

EB929
LIN
Solution
Course Notes

© Copyright Matrix Multimedia Limited 2009

1.	Teacher's notes	4
2.	LIN BUS - Introduction	5
2.1	Features	6
2.2	Communications	6
2.3	Sync Break	6
2.4	Sync Field	7
2.5	ID field	7
2.6	Data field	8
2.7	Checksum field	8
2.8	Bus interface	9
3.	LIN training solution	10
3.1	Hardware	10
4.	Flowcode components	11
4.1	LIN Master	11
4.1.1	Configuration	11
4.2	LIN_Slave	3
4.2.1	Configuration	3
4.2.2	Functions	2
4.3.	Demonstrations and Exercises	3
5.	Exercise 1 - Master node data transmission	4
5.1	Introduction	4
5.2	Theory	5
5.3	Objectives	6
5.4	Requirements	6
5.5	Implementation	7
5.5.1	Master node	7
5.5.2	Slave C node	7
5.6	Learning outcome	9
5.7	Further work	9
6.	Exercise 2 - Multiple slave nodes	10
6.1	Introduction	10
6.2	Theory	10
6.2.1	Network operation	10
6.2.2	Message format	11
6.2.3	Message ID	11
6.2.4	Bus master	11
6.2.5	Bus slaves	11
6.3	Objectives	13
6.4	Requirements	13
6.5	Implementation	14
6.5.1	Master Node	14
6.5.2	Slave A node	14
6.6	Learning outcome	15
6.7	Further work	15
7.	Exercise 3 - Slave data transmission	16
7.1	Introduction	16
7.2	Theory	16
7.2.1	Master node	16
7.2.2	Slave node	17
7.2.3	Network theory	18
7.3	Objectives	19
7.4	Requirements	19
7.5.1	Slave B node	20
7.5.2	Master node	20
7.6	Learning outcome	20
7.7	Further work	20
8.	Exercise 4 - Slave to Slave transmission	21
8.1	Introduction	21
8.2	Theory	21
8.3	Objectives	22

8.4	Requirements	22
8.5	Implementation.....	23
8.5.1	Radio Station Display Node.....	23
8.5.2	Master Node	23
8.6	Learning outcome.....	23
8.7	Further work	23
9.	Exercise 5 - LIN-CAN interface (Demonstration)	24
9.1	Introduction.....	24
9.2	Theory	25
9.3	Requirements	26
9.4	Preparation.....	26
9.5	Operation.....	27

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.

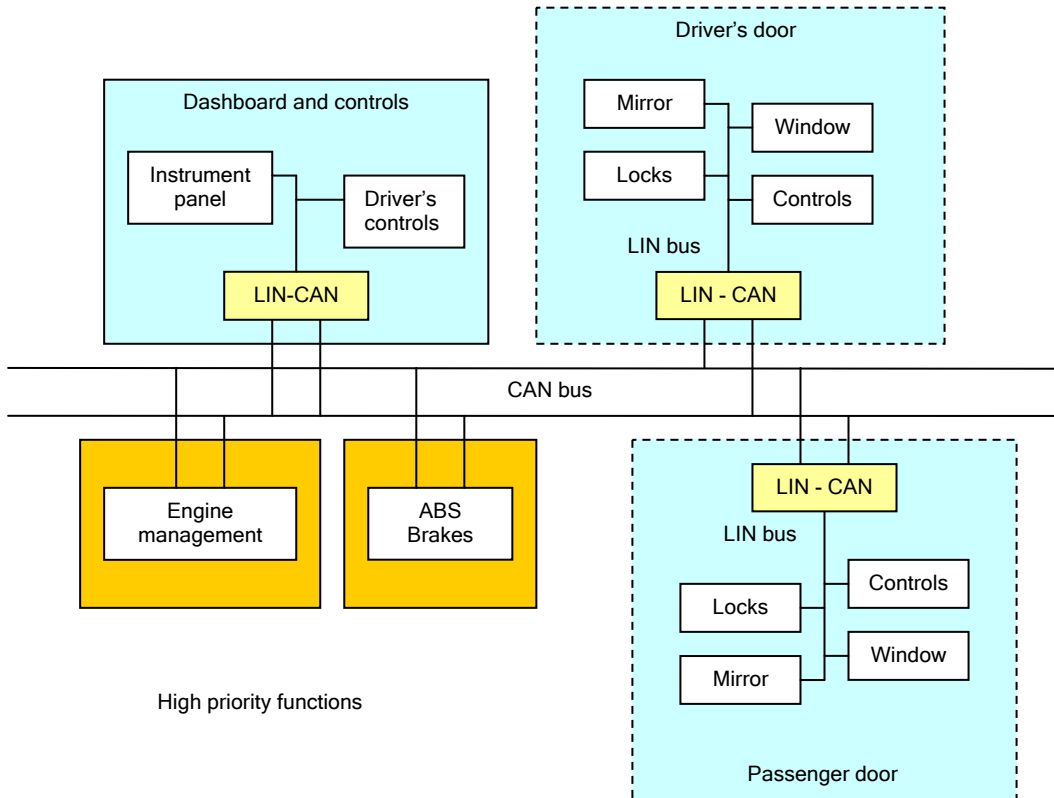
2. LIN BUS - Introduction

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.

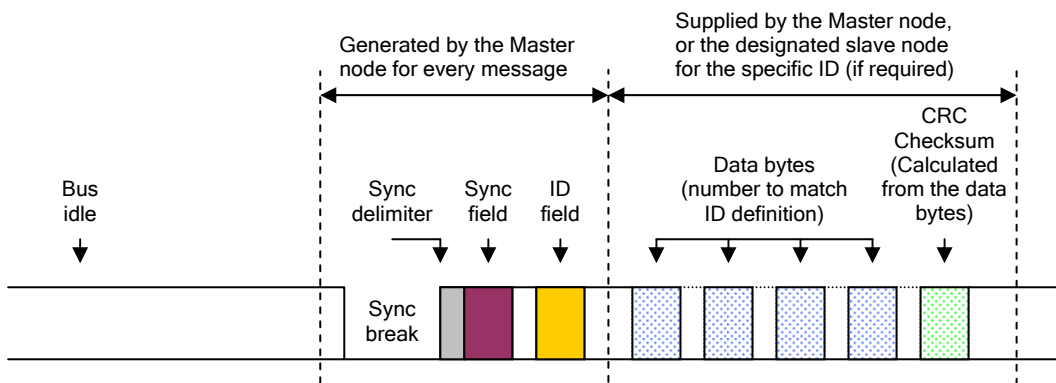
2.1 Features

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:

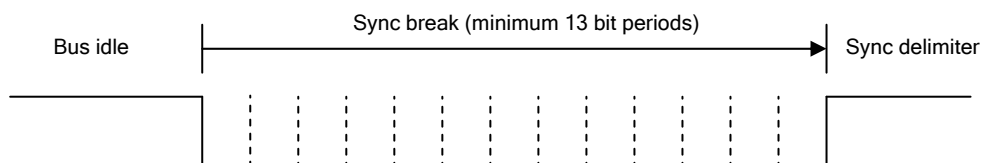
- A synchronization break
- A synchronization field
- An identifier field - including two parity bits calculated from the identifier value and added to the message; this is used by the receiving node(s) to test the integrity of the identifier value.
- Up to 8 bytes of data
- A checksum byte - calculated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



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.

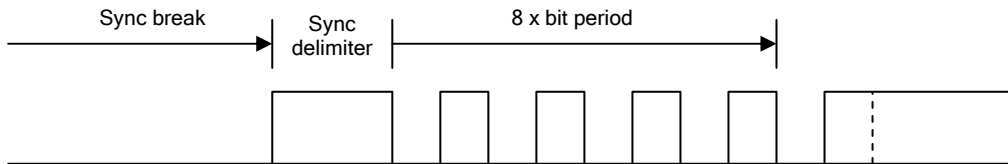
2.3 Sync Break

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).

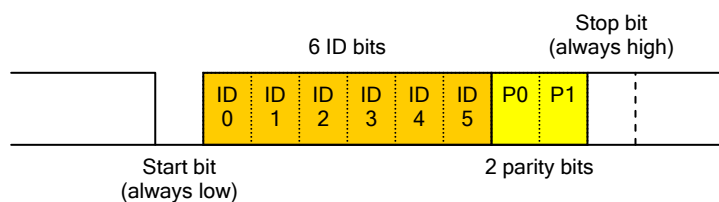
2.5 ID field

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)

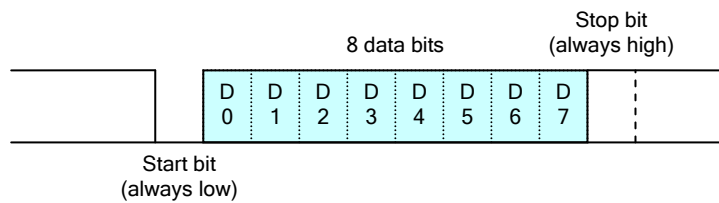
The 6 ID bits allow a maximum of 64 message ID values to be allocated (four of these values have been reserved by the LIN consortium for special functions).

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.



