Local InterConnect Network (LIN)

* Introduction:-

→ LIN is a Low-cost serial communication protocol used in automotive applications.

→ LIN bus is a subsystem (sensors, actuators) inside the vehicle and act as a complementary to the CAN bus.

→ All LIN nodes connected over a single wire bus.

→ LIN bus is used for applications that don't require high-speed data transmission, such as

- Door → Mirrors, door locks, and window control
- Climate → control panel, small motors
- Seat → seat positions and occupancy sensors
- roof → rain sensor, sun roof, and light sensors.... etc

→ CiA (CAN in Automation) standardized LIN as ISO 17987:2016

* LIN bus specifications:

→ Low cost option (if speed / fault tolerance are not critical.)

→ single master, multiple slaves (up to 15 slaves)

- a master often be MCU and a slave often be sensor or actuator.
- a group of single master and up to 15 slaves, called "LIN cluster"

→ single wire bus connected all devices on the LIN bus → open drain

→ speed start from 1kbit/s up to 20 kbit/s

→ total length of bus line is 40 meters.

→ Byte oriented protocol, and the data length can vary between 2, 4 or 8 bytes

→ operating voltage of 12 Volt.

→ LIN supports error detection, checksums and configurations.

→ it is a deterministic protocol

- it is mean that it does not involve bus arbitration.
- instead, it employs time-triggered scheduling with guaranteed latency time.

→ sleep mode and wake up support.

→ self-synchronization protocol (in frame synchronization)

* Basic layout :-

→ in a LIN cluster, there is a single wire that connects a master node and up-to 15 slave nodes.

→ the master node controls the bus access, while the slave nodes can send and receive information.

→ instead of implementing a dedicated communication controller, a microcontroller is programmed with the LIN protocol

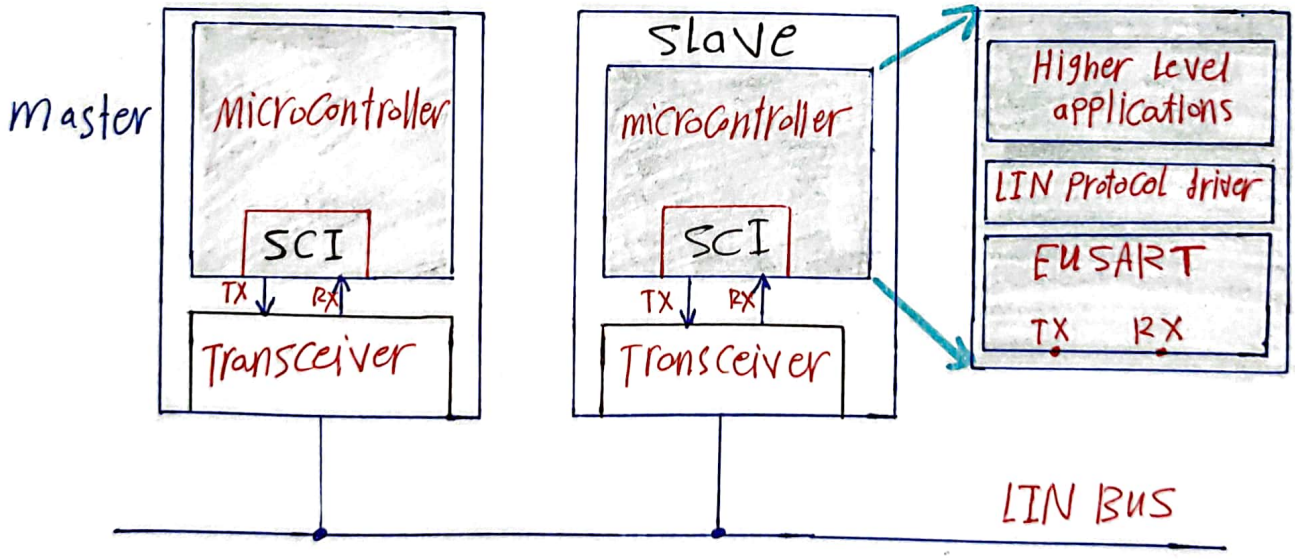
→ the microcontroller drives the communication to the transceiver through the serial communication interface (SCI)

→ SCI has replaced UART in most LIN protocol applications, reducing installation efforts.

→ SCI and UART are typical of most microcontrollers

→ all nodes are passively connected to the bus, meaning they don't actively drive the bus signal.

→ a pull-up resistor is employed to ensure the bus remains at the supply voltage level when nodes are in the off-state (open drain)



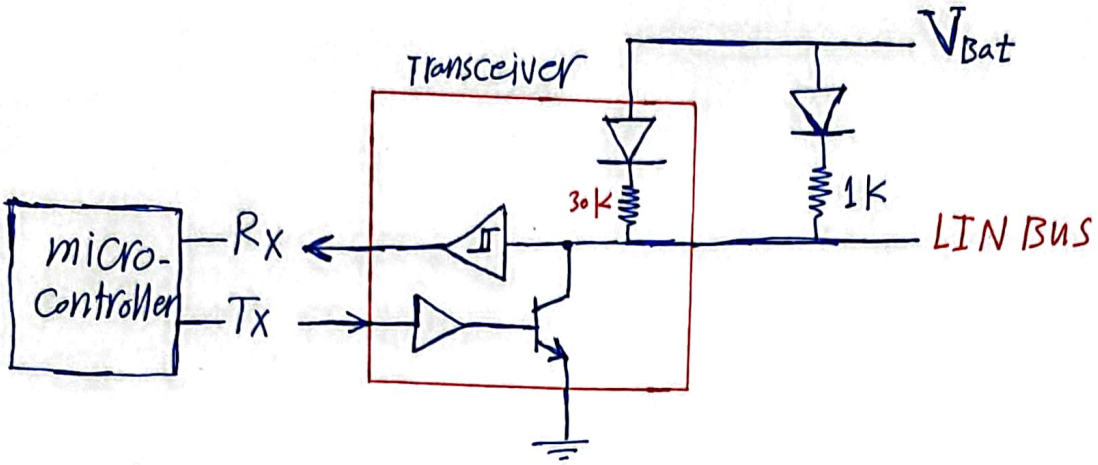
* physical layer:

→ in the OSI model, the physical layer is concerned with "putting the bits on the wire"

→ the LIN physical layer is based on ISO 9141 (the K-Line bus)

→ the LIN consists of the bidirectional bus Line which is connected to the transceiver of every bus node, and is connected via a termination resistor and a diode to the battery voltage (V_{Bat}).

→ the LIN Bus operates between 9V and 18V.



→ Slave nodes on the LIN Bus typically incorporate a termination mechanism by being pulled up from the LIN bus to the voltage supply (V_{sup}) using a 30 kΩ resistor and a serial diode.

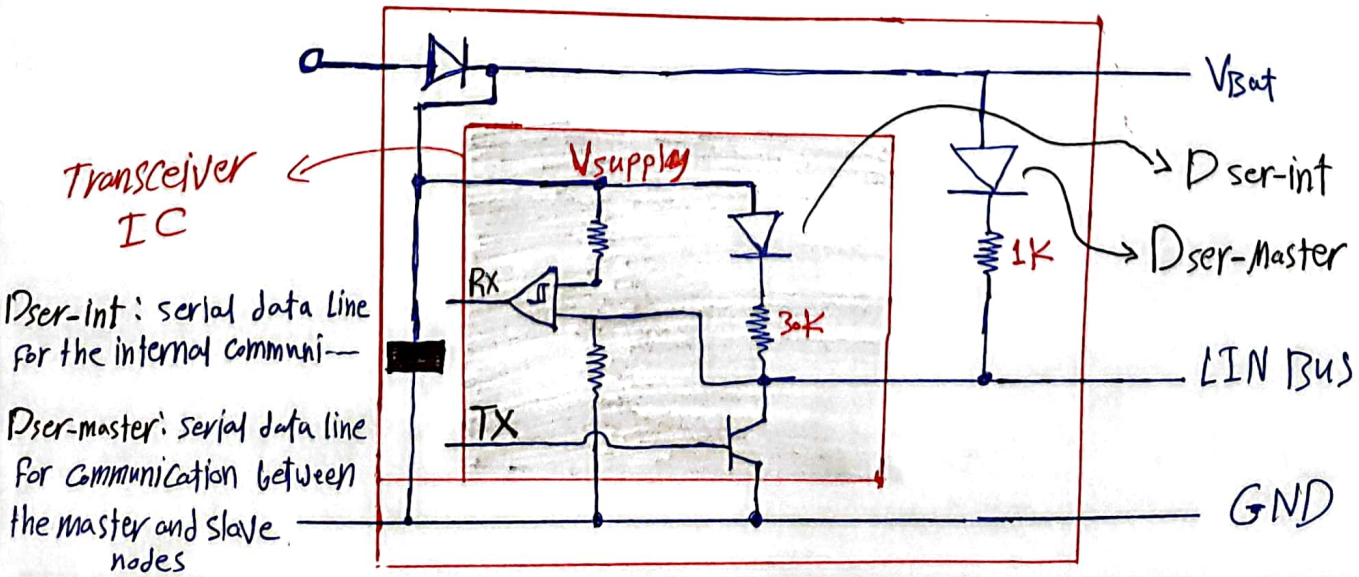
→ the termination arrangement is usually built-in the transceiver package of the slave node.

