

## LIN Slave example project

1.10

### LIN Features

- LIN 2.1 Slave Node implementation
- Automatic baud rate synchronization
- Transport Layer Raw API
- Automatic Configuration Services handling
- Usage demonstration of both static and dynamic API

### General Description

This example project demonstrates the LIN Slave component operation. The unconditional frame is received with the new value of the scalar signal. The new value of this signal is read from frame, incremented and then written to the scalar signal located at another unconditional frame. This makes this signal's value available to LIN Master.

LIN Slave example also demonstrates Transport Layer (Raw API) and Automatic Configuration Requests Handling operation.

The operation of Raw API is demonstrated in the following way: when MRF is received, the SID byte of the request is incremented and response is sent to LIN Master with the requested SRF. The operation of Automatic Configuration Requests Handling is demonstrated with two services - 0xB0 (Assign NAD) and 0xB2 (Read by identifier).

As a result of MRF reception with the 0xB0 service request, the LIN Slave NAD value is updated with the NAD assigned by LIN Master. The result of request is sent to LIN Master in SRF. After successful execution of 0xB0 service LIN Slave will ignore all the MRF frames with old NAD.

Service 0xB2 performs LIN Slave identification using supplier and function IDs. As a result of a successful "Read by identifier" service execution is a positive response that sent to Master inside of requested SRF.

**Note.** If some failure will occur on the Transport Layer level, it will be reinitialized to its default state. This means current LIN Slave NAD will be overwritten initial NAD. This may cause frames that addressed with a changed NAD to be ignored.

## Development kit configuration

1. This project is written for Cypress kit CY8CKIT-001 PSoC® Development Kit equipped with a 2x16 alphanumeric LCD module and CY8CKIT-017 CAN/LIN Expansion Board Kit.
2. Jumpers on the CY8CKIT-001 PSoC Development Main Board have a default setting for 3.3V operation. Configure the board to 5V operation by moving SW3 - VDD Select to 5V position (up position).
3. The CAN/LIN Expansion Board connects to the CY8CKIT-001 PSoC DVK through the 20x2-pin connector. It hooks up to the DVK through the Port A. Jumpers on the CY8CKIT-017 CAN/LIN Expansion Board have a default setting for 5V operation.
4. Place the PSoC 3 or PSoC 5 processor module on the CY8CKIT-001 DVK. Set corresponding device in the example project under "Device Selector" item of the context menu.
5. Power the DVK using either battery connections or a wall power unit.
6. Connect the MiniProg3 JTAG cable to the JTAG connector, both on MiniProg3 and the PSoC 3 processor module. Connect the MiniProg3 to a host PC USB high speed port using a USB cable.
7. Build the project and program the hex file on to device using MiniProg3.
8. Power cycle the device and observe the results on the LCD, refer to the Result section for more information.

## Importing Node Capability File (NCF)

Place one more LIN component macro onto schematic to verify correctness of configuration import from the NCF. Note that only one LIN component per design is expected to be used.

Open customizer of the newly added LIN component, select Import File, select "Node Capability File, (\*.ncf)" extension in the opened window, browse to the example project location You have selected while opening project and point to the "LIN\_Slave\_Example.ncf" file. Observe import results at "Import NCF File Status" window. Manually configure options that were not configured during import. Click "OK" button to finish configuration import process.

Compare two LIN component configurations. They have to be equal, with the following exceptions: Bus Inactivity, Break Detection Threshold, frames Default ID field and Transport Layer API format. These options should be set manually as, according to the LIN 2.1 specification NCF file has no possibility to save these configuration options.



Remove originally existent LIN component and rename newly placed from LIN\_1 to LINS. Build project and verify its operation correctness according to the guidance provided under General Description and the Expected Results sections.

## Importing LIN Description File (LDF)

Place one more LIN component macro onto schematic to verify correctness of configuration import from the LDF. Note that only one LIN component per design is expected to be used.

Open customizer of the newly added LIN component, select Import File, select "LIN Description File, (\*.ldf)" extension in the opened window, browse to the example project location you have selected while opening project and point to the "LIN\_Slave\_Example.ldf" file. Click "Open" button. Select node configuration you want to import and press "OK". Observe import results at "Import LDF File Status" window. Manually configure options that were not configured during import. Click "OK" button to finish configuration import process.