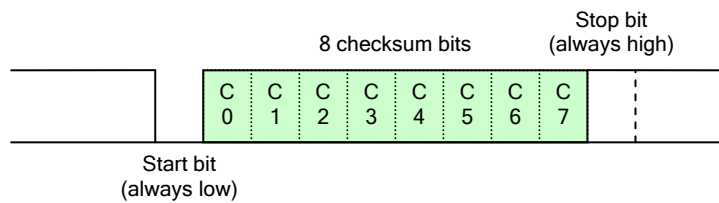
2.6 Data field

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

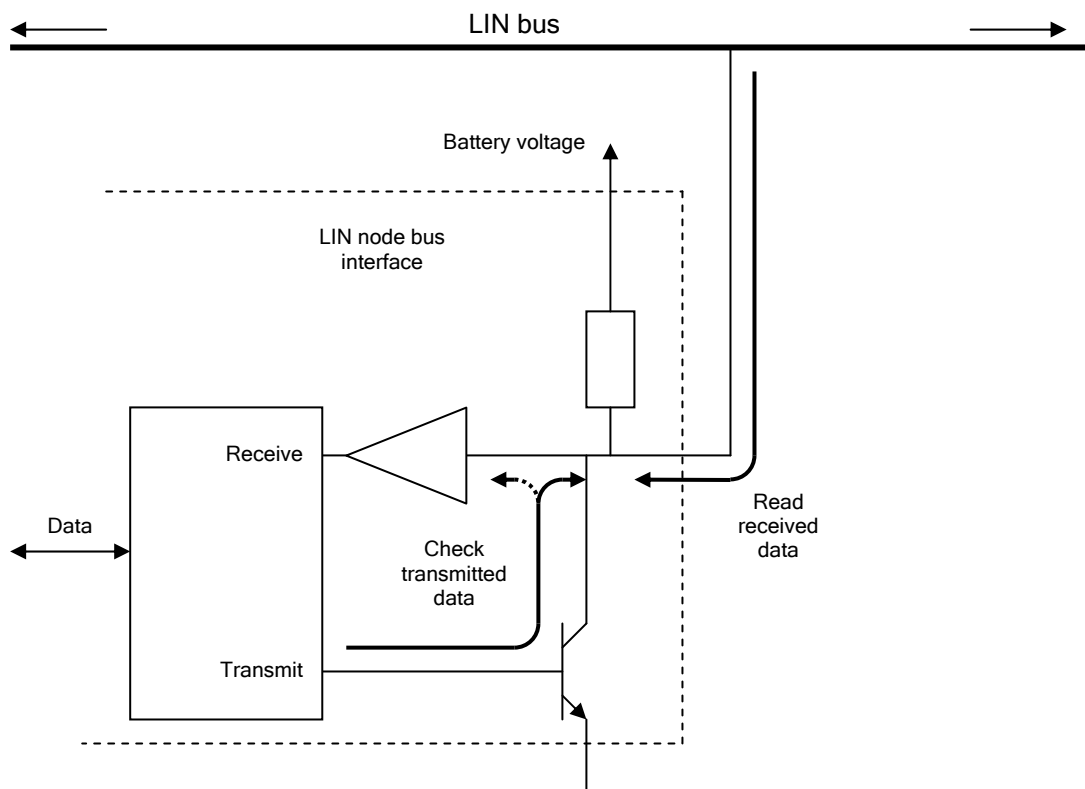
Every message that includes data must also have a checksum field added. This is an 8-bit value that is calculated from the data values and added to the message; Flowcode performs this function automatically. All nodes reading the message should match the checksum with a pattern generated from the message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



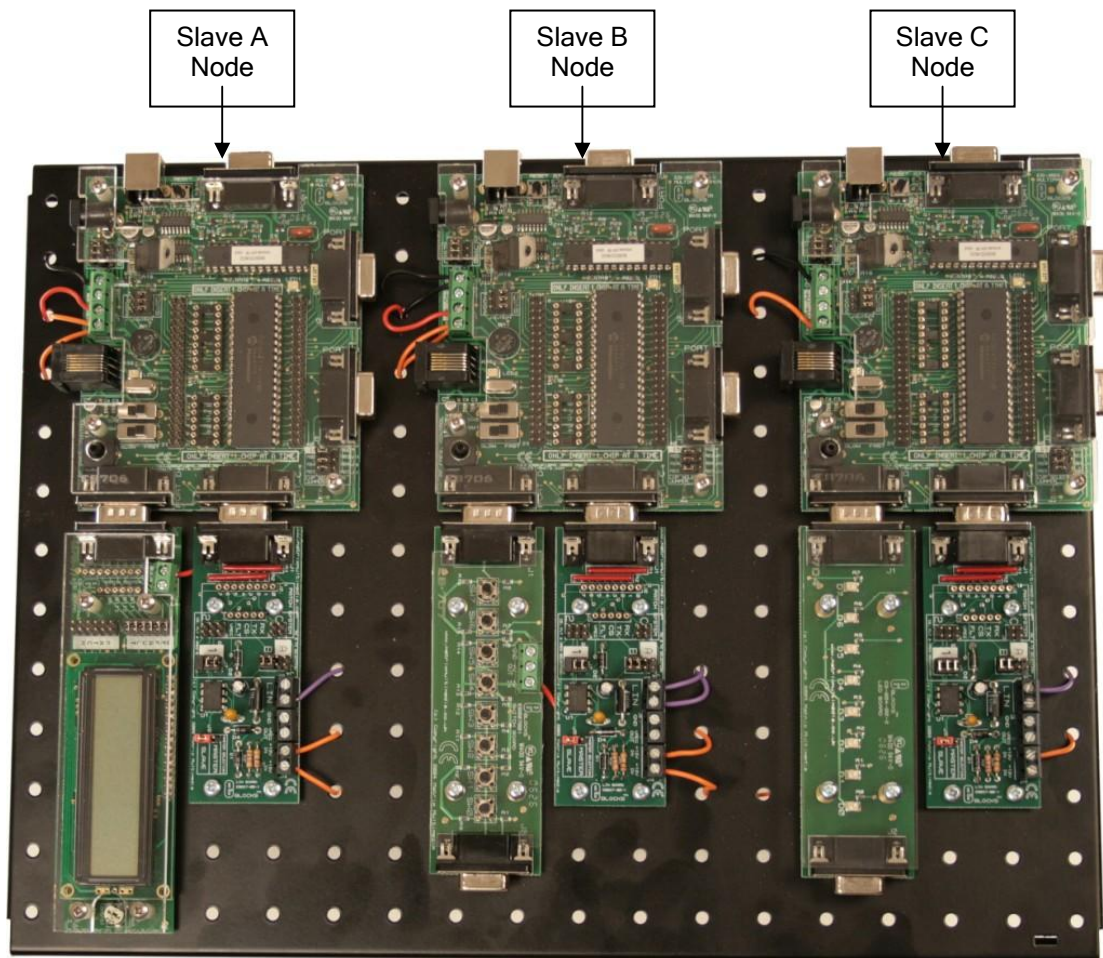
LIN Bus interface schematic

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.



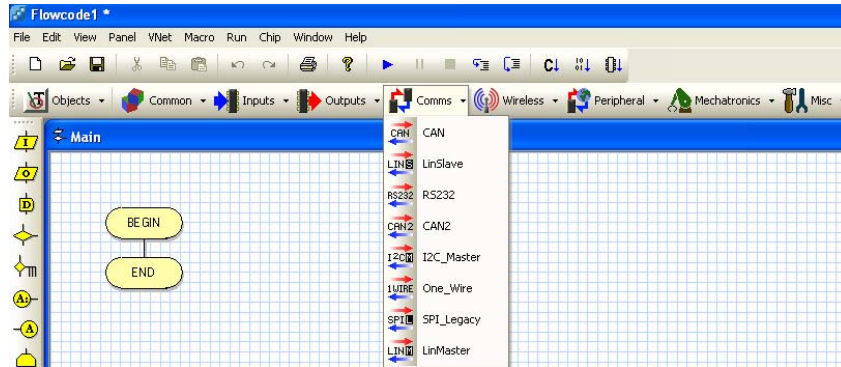
LIN slave nodes

4. Flowcode components

To assist with the understanding and development of practical LIN bus applications, Flowcode has been supplied with two components that control the LIN bus interfaces, support the necessary software functions, and provide a user-friendly programming interface.

4.1 LIN Master

The LIN Master component can be found in the Comms section of the Component Toolbar



The LIN Master component is added to the flowchart by clicking on its name or menu icon.



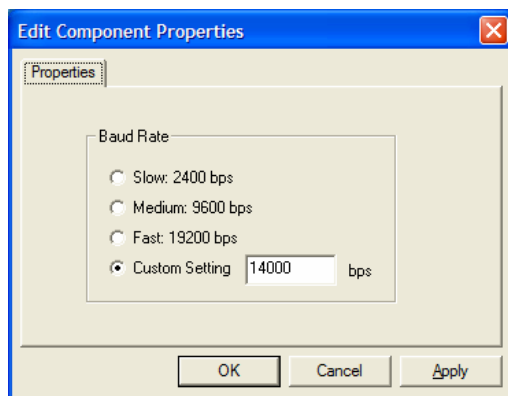
The Panel icon of the LIN Master component

This component must be added into programs that are written for a LIN bus master node (there must only be one master node on any LIN bus). The software functions and configuration options provided are specific to master node operation.

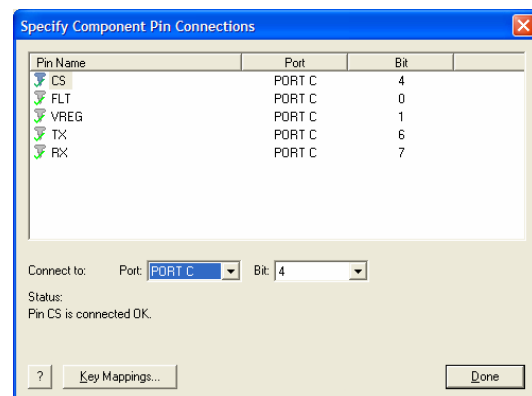
4.1.1 Configuration

The LIN Master component can be selected by clicking on its Panel icon, or by selecting it from the drop down menu at the top of the Properties window.

The following windows can then be activated by clicking on the three dots next to Ext Properties or Connections.

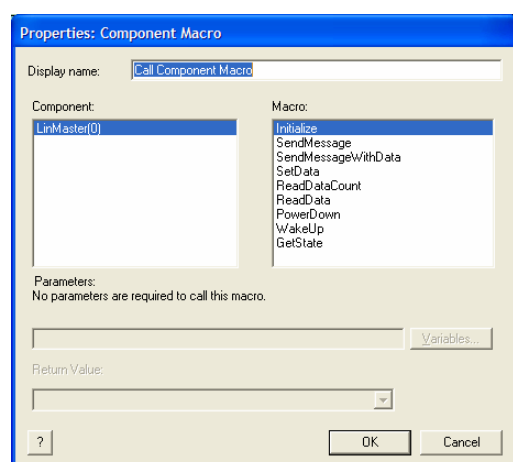


Property page - Baudrate selection



Component connection window

4.1.2 Functions



Function Selection panel.
Appears when a program Component Macro
box is selected

Functions:

Initialize

Initializes LIN Master operation. Must be executed before any other LIN functions.

