Local Interconnect Network (LIN)

Syed Rafsan Ishtiaque
Muhammad Rohail Usman
Yigitcan Aydin

Background

LIN, Local Interconnect Network, that tries to provide

everything that we need to build a simple, low-cost bus

tailored to the needs of the automotive and other industry/ applications.

LIN is widely popular and present in many vehicle platforms, with

the usage of 20+ LIN nodes per vehicle is not an exception.

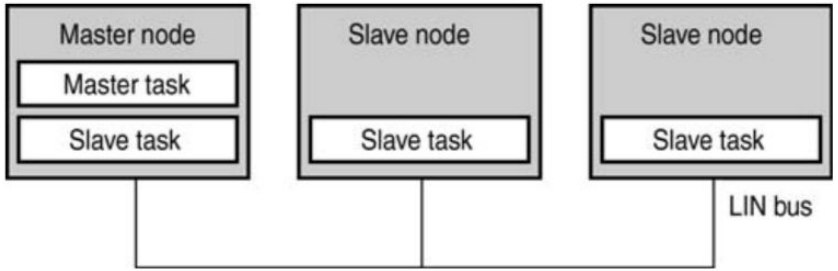
Reason for selecting LIN

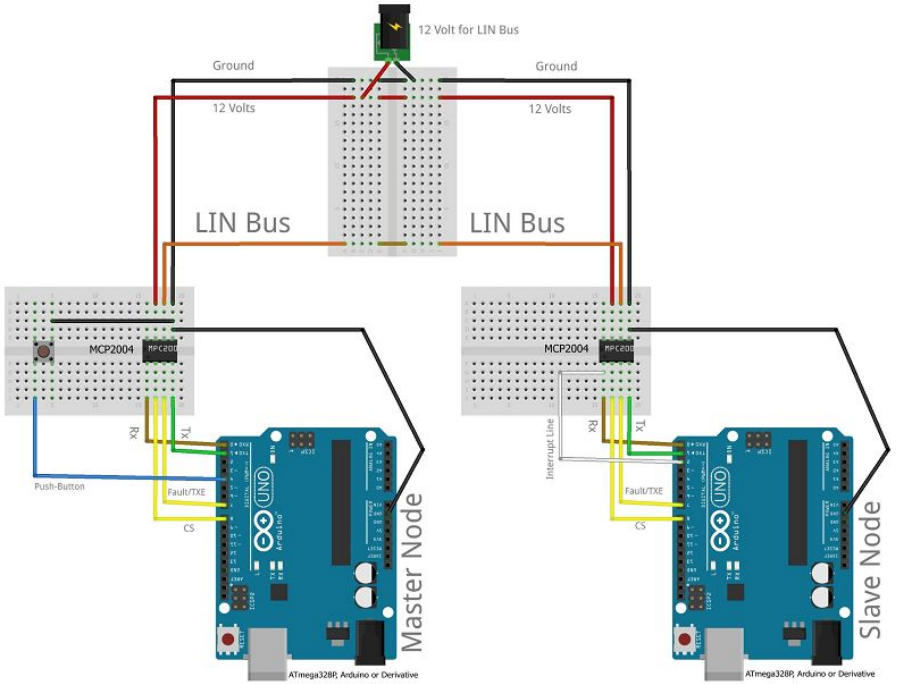
Aim: cost efficiency, small sensor, actuators, and control networks. It is widely used where bandwidth is not the main criteria rather low-cost and accessibility

We can run LIN on very low-cost ECUs. Theoretically, even without a quartz clock but only with an RC element

The architecture is a master-slave setup with up to 16 nodes, one master and up to 15 slave nodes.

Comparison with our project





Where to implement our project in real life scenario

Car radio, air conditioning, window functionalities, mirrors, doors, lights, interior, remote controlling of the car’s radio and so on

LIN is targeted at vehicle body controller

Even we can use it in any project where communication is necessary, such as our advanced embedded lab project

LIN vs UART (Universal Asynchronous Receiver-Transmitter)

Connecting 16 participants requires one wire plus power related cabling for LIN because it is a shared medium.

