LIN – A real “Plug 'n' Play” Bus System?

Standardized application functions enable the possibility for flexible, fast and cost effective LIN developments. Therefore Plug ’n’ Play will also be possible for automotive LIN applications.

The continuous growth of electronic content in cars allows improvements in comfort, safety and fuel economy. Integrating these functions requires the application of multiple bus systems to guarantee that the applicable modules can communicate at the required speed and safety levels. The vehicle manufacturers have developed different bus network standards to minimize cost while maximizing the performance.

The latest development in this area is now accepted on all new development platforms: The LIN bus (local interconnect network) - focusing primarily on the comfort area. Thanks to LIN it is possible to affordably interconnect the steadily increasing number of comfort functions. Additionally LIN can be implemented in applications that were previously impossible to network due to the prohibitive incremental cost of the networking features.
New demands require new solutions. A pure IC development for LIN slaves now becomes a complete system development.

The “typical” development path for LIN slaves
Typical LIN slave developments have many common points. Even though LIN slaves are usually ‘smart actuators’ or ‘smart sensors’, they are normally developed as electronic control units.

The supplier selects from familiar microcontroller choices the cheapest one with the appropriate performance and memory size. Based on the OEM specification, the applicable hardware is built around this microcontroller. A certified LIN2.x software API has to be applied to guarantee interoperability. This way LIN slaves become ‘plug and play’ on the network side.

However what does this mean on the application side? On the application side, required functions are implemented in software, meaning that for each LIN slave (see fig 1.) new application software has to be developed. Various suppliers end up with different application software on different microcontroller cores. Multiple developments stack up, creating a large demand on supplier resources for development. Each of these implementations needs to be thoroughly tested, with a risk of software bugs remaining.

The LIN conformance test only analyzes the LIN physical layer down to the API-function. Each new implementation of a LIN slave has to be checked to confirm that the mandatory API support functions have been implemented correctly. As with any software development flow, it is possible that bugs will appear late in the development flow due to limited test coverage. The consequence is that OEMs demand by default Flash controllers, adding additional IC complexity, memory size and development cost, to allow for fast recovery from software bugs. In the LIN standard, and in most OEM implementations, Flash reprogramming via the LIN bus has not been foreseen. Therefore this feature is not standardized and requires significant test resources for each of the different microcontroller concepts. The higher costs for certifying one class of microcontrollers lowers the interest for suppliers to change towards a cheaper competitive offer.

Discrete developments using standard microcontrollers require verification for each new application that the reaction time for LIN messages is guaranteed. The interrupt load from the application can only influence the LIN communication up to the extent that the required response timings are met. This may lead to the selection of a more powerful microcontroller than the application itself requires. A possible solution to guarantee timings is through the use of operating systems, which leads once more to additional demands on the microcontroller resources, processing power and memory. The alternative is the use of dual core microcontrollers that split the LIN communication task from the application task on the hardware level.

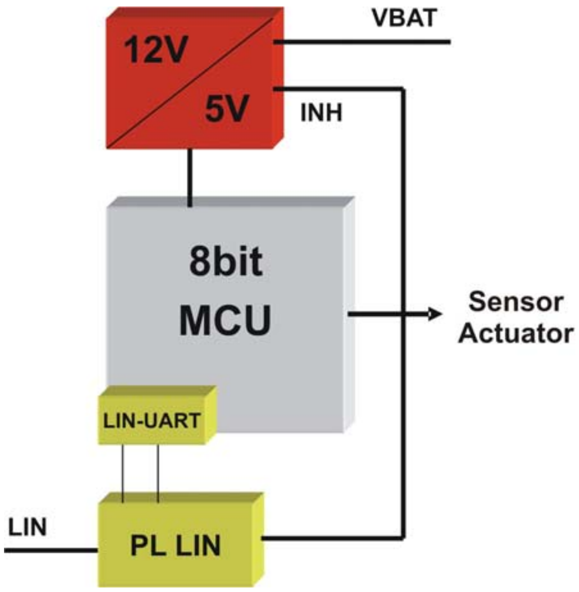


Figure 1 – Block diagram "typical" LIN Slave

In summary, when considering LIN slave developments in more detail, it becomes evident that each new development with slight differences in requirements forces some changes in software or hardware. Therefore LIN slaves are no longer plug and play, adding to system costs and risks.