→ the master node, on the other hand, the master node needs a separate termination setup, which includes adding a 1 kΩ resistor and a serial diode between the LIN Bus and the voltage supply (V_{sup}). unlike the slave node, the master node's termination is not built-in with the transceiver package.

→ A cluster in LIN corresponds to an open collector circuit.

→ A pull-up resistor ensures that the bus level matches the supply voltage when Tx transistors are off

→ When a Tx transistor circuit conducts, the bus Level is pulled to ground, representing a dominant LoW state.



→ the diode is mandatory for preventing an uncontrolled power supply of the ECU from the bus in case of a "loss of battery".
→ the LIN protocol utilizes a conventional single-wire line for physical signal transmission.
→ this means that no differential voltage signals are used.
→ logic one and logic zero are represented by defined voltage levels.
→ to ensure sufficient noise immunity, the reference potential for the bus level is derived from the supply voltage of the ECU electronics and vehicle ground.
→ Levels below 40% of the supply voltage are interpreted as logic zero by the receiver
→ Levels above 60% of the supply voltage are interpreted as logic one by the receiver
→ assuming $V_{BAT} = 12\,V$ → logic 0 → $V < 4.8$
↳ logic 1 → $V > 7.2$
→ senders transmit voltage levels below 20% for logic zero and above 80% for logic one

* LIN Transceiver (physical Layer):

→ the LIN bus operates within the voltage range of 9V to 18V
→ in many cases, microcontrollers operate at lower voltage levels, such as 3.3V or 5V for their I/o pins. However, the LIN bus itself operates at higher voltage levels.
→ To enable communication between the microcontroller and the LIN bus, a transceiver is used.
→ this transceiver adapts the voltage levels of the microcontroller's I/o pins to match the higher voltage levels of the LIN bus. This allows the MCU to send and receive data on the LIN bus effectively, despite the voltage disparity.

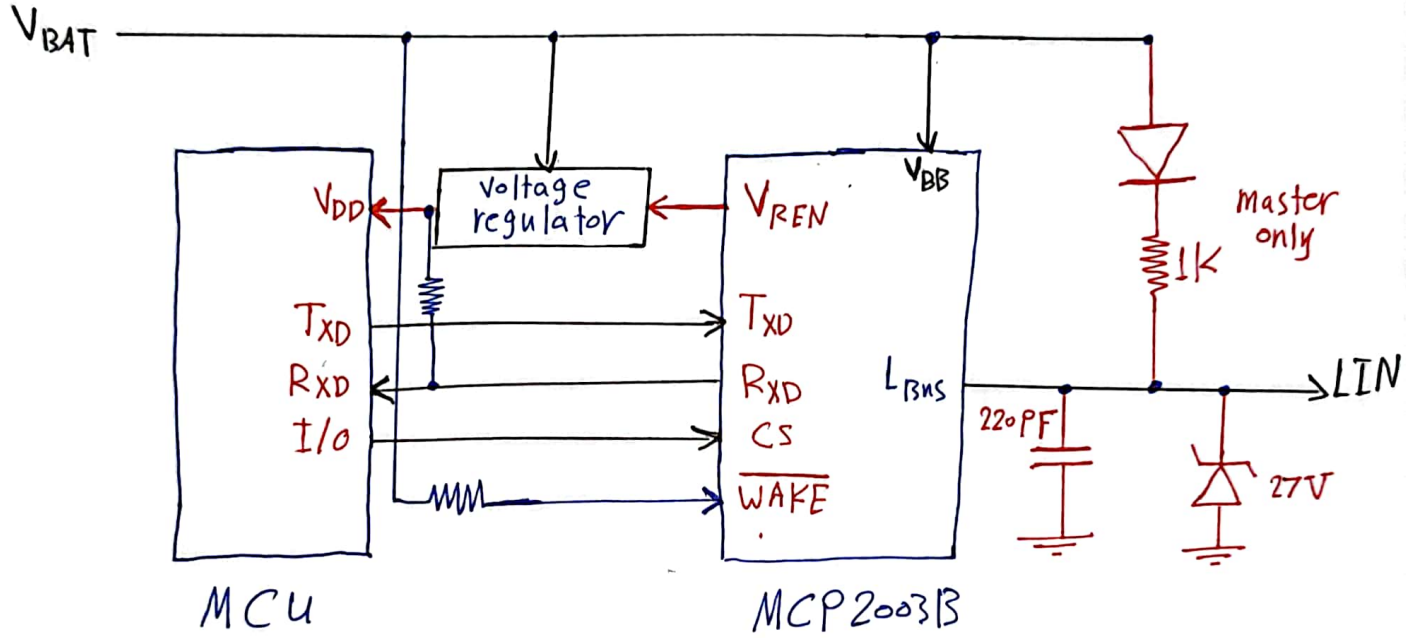
→ there are several typical configurations for LIN nodes.
→ these configurations include using transceivers such as:
1- MCP2003B → stand Alone transceiver
↳ it requires a LIN-capable UART peripheral in the MCU

2- MCP2021A → stand alone transceiver with built-in voltage regulator
  → often referred to as "Mini LIN system Basis chip"

3- MCP 2050 → stand alone transceiver with built-in voltage regulator and watchdog timer
  → Commonly known as "LIN system Basis chip"

4- System-in-Package (SIP):
  → example: PIC16F1829LIN, ATA6616C
  → Combines an MCU and complete LIN system Basis chip functionality

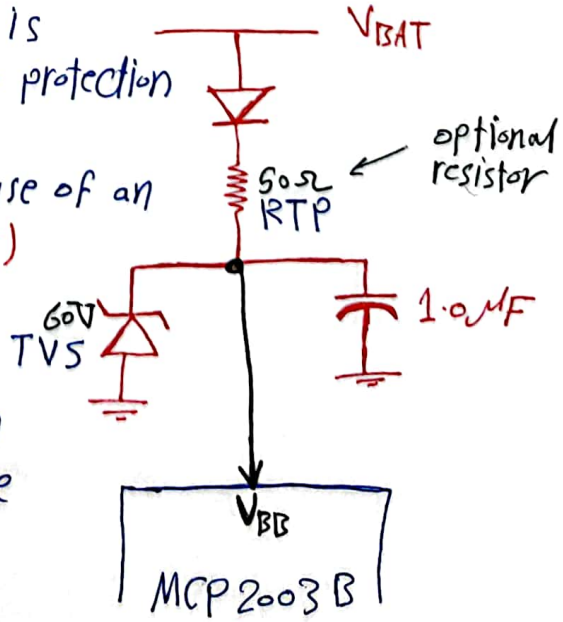
ex: MCP2003B



* Transient Voltage protection (Load Dump)

→ To protect device against power transients and electrostatic discharge (ESD) events, it is recommended to include a transient voltage protection mechanism in the circuit.

→ this protection is achieved through the use of an external 60V transient suppressor (TVS) diode connected between the V_{BB} pin and ground. Additionally, a 50Ω transient protection resistor (RTP) is placed in series with the battery supply and the V_{BB} pin.



