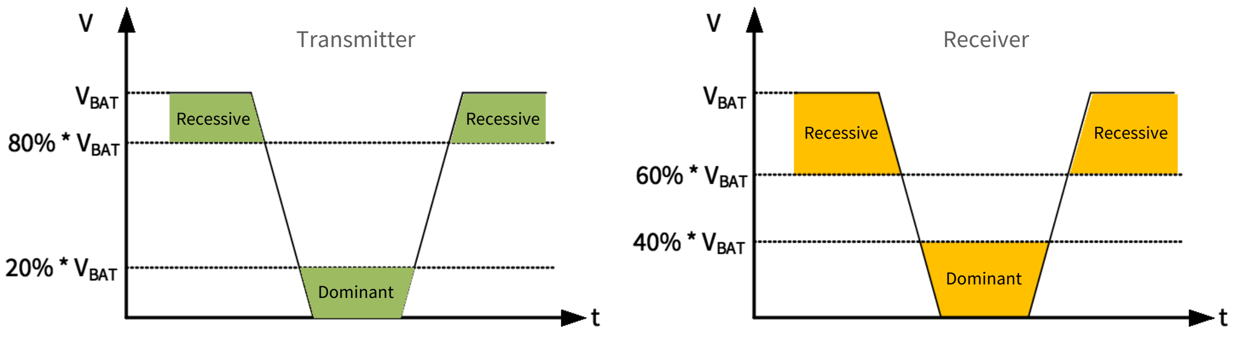


Figure 3. 4 LIN Transceiver Threshold Levels

3.2.4. Bit Rate Deviation

In a LIN network, only the master node requires a quartz or ceramic oscillator to provide a high-precision clock source. The slave nodes primarily rely on the Sync Field to achieve bit rate synchronization. Apart from the inherent deviation of the clock sources, temperature changes and voltage offsets can also affect the bit rate. Table 3. 1 defines the bit rate deviation requirements of LIN networks.

Bit Rate Deviation Type	Allowed Deviation Range
Master node deviation from standard bit rate	±0.5%
Deviation of slave node using a high-precision clock instead of Sync Field from standard bit rate	±1.5%
Deviation of slave node using the Sync Field instead of a high-precision clock before synchronization from standard bit rate	±14.0%[1]
Deviation of slave node using the Sync Field after synchronization from master node's bit rate	±2.0%
Bit rate deviation between any two communicating slave nodes	±2.0%

Table 3. 1 Bit Rate Deviation Requirements

[1]14% deviation is typical for low-cost on-chip oscillators. Internal calibration can help achieve precision above ±14%.

3.2.5. Bit Sampling

After synchronization via the Sync Field of the frame header sent by the master node, slave nodes must accurately sample each bit. This process is illustrated in Figure 3. 5.

（1）The synchronization occurs at the falling edge of the start bit of a byte field. The typical and maximum byte field sync time t_{BFS} are 1/16 t_{bit} and 2/16 t_{bit}, respectively.

（2）After synchronization, the bit sampling must occur between the earliest sampling time (t_{EBS}) and the latest sampling time (t_{LBS}). t_{LBS} depends on t_{BFS}, and the formula for calculating t_{LBS} is t_{LBS} = 10/16 t_{bit} - t_{BFS}.

（3）For subsequent bits, the sampling requirement remains consistent with the initial sampling. The sampling window repetition time (t_{SR}) is defined as the interval between adjacent bits' t_{EBS} or t_{LBS}, and is calculated using the formula: $t_{SR} = t_{EBS(n)} - t_{EBS(n-1)} = t_{LBS(n)} - t_{LBS(n-1)} = t_{bit}$.

Time Point	Min.	Typical	Max.	Remarks
t_{BFS}		1/16 t_{bit}	2/16 t_{bit}	
t_{EBS}	7/16 t_{bit}			$t_{EBS} \le t_{LBS}$
t_{LBS}				$t_{LBS} \ge t_{EBS}$

Table 3. 2 Bit Sampling Requirements

Table 3. 3 provides two examples of bit sampling:

t_{BFS}	t_{EBS}	t_{LBS}
1/16 t_{bit}	7/16 t_{bit}	9/16 t_{bit}
2/16 t_{bit}	8/16 t_{bit}	8/16 t_{bit}