While in UART, connecting point-to-point 16 devices over a UART could easily become its own management challenge and certainly will need 240 connections without counting for the power related cabling, 120 connections for transmission and same for reception.

Resolving communication collision

If there are multiple people in group call and everyone speaks simultaneously then no one will clearly be able to listen each other.

In LIN nomination of one master like managing the group call. The master acts as a conductor. Based on a defined arrangement our schedule table, the master requests each individual communication participant to speak up.

Now everyone can listen to the information and speak up if there is a turn agreed in advance, this is how LIN resolves any communication collisions.

Master and Slave node

The master task is a routine that is only contained in the LIN master node. Its job is to read the schedule table entries and call out all the slave nodes in defined order.

The slave task despite its name, resides in all nodes also the master node. Its job is to listen to the bus for a header message and react accordingly depending on the identifier

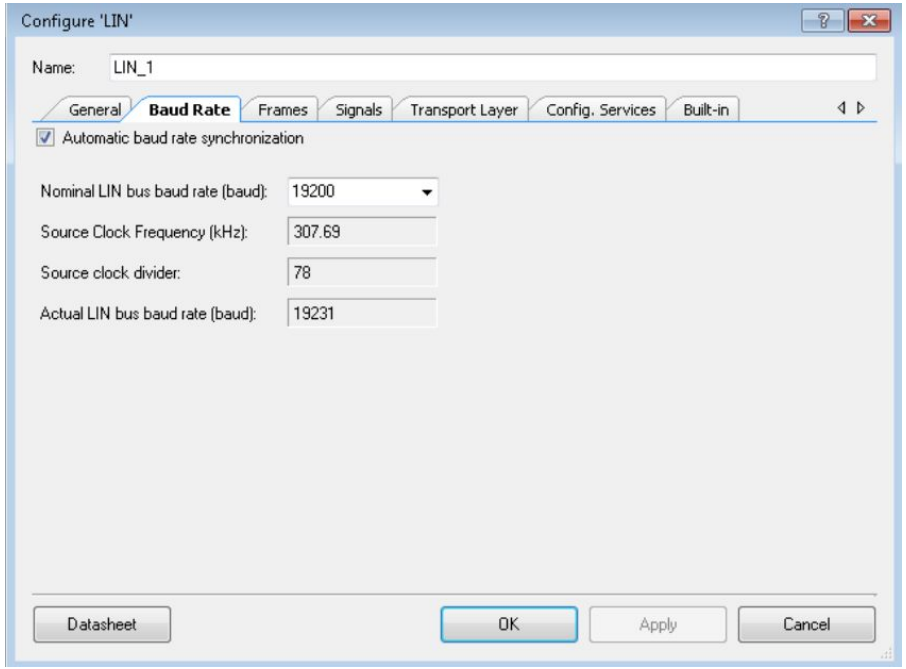
Baud Rate

The baud rate is the rate at which information is transferred in a communication channel.

Baud rate is commonly used when discussing electronics that use serial communication.

In the serial port context, "9600 baud" means that the serial port is capable of transferring a maximum of 9600 bits per second

Automatic Baud Rate Synchronization



This option enables the component measures the exact baud rate of the bus from the sync byte field of each LIN frame header.

Example: Max 20k, Min 1k, when ABRS on, the actual BR can be between 20k and 1k

LIN Frame

A LIN frame consists of header and response parts

Master node sends messages to the slaves using header part of the frames

Slave nodes responds the requests from master node using response part of the frame. This response message contains up to 8 bytes of data

LIN Frame: Header

Break (or SBF, Sync Break Field): min 14 bits long but usually it is 20 bits long in use. This field is considered as the start of the frame