SendMessage

Initiates a LIN message but does not transmit any data - a slave node is expected to respond with appropriate data.

SendMessageWithData

Initiates a LIN message and transmits data. See the SetData function.

SetData

Sets eight data values for transmission by the SendMessageWithData function.

ReadDataCount

Returns the number of data bytes received with the last message.

ReadData

Read a specified byte from the most recently received message.

PowerDown

Disable operation of the LIN interface device - Power saving.

WakeUp

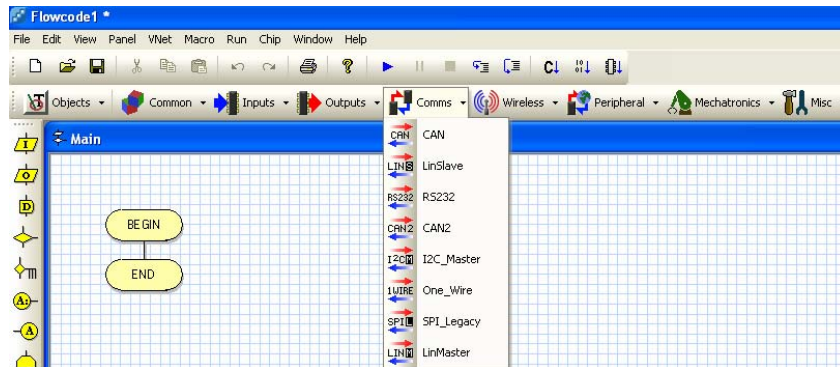
Enable operation of the LIN interface device.

GetState

Read the current state of LIN operation. Includes fault conditions and operating states.

4.2 LIN_Slave

The LIN Slave component can be found in the Comms section of the Component Toolbar



The LIN Slave component is added to the flowchart by clicking on its name or menu icon.



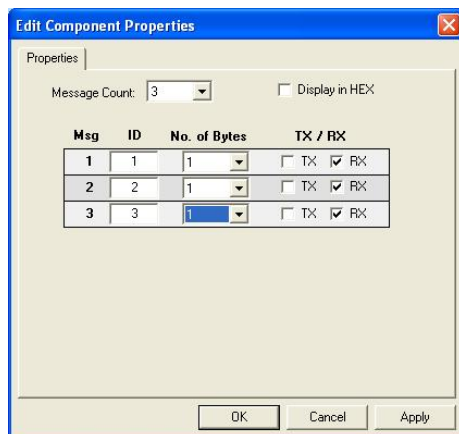
The Panel icon of the LIN Slave component

This component must be added into programs that are written for a LIN bus slave node. The software functions and configuration options provided are specific to slave node operation.

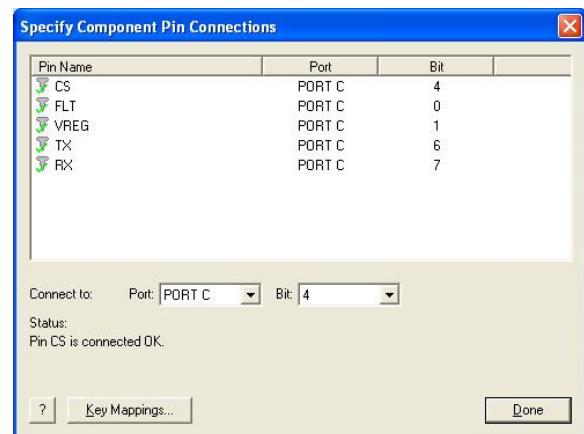
4.2.1 Configuration

The LIN Slave component can be selected by clicking on its Panel icon, or by selecting it from the drop down menu at the top of the Properties window.

The following windows can then be activated by clicking on the three dots next to Ext Properties or Connections.

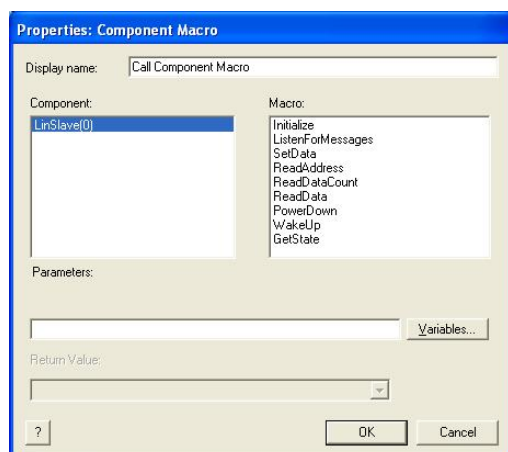


Properties - Message configuration



Component connection window

4.2.2 Functions



Function Selection panel
Appears when a program Component Macro box
is selected.

Functions:

Initialize

Initializes LIN Slave operation. Must be executed before any other LIN functions.

ListenforMessages

Waits for a LIN message:

- Transmits the appropriate data if a TX message ID is recognized.
- Reads the data if an RX message is recognized.

Returns the Msg value of the message (see the Message Configuration panel).

SetData

Sets eight data values for a selected message.

ReadAddress

Read the ID value of the last message received.

ReadDataCount

Returns the number of data bytes configured for a selected message.

ReadData

Read a specified data byte from the most recently received copy of a specified message.

PowerDown

Disable operation of the LIN interface device - Power saving.

WakeUp

Enable operation of the LIN interface device.

GetState

Read the current state of LIN operation. Includes fault conditions and operating states.

4.3. Demonstrations and Exercises

A number of exercises have been provided to introduce the students to some of the important features of LIN. The knowledge gained can be reinforced by undertaking the suggested programming tasks, or by downloading the supplied programs and observing the system in operation.

Each exercise introduces a feature of the LIN system and guides the student through a practical application.

Example solution programs for each exercise are provided on the accompanying CD-ROM. These can be analysed by students who are required to write their own programs, or can be used as demonstration programs for students wishing to study the bus signals and network operation

Examples/Demonstration programs:

Exercise 1 - Master node to single Slave node data transmission:

Master Node:	LIN_EX1_M.fcf	Send steering column switch simulator
Slave A node:	LIN_EX1_SA.fcf	No function
Slave B node:	LIN_EX1_SB.fcf	No function
Slave C node:	LIN_EX1_SC.fcf	Dashboard lamps

Exercise 2 - Master node to multiple Slave node data transmission:

Master Node:	LIN_EX2_M.fcf	LIN_EX1_M + Radio station selection
Slave A node:	LIN_EX2_SA.fcf	Radio station display
Slave B node:	LIN_EX2_SB.fcf	No function
Slave C node:	LIN_EX2_SC.fcf	Dashboard lamps

Exercise 3 - Slave node to Master node data transmission:

Master Node:	LIN_EX3_M.fcf	LIN_EX2_M + Remote radio selection
Slave A node:	LIN_EX3_SA.fcf	Radio station display
Slave B node:	LIN_EX3_SB.fcf	Radio station selector
Slave C node:	LIN_EX3_SC.fcf	Dashboard lamps

Exercise 4 - Slave node to Slave node data transmission:

Master Node:	LIN_EX4_M.fcf	LIN_EX3_M + Improved data flow
Slave A node:	LIN_EX4_SA.fcf	Radio station display
Slave B node:	LIN_EX4_SB.fcf	Radio station selector
Slave C node:	LIN_EX4_SC.fcf	Dashboard lamps

Exercise 5 - LIN-CAN interface demonstration (Requires CAN Solution)

CAN Solution node D:	LIN_EX5_CAN_D.fcf	LIN-CAN translator
Slave A node:	LIN_EX5_SA.fcf	Radio station display
Slave B node:	LIN_EX5_SB.fcf	Radio station selector
Slave C node:	LIN_EX5_SC.fcf	Dashboard lamps

CAN system demonstration software folder

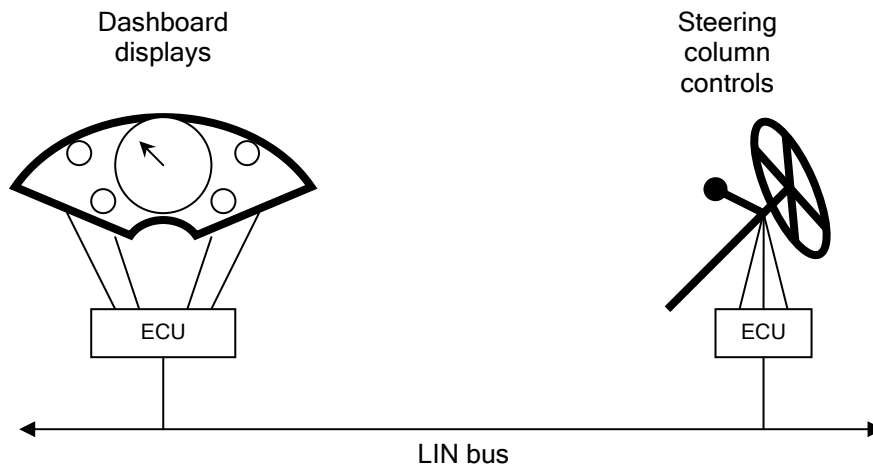
5. Exercise 1 - Master node data transmission

5.1 Introduction

In this exercise we will simulate some of the controls and indicators found in a modern car instrument panel and steering column.

A car dashboard typically contains a number of illuminated icons that display the state of some of the main driving controls: headlights, fog lights, indicator lights, etc. Many of the controls are mounted on the steering column and have traditionally required a significant amount of cabling to connect them to their controlled devices, and provide a dashboard display of their state. The use of LIN bus allows groups of functions to be connected to local ECUs (Electronic Control Units), and for information to be transmitted between ECUs on a single wire in the form of messages. Multiple ECUs can be attached to a single LIN bus (up to 16), and each ECU is referred to a BUS NODE.