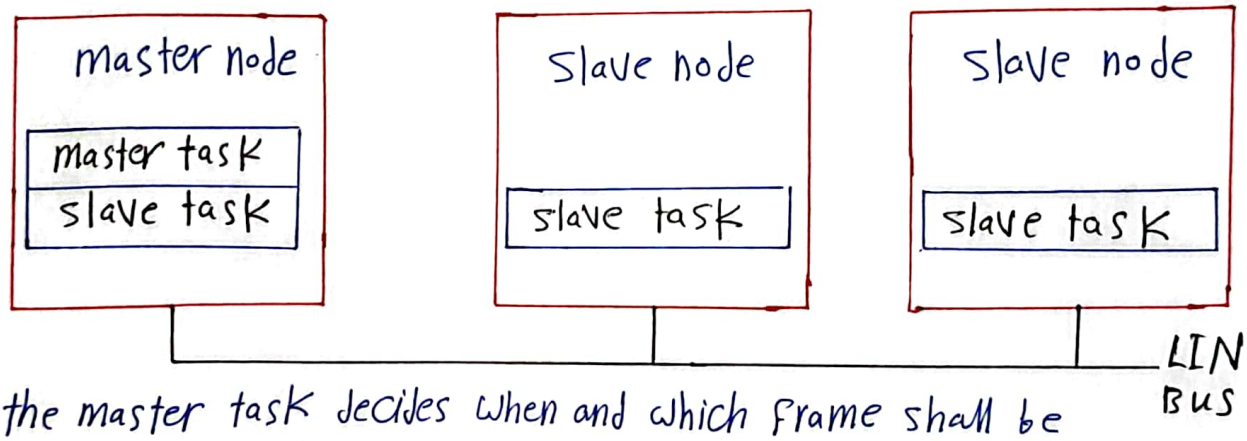
→ the TVS diode acts as a voltage clamp, diverting excessive voltage spikes to the ground, preventing them from damaging the device
→ the RTP resistor limits the current during transients, providing further protection against voltage surges.

---

* LIN protocol Frames:-

→ Master and slaves:

- a cluster consists of one master node and several slave nodes.
- the master node contains the master task as well as a slave task
- all other slave nodes contain a slave task only.
- a node may participate in more than one cluster
- example: a sample cluster with one masternode and two slave nodes



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

→ LIN Frames:

- A Frame consists of a header (provided by the master task) and a response (provided by a slave task).
- the header consists of a break field and sync field followed by a frame identifier.
- the frame identifier uniquely defines the purpose of the frame.
- the slave task appointed for providing the response associated with the frame identifier transmits it.
- the response consists of a data field and a checksum field.
- the slave tasks interested in the data associated with the frame identifier receives the response, verifies the checksum and uses the data transported.

* LIN Description File (LDF):

→ in the LIN system, each slave has an LDF. this file provides information to each slave about whether it will be the sender or receiver/listener of specific data.

→ the LDF allows each slave to determine its role in the communication process. if a slave is designated as a sender for certain data, it will transmit that data

→ on the other hand, if a slave is assigned as a receiver or listener, it will receive and process the data

→ the slave can't be both a sender and a receiver at the same time.

→ ex:

T1: Frame 1	ID = 0X10	sender → slave 2	receiver → slave 1
T2: Frame 2	ID = 0X12	// → slave 1	// → master

..... and so on.

---

* LIN Frame:

→ the LIN Frame is divided into two parts: the header and the response.

→ the header is always transmitted by the master node since the master controls the LIN bus, it is transmitted based on schedule table which specifies the identifier for each header and the time interval between start of two consecutive frames.

→ the header consists of three distinct fields:

1- Break
2- Sync
3- Protected Identifier Field (PID)

1- Break Field is divided into two parts: (13+1) bits / (18+2) bits

1- Break, which consists of (13 bits ~ 17 bits) dominant bits that signal the begining of a new frame.
2- the delimiter, which is a recessive bit

→ a break field is always generated by the master task in the master node.

→ this is longer than the worst case data pattern of all dominant, thus ensuring that the bus will see it as a break.

→ the periodicity of a slave task must be shorter than the time it takes to transmit the break field in a LIN cluster. specifically, the shortest periodicity for any task should be less than the duration of the break field

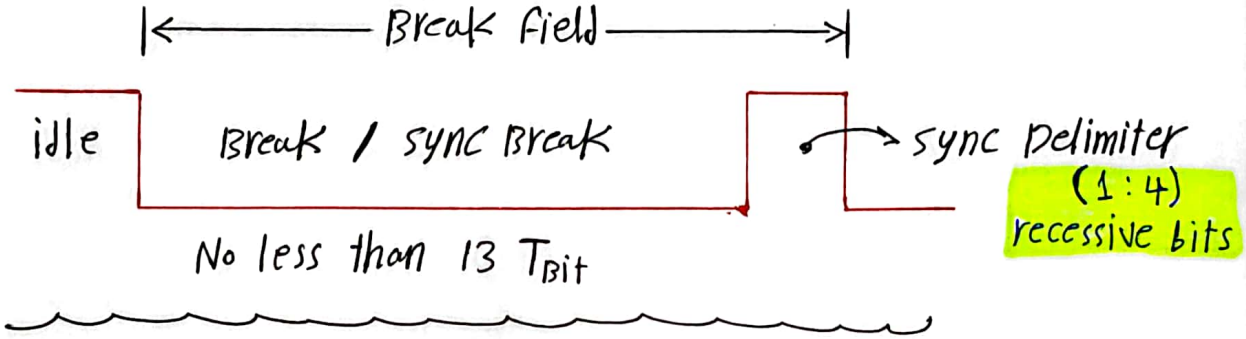
→ this guarantees that even the slowest slave task gets a chance to perform its checks within each break field period. otherwise, communication errors or missed tasks could occur.

→ the duration of the break field = (13 : 18) * bit time

→ assume that the Freq = 20 kbps

→ then the bit time = 1 / 20 kbps

→ it is crucial that the total duration of the break field is greater than the shortest periodicity of any task.



2- Sync field :.

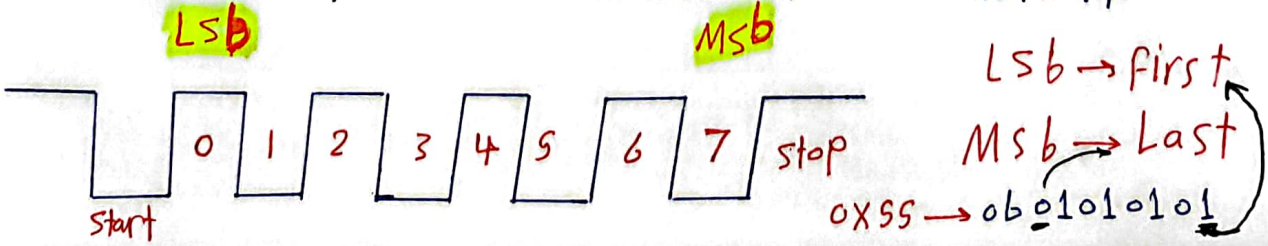
→ the sync field is an 8-bit square signal sent in the form of a byte field structure, which are typically 10-bits.

→ to ensure that all slaves send and receive at the same clock time, the master transmits the sync field after the sync break field.

→ the sync field contains the value 0x55.

→ to drive the clock, the slaves measure the time between the first and last falling edge of the sync field and divide it by eight. the result corresponds to one bit time (In Frame synchronization)

→ this makes the LIN protocol supports dynamic data rate adaptation, allowing nodes with different processing capabilities or communication requirements to coexist on the same network



3) Protected Identifier Field (PID) :-

→ the PID consists of two parts :

- a six bit identifier, which used to identify each frame on the bus and describe its contents.
- a two bit parity. the computation of parity bits P0 and P1 is based on exclusive OR (XOR) logic, where the P0 represents even parity and P0 represents odd parity.