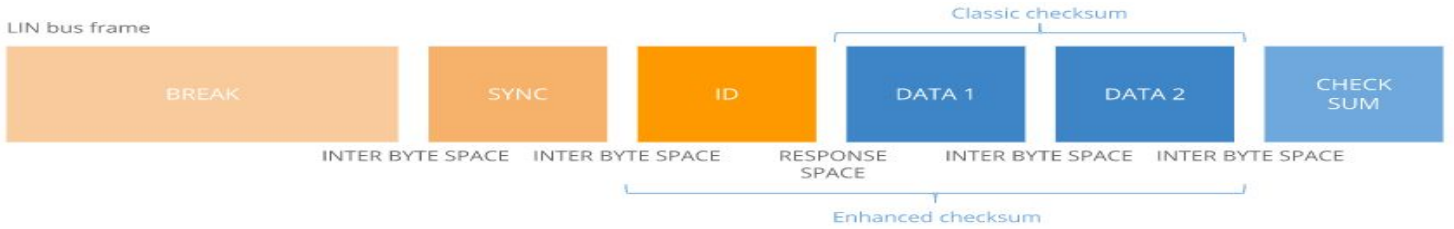
Sync field: Contains 8 bits to determine the baud rate of the master node. This field helps to synchronize all the slave nodes

Identifier (or ID field): Contains 6 bits (64 IDs) which determines which slave nodes must respond to the header. As a plus it contains 2 parity bits that helps slaves to validate the messages

LIN Frame: Response

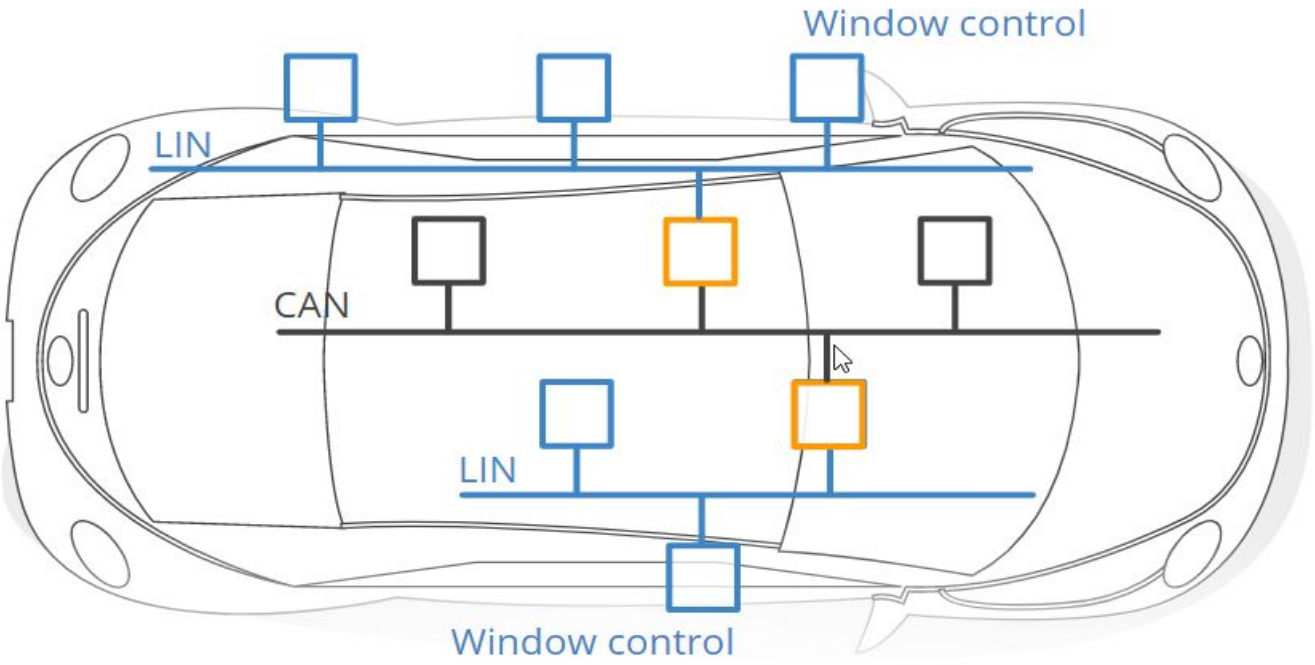
Data field: Data field contains the information to be responded to the master. May transmit 2, 4 or 8 bytes of data