The road to Plug and Play ready LIN slaves

How can Plug and Play LIN slaves be realized?

To answer this question, one should consider the commonalities between different applications. Typically the LIN slave consists of a communication task that manages the data exchange according to the LIN protocol, and an application task, that performs the actual function of the slave.

The interoperability of the LIN task is defined in the LIN 2.x specification, and is guaranteed through conformance tests, as well as additional tests at the vehicle manufacturer. This assures plug and play on the LIN side.

How does this look on the application side? To realize standardization on this level, the slave application should not be considered by itself, but in comparison to applications with very similar demands.

When running this exercise it becomes obvious that many slaves have similarities. As example we review one of the main LIN slave applications: LIN based switch modules. A switch, controlled through LIN, is represented in each application as a single bit of data. The switch has to be debounced according to an algorithm, in which the debounce times are defined by the applied type of switch. Differences come only from the way the switch is implemented, i.e. to ground or to supply, or as in a matrix configuration. Additional switch module functions include PWM control for background illumination or function lighting.

Real plug and play is only possible when, identical to the LIN task, the applications are standardized and configurable within practical limits.

Standardizing the LIN slave application functions

The ‘typical path’ for LIN slave development leads to a software development that can only be used for one specific application. LIN slaves from different suppliers executing exactly the same functionality look completely different. Today this is the standard practice. The individual development costs of each unique LIN module implementation have to be written off over the volumes. Thereby LIN may become prohibitively expensive for small volume platforms which should have been one of the benefits of a LIN architecture.

How can we resolve this inefficiency, and make the adoption of LIN more attractive on a wider scale?

First it is necessary to leave the ‘module level’ or even the sub-net level, and summarize the common requirements of LIN slaves in different local networks. Clearly this needs a view of the full car architecture and can only be executed with the vehicle manufacturer. The summary of all slave requirements across the different sub-networks can then be analyzed. Specific differences between the modules can be chosen as configurable on the slave side, or can be implemented on the master side. This results in a totally new methodology of defining LIN slaves and application functions. The initial effort required to consider the definition at a higher level will result in future reductions of component cost. The reduction in development cost and risk offers additional flexibility which will lead to faster time to market, and additional possibilities for product differentiation and branding for the vehicle manufacturer.