1) Frame identifier :-

→ the most significant bits (ID4 and ID5) are used for data length control. For example,

ID range	Hex	Data length
0 : 31	0X (00 : 1F)	2
32 : 47	0X (20 : 2F)	4
48 : 63	0X (30 : 3F)	8

→ the Frame identifiers are split in three categories:

decimal Value	Hex Value	Description
0 : 59	0X (00 : 3B)	used for signal carrying Frames
60-61	0X (3C : 3D)	used to carry diagnostic and configuration data
62-63	0X (3E : 3F)	reserved for Futur enhancements

2) Parity Part : is calculated from the ID bits using this XOR equations.

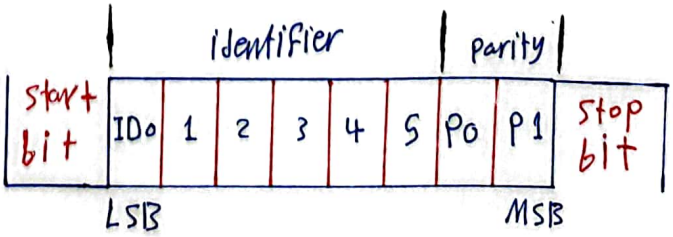
$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4$$
$$P1 = !(ID1 \oplus ID3 \oplus ID4 \oplus ID5)$$

→ So all slave tasks continnally listen to ID field and verify its parity.

→ Accordingly, the slave task decided to read data, send response, or do nothing.

ex: 0X26 ⟶ 10 0110

$$P0 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$
$$P1 = !(1 \oplus 0 \oplus 0 \oplus 1) = 1$$



PID = 1010 0110

P1 P0 ID5 ID0

* LIN Frame response:

→ it is transmitted by a slave task and is divided into two fields:
- Data and checksum

1) the Data field:
→ once the header sent by the master node has been transmitted, it's time for the transfer of actual data to begin.
→ Based on the transmitted identifier, a slave will recognize that it has been addressed, and reply with a response in the data field
→ several signals can be packed into one frame
→ Data bytes are transmitted LSb first
→ the data field consists of 1 to 8 bytes that represent the data payload to be transmitted
→ the data length is controlled by the most significant 2 bits of the header (ID 4 and ID 5).

2) checksum field:
→ it consists of eight bits that are used for data validation.
→ it can be calculated by two methods: classic and enhanced.
→ the master node manages the use of classic or enhanced checksum, which based on the frame identifier
→ communication with LIN 1.X slave nodes uses the classic checksum, while the communication with LIN 2.X slave nodes uses the enhanced checksum.
→ the frame identifier 60 (0X3C) to 61 (0X3D) are always required to use the classic checksum.

1) classic checksum:
→ the classic checksum is calculated by summing the data bytes only, as specified in version 1.3 of the LIN specification.
→ it is used with master request frame, slave response frame, and communication with LIN 1.X slaves

→ the calculation involves summing all the values and subtracting 255 whenever the sum is greater than or equal 256

ex: Data field ⟶ 1: 0x10 2: 0x20 3: 0x30 4: 0x40

Data byte 1 : 0001 0000
Data byte 2 : 0010 0000

0011 0000
Data byte 3 : 0011 0000
1 1

0110 0000
Data byte 4 : 0100 0000
1

1010 0000

Checksum = inverse (1010 0000) = 0101 1111

2) enhanced checksum :

→ it is calculated by summing the data bytes and the protected identifier byte, as specified in version 2.0 of LIN specification.

→ it is used for communication with LIN 2.x slaves.

→ the calculation is similar to the classic check sum, but includes the PID byte in the sum.

ex: PID = 0x30 Data field : 0x10 , 0x20, 0x30, 0x40

PID : 1111 0000
Byte 1 : 0001 0000
1111

1 0000 0000
carry bit ⟶ 1

0000 0001
Byte 2 : 0010 0000

0010 0001

0010 0001
Byte 3 : 0011 0000
1

0101 0001
Byte 4 : 0100 0000
1

1001 0001

Checksum = inverse (1001 0001)
= 0110 1110 #

* LIN Frame types :

1) unConditional frame: ⟶ (ID 0:59)

⟶ it is the normal type of LIN Communication and typically used to transmit useful data

⟶ it consists of a header and a response.

⟶ the master transmits the header onto the bus as a request. then a defined slave answers with the response

⟶ a slave task recognizes, based on the identifier (ID), whether it needs to send a response

⟶ the slave task of the master can also be addressed by the ID and send a response

⟶ the specific node responsible for the response is described in the LIN Description file (LDF).

⟶ unconditional LIN Frame prevents collisions :

- as it establish a unique relationship between the header and the response (one-to-one relationship), ensuring that only one node can use a slot at a time.
- utilizing ID-based recognition, and configuring the LDF. this ensures that only the intended slave node sends a response, preventing multiple nodes from responding simultaneously.
- unconditional frames are therefore ideally suited for periodic data transmission, in which the time when a message appears on the bus is predictable.

* LIN Schedule Table :.

⟶ in the LIN cluster's master-slave architecture, the master node governs communication through a mechanism called delegated token bus access.

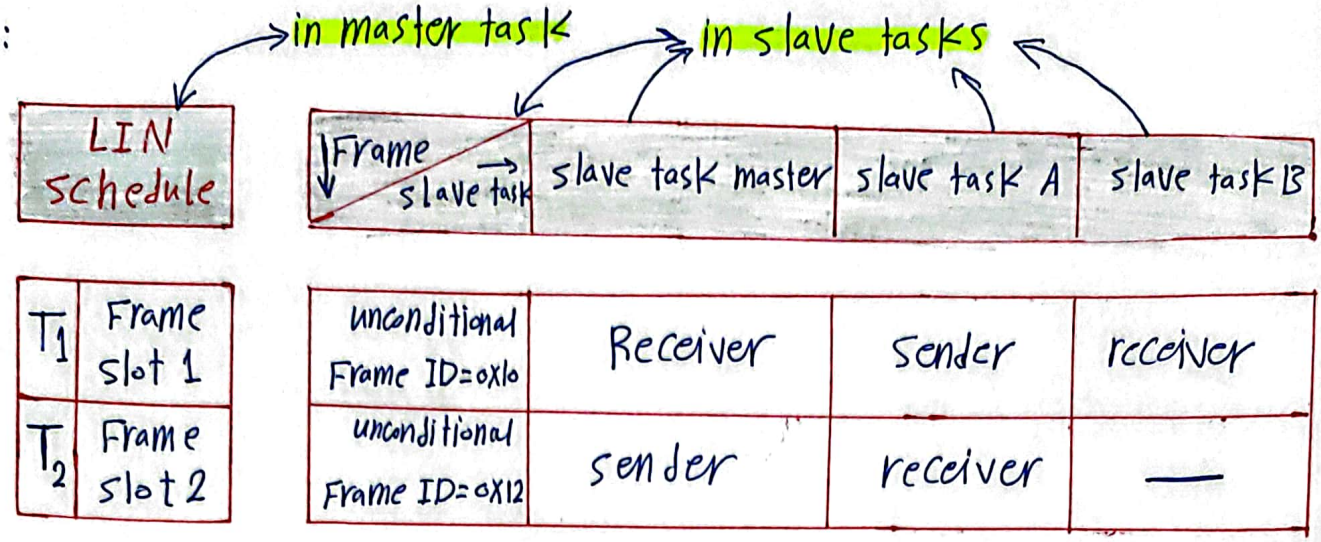
⟶ this approach ensures orderly access to the communication bus, promoting nearly collision-free communication and predictable data transmission.

⟶ the LIN protocol uses schedule tables, which help prevent the communication bus from getting overloaded.

⟶ the LDF contains the schedule table, defining the sequence and time grid for message transmission.

⟶ after completing the table, the master node restarts the sequence from the begining.

ex:



T_1	Frame slot 1
T_2	Frame slot 2

unconditional Frame ID=0x10	Receiver	Sender	receiver
unconditional Frame ID=0x12	sender	receiver	—

→ the master node follows a predetermined sending scheme that is planned by the system engineer and described in the LDF, this makes communicatio in the network predictable, because there is a fixed time schedule.

→ the schedule is divided into slots, dedicated to transmitting one frame each.

→ Mini-slots, defined within the LDF and implemented in master node, determine the size of the slots and represent the cycle with which the master task processes the schedule.

→ the duration of mini-slots forms the time base for the ongoing communication, ensuring a predictable transmission process.

