E BLOCKS®
LEARN·DESIGN·BUILD

LIN Bus Communications

EB929-80-03

MATRIX

www.matrixmultimedia.com

EB929

LIN Solution

Course Notes

1. Teacher's notes

LIN - Local Interconnect Network

These teacher's notes have been written as a brief introduction to the LIN bus system and the contents of the Matrix Multimedia LIN Solution.

This manual is divided into three main sections:

- A brief overview of some of the main features of LIN
- An introduction to the Matrix Multimedia Flowcode implementation of LIN and the hardware components of the LIN Solution.
- A series of exercises and demonstrations to help reinforce the learning process through practical experience.

The exercises are intended to be carried out in the order they are presented. Each one builds on the knowledge gained from the previous one, resulting in the development of a functional network that demonstrates many of the capabilities of LIN.

The final exercise is intended to be used as a demonstration and is not essential to the understanding of LIN. It also requires the availability of a complete Matrix Multimedia CAN Solution, and demonstrates the integration of the two networks.

Students who are not required to undertake the programming tasks in the exercises may download the example programs from the CD-ROM and use them to provide practical demonstrations of LIN bus operation. In this case the exercise material in the manual should be treated as a description of the associated programs and the features being demonstrated.

Although it is not necessary for all students to be proficient in Flowcode programming, it is recommended that a supervisor should have some Flowcode experience for debugging purposes.

Some students might gain extra insight into the operation of LIN by examining the bus signals with an oscilloscope.

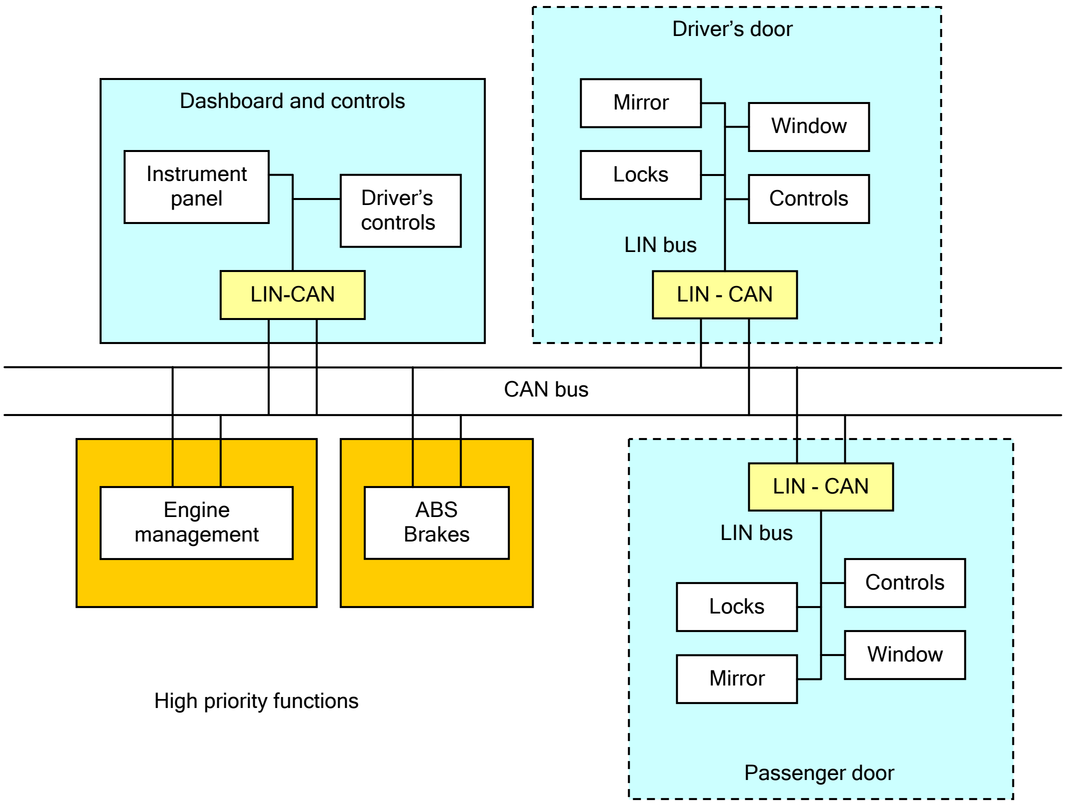
2. LIN BUS - Introduction

LIN (Local Interconnect Network) is a bus system developed by a group of vehicle manufacturers – the LIN consortium - and is aimed mainly at the automotive industry. It has been designed to provide a low-cost networking solution for low-speed, low-risk applications as an alternative to the high speed, high integrity, higher cost CAN (Control Area Network) bus which is required for engine management, braking and safety features.