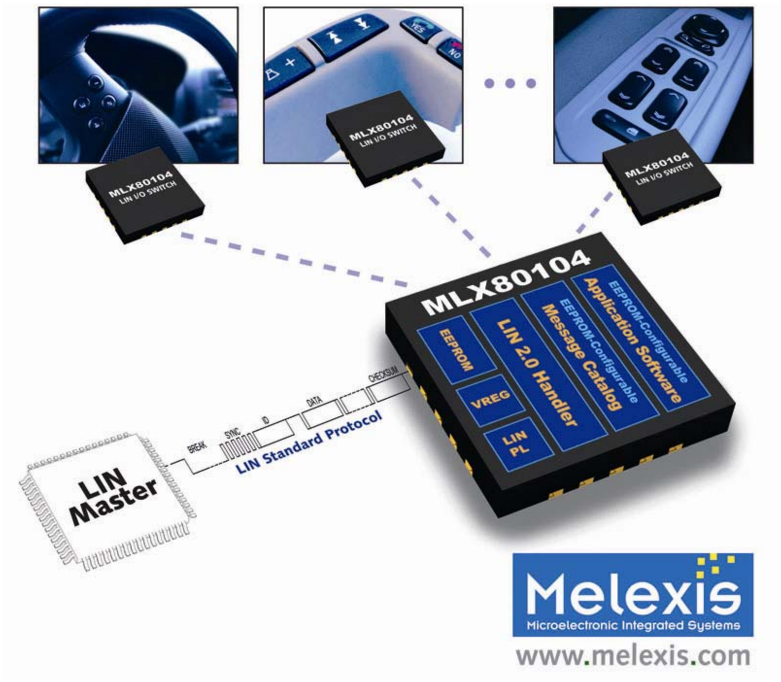


Figure 2 – Definition of a “Plug and Play” LIN Slave

Melexis (www.melexis.com) has applied this system level approach for LIN slave definition in collaboration with one of the major European vehicle manufacturers for the definition of a new platform. The requirements from several switch modules (door, seat, steering wheel, dashboard, roof, light switch) have been assembled. Additionally a message catalog has been defined and slave specific functions have been implemented on the master. This resulted in the MLX80104 integrated LIN slave that is now configurable for a very wide range of switch module applications. It can be applied on all these applications without the need for software development and re-qualification.

Implementing a switch module as a genuine Plug and Play system
LIN switch modules typically consist of a microcontroller, a voltage regulator and a LIN transceiver. The microcontroller should be capable to perform the different switch reading functions: single switch, matrix, resistive encoded, and rotating encoder, with the appropriate debouncing. Additionally it should be able to drive pulse width modulation (PWM) outputs for background lighting and function lighting LEDs. Finally, besides these application specific tasks it should be able to handle the different LIN dataframes. The above mentioned MLX80104 realizes all this in a single chip LIN slave (fig 3).
The combination of LIN transceiver + voltage regulator and LIN handler realizes a LIN slave that is certified from hardware as well as protocol point of view. This basic LIN subsystem is complemented with high voltage I/O's to realize I/O-extension applications like switch modules.

What makes this MLX80104 a real application specific standard product (ASSP)? How is it optimized for low cost LIN I/O extension applications, and for switch reading in particular?

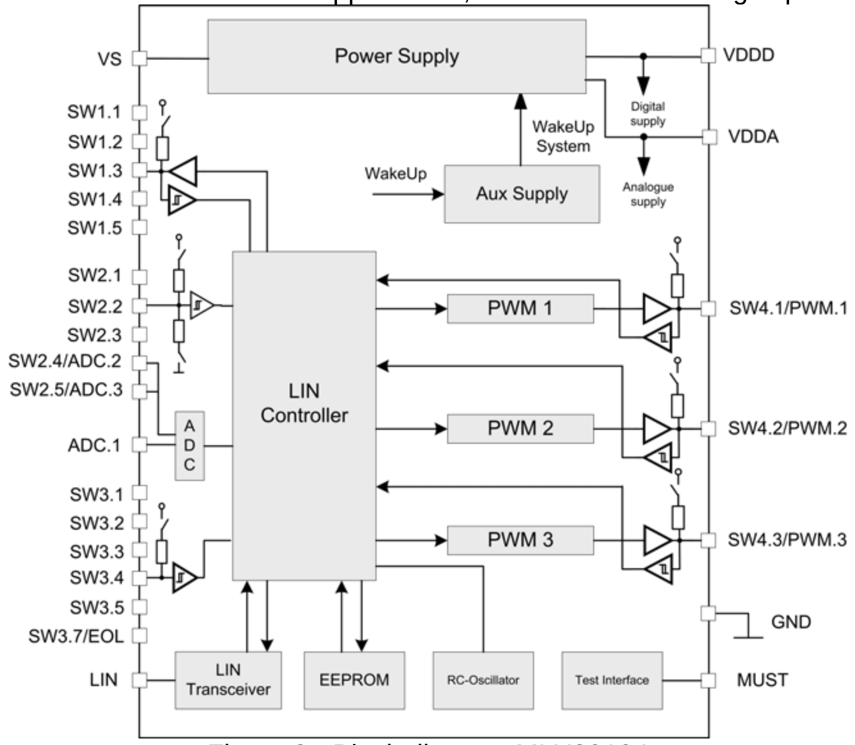


Figure 3 - Block diagram MLX80104

The unique concept of the MLX80104 resides in its dual core microcontroller. The dual core is optimized for handling the LIN protocol task independently on hardware level from the application task. Data exchange between the two tasks is synchronized by a hand shake protocol. This is similar to an operating system but realized in hardware, eliminating the cost premium.