In this example the headlight control switch on the car steering column must control the headlight icon on the dashboard. The steering column and dashboard will contain separate bus nodes and the headlight switch information must be transmitted from one to the other.



Some important features of LIN bus are:

- A node is any device (ECU) connected to the LIN bus that is capable of reading or writing data.
- Unlike many bus systems, not all nodes on a LIN bus nodes are of equal significance: LIN bus requires a Master node to be designated to control all the communications on the bus.
- Unlike many bus systems, LIN bus nodes do not have individual network addresses. Data is broadcast on the bus in the form of messages, with each message containing an identifier (ID) value that identifies its format and function.

In this exercise the Master node is the one in the steering column. It reads the control settings, simulated by the attached push-buttons, and transmits the status information as a message on the LIN bus.

The Slave C node will simulate the dashboard display by listening for the appropriate message identifier, reading the data contained in the message, and controlling the dashboard icons - simulated by the attached LEDs.

We will add the Flowcode LIN Master component to the program in the node representing the steering column ECU, and use it to configure and control message transmission. The LIN Slave component will be added to the program in the node representing the dashboard ECU, allowing messages to be configured and read.

Note:

It is important at this stage to recognize the master node because it must be configured with the Flowcode LIN_Master component. ALL other nodes on a LIN bus are Slave nodes. These must be configured with the Flowcode LIN_Slave component.

5.2 Theory

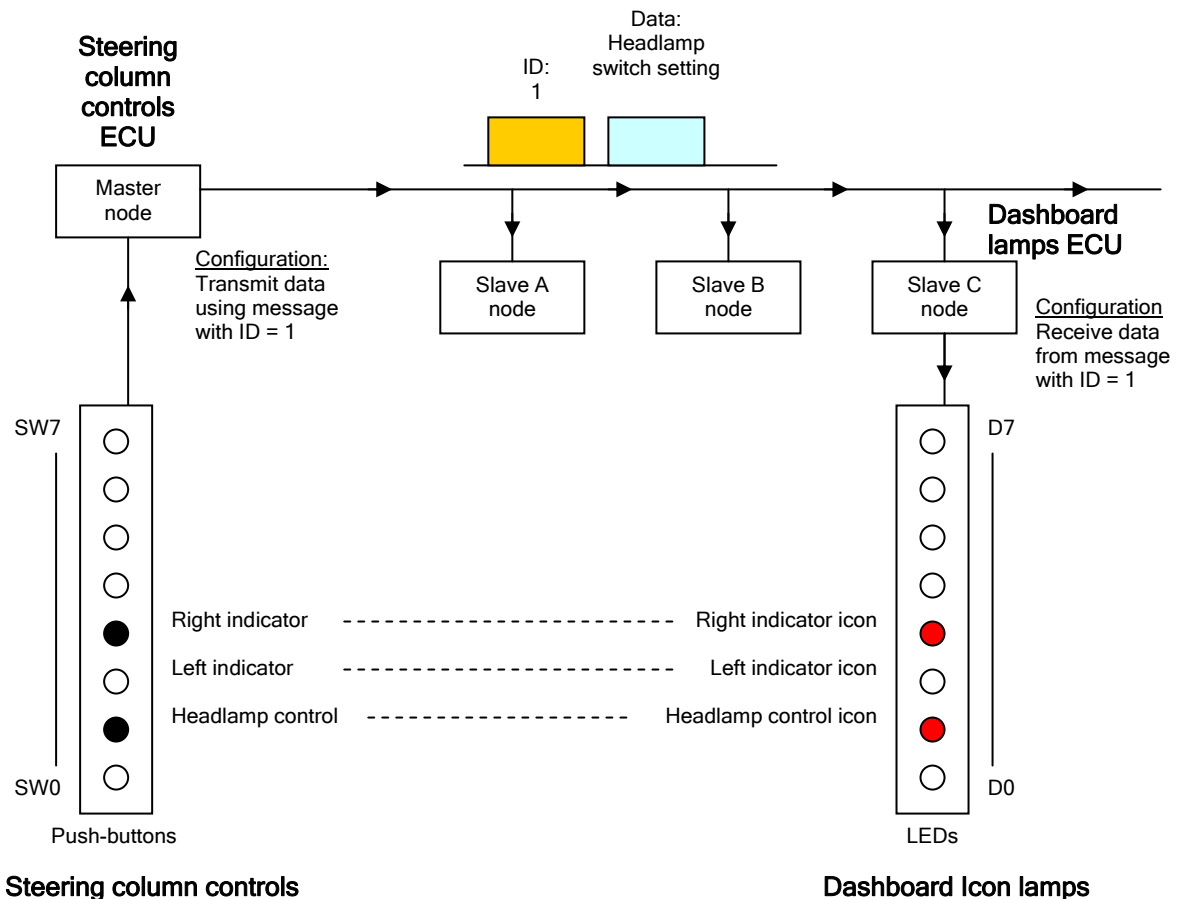
Data is transmitted on the LIN bus in the form of messages. At the programming level, a message consists of two components:

- A message identifier (ID)
- Up to eight bytes of data

We will start by using a message with the identifier 1 to transmit 1 byte of data from the Master node (Steering column controls) to one of the Slave nodes - Slave C node (Dashboard display icon lamps).

A program in the Master node will continuously read the status of the steering column controls attached to Port D of the Master controller, and use the functions of the LIN Master component to transmit the control settings on the LIN bus.

A program in the Slave C node will use functions from the LIN Slave component to monitor the LIN bus for reception of a message with identifier 1, retrieve the received data, and use it to control the dashboard lamps. - simulated by the LEDs attached to Port D of the Slave C controller.



Control switch data transmission

5.3 Objectives

- Demonstrate the significance of the bus master and slave nodes.
- Configure two of the network nodes.
- Achieve data transmission from the master node to one of the slave nodes.

5.4 Requirements

- Non programmers:
Copies of the supplied programs for Exercise1, to be downloaded to the LIN controller boards.

5.5 Implementation

5.5.1 Master node

Connect the programming lead to the Master node

Open a new Flowcode program and add the LIN Master component.

Use the Flowcode LIN Master component properties to set the baud rate to 9600

- Write a simple Flowcode program to read the status of the push-button representing the headlight control (SW1).
- Use the LIN Master: SetData and SendMessageWithData functions to transmit the state of the switch using message ID 1. Use the data value 0 to represent the switch open and 255 to represent the switch closed (pressed).
- Add a 50ms delay between each transmission.

5.5.2 Slave C node

Connect the programming lead to the Slave C node

Open a new Flowcode program and add the LIN Slave component

Use the Flowcode LIN Slave component properties to define messages with the following properties:

Msg	1
ID	1
No. of Bytes	1
TX/RX	Rx (Receive)

Msg	2
ID	2
No. of Bytes	1
TX/RX	Rx (Receive)

Msg	3
ID	3
No. of Bytes	1
TX/RX	Rx (Receive)

- Use the LIN_Slave: ListenforMessage and ReadData functions to write a simple Flowcode program that continually checks for the reception of a message with 1 as its Msg value, and retrieve the associated single data byte.

Note:

The LIN Slave: ListenforMessage function returns the Msg value of a recognized message, not the ID value. A value of 0 is returned if no message is received within the assigned waiting period.

- Use the received Msg value and associated data byte to control the state of the headlight icon LED (D1).

5.6 Learning outcome

This exercise has introduced the basic features of the LIN bus system and the use of the Flowcode LIN components to configure the nodes and define a message. The student should now have an understanding of:

- The interaction between the Master and Slave nodes.
- The configuration of the nodes to enable communication.
- The structure of the messages being transmitted on the bus.
- The processes required to transmit and receive a specific message.

5.7 Further work

The current network only handles a single control setting and represents an inefficient use of resources.

- Modify the Steering column node program to include controls for the left and right indicators (use SW2 and SW3 respectively).
- Use messages with IDs 2 and 3 to transmit the left and right indicator status values respectively, and add a 50ms delay between each transmission. The dashboard node has already been configured to recognize these messages.
- Modify the Dashboard node program to allow detection of the left and right indicator messages to control the appropriate icon lamps (use D2 and D3 respectively).

The Baud Rate property is only available for the master node. Change the value and see how the slave node reacts.

Note:

If you have access to an oscilloscope you will be able to confirm the effect the Baud Rate value changes have on the signal waveforms.

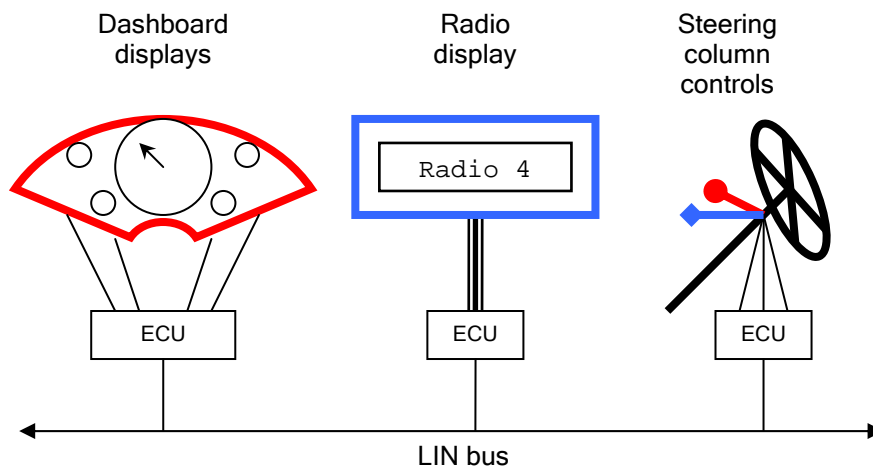
Change the message IDs transmitted by the master node, or expected by the slave node, and test the effect this has on the slave nodes.