The development of LIN allows the creation of low cost networks of localized functions. These would typically be made up of groups of controls and actuators that do not require the sophistication and cost of a direct CAN bus connection, e.g. the control switches and motors required to operate the window and adjust the wing mirror on a car door.

A complete LIN system can be connected to the main CAN bus as a single node

Although it is expected to work in conjunction with CAN bus, the LIN bus it is not directly compatible and requires a LIN-CAN translator to allow the transfer of data between the two networks.



Some of the main application specific features of the LIN bus are:

- Development for implementation using low cost hardware. Data is transmitted on a single wire, relying on the system ground to provide a return path for the signals and thus saving money on wiring and connectors.
- The transmission protocol includes synchronization bits that allow the slave nodes to measure the bit rate of each message from the master node, on a message-by-message basis. This counteracts the effects of component inaccuracies and allows low-cost frequency reference components to be used in place of the expensive crystals required for more time critical systems.

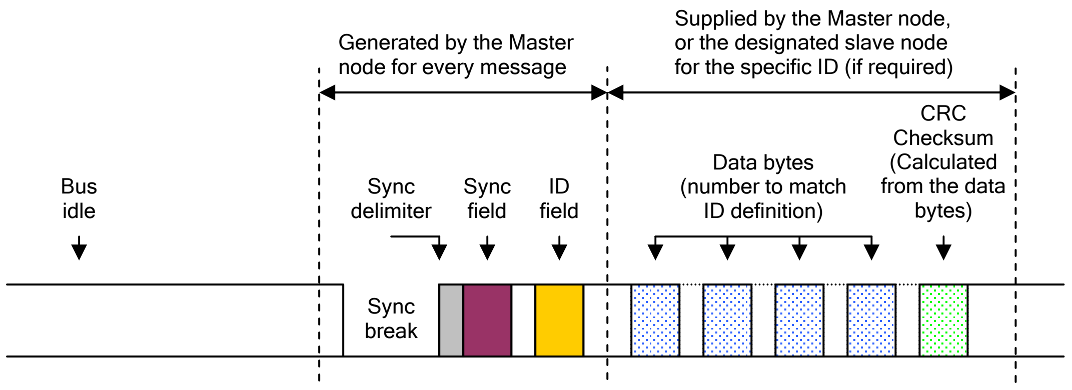
2.1 Features

LIN has been developed specifically for low-cost automotive applications. A number of features of both the hardware design and communication protocol have been included to assist in this aim.

2.2 Communications

LIN bus messages consist of:

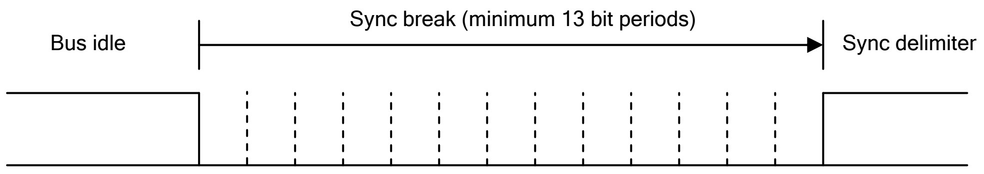
- A synchronization break
- A synchronization field
- An identifier field – including two parity bits calculated from the identifier value and added to the message; this is used by the receiving node(s) to test the integrity of the identifier value.
- Up to 8 bytes of data
- A checksum byte - calculated from the data bytes and added to the message; this is used by the receiving node(s) to test the integrity of the received data.