Checksum field: Contains 8 bits to validate LIN frame. Validation is done by summing up the data bytes. In enhanced checksum, bits in the data field and ID field are summed up to get the validation



[1]

LIN and CAN



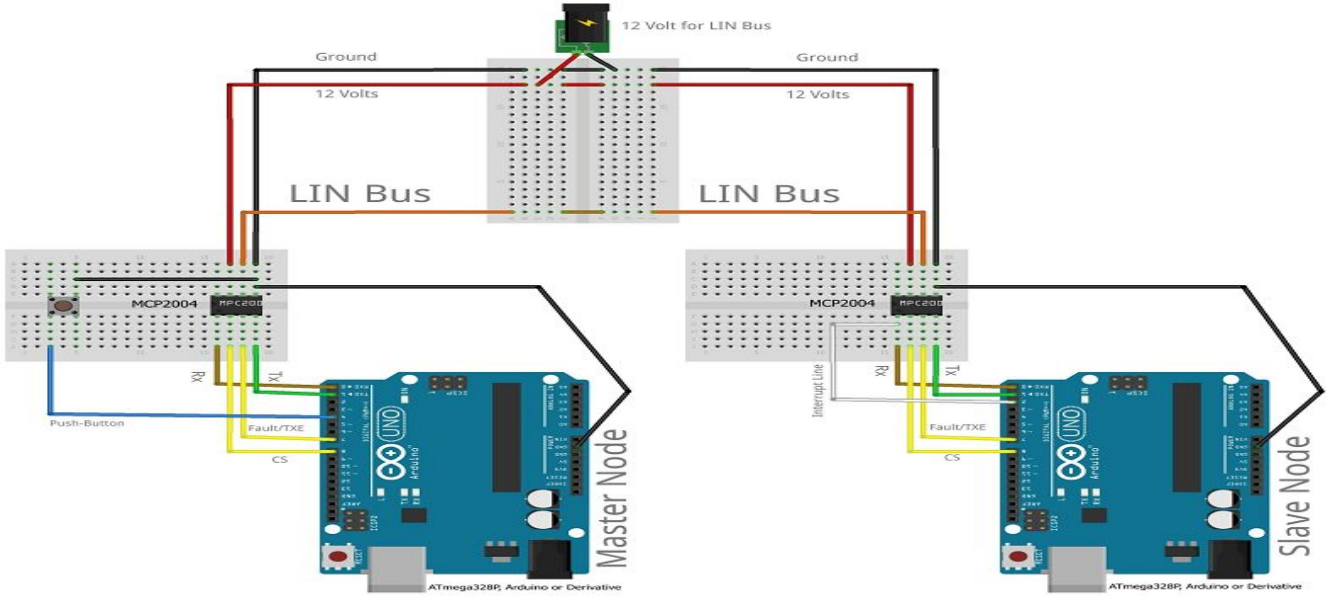
[1]

LIN Usage

In automotive industry:

- Window
- Engine
- Door
- Seats
- Wipers

Hands on approach



Implementation of a Simple Hello World in LIN Network.

With the Master Node in charge, LIN frames are exchange within scheduled slots and Frame Identifer 0x30

Software Workflow - Master Node:

Master Node Scheduler

```
#include "JRLinMaster.h"
#include "JRLinFrameBuffer.h"
#include "JRLinHelper.h"
#include <AltSoftSerial.h>

#define DEBUG_MODE_MAIN (0)

/**PINS */
#define FAULT_PIN (8)
#define CS_PIN (7) // not recommended to be hardwired in schematic; operation
#define ONBOARD_LED (13)
#define BUTTON_PIN (4)

/* Schedule size euqals 500ms; 5ms per slot; 100 slots*/
#define SCHEDULE_MAX_SLOTS (100)
#define SCHEDULE_TBASE_MILLIS (5)  //milliseconds, x slots = cycle time

#define SCHEDULE_ENTRIES (1)
#define SLOT_TIMEOUT_MICROS (25000) //25000 microsec = 25 ms

AltSoftSerial altSerial;

JRLINMaster lin;
JRLINFrameBuffer framelist[SCHEDULE_ENTRIES];
static JRLINFrameBuffer *ptrCurrentFrame = NULL;

// timer to aboard waiting for a frame
volatile unsigned long frame_start = 0;

// used for creating a schedule grid
unsigned long delayStart = 0; // delay start time
bool timerRunning = false; // true if still waiting for delay to finish
uint8_t slotCount = -1;

enum MasterStates {
  sendId      //ID request
  , receiveId // reply from nodes
} masterState;
```