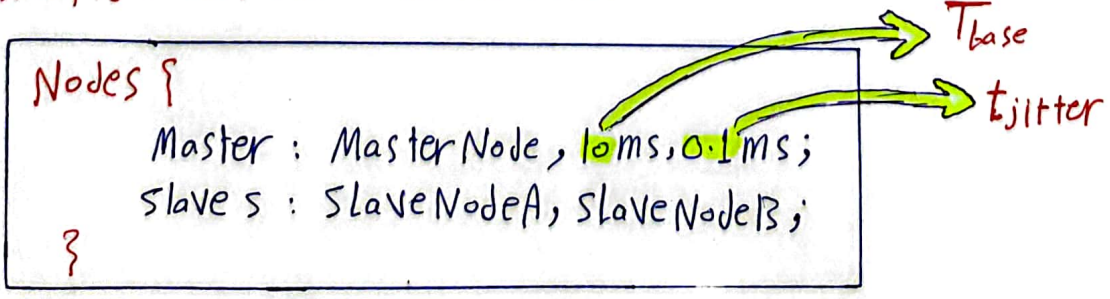
→ the master node is responsible for ensuring that all frames relevant in a mode of operation are given enough time to be transferred

→ the minimum time unit used in a LIN cluster is the time base (T_{base}). it acts as a reference for timing operations in the cluster

→ the time base is implemented in the master node and used to control the timing of the schedule table. the frames' timing within the schedule table is based on this time base.

→ the time base commonly used in LIN cluster is usually set to 5, 10, 15 ms. this value provides a standard reference for scheduling and coordinating frame transmission.

→ example : LDF File → Master node

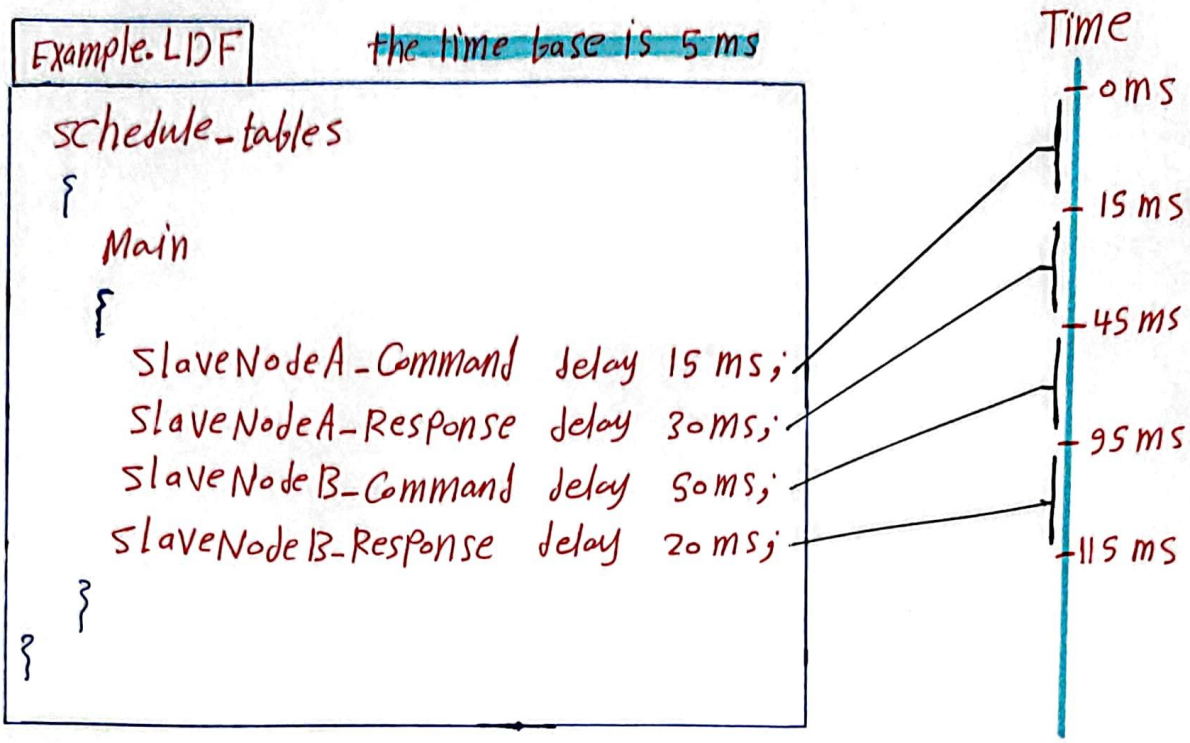


* example:.

→ example of a LIN schedule table depicted within a LDF:

Example.LDF — the time base is 5 ms

```
schedule_tables
{
  Main
  {
    SlaveNodeA_Command  delay 15 ms;
    SlaveNodeA_Response delay 30ms;
    SlaveNodeB_Command  delay 50ms;
    SlaveNodeB_Response delay 20 ms;
  }
}
```



→ the schedule table consists of a "Main" sub-table that defines four Frame slots

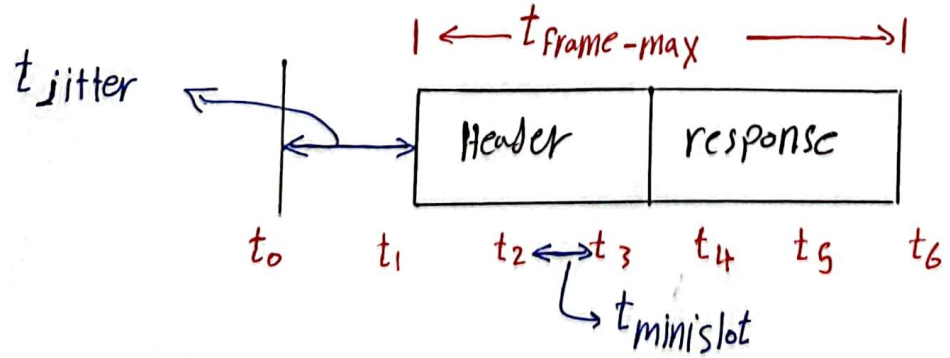
→ each slot represents a specific period of time allocated for the transmission of a particular frame.

→ it is important to ensure that the value assigned to each slot is a multiple of the time base.

---

* Jitter :.

→ it is the potential time difference between the nominal begining of a slot and the actually starting point, or it is a small delay tolerance is specified for the master node, in order to create the final frame slot width.



→ the jitter specifies the difference between the maximum and minimum delay from time base tick to the header sending start point (falling edge of break field)

→ in computing the size of a slot, a jitter value is added

→ if a Frame does not completely fill out the slot, the rest of time period must be waited until the next slot is available. this time period is known as the IFS (inter frame space).

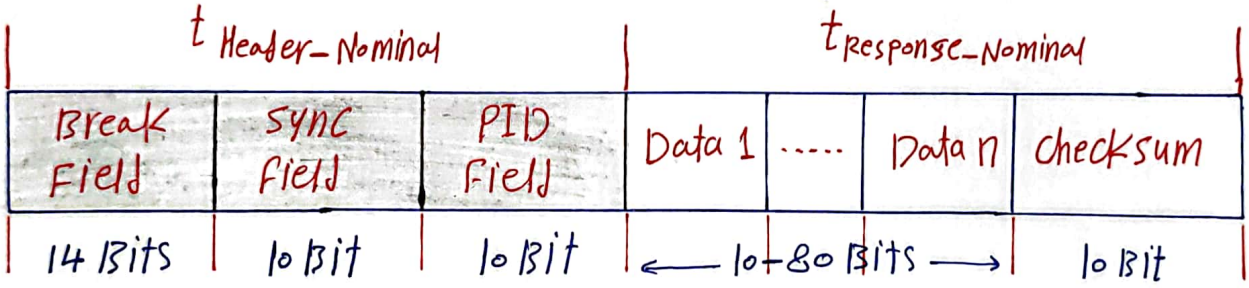
→ LIN slaves are typically implemented using low-cost, lower speed MCUs. thus, the specification allows a time reserve of up to 40% to be added to each SCI Frame byte.

→ the Response Time (RS) space: is the inter-byte space between the PID field and the first data byte in the response.