Generated by the Master node for every message
Supplied by the Master node, or the designated slave node for the specific ID (if required)
Bus idle
Sync delimiter
Sync field
ID field
Data bytes (number to match ID definition)
CRC Checksum (Calculated from the data bytes)
Sync break

The master node initiates **all messages** by transmitting the Sync Break, Sync field, and ID. If data is required for the message, one node must respond by writing the data and the CRC byte to the bus; this can be either the master node or one of the slave nodes. Any nodes configured to read the message will simply detect a complete message on the bus and will not be aware of the source of the data.

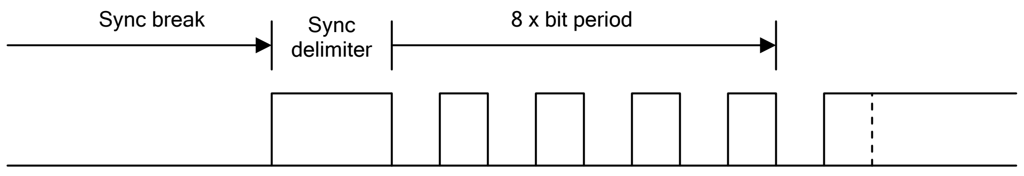
2.3 Sync Break

LIN bus has a single master node that initiates all communications. A message sequence is initiated by transmitting a Sync Break. This is a continuous low-level signal for a minimum of 13 data bit intervals (the bus voltage is high when idle). This must be detected as a special event by the slave nodes and used to force them to start the detection process for the following message. The slave nodes must time this interval using their local frequency references and compare the duration with the information from the following synchronization field. The bus voltage returns to the high level for a short period to signal the end of the sync break.



2.4 Sync Field

The sync field is transmitted a short time after the sync break and consists of a sequence of ten alternating low and high bits, providing a clock reference for the slave nodes to synchronize with.



Starting with the first falling edge, and timing the 4 subsequent falling edges, a value for 8 bit periods can be measured - allowing simple division to find a single bit period. If the timing reference of a slave node does not match that of the master node (due to component tolerances, thermal drift etc) this sequence should be sufficient to allow the slave node to acquire a bit period measurement for each individual message, based on its own timing reference.

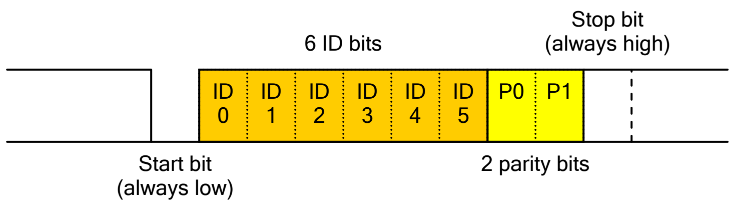
The calculated bit period must be compared with the preceding Sync break period to confirm that the sync break was valid (minimum of 13 bit periods).

2.5 ID field

LIN bus does not send messages directly between nodes using node addresses. Instead, all messages contain a message identifier (ID) field. Each node that is required to recognise a particular message must be configured to respond to the detection of the ID, and to either read or write any associated data. Only one node must be configured to write the data and CRC field for any individual message ID, and all nodes recognizing the ID must be configured to expect the correct number of data bytes for that message.

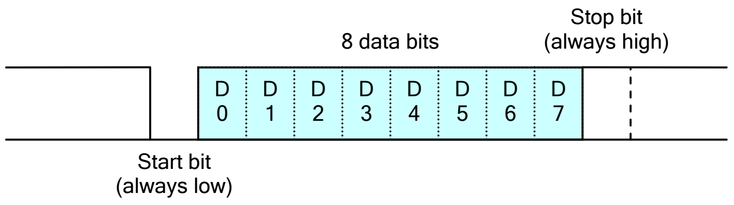
The ID field contains eight data bits: 6 ID bits and 2 Parity bits (P0, P1)
The 6 ID bits allow a maximum of 64 message ID values to be allocated (four of these values have been reserved by the LIN consortium for special functions).

The two parity bits are calculated from the six ID bits and added to them to form an 8-bit byte; Flowcode does this automatically. All slave nodes must confirm that the received parity bits are correct for the received ID bits before responding to a message; Flowcode also performs this function automatically.