Master Node I/O PIN and Memory Allocation

```
void setup() {
  pinMode(CS_PIN, OUTPUT);
  digitalWrite(CS_PIN, HIGH);

  pinMode(FAULT_PIN, OUTPUT);
  digitalWrite(FAULT_PIN, HIGH);

  pinMode(BUTTON_PIN, INPUT_PULLUP);

  pinMode(ONBOARD_LED, OUTPUT);
  digitalWrite(ONBOARD_LED, LOW); //dark

  /**
```

Master Node LIN Frames

```
#if DEBUG_MODE_MAIN >= 1
  altSerial.println("HW UART LIN Master Node");
#endif

  /*
     Configure frame list schedule
  */

  framelist[0].init(0x30, fb_id_send, 4, 10);
  framelist[0].data[0] = 0xFF;
  framelist[0].data[1] = 0x00;
  framelist[0].data[2] = 0xFF;
  framelist[0].data[3] = 0x00;
  framelist[0].len = 4;

  lin.setFrameList(framelist, SCHEDULE_ENTRIES);
  ptrCurrentFrame = NULL;

  /**
```

Software Workflow - Slave Node:

Slave Node Scheduler

```
void loop()
{
  // one break has to have occured, compute() time vs frame parse/protocol time
  if (lin.breakSymbol != valid) {
    compute();
    return;
  } else {

    static SlaveStates ret;
    ret = lin.slave_receive(frame_start, SLOT_TIMEOUT );

    if ( ret == frameProcessing) {
      sRetStatus = 1000;

    } else if ( ret == frameValid) {
      sRetStatus = 2000;
#if DEBUG_MODE_MAIN >= 1
      digitalWrite(DEBUG_PIN, LOW); // frame finished
#endif
```

Slave Node I/O PIN and Memory Allocation

```
void setup() {
  pinMode(ONBOARD_LED, OUTPUT);
  digitalWrite(ONBOARD_LED, LOW); //dark

  // Initialise here the PINS
  pinMode(CS_PIN, OUTPUT);
  digitalWrite(CS_PIN, HIGH);

  pinMode(FAULT_PIN, OUTPUT);
  digitalWrite(FAULT_PIN, HIGH);

#if DEBUG_MODE_MAIN >= 1
  pinMode(DEBUG_PIN, OUTPUT);
  digitalWrite(DEBUG_PIN, LOW);
#endif
```

Slave Node Break Symbol Identifier

```
// Interrupt routine
void pinchange() {
  if (digitalRead(INT_PIN) == LOW) { //pin logic moved to LOW
    _pin_down_time = micros();
    if (lin.breakSymbol != valid) lin.breakSymbol = pindown_indicated;
  } else { // pin logic moved to HIGH, check if valid break
    if ((micros() - _pin_down_time) >= PIN_BREAK_TIME) {
      lin.breakSymbol = valid;
      frame_start = micros(); // frame has started
#if DEBUG_MODE_MAIN >= 1
      digitalWrite(DEBUG_PIN, HIGH); // frame started
#endif
    }
  }
}
```

Thanks For Your Attention

Reference

[1] CSS Electronics, LIN Bus Explained - A Simple Intro (2022), https://www.csselectronics.com/pages/lin-bus-protocol-intro-basics, Accessed on 01.06.2022

[2] Everything is Uploaded to the GitHub : https://github.com/EhamRafsan/Local-Interconnect-Network-LIN