The MLX80104 can be configured to match the requirements of a specific application through the setting of its EEPROM. The EEPROM is programmed End-Of-Line (EOL) using a dedicated EOL programming protocol. The stored data is complemented with a hamming encoding to ensure data integrity.

The LIN Task

The LIN task is responsible for the proper communication with the LIN master. The MLX80104 LIN protocol is certified according to the LIN2.0 standard, the baud rate can be set up to 20 kbaud, and diagnostic functions according to LIN2.0 are supported.

The initial LIN-IDs are configurable End-Of-Line in the EEPROM and can be modified by the master via the “Node Configuration Service” (Assign NAD). Besides the “normal” frames, bandwidth-saving “event triggered” frames are also supported.

The full message catalog has been defined in collaboration with the above mentioned vehicle manufacturer. The table shows an example of the LIN frame “Switch Status”.

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte 7	8Bit AD Signal (ADC.3)							
Byte 6	8Bit AD Signal (ADC.2)							
Byte 5	8Bit AD Signal (ADC.1)							
Byte 4	Node Error		SW3.5	SW3.4	SW3.3	SW3.2	SW3.1	SW25
Byte 3	SW24	SW23	SW22	SW21	SW20	SW19	SW18	SW17
Byte 2	SW16	SW15	SW14	SW13	SW12	SW11	SW10	SW9
Byte 1	SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1
Byte 0	LIN PID Event Triggered							

Figure 4 - LIN Frame "Switch Status"

Every I/O signal in the MLX80104 has been assigned to a dedicated set of bits in the message catalog. The assignment of signals on application functions is controlled by the LIN Master.
The sleep mode timer is also settable End-Of-Line. The MLX80104 can be configured to wake up from the LIN bus, from switch inputs, and from analog inputs.

The Applications Task

At the heart of the component resides the application task. The MLX80104 is configurable to meet a variety of application requirements.

Up to 15 I/O’s can are available to flexibly read switches. These IO’s can be configured End-Of-Line as single switches or as a matrix. Several I/O’s are high voltage, which allows the reading of remote switches while protecting the IC from short circuits to supply or to ground. The I/O’s can also be configured as output, for instance to control relays. Single switches and matrix switches can be configured to detect wake up events. Debounce timings are set by the user according to the needs of the specific switch types.

For background and function lighting 3 PWM outputs are available. The base frequency is the same for all 3 PWMs and can be programmed End-Of-Line in a wide range to meet standard automotive demands. Three 8bit Analog inputs are available to decode resistive encoded switches.