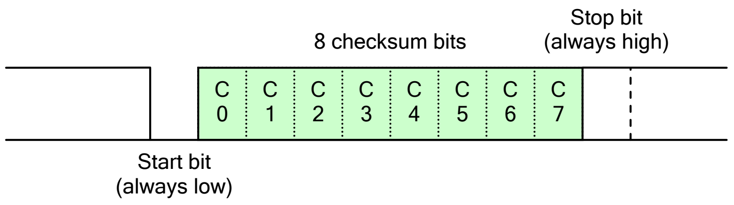
2.6 Data field

Every message type allocated a message ID must also be configured, in every node recognizing it, to contain a fixed amount of data - 0 to 8 bytes. Each data byte is formatted with 1 start bit, 8 data bits and one stop bit. Data can be transmitted by the master node, or a designated slave node, depending on the message ID.



2.7 Checksum field

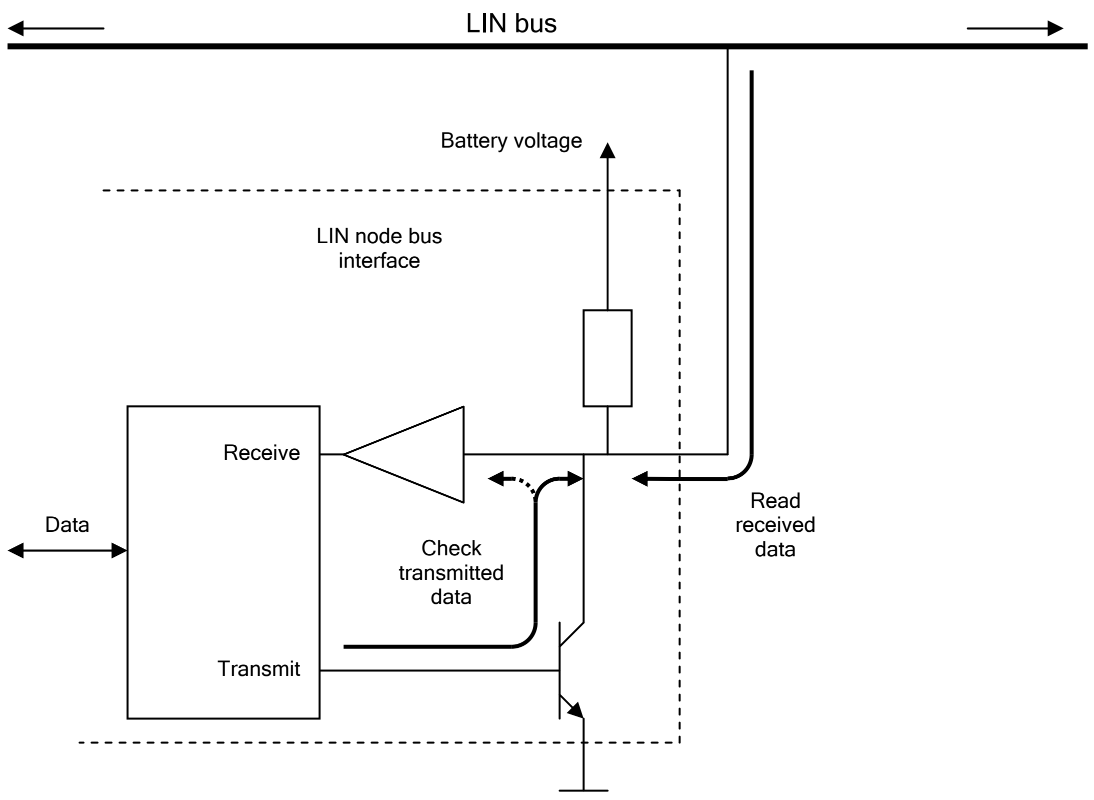
Every message that includes data must also have a checksum field added. This is an 8-bit value that is calculated from the data values and added to the message; Flowcode performs this function automatically. All nodes reading the message should match the checksum with a pattern generated from the message data to test the integrity of the received data; Flowcode performs this function automatically.



2.8 Bus interface

The LIN bus uses a single wire to transmit all control and data signals, saving component and assembly costs. The bus can operate at vehicle battery voltage levels (8-18v), via pull-up resistors at each node. The signal line is passively pulled high by the resistors and can be actively pulled low by a transistor in each of the network nodes. If more than one node attempts to transmit on the bus at any time, a node transmitting a 0 (low) will override any nodes transmitting a 1 (high): 0 = Dominant, 1 = Recessive.