* Frame slot calculation :-

→ the nominal value for transmission of a Frame exactly matches the number of bits sent (no response space and no inter-byte space)

→ the nominal break field is 14 nominal bits long (13 + 1 delimiter)



$$t_{bit} = \frac{1}{Freq} = 1 / (19.2\,Kbps) = 52.1\,\mu s$$

$$T_{Header\_Nominal} = (14 + 10 + 10) * t_{bit} = 34 * t_{bit}$$

$$T_{Response\_Nominal} = 10\,(N_{Data} + 1) * t_{bit}$$ (+1 → for checksum)

$$T_{Frame\_Nominal} = T_{Header\_Nominal} + T_{Response\_Nominal}$$

→ as we reserve of up to 40% for message transmission

$$T(Header\_Maximum) = 1.4 * T(Header\_Nominal)$$
$$T(Response\_Maximum) = 1.4 * T(Response\_Nominal)$$

$$T(Frame\_Maximum) = 1.4\,(T(Header\_Nominal) + T(Response\_Nominal))$$
$$= T(Header\_Maximum) + T(Response\_Maximum)$$

* Signals :.

→ signals are carried within the data field of a Response Frame.
→ each signal has a single publisher, meaning it is written by a specific node within the cluster.
→ the signal represent a specific value or parameter within the cluster, i.e. → push button state , temp value .....
→ the signal can be represented by one bit or more than one bit.
→ all signals have an initial values that remain valid until updated.

* signal types:

1) a scalar signal :.

→ a scalar signal is between 1 and 16 bits long
→ a 1-bit scalar signal is called a boolean signal
→ scalar signals in the size of 2 to 16 bits are treated as unsigned integers.

2) A Byte array signal :.

→ it is an array of 1 to 8 bytes, allowing for the transmission of larger data chunks.

→ For any signal larger than 1-bit, LSB sent first

* signal consistency (تناسق) :.

→ to ensure signal consistency in LIN Frames, the reading or writing signals (scalar or bytearray) should be atomic operation. this means that it should not be possible for an application to receive a signal value that is partially updated.

→ the atomic operation refers to actions that are non-divisible. it ensures that an operation is executed as a single, unbreakable unit. for example, when reading or writing a signal value, the entire operation should occur without interruption.

* signal packing :.

→ a signal is transmitted with the LSB first and the MSB Last
→ for byte arrays, which consist of sequences of bytes, each byte in the array is mapped to a single frame byte.

→ the mapping starts with the lowest numbered data byte within the frame.
→ several signals can be packed into one frame as long as they don't overlap each other.
→ the same signal can be packed into multiple frames as long as the publisher of the signal is the same.

---

2) Event Triggered Frame:.

→ the purpose of an event triggered frame is to increase the responsivity of the LIN cluster without assigning too much of the bus bandwidth to the polling of multiple slave nodes with seldom occurring events.
→ the event triggered frame is equivalent to a standardized unconditional frame.
→ the difference is that multiple slaves may send a response to a header frame.
→ when the master sends a header with the identifier for an event triggered frame, the assigned slaves may append (probable) their specific responses.
→ only one response can be sent after the header, and the first data byte contains an additional PID to identify the node that sent the response
→ to ensure that the length of an event triggered frame is clearly defined, all potential responses have the same number of data bytes.

→ in a multiple node system, collisions can occur when multiple nodes generate a response to an event-triggered frame simultaneously
→ the master resolves these collisions with a collision resolving schedule
→ the collision resolving schedule (CRS) is a special sending scheme used by the master to handle collisions. it involves polling the responses of the slaves again and treating them as regular unconditional frames.
→ the master node generates a regular unconditional frame for each slave node involved in the collision.
→ each unconditional frame is specifically identified by a PID allowing the master node to address individual slaves.
→ By using the polling mechanism with individual unconditional frames and unique PIDs the master node resolves the conflict caused by the collision
→ this ensures that each slave node's response is correctly identified and processed.

→ By sequentially handling collided responses and then resuming the regular schedule, the master ensures that no valid responses are left untransmitted.

Ex:

→ there are multiple slave nodes that receive an event trigger frame from the master node.

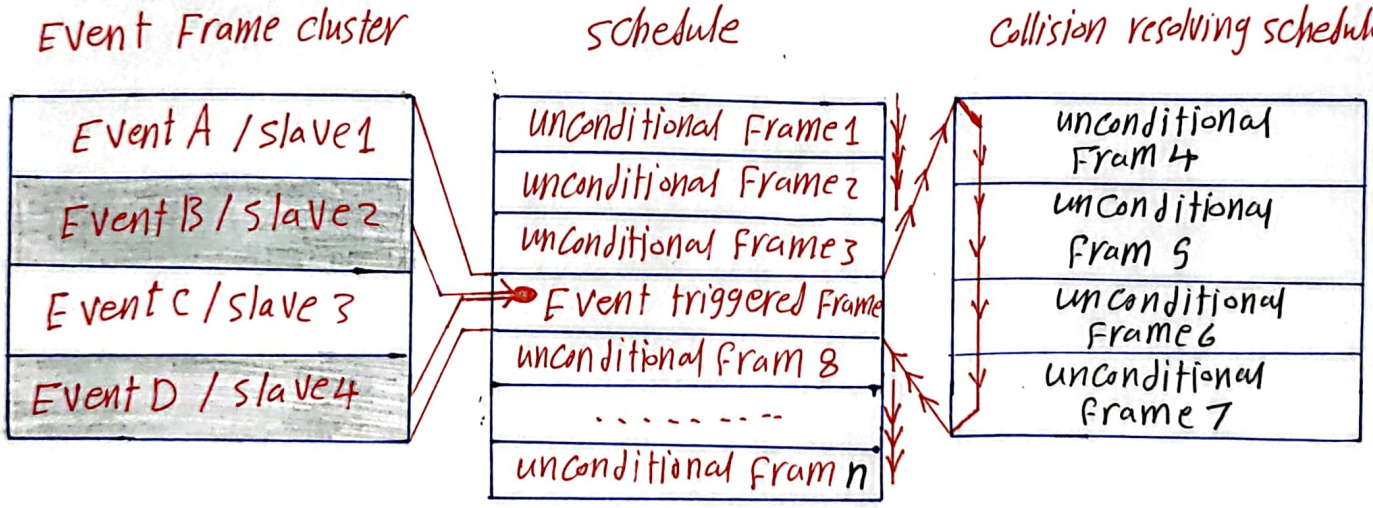
→ Both slave2 and slave4 detect the header and have events present.

→ as a result, both slave nodes publish their responses. However, this simultaneous publishing of responses leads to a checksum failure, indicating a collision.

→ the master node detects the collision by checking the checksum failure

→ to resolve this, the master node will ask each slave individually for their events separately using the schedule resolution schedule table

→ By doing so, the master node can gather the correct and individual responses from each slave node.



3) Sporadic Frame :-

→ in the LIN protocol, communication is typically deterministic, with a predefined fixed schedule table defining precisely when each node transmits. this real-time determinism is important for ensuring predictable and reliable operation.

→ However, there is some times a need to send unpredictable message in a dynamic way when certain events occur. For example, a sensor may need to report a measurement when a threshold is crossed, but this cannot be scheduled explicitly.

→ Sporadic Frames provide a way to accommodate such dynamic signaling needs without hold up the overall deterministic nature of the LIN bus. they do this by reserving specific slots conditionally:
  → the schedule table allocates certain slots that may be used for sporadic frames.
  → these slots are still scheduled deterministically
→ a sporadic frame is transmitted by the master only when it knows a slave node has updated information to publish.
→ the master uses sporadic frames to send rarely used information → (application-based).
→ structurally, a sporadic frame shares a schedule slot like a standard unconditional frame
→ Logically, a sporadic frame functions similarly to an event triggered frame but initiated by the master
→ the master only transmits a sporadic frame header when it knows a related slave signal has changed.
→ if no signals require updating, the corresponding schedule slot remains empty.