Change the number of data bytes transmitted by the master node, or expected by the slave node, and test the effect this has on the slave nodes.

6. Exercise 2 - Multiple slave nodes

6.1 Introduction

Many dashboard instruments now display information using text instead of gauges and lamps. In this exercise we will add control of the car radio to the LIN system. Radio station selection will be available from the steering column, so the controls will be added to the steering column ECU. The radio station text display is usually mounted on the main console and will have a dedicated ECU. We will modify the Steering Column program from exercise 1 so that some of the unused push-buttons can be used as radio station selectors, and the radio station information will be transmitted in a new LIN message. The radio display ECU (Slave A node) will be configured to recognize the station select message, and programmed to produce a display of the required information on the attached LCD.



6.2 Theory

6.2.1 Network operation

The nodes on a LIN bus are not allocated individual addresses, so data is not targeted at specific recipients. Instead, each message is broadcast on the network and contains an identification field (ID) that indicates the message contents.

Messages are differentiated by:

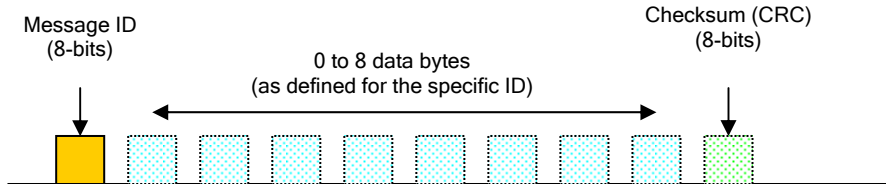
- The data source node. (LIN requirement)
Only one node must supply data for any message ID.
- The number of data bytes. (LIN requirement)
Every message ID used must have a defined number of data bytes.
- The data. (Programming option)
Messages with the same data source node and number of data bytes, but with different data content, can (and probably should) be allocated different IDs by the network designers.

Individual nodes can be configured to respond to, or to ignore, messages with any of the IDs used on the network.

If a node is required to respond to an ID, either by reading or writing data, the number of data bytes transferred must match the configuration of all the other nodes using that message.

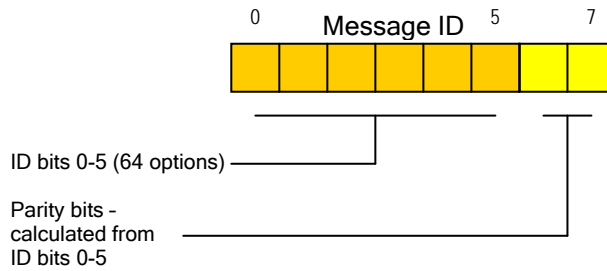
6.2.2 Message format

For every message ID allocated for use on an individual LIN network, the associated message format must be defined. Any node required to operate with a particular message ID must have the message configuration set to match the network definition.



6.2.3 Message ID

The message identifier consists of 8 bits: 6 ID bits and 2 parity bits. The parity bits are calculated and added automatically to allow detection of transmission errors.



The six available ID bits allow 64 unique message IDs to be defined. With the exception of 4 message IDs reserved by the LIN consortium (0x3C, 0x3D, 0x3E, and 0x3F), all available IDs can be freely allocated and configured by the developers of individual networks - within the constraints of the bus specification.

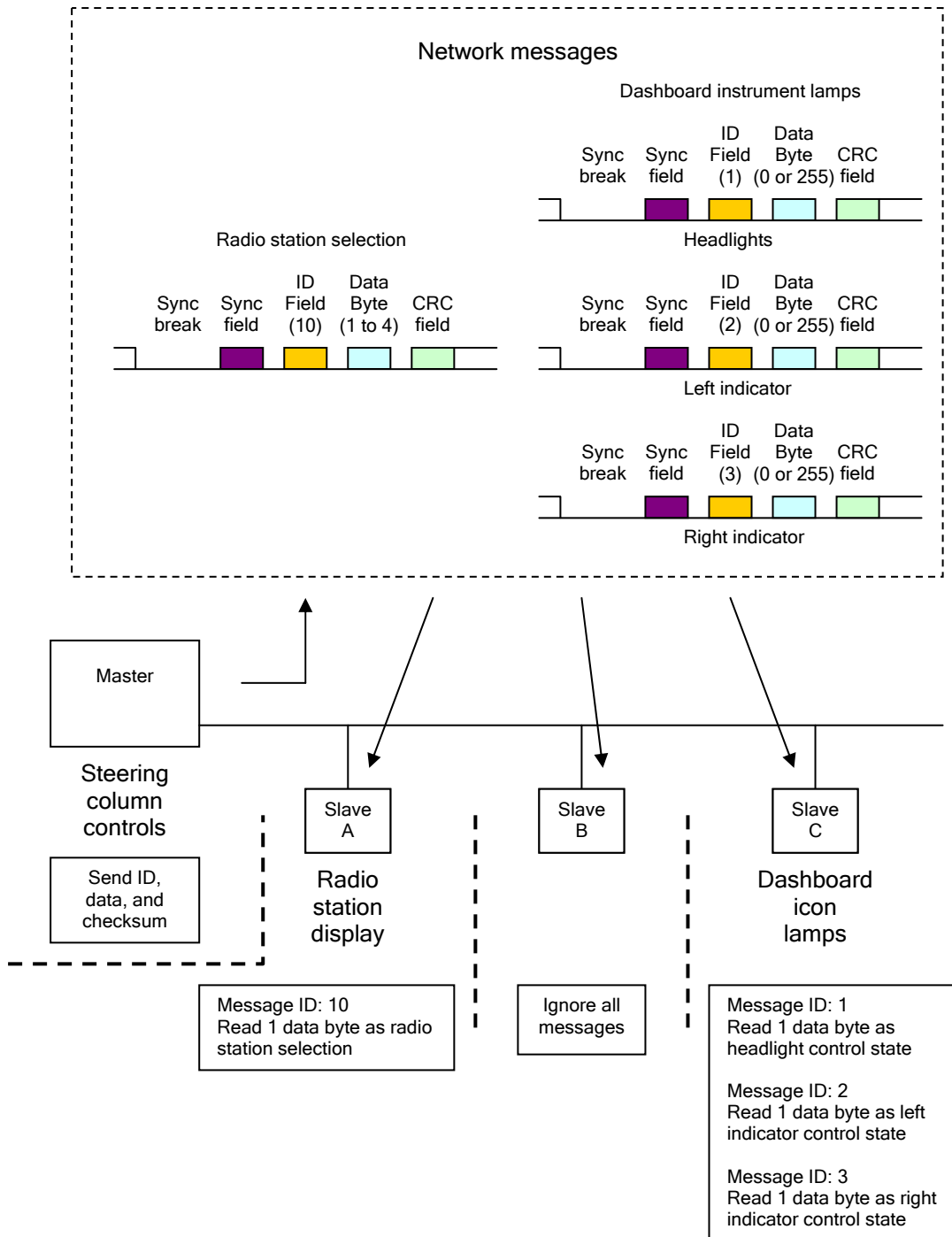
An individual message can contain a maximum of 8 data bytes.

6.2.4 Bus master

A LIN bus network can only have one master node that controls all communications on the bus.

6.2.5 Bus slaves

Slave nodes must be configured to detect the IDs of any messages they are required to operate with, and to read or write the accompanying data in the format defined by the overall network specification.



Network messages and node configurations

6.3 Objectives

- Introduce a new message onto the network with different data content to the original messages.
- Configure a new node to respond to the new message.

6.4 Requirements

- Programmers:
A solution to Exercise 1, or copies of the supplied programs for Exercise 1, to be downloaded to the LIN controller boards and modified in this exercise.
- Non-programmers:
Copy of the supplied programs for Exercise 2, to be downloaded to the LIN controller boards

6.5 Implementation

6.5.1 Master Node

Connect the programming lead to the Master node

Make a copy of the Flowcode program from the previous exercise

- Modify the program to read the states of switches SW4, SW5, SW6, and SW7, as selectors for Radio1, Radio2, Radio3, and Radio4, respectively.
- Use the LIN Master: SetData and SendMessageWithData functions to transmit the value of the selected radio station as the single data byte of a LIN message, with ID set to 10, in addition to the original message transmissions (include a 50ms delay between each transmission).

As none of the Slave nodes have been configured to recognize a message with an ID of 10 the full results of the modification will not be obvious at this stage.

6.5.2 Slave A node

Connect the programming lead to the Slave A node

Write and download a Flowcode program that includes the LIN Slave component, and is correctly configured to recognize and read messages being sent by the Master node with an ID value of 10.

Msg	1
ID	10
No. of Bytes	1
TX/RX	Rx (Receive)

- Use the LIN Slave: ListenforMessages function to detect transmission of the message
- Use the LIN Slave: ReadData function to read the transmitted data and generate a display of the selected radio station on the attached LCD.

Remember:

The Flowcode LinSlave: ListenForMessages function returns the associated value of Msg from the configuration panel when a message is recognized, not the ID value. With the suggested configuration values the value 1 will be returned when a message with an ID of 10 is received. This value must be used with the LIN Slave: ReadData function as it represents an internal reference (index) to the message. The LIN ID of a message can be read using the LIN Slave: ReadAddress function with the associated Msg value

6.6 Learning outcome

This exercise has demonstrated the significance of the message identification field to LIN bus operation, and its use in defining the purpose and format of the accompanying data.

In addition to the creation of a new network message, this exercise has also added a new slave node that has been configured to read and react to the new message, adding new functionality to the network without affecting the existing slave node.

6.7 Further work

Experiment with different message IDs and data formats.

(Advanced)

The ListenForMessages function in the LIN Slave component returns a value to the main program whenever it is executed. Modify the Slave A node program to display the function values returned when an unrecognized or incorrectly formatted message is detected.