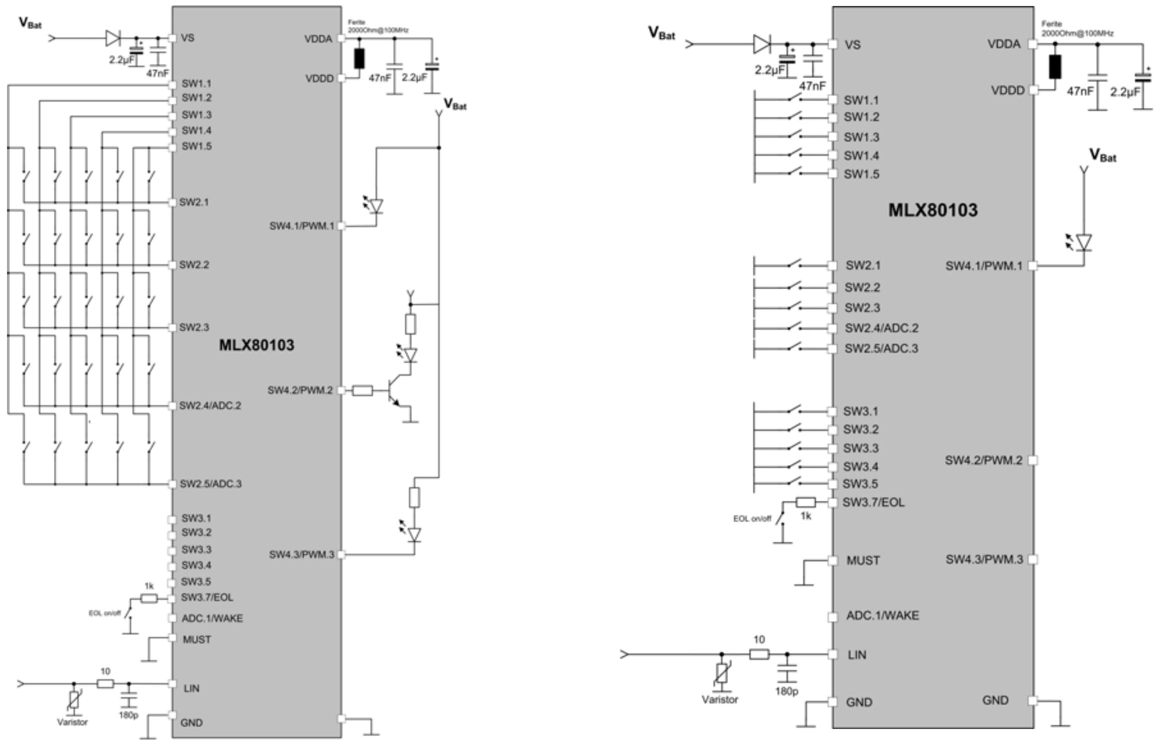


Figure 5 – Application Samples

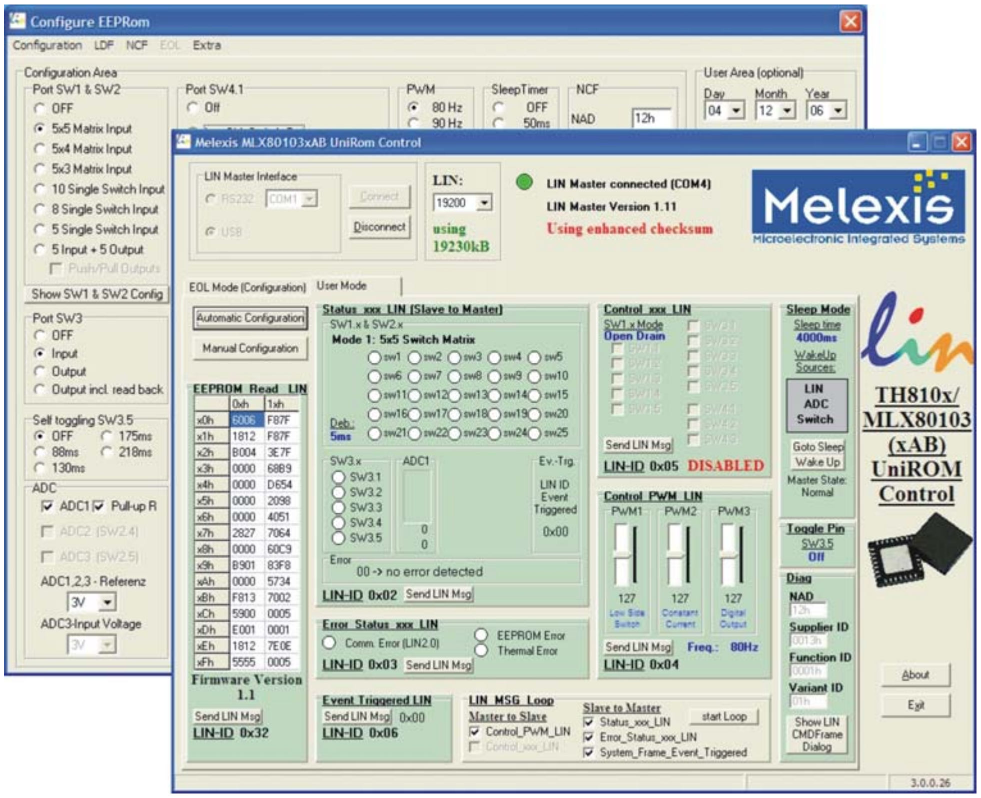
Customer specific data like production date, serial number etc, can be programmed in a dedicated area of the EEPROM during End-Of-Line.