---

4) Diagnostics Frame

→ Diagnostics concept: a diagnostic check is a process of checking a vehicle's systems and components to help identify issues and fix them. modern vehicle are equipped with extensive computer networks that can provide information on the engine, brakes, and other related systems.
→ Diagnostics frames are used in the transport layer specification section of the LIN v2.2A specification.
→ the diagnostics frames primary function is implementing diagnostics and configuration function within a LIN network.
  → (carry diagnostic / configuration data)
→ they are always 8 bytes long and always use the classic checksum.
→ the LIN protocol defines two diagnostic frames:

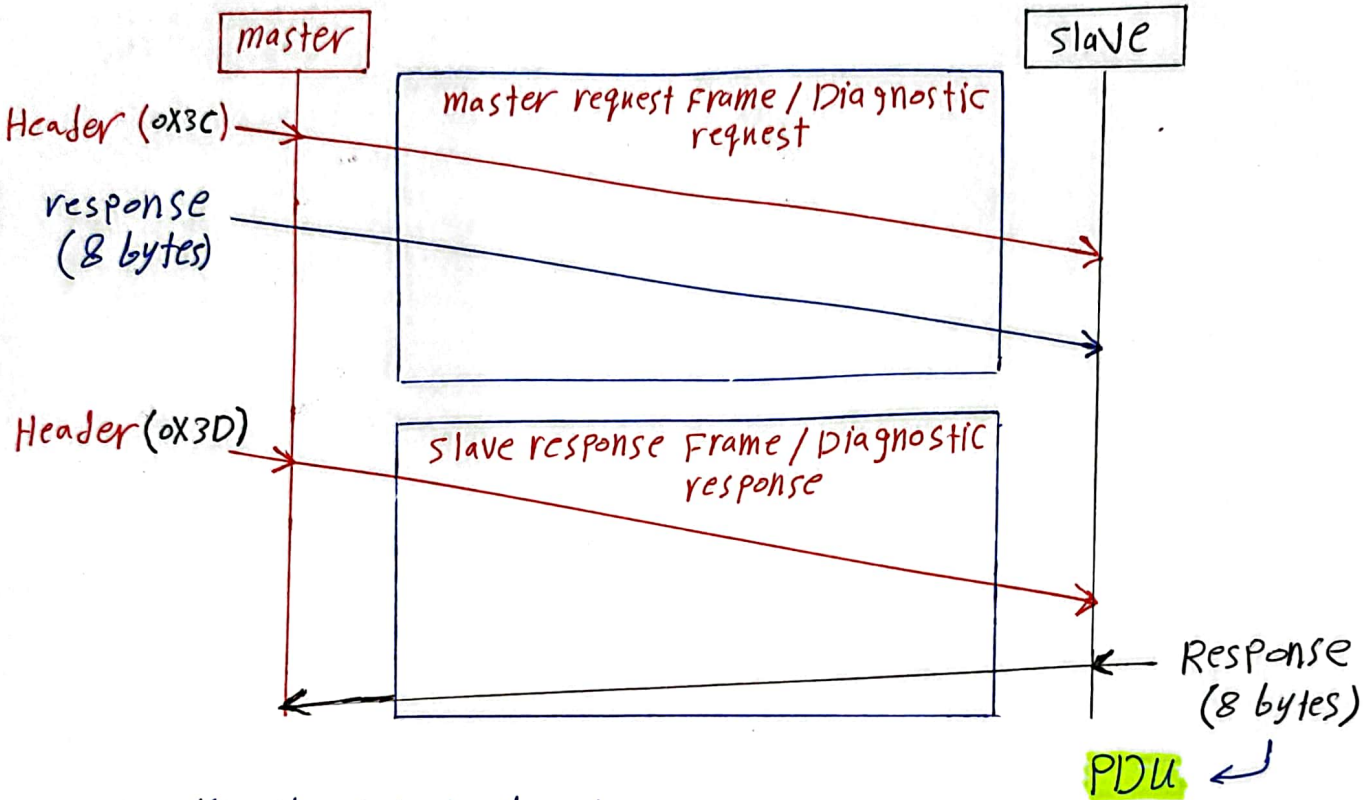
1) the master request frame
   → 0X3C (60)
→ used as a diagnostic request or to configure slaves.

2) the slave response frame
   → 0X3D (61)
→ used as a diagnostic response.

Diagnostic schedule

Master request Frame ID = 0X3C	Frame slot1
slave response Frame ID = 0X3D	Frame slot2

LIN Master ↓	LIN slave1 ↓	slave2 ↓
Header + response		
Header	response	



→ normally, the data in the slave response called PDU (Packet Data unit)

* the key difference between the internal and external diagnostic frames:

1- initiation of Diagnostic request frame:

→ in internal diagnostics, the master task in the master node sends a header with ID = 0X3C to initiate the diagnostic request.

→ while in external diagnostics, an external tool sends the diagnostic request to the system

2- Response generation: → internally

→ During the Diagnostic request phase the slave task in the master node sends the diagnostic response to the header with ID = 0X3C.

→ During the Diagnostic response phase the slave task in the slave node sends the diagnostic response to the header with ID = 0X3D

→ in external diagnostics, during the diagnostic request phase, the slave task in the master node generates the diagnostic response in response to the diagnostic request received from the external tool.

→ the PDU (packet Data unit) can consist of more than 8 data bytes
→ ex: the 24 bytes → 3 consecutive slave responses can form the response data (24 bytes)

* the response Frame

→ Node Address for Diagnostics (NAD):

- the first byte of the frame response always contains the NAD, regardless of whether it is a master frame response or a slave frame response
- the NAD represents the formal node address of the slave that is being configured or diagnosed.
- When a master sends a diagnostic request, it uses the NAD as the address to which the slave should respond.
- the slave device recognizes the initial NAD and sends its response accordingly.
- By assigning a specific NAD, each slave device in the network can be uniquely identified and addressed during diagnostic or configuration operation.

NAD Value	Description
0	reserved for go to sleep command.
1-125 (0X7D)	Slave node addresses
126 (0X7E)	functional node address (NAD), only used for diagnostics (using the transport layer)
127 (0X7F)	slave node address broadcast (broadcast NAD)
128 (0X80) -255 (0XFF)	Free usage Diagnostic frames with the first byte in the range 128 to 255 are allocated for free usage since LIN 1.2 standard. see user defined diagnostics section 5.2.6.

2 → Protocol control information (PCI)

- PCI is present in the second byte of the frame response and provides information about the transfer mode
- in the ISO transport protocol, there are two types of data transfer:
  1- unsegmented data 2- segmented data

1- unsegmented data transfer:

→ it consists of a single master request frame and a single slave response frame (SF) → single frame

→ when we want to configure a node, the request we send to it for diagnostics is usually unsegmented (SF)

→ single Frame (SF): this type indicates that the transported message fits into a single PDU (Package Data Unit). this type of Frame can contain a maximum of five data bytes, plus one additional byte for the service ID (SID / RSID) byte.

NAD	PCI	SID	D1	D2	D3	D4	D5

PCI_type = SF

Type	B7	B6	B5	B4	B3	B2	B1	B0
	PCI type				additional info			
SF	0	0	0	0	length			

← PCI Byte →

Length field → Max = 5 for 5 data bytes will be transfer = 0b0101 + SID byte = 0b0110.

2- segmented data transfer:

→ it involves multiple Frames to transmit a message, with Frames types FF (first Frame) and CF (Consecutive Frame).

→ First Frame (FF): is used to indicate the start of a multi PDU message.

→ the following Frames will be of the Consecutive Frame (CF) type

→ total number of data bytes in the message plus one (for SID or RSID) shall be transmetted as length. 4 bits of PCI and 8 bits of LEN byte, so the maximum is → 0xFFF = 4095 bytes.

→ Consecutive Frame (CF): is used for multi PDU message.

Type	PCI type				additional info			
	B7	B6	B5	B4	B3	B2	B1	B0
SF	0	0	0	0	length → max = 5+1			
FF	0	0	0	1	length /256			
CF	0	0	1	0	Frame Counter			

Length → 0 ··· 6, which is the maximum payload Length in single Frame message.