Any node transmitting on the bus must use its receiver to monitor the bus. If the signal level being transmitted does not match the signal level being received, due to transmission from another node or a bus fault, the node must abort the transmission.



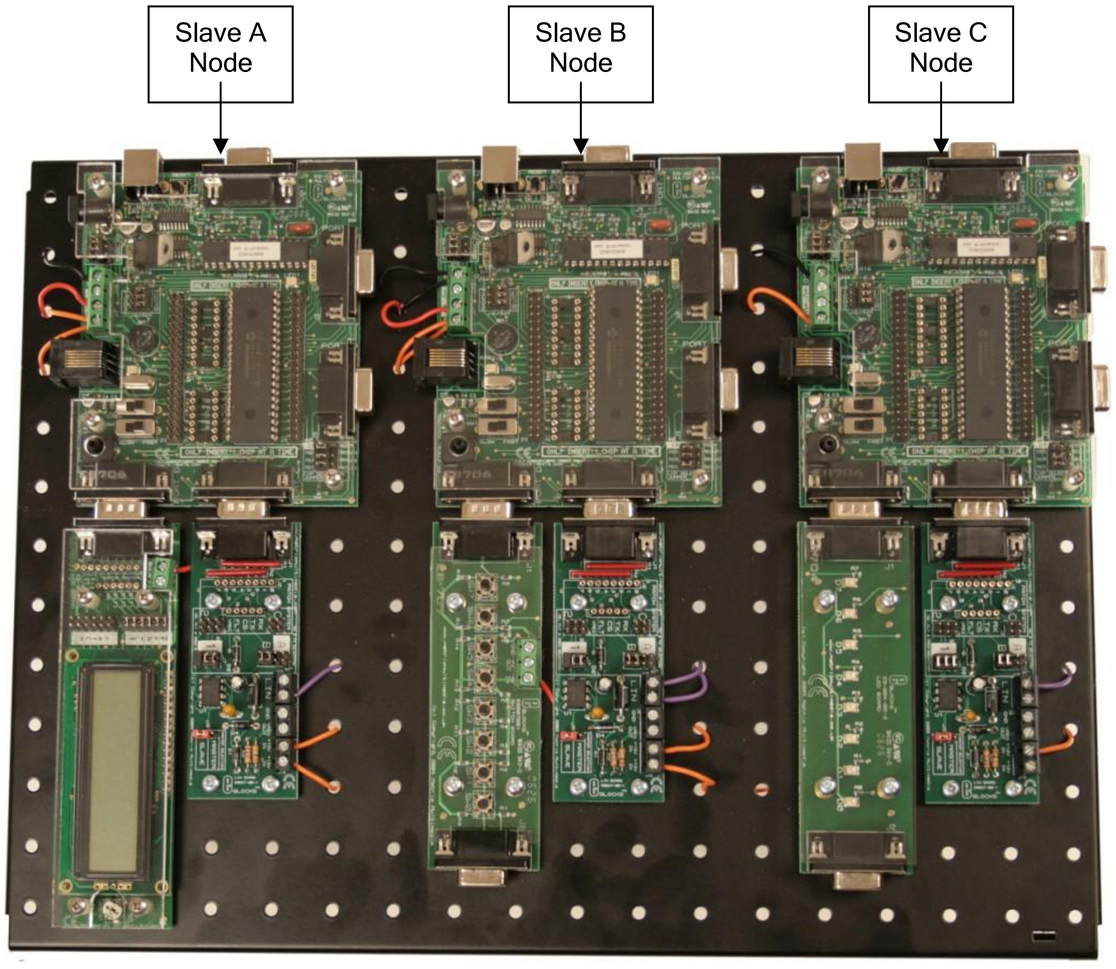
LIN Bus interface schematic

3. LIN training solution

3.1 Hardware

This course is based around a LIN training solution that contains four LIN nodes. Each node represents an automotive ECU (Electronic Control Unit) and consists of a multi-programmer board with a LIN interface e-block and input or output device e-block. This allows genuine applications to be developed, simulated and demonstrated.

One node is designated as the bus master node and the others are slave nodes.



LIN slave nodes