Remove originally existent LIN component and rename newly placed from LIN\_1 to LINS. Build project and verify its operation correctness according to the guidance provided under General Description and the Expected Results sections.

## Using a LIN Bus Analyzer Tool

This example project demonstrates functionality of the LIN Slave device, so the LIN Master device must be also used. The LIN bus analyzer or emulator tool can be used as LIN Master device. Or, it is even possible to use any other LIN Master node to communicate with this example project.

If you use a LIN bus analyzer or emulator tool to communicate with this example project, then the tool must be set up to send and receive frames with proper length and ID and at a proper baud rate.

## Expected Results

When the project is started the LIN component is initialized and, in case of successful startup procedure, the “LIN Example, Init succeed” message will be displayed on the LCD. When the frame with ID equals 01 is received, the value of the **SignalIn** and **SignalOut** signals will be displayed – “SignalIn: 01, SignalOut: 03”, where **SignalOut** value is incremented by 2 value of **SignalIn**. The same display scheme will be observed when next frame with ID equal 01 will be received by LIN Slave.

When MRF frame is received by LIN Slave, its value is displayed after “Received frame:” message. Then this text is replaced with “Sent frame:” text with the frame data that will be sent to LIN Master as a result of SRF. For the Automatic Configuration Request Handler frames LCD will hold the last transmitted frame or signal.

The example project functionality demonstration is depicted below on Figure 1.

Figure 1. LIN Slave communication diagram.

Protected Identifier	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Checksum
0xC1	SignalIn = 0x01	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF		0x3D (enhanced)
0x80	0xFE (RE)	SignalOut = 0x03	0xFF	0xFF	0xFF	0xFF			0x7D (enhanced)
0xC1	SignalIn = 0xAA	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF		0x93 (enhanced)
0x80	0xFE (RE)	SignalOut = 0xAC	0xFF	0xFF	0xFF	0xFF			0xD3 (enhanced)
0x3C	NAD = 0x10	Data Len = 0x06	USER SID = 0x12	0x34 (D1)	0x56 (D2)	0x78 (D3)	0x9A (D4)	0x00 (D5)	0x3A (classic)
0x7D	NAD = 0x10	DataLength = 0x06	USER RSID = 0x13	0x34 (D1)	0x56 (D2)	0x78 (D3)	0x9A (D4)	0x00 (D5)	0x39 (classic)
0x3C	NAD = 0x10	DataLength = 0x06	ACRH SID = 0xB0	0xFE (SP ID LO)	0x7F (SP ID HI)	0x41 (FNC ID LO)	0x48 (FNC ID HI)	0x20 (NEW NAD)	0x11 (classic)
0x7D	NAD = 0x10	DataLength = 0x01	ACRH RSID = 0xF0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFD (classic)
0x3C	NAD = 0x20	DataLength = 0x02	USER SID = 0x0F	0x11 (D1)	0xFF	0xFF	0xFF	0xFF	0xBD (classic)
0x7D	NAD = 0x20	DataLength = 0x02	USER RSID = 0x10	0x11 (D1)	0xFF	0xFF	0xFF	0xFF	0xBC (classic)
0x3C	NAD = 0x20	DataLength = 0x06	ACRH SID = 0xB2	0x00 (ID)	0xFE (SP ID LO)	0x7F (SP ID HI)	0x41 (FNC ID LO)	0x48 (FNC ID HI)	0x11F (classic)
0x7D	NAD = 0x20	DataLength = 0x06	ACRH SID = 0xF2	0xFE (SP ID LO)	0x7F (SP ID HI)	0x41 (FNC ID LO)	0x48 (FNC ID HI)	0x00 (Variant)	0xDE (classic)

Legend:

Data published by LIN Master

Data published by LIN Slave

RE – Response Error signal  
 SP ID LO – Supplier ID LSB  
 SP ID HI – Supplier ID MSB  
 FNC ID LO – Function ID LSB  
 FNC ID HI – Function ID MSB

© Cypress Semiconductor Corporation, 2009-2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