Hint:

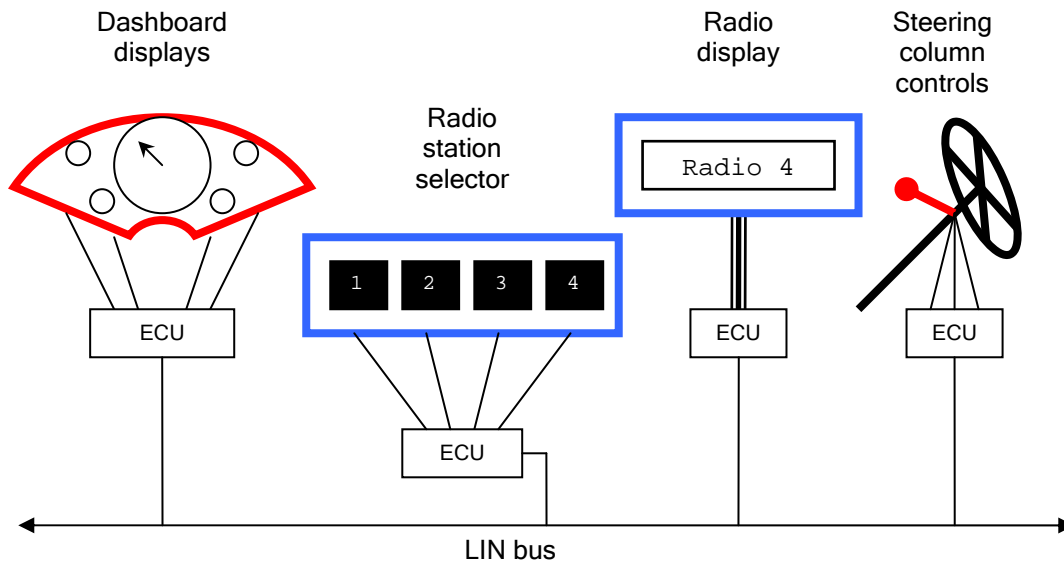
Change the message configuration properties in the LIN Slave component of the Slave A program to cause the errors to be generated, and observe the resulting function values. Develop a strategy to deal with problems caused by network faults or incorrectly configured nodes.

7. Exercise 3 - Slave data transmission

7.1 Introduction

In the previous examples only one node (ECU) has been supplying data on the network. Most networks will require a number of nodes to supply data, due to the I/O limitations of individual ECUs and the remote locations of some of the data sources. The additional data sources will require connection to local ECUs that are connected as slave nodes onto the LIN bus.

In this exercise the radio station selector buttons will be moved from the main steering column control node to a dedicated In-Car-Entertainment node. This node is required to supply data onto the network but, as a slave node, has no control over the network and can not initiate a data message transmission. The message must be transmitted onto the LIN bus under the control of the bus master node (Steering column control ECU).



This exercise will demonstrate the procedure required to retrieve data from a slave node by configuring it write to the bus in response to detection of a specific message ID.

7.2 Theory

The Master node must initiate all data transfers on the LIN bus. If there is a network requirement for data from one of the Slave nodes, the master node must initiate the data transfer sequence by transmitting the first part of a recognized message, then release the bus so that the appropriate slave node can complete the message by supplying the data and checksum.

7.2.1 Master node

When data is required from a Slave node, the Master node must transmit the first part of the network message up to and including the message ID. When the message ID has been transmitted, the Master node must release control of the LIN bus to allow a Slave node to respond by completing the message.

The Flowcode LIN Master: SendMessage function allows a message ID to be transmitted without any accompanying data (as an alternative to the SendMessageWithData function). The master node then listens for the correct number of data bytes and the checksum byte being transmitted by one of the Slave nodes.

7.2.2 Slave node

One (and only one) of the network Slave nodes must be configured to respond to each of the messages requiring slave node data. The appropriate slave node must be configured to recognize the message ID and respond by transmitting the correct amount of data and a checksum byte.

The Flowcode LIN Slave component allows the properties of each recognized message to be configured. One of the options allows a message to be configured for Transmission (TX) or Reception (Rx). This only applies to the data and checksum fields of the message; the ID is always transmitted by the Master node and received by all the Slave nodes.

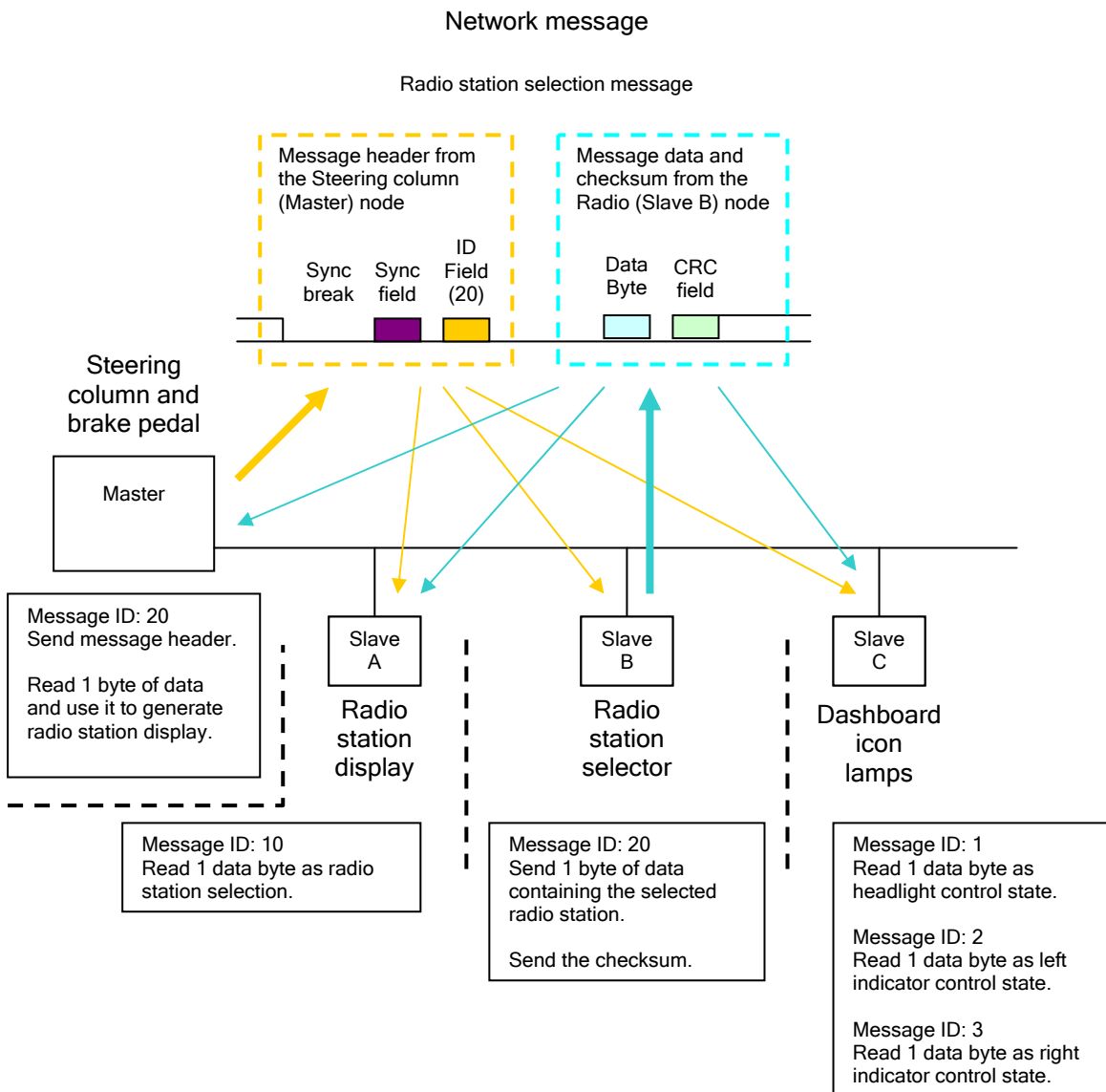
7.2.3 Network theory

In this exercise, a new message with ID set to 20 will be used to write the value of the selected radio station from the Radio station selector node.

The LIN Bus master node (Steering column control ECU) initiates the data transfer by transmitting the first part of a message with an ID of 20. All the slave nodes receive this part of the message and determine their responses.

- The nodes controlling the dashboard lamps and the radio station display are both configured to ignore the message.
- The node attached to the radio selector buttons is configured to respond to the message ID by writing the selected radio value and a checksum onto the bus.

When the radio station data has been received by the Master node, it is transmitted to the Radio station display node using the ID: 10 message from the previous exercise.



7.3 Objectives

- Describe the process required to retrieve data from slave nodes.
- Add another message type to the network, configured to allow a slave node (Radio buttons) to respond to a prompt from the master node (Steering column) with data.
- Configure the participating nodes to respond correctly to the new message type.

7.4 Requirements

- Programmers:
A solution to Exercise 2, or copies of the supplied programs for Exercise 2, to be downloaded to the LIN controller boards and modified in this exercise.

Non-programmers:

Copies of the supplied programs for Exercise 3, to be downloaded to the LIN controller boards

7.5 Implementation

7.5.1 Slave B node

Connect the programming lead to the Radio station selector node (Slave B).

Open a new Flowcode program and add the LIN_Slave component

Use the Flowcode LIN Slave component properties to define a message with the following properties:

Message

Msg	1
ID	20
Data Bytes	1
Direction	Tx

- Write a simple Flowcode program to read the status of the push-buttons, SW4 to SW7, attached to Port D of the controller, on a 250-millisecond cycle.
- Produce a single data value representing the most recent station selected (1 to 4).
- Use the LIN_Slave: SetData and ListenForMessages functions to prepare the message data and listen for reception of the ID: 20 message header.

7.5.2 Master node

Connect the programming lead to the Master node

- Modify the Steering column (Master node) program from Exercise 2 to regularly request data from the Radio station selector node. Use the LIN Master: SendData function to initiate a message with ID: 20 - expect one data byte to be returned. This will generate the required message header and initiate a response from the Radio station selector node.
- Send the received radio station data, using message ID: 10, to update the radio station display.