Figure 5 shows two different application samples, both built around the MLX80104 with only a change in the EEPROM configuration.

The Development System

An evaluation kit is available to minimize the implementation effort around the component. This allows the realization of complete switch module A-samples in a matter of days.

The evaluation kit consists of PC software (Figure 6) that communicates via USB with a Melexis LIN Master that communicates via LIN with the switch module. These tools allow the user to configure the component in End-Of-Line mode, as well as to communicate in the final application mode.


Configure EEPRom

Configuration Area

Port SW1 & SW2: OFF [✓ 5x5 Matrix Input] 5x4 Matrix Input, 5x3 Matrix Input, 10 Single Switch Input, 8 Single Switch Input, 5 Single Switch Input, 5 Input + 5 Output

Port SW3: OFF [✓ Input] Output, Output incl. read back

Self toggling SW3.5: [✓ OFF] 175ms, 88ms, 218ms, 130ms

ADC: ☑ ADC1 ☑ Pull-up R

ADC1,2,3 - Referenz: 3V

PWM: [✓ 80 Hz] 90 Hz

SleepTimer: OFF, 50ms

NCF: NAD 12h

User Area (optional): Day 04, Month 12, Year 06

Melexis MLX80103xAB UniRom Control

LIN: 19200, using 19230kB

LIN Master connected (COM4)

LIN Master Version 1.11

Using enhanced checksum

EEPROM Read LIN

	0xh	1xh
x0h	6006	F87F
x1h	1812	F87F
x2h	B004	3E7F
x3h	0000	68B9
x4h	0000	D654
x5h	0000	2098
x6h	0000	4051
x7h	2827	7064
x8h	0000	60C9
x9h	B901	83F8
xAh	0000	5734
xBh	F813	7002
xCh	5900	0005
xDh	E001	0001
xEh	1812	7E0E
xFh	5555	0005

Firmware Version 1.1

LIN-ID 0x32

Status xxx LIN (Slave to Master): Mode 1: 5x5 Switch Matrix, Deb.: 5ms

Error: 00 -> no error detected

LIN-ID 0x02

Error Status xxx LIN: LIN-ID 0x03

Control xxx LIN: SW1.x Mode Open Drain, LIN-ID 0x05 DISABLED

Control PWM LIN: PWM1 127, PWM2 127, PWM3 127, Freq.: 80Hz, LIN-ID 0x04

Event Triggered LIN: 0x00, LIN-ID 0x06

Sleep Mode: Sleep time 4000ms

Toggle Pin: SW3.5 Off

Master State: Normal

TH810x/ MLX80103 (xAB) UniROM Control

3.0.0.26

Figure 6 – Configuration and Test Software

The communication uses the standard LIN protocol; therefore other LIN tools can be used to communicate with the switch module as well. For serial production the programming sequence is easy to implement on the user’s End-Of-Line tester.

Summary

A LIN slave developed in the ‘typical’ way bares additional costs due to the need for software development and testing. To avoid this, a new LIN slave definition methodology is required. For applications suitable to this ‘high level’ approach the specification should be done at the vehicle architecture level in order to fully leverage the benefits of LIN as a low cost network solution. To summarize the application requirements from the individual sub networks, collaboration with the vehicle manufacturer is indispensable.

Michael Bender
Product Marketing Manager

Or for additional information contact Melexis Direct:

Europe, Africa, Asia:	America:
Phone: +32 1367 0495	Phone: +1 603 223 2362
E-mail: sales_europe@melexis.com	E-mail: sales_usa@melexis.com

ISO/TS 16949 and ISO14001 Certified