Table 3. 3 Bit Sampling Examples

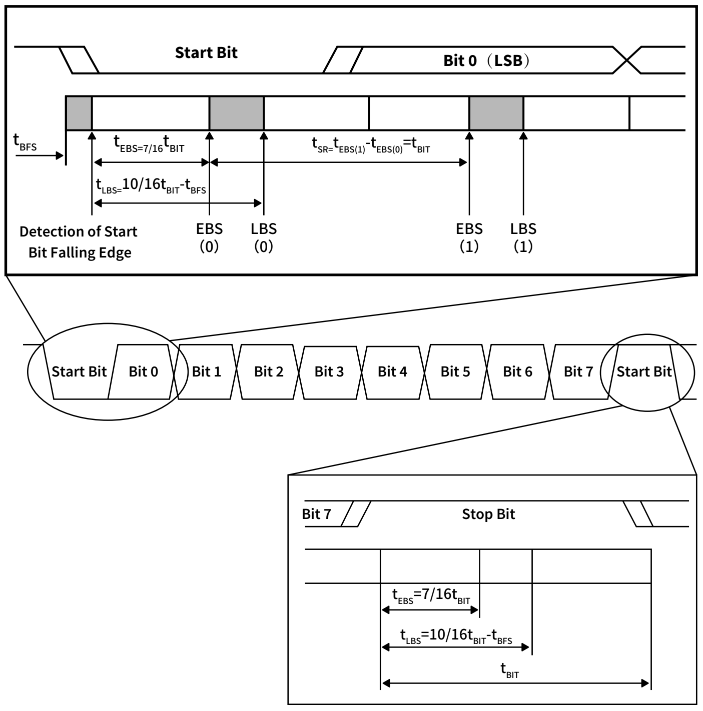


Figure 3. 5 Bit Sampling Diagram

3.2.6. Duty Cycle Requirements

To ensure that LIN messages can be transmitted and parsed properly, the transmitter of the LIN transceiver must maintain bus levels conforming to design logic, and the receiver must be able to sample levels correctly. This means that the duty cycle from TXD to the LIN bus and from the LIN bus to RXD must remain undistorted. Table 3. 4 provides duty cycle requirements for communication rates of 20 kbps and 10.4 kbps. Figure 3. 6 is the duty cycle diagram.

Parameter	Variable	Min.	Typical	Max.	Unit	Remarks
Bus Load Condition (CBUS / RBUS): 1 nF / 1 kΩ, 6,8 nF / 660 Ω, 10 nF / 500 Ω)						
Duty Cycle 1	D1[1]	0.396				$V_{th(rec)(max)} = 0.744 \times V_{BAT}$ $V_{th(dom)(max)} = 0.581 \times V_{BAT}$ V_{BAT} = 7 V to 18 V, t_{bit} = 50 µs[2]
		0.396				$V_{th(rec)(max)} = 0.76 \times V_{BAT}$ $V_{th(dom)(max)} = 0.593 \times V_{BAT}$ V_{BAT} = 5.5 V to 7 V, t_{bit} = 50 µs[2]
Duty Cycle 2	D2[1]			0.581		$V_{th(rec)(min)} = 0.422 \times V_{BAT}$ $V_{th(dom)(min)} = 0.284 \times V_{BAT}$ V_{BAT} = 7.6 V to 18 V, t_{bit} = 50 µs[2]
				0.581		$V_{th(rec)(min)} = 0.41 \times V_{BAT}$ $V_{th(dom)(min)} = 0.275 \times V_{BAT}$ V_{BAT} = 6.1 V to 7 V, t_{bit} = 50 µs[2]
Duty Cycle 3	D3[1]	0.417				$V_{th(rec)(max)} = 0.778 \times V_{BAT}$ $V_{th(dom)(max)} = 0.616 \times V_{BAT}$ V_{BAT} = 7 V to 18 V, t_{bit} = 96 µs[2]
		0.417				$V_{th(rec)(max)} = 0.797 \times V_{BAT}$ $V_{th(dom)(max)} = 0.63 \times V_{BAT}$ V_{BAT} = 5.5 V to 7 V, t_{bit} = 96 µs[2]
Duty Cycle 4	D4[1]			0.59		$V_{th(rec)(min)} = 0.389 \times V_{BAT}$ $V_{th(dom)(min)} = 0.251 \times V_{BAT}$ V_{BAT} = 7.6 V to 18 V, t_{bit} = 96 µs[2]
				0.59		$V_{th(rec)(min)} = 0.378 \times V_{BAT}$ $V_{th(dom)(min)} = 0.242 \times V_{BAT}$ V_{BAT} = 6.1 V to 7 V, t_{bit} = 96 µs[2]

Table 3. 4 Duty Cycle Requirements

[1]D1, D3= $t_{bus(rec)(min)}$ / (2*t_{bit}), D2, D4= $t_{bus(rec)(max)}$ / (2*t_{bit}).

[2]t_{bit}=50 µs (20 kbps), t_{bit}=96 µs (10.4 kbps).