7.6 Learning outcome

This exercise has demonstrated the configuration options and the communication sequence required to retrieve information from a Slave node under the control of the Master node. Many alternative bus systems use complex protocols and hardware to maximize efficiency by allowing all nodes to compete for control of the bus, detecting and reacting to collisions when they occur. The involvement of the master node in all communications reduces the efficiency of the LIN bus (not a serious problem) but allows it to remain simple (a big advantage).

7.7 Further work

As the amount of data being transmitted on a network increases it becomes possible to overload the system, or increase response times, especially when one network node is responsible for the control of all network operations. Modify the Master node program to transmit data only when conditions change, and request data at an appropriate rate.

8. Exercise 4 - Slave to Slave transmission

8.1 Introduction

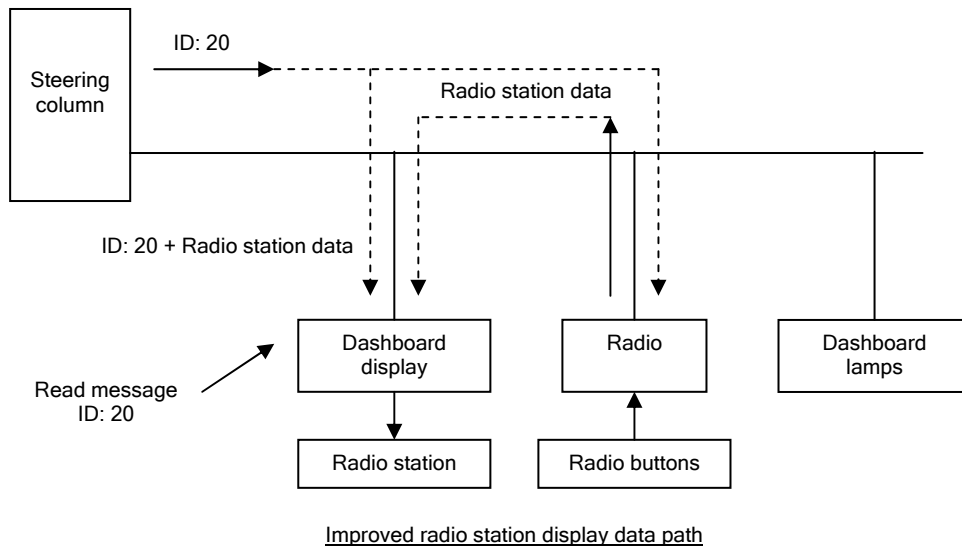
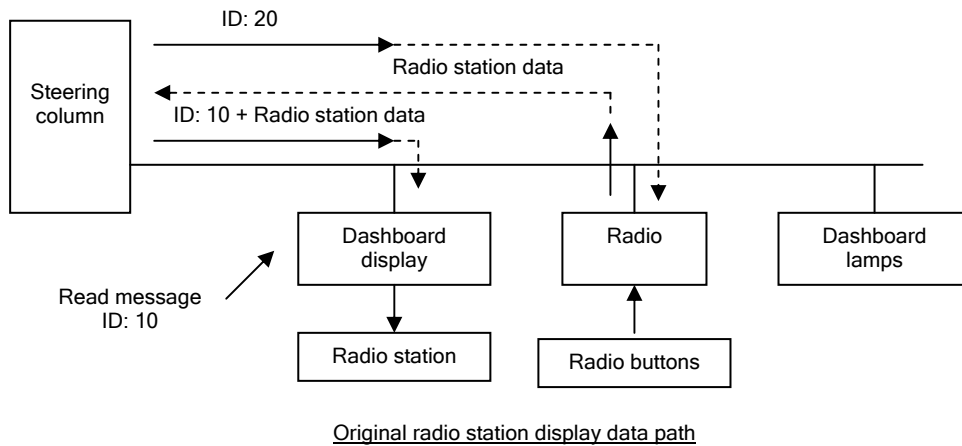
In the previous exercise the process to display the radio station information is slightly inefficient. The master node (Steering column) requests the information from the Radio selector node by initiating a message with ID: 20 and reading the data returned by the radio node. It then transmits the radio station information to the dashboard display using a message with ID: 10.

An important feature of LIN is that all information is broadcast and can be received by any node. Individual nodes can be configured to respond to any properly detected message on the bus, even if it is initiated by the Master node and completed by one of the other Slave nodes.

This exercise involves simple modifications to some of the programs from the previous exercise to allow direct communication between Slave nodes (under the control of the Master node), and the removal of a redundant message definition.

8.2 Theory

In the previous exercise the radio station data retrieved from the Radio Selector node was not required by the Master node and was simply re-transmitted to the Radio Display node using a different message ID. This process can be simplified if the Radio Display node is re-configured to detect the data being supplied by the Radio Selector node directly.



8.3 Objectives

- Demonstrate the process that allows direct Slave node to Slave node data transfer, under the control of the Master node.

8.4 Requirements

- Programmers:
A solution to Exercise 3, or copies of the supplied programs for Exercise 3, to be downloaded to the LIN controller boards and modified in this exercise.
- Non-programmers:
Copies of the supplied programs for Exercise 4, to be downloaded to the LIN controller boards

8.5 Implementation

8.5.1 Radio Station Display Node

Connect the programming lead to the Radio station display node (Slave A Node), Start Flowcode and open the LIN_EX3_SA program.

Select the LIN Slave component properties and change the configured message ID from 10 to 20 (no other changes are necessary).

Download the modified program to the Radio display controller.

When the download is complete, the network should function as it did before the modification. What is not apparent is the fact that the Radio display node is now reading its data from the message with ID: 20, directly from the Radio selector node, and is ignoring the message with ID: 10 being generated by the Master node.

8.5.2 Master Node

Connect the programming lead to the Steering column node (Master node), Start Flowcode and open the LIN_EX3_M program.

To confirm the redundancy of the ID: 10 message, remove the program blocks that read the data from the ID: 20 message and send the ID: 10 message. (Leave the block that sends the ID: 20 message in place).

Download the program to the Master controller.

When the download is complete the network should function as it did before the modification. The Master node is now only initiating the data transfer between the Radio station selector and Radio display nodes, and is no longer reading or re-transmitting the data.

8.6 Learning outcome

This has been a relatively simple exercise, but has demonstrated an important feature of LIN bus operation. The ability to transfer data directly between slave nodes significantly reduces the workload of the master node and the amount of network traffic.

8.7 Further work

(Advanced)

- Modify the ID: 20 message configuration in the radio station selector and display nodes so that it includes 8 data bytes.
- Modify the master node program to expect 8 data bytes when it initiates the ID: 20 message.
- Modify the radio selector node program to transmit the full name of the selected radio station as individual characters.
- Modify the radio station display node program to display the received radio station text.

The resulting system should be more flexible because the radio station selector can generate any station name (up to 8 characters) and the radio station display can become a general purpose text display.

9. Exercise 5 - LIN-CAN interface (Demonstration)

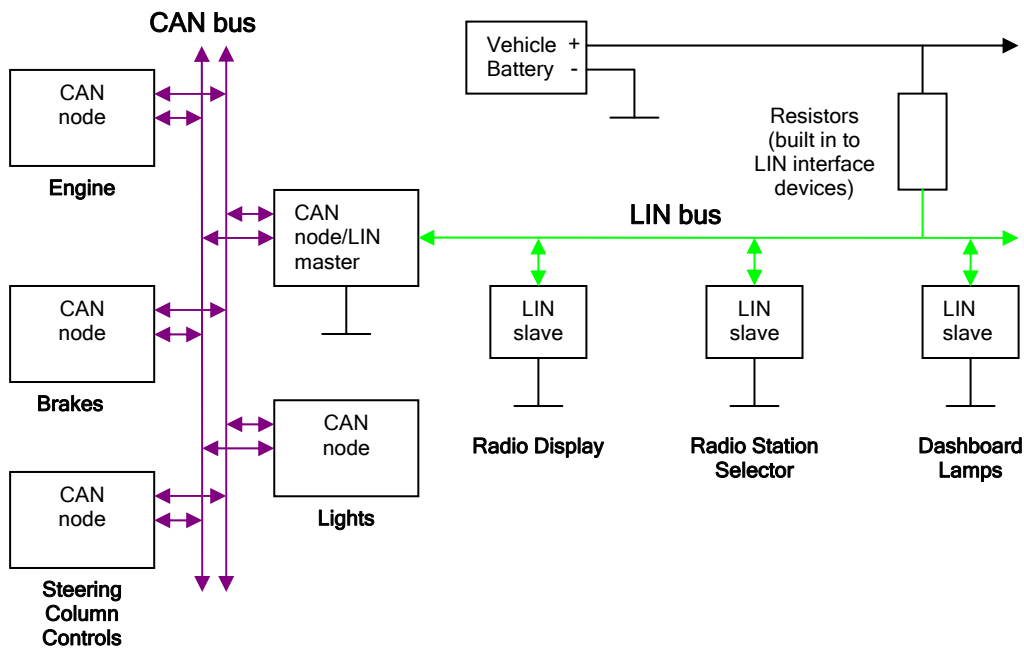
9.1 Introduction

This demonstration requires the availability of the full Matrix Multimedia CAN solution, running the DM02 demonstration - also supplied in the CAN sub-folder of the LIN Exercise 5 software.

LIN bus has been designed to allow the creation of low cost automotive networks. It is not suitable for performing many of the tasks that the other automotive bus system - CAN bus - can perform, so the two networks must be able to communicate with each other if they are to be used together in an integrated vehicle control system. Although LIN is expected to work in conjunction with CAN, the two are not directly compatible and will require a CAN-LIN translator node to allow the transfer of data between the two networks.

LIN bus is slower and less reliable than CAN bus, so it intended to be used for localised control of functions that are neither time critical nor safety critical.

The main advantage of LIN is that it has been developed to be implemented using low cost hardware. Data is transmitted on a single wire, relying on the common 0v of the system to provide the signal return path and thus saving money on wiring and connectors.