→ Frame Counter in first CF = 1, in second CF = 2 ··· etc, if more than 15 Frames, Frame Counter warps around and Continues with 0, 1, 2 ····

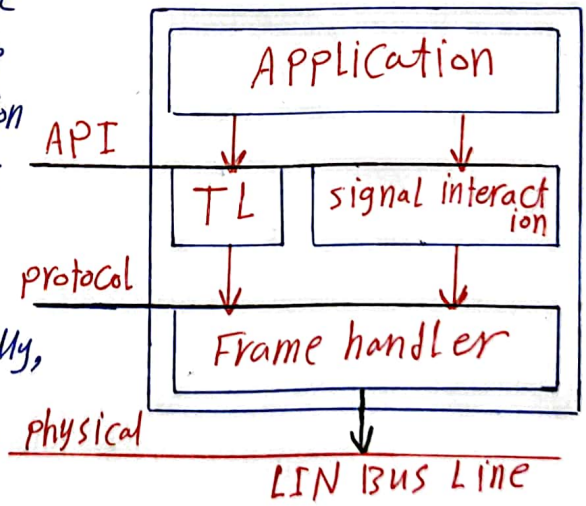
SF	NAD	PCI = SF	D0	D1	D2	D3	D4	D5
FF	NAD	PCI = FF	LEN	D0	D1	D2	D3	D4
CF	NAD	PCI = CF	D0	D1	D2	D3	D4	D5

→ 8 bytes each.

D0 → SF & FF → SID

* LIN Diagnostics support

→ in a cluster, a node connects to the physical bus wire using a Frame transceiver. However, the application does not directly access the frames on the bus. instead, there is a single-based interaction Layer added as an intermediary. additionally, there is a transport layer interface that exist between the application and the frame handler.



→ the TL and signal interaction, enable the application to indirectly access and interact with the Frames on the bus, ensuring proper Communication within LIN network.

*·3→ Service Identifier (SID): specifies the type of request that a slave node should perform when addressed

→ 0 to 0XAF and 0XBB to 0XFE are used for diagnostics while 0XB0 to 0XB7 are used for node Configuration.

→ Response service Identifier (RSID): specifies the contents or type of response that the slave node will provide to the master node after processing the request.

→ the RSID for a positive response is always → RSID = SID + 0X40

→ some of available services:

SID	service	type
0-0XAF	reserved	reserved
0XB0	Assign NAD	optional
0XB1	Assign Frame identifier	obsolete (outdated)
0XB7	Assign Frame identifier range	Mandatory
0XB8-0XFF	reserved	reserved
0XB2	Read by identifier	Mandatory

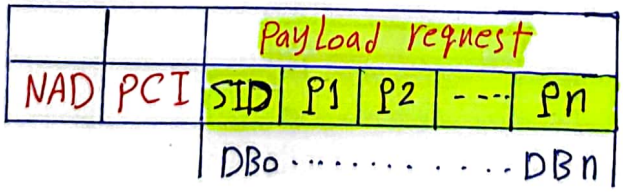
→ only the services 0XB2 and 0XB7 must always be supported by a slave, the other are optional

→ the service 0XB1 was deleted in LIN spec V.2.1, but we often have to implement it in a master if we have LIN V.2.0 slaves connected there. and it was replaced in LIN V.2.1 by the service 0XB7

* DTL Standard payload Layout:-

→ DTL (Diagnostic Transport Layer) Request service:-

- the SID is always located in the first byte of the payload



- When a positive response is received:
  - → RSID = SID | 0X40
  - → the RSID also located in the first byte of the payload (DB0).
- When a negative response is received:
  - → the first byte of the payload is set to 0X7F, indicating a negative response
  - → the second byte of the payload contains the original SID

→ the third byte of the payload provides an error code that further describes the reason for the negative response.

		Payload Negative response			
		DB0	DB1	DB2	DB3 ..... DBn
NAD	PCI	0X7F	SID	error code	not used .....

---

* LIN Product identification :.

→ According to LIN specification, each LIN V.2.X node has a unique product identification.

→ the product identification is composed of three values:

1- Supplier Id:

→ this is a 16-bit number assigned to the manufacturer by the CIA.

→ the most significant bit of the supplier Id is always set to 0.

2- Function Id:

→ this is a 16-bit manufacturer-specific number that identifies a specific product.

→ products that differ in LIN communication or in their properties at the interfaces should have different function Ids.

3- Variant:

→ this is an 8-bit number that should be changed whenever the node undergoes functional changes.

→ it helps distinguish different versions or variants of the same product.

→ supplier Id and function Id are required in some diagnostic services as parameters in the master request

→ Node configuration and identification service is transported by the transport layer only in a single Frame.

→ Remember:

NAD → 0X7F → broadcast NAD

SID → 0X B0 → Assign NAD

PCI → transfer mode and Length of data.

---

*Diagnostic service Assign NAD: → SID = 0XB0

→ Diagnostic request → the header with ID 60 (0X3C)

NAD	PCI	SID	D1	D2	D3	D4	D5
initial NAD	0X06	0XB0	X	y	Z	W	new NAD

X → supplier ID LSB
y → supplier ID MSB
Z → Function ID LSB
W → Function ID MSB

→ if only slave is connected, you can also use the wildcard NAD 0X7F, which allows the master node to communicate with the single slave without knowing its specific address.

→ Wild cards

NAD → 0X7F supplier ID → 0X7FFF

Function ID → 0X FFFF

→ Not all slaves allow the reconfiguration of the NAD.

→ A positive response → the header with ID 61 (0X3D)

NAD	PCI	RSID	unused				
initial NAD	0X01	0XF0	0XFF	0XFF	0XFF	0XFF	0XFF

*Diagnostic service Read data by ID: → SID = 0XB2

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	0X06	0XB2	Identifier	W	X	y	Z

W → supplier ID LSB
X → supplier ID MSB
y → function ID LSB
Z → function ID MSB

→ the layout of the response data depends on the requested identifier.

→ the SID = 0XB2 for positive response → RSID = 0XB2 | 0X40
= 0XF2

For example :-

Identifier	interpretation	Length of response
0	LIN product identification	5 + RSID
1	Serial number	4 + RSID
2-31	Reserved	-
32-63	user defined	user defined
64-255	Reserved	-

→ for example

ID	NAD	PCI	RSID	D1	D2	D3	D4
0	NAD	0X06	0XF2	SUP-ID LSB	SUP-ID MSB	Fun-ID LSB	Fun-ID MSB
1	NAD	0X05	0XF2	serial 0 LSB	serial 1	serial 2	serial 3 MSB
32-63	NAD	0X04	0XF2	user defined	user defined	user defined	user defined

0X02-0X06

D5
Variant
0XFF
user defined

unused

---

* Diagnostics Frame → Network Management Frames :

→ Network management in a LIN cluster refers to cluster wake-up and go-to-sleep operations.

→ the master initiates Bus sleep mode by transmitting a go-to-sleep command.

→ the go-to-sleep command is a master request frame with the first data field set to 0 and the rest set to 0XFF.

Data 1	Data 2	.....	Data 8
0	0XFF	0XFF	0XFF

→ go-to-sleep command

→ the slave nodes shall ignore the data fields 2 to 8 and interpret only the first data field.

→ in case of bus inactivity, a slave node must be able to receive / transmit frames for 4 seconds.

→ the slave node shall automatically enter bus sleep mode earliest 4 sec and Latest 10 sec of bus inactivity.

→ Bus inactivity is defined as no transitions between recessive and dominant bit values. Bus activity is the inverse.

* LIN Wakeup :.

→ Wakeup is one task that may be initiated by any node on the bus (a slave as well as the master)

→ Per the LIN 2.0 specification, the wake up request is issued by forcing the bus to the dominant state for 250 microseconds to 5 milliseconds.

→ each slave should detect the wakup request and be ready to process headers within 100 milliseconds.

→ the master should also detect the wake up request and be ready to send ~~process~~ headers within 100 to 150 ms after receiving the wakeup request and when the slave nodes are ready.

→ if the node that transmitted the wake up signal is a slave node, it will be ready to receive or transmit frames immediatly.

→ if the master node not issue headers within 150 ms after receiving the first wakeup request, then the slave requesting wakeup may try issuing a second wakeup request (and waiting for another 150 ms)

→ if the master still does not respond, the slave may issue the wakeup request and wait 150 ms a third time

→ if there is still no response, the slave must wait for 1.5 seconds before issuing a forth wakeup request.