LIN (Local Integrated Network)

maximum speed: 20kbps
length: 40m depending on data rate

- Created for low cost, low-end multiplexed communication in automotive
- CAN network addresses the need for high bandwidth / advanced error handling, but h/w & s/w cost of implementation make CAN less popular for low performance devices like power window / seat controllers

Why LIN if CAN ??

- LIN used where cost efficient communication where versatility of CAN is not required
- can implement LIN relatively inexpensively using low cost UART into most modern 8 bit microcontrollers

No need for a seperate chip like CAN →

modern automotive system ⟹ (examples)

LIN	Low cost applications	Body electronics
CAN	mainstream applications	powertrain & Body commn.
FlexRay	High speed synchronized commn	active suspension

imp

- one master multiple slave
- (transmitted by master) Header / Response (transmitted by slave resides in master node or slave node) (2,4,8 byte)

once id received & identified by slave, slave begins response

Break	Sync	Identifier	Data 1 - 8 byte	Checksum 8 bit

(LIN message format)

- Break: SOF
- Sync: (0x55) to sync clock with slave (used by slave) by setting the same baud rate
- Identifier: 6 bit message ID + 2 bit parity, total 8 bit
- Response: once slave identifies, sends response with 1-8 byte data & 8 bit checksum

Imp *

* LIN Message Format :-

- LIN bus is a polled bus (imp)
- One master & one or more slave
- master device contains both master task & slave task ← whereas each slave contains only slave tasks
- slave contains only slave task

- Communication is controlled entirely by master
- master using master task, sends header
- slave responds using response. Response can be given by slave task or master task

imp

imp

Operational Flow

- Normally master task polls each slave task using header
- Prior to starting LIN, each slave task is configured to
  (1) publish data to the bus OR
  (2) subscribe to data in response to each received header ID

How slave determines to publish or subscribe?
- It verifies ID parity and ID, to determine if it needs to publish or subscribe

If slave needs to publish :
↳ transmits 1-8 data byte to bus + 8 bit checksum

If slave needs to subscribe :
↳ reads data payload + checksum & does internal action

(most simple explanation)

* How LIN works :

① Initialisation : A node is initialised during power up or reset. Awaits break signal from master

② Header : Master sends frame header, consisting of break, sync, and ID

③ Response : After the header, corresponding slave responds with data and checksum

* LIN in Embedded system :

① Microcontroller with UART :

② LIN Transceiver : UART only is fine but its better to use LIN transceiver for better signal integrity and voltage levels

③ Software : Implement a LIN protocol stack
Libraries within microcontroller vendors available

④ Master node : Choose master and implement control logic for communication timings

⑤ Slave nodes : Implement slave functionality in nodes that will respond to the masters requests

* LIN is deterministic meaning master decided when data is transmitted and which slave will respond. But schedule (meaning which msg to be send is defined prior and known to all the nodes

* Communication flow:

① Initialisation:

- On system power up/reset, all nodes start in an initialised state
- They remain silent, waiting for master to initiate communication

② Master sends Header:

- Send: BREAK field
  - prolong dominant signal (logic low)
  - signify start of new frame and inform all other nodes
- Sync field
  - 0x55
  - A known pattern that helps slave to synchronise with master by updating baud rate
- Sends ID
  - This ID determines which data/message to be communicated

③ Slave Response:

- a data field
  - Based on ID specific slave responds
  - sends upto 8 bytes of data or it remains silent if its a master request frame (i.e. master broadcasting data
- checksum

④ Sleep mode / Wake up

- if no transmission for certain duration, nodes transition to sleep mode
- prolonged dominant state (logic low) to wake up

⑤ Error Handling:

- No advance error handling
- detects error using checksum in response
- Contains local error counter