CAN-LIN Interface

In this demonstration the Steering column node (Master node) has been completely removed from the LIN bus due to vehicle modifications. The control data required for the dashboard icon lamps is still available in the system, but must now be accessed via the vehicle's main CAN bus. The original LIN master Node must be replaced by a LIN-CAN translator node that acts as the LIN master (sending data, reading data, and initiating direct data transfers), and also communicates with the CAN bus, listening for messages containing data relating to the dashboard icon lamps. In addition, the radio station selector ECU contains an input that can be used to activate the vehicle horn, possibly for alarm purposes.

9.2 Theory

A single device (ECU or dedicated translator) can be used as an interface between a CAN bus and a LIN bus. If the device contains sufficient processing power it can also perform other tasks.

In this demonstration the translator acts as the LIN bus master node, but it would also be possible for it to be one of the slave nodes (depending on the network function).

The CAN bus does not support Master and Slave nodes, so the LIN-CAN translator is a general purpose node on the CAN bus.

The LIN-CAN translator must listen for relevant messages on the CAN bus and send the appropriate LIN messages in response. In this case both busses use the data value 255 to represent ON, and 0 to represent OFF. The message identifiers on the two busses are different.

CAN ID	Direction	LIN ID	Function
101	->	1	Lights
102	->	2	Left indicator
103	->	3	Right indicator
104	->	4	Fog lights
106	->	6	Passing lights
-	-	20	Radio station
107	<-	21	Horn

The CAN-LIN Translator Node must:

- Convert any received CAN message IDs to the corresponding LIN IDs, and send them on the LIN bus with the accompanying CAN data value unchanged.
- Repeatedly initiate the LIN ID = 20 message, to transfer radio station data from the Radio station selector node to the Radio display node.
- Repeatedly initiate the LIN ID = 21 message to test the status of the horn input on the Radio station selector node. When a change in state of the horn control is detected, a message with ID = 107 must be sent on the CAN bus, accompanied by the data representing the new state of the control (0 = OFF, 255 = ON).

Note:

This is the same message ID used by one of the original CAN nodes to control the horn. The added complexity of CAN allows nodes to use the same message IDs if they are supplying the same network data (where this does not cause confusion).

9.3 Requirements

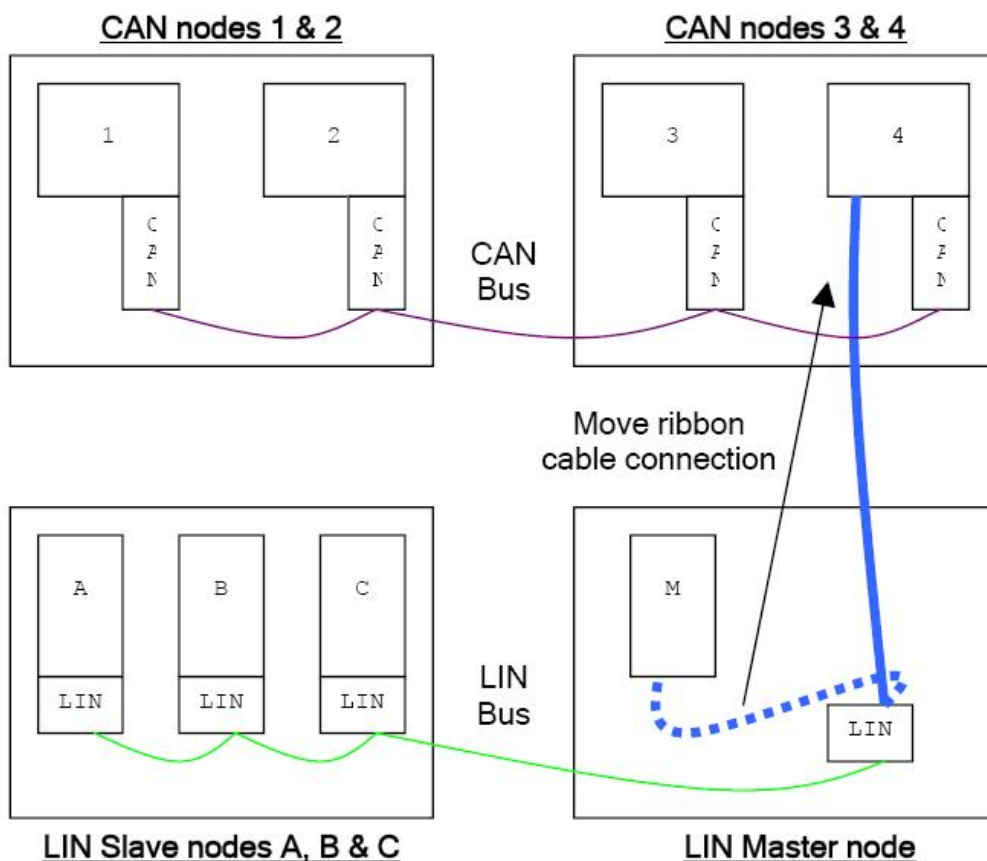
- A full Matrix Multimedia CAN Solution
- A full Matrix Multimedia LIN Solution
- The Exercise 5 programs from the LIN Solution CD
- The CAN DM02 programs from the Exercise 5 sub-folder, or the CAN Solution CD
- Ribbon cable extension

9.4 Preparation

The following setup procedure is required to enable the CAN and LIN systems to communicate with each other.

- Download the CAN DM02 programs to the CAN nodes 1-3
- Download the LIN_EX5_CAN_4 program to the CAN node 4.
- Download the remaining LIN_EX5 programs to the appropriate LIN nodes.

Disconnect the LIN e-BLOCK ribbon cable from the LIN bus master node and connect it to PORT D of the CAN node 4 controller. (The LIN e-Block can be physically moved onto the CAN system base, or can be connected to the CAN node using the supplied ribbon cable extension).



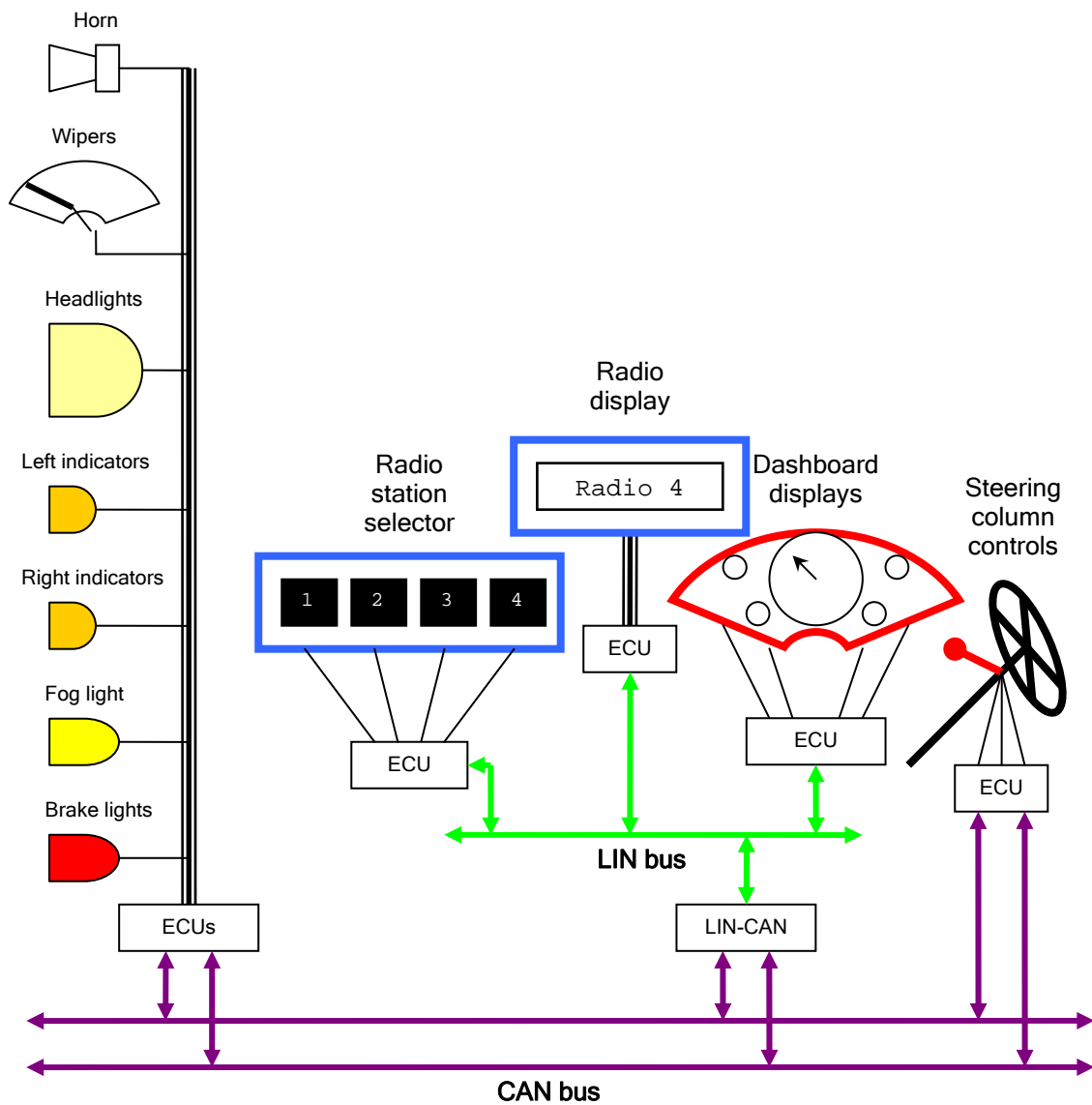
9.5 Operation

The completed system should now function as a fully integrated control system.

The steering column controls, with some additions, are now as indicated on CAN node 2.

The main light clusters, horn, and wipers are simulated on CAN node 3, with the LIN C node simulating the dashboard display for most of these functions.

The radio station display and selection functions are on the LIN A and B nodes respectively, but the LIN B node can also control the horn, on CAN node 3, using SW 0.



Integrated LIN-CAN control system