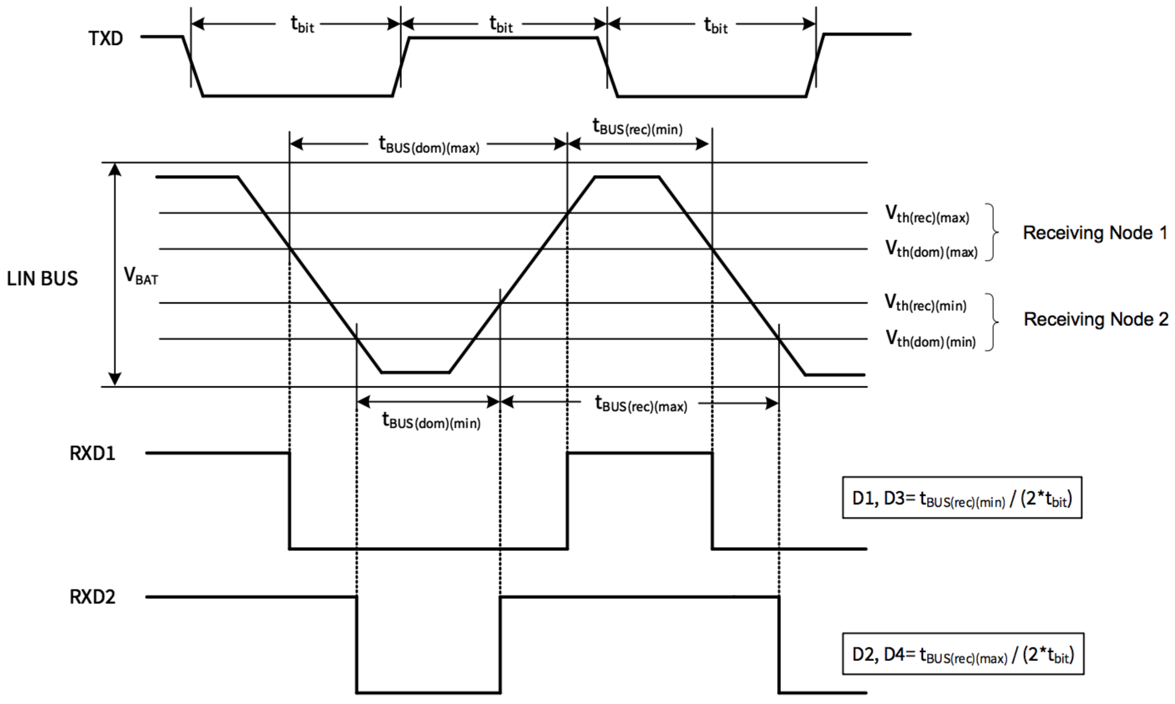


Figure 3. 6 Duty Cycle Diagram

3.2.7. Failure Mode Performance

（1）When a LIN transceiver loses its connection to V_{BAT} or GND, it should not affect the normal communication of other nodes on the bus. Once power is restored at the failing node, it should be able to rejoin the network communication.
（2）When the LIN bus is shorted to V_{BAT} or GND, communication across the bus may be disrupted. After the fault is resolved, the bus must resume normal communication.

3.2.8. TXD Dominant Timeout

TXD dominant timeout is a common protection feature in LIN transceivers. Under normal operation mode, if the transceiver detects that TXD has been driven low for an extended period, it triggers a timeout protection mechanism. This mechanism disables the transmitter and pulls the bus to a recessive state. The timer starts counting on the falling edge of TXD, and resets on the rising edge of TXD. This function is designed to prevent excessive battery drain due to hardware/software faults persistently holding TXD low.

3.3.Line Characteristics

For LIN networks, as the number of nodes increases, the bus-to-ground capacitance increases, while the equivalent pull-up resistance decreases. The bus rise time is determined by the time constant τ, with detailed parameters listed in Table 3. 5. The calculation formula is $\tau = R_{BUS} \times C_{BUS}$, where R_{BUS} and C_{BUS} are calculated as follows:

$C_{BUS} = C_{MASTER} + n^* C_{SLAVE} + LEN_{BUS} {}^*C'_{LINE}$

$R_{BUS} = R_{MASTER} || R_{SLAVE1} ||\ldots|| R_{SLAVEN}$

Parameter	Min.	Typical	Max.	Remarks
C_{MASTER}		220pF		Master node capacitance
C_{SLAVE}		220pF	250pF	Slave node capacitance
C'_{LINE}		100pF/m	150pF/m	Line capacitance (length-dependent)
LEN_{BUS}			40m	Bus length
τ	1μs		5μs	LIN network time constant

Table 3. 5 LIN Line Characteristics

4.Revision History

Revision	Description	Author	Date
1.0	First draft	Xiutao, Lou; Lele Zhang	2023/12/26

Sales Contact: sales@novosns.com;Further Information: www.novosns.com