Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE96999 - Basic LIN Slave Implementation in PSoC® 4

Objective

These code examples demonstrate the implementation of LIN slave communication in PSoC® 4.

Overview

These code examples show how to implement a LIN slave using the LIN Component in PSoC 4. PSoC 4 devices support LIN slaves with both LIN v1.3 and LIN v2.1/2.2 protocol specifications. Two examples are provided:

Example1 - LIN Slave Communication:

In this code example, PSoC 4 responds to LIN master commands to:

- Set the RGB LED color on PSoC development kits such as CY8CKIT-042 and CY8CKIT-044.
- Report the current RGB LED color setting

Example2 – Multiple-Instance LIN Slave Communication:

In this code example, PSoC 4 is initialized to have two LIN slave Components, which are connected to two different LIN networks (masters). PSoC 4 responds to both LIN master (LIN master1 and LIN master2) commands to do the following:

Respond to LIN master1 commands:

- Store the CapSense® linear slider centroid position
- Report the saved CapSense linear slider centroid position value

Respond to LIN master2 commands:

- Set the RGB LED color on PSoC development kits such as CY8CKIT-042
- Report the current RGB LED color setting

Requirements

Tool: PSoC Creator 3.3 SP1 or later

Programming Language: C (ARM® GCC 4.9.3 and ARM MDK compilers)

Associated Parts: All PSoC 4100, 4200, 4100-M and 4200-M parts

Related Hardware: CY8CKIT-042, CY8CKIT-044, CY8CKIT-026, Model 9011 LIN to USB Data Converter or equivalent, Jumper wires[1].

[1] Wires that are used to connect from CY8CKIT-026 Arduino™ headers to LINx Tx, LINx Rx and LINx NSLP pins on the same board.

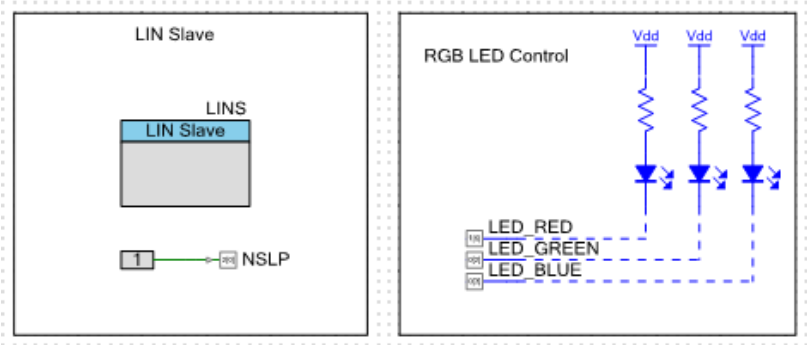
Example1 – LIN Slave Communication

Design

In this example, PSoC 4 acts as a simple LIN slave. The slave monitors for data, which is transmitted from the LIN master (analyzer). If a predefined frame is received from the master, the slave controls the RGB LED color as per the data that is available in the received frame. The LIN master can get the RGB LED status by sending a frame with a predefined frame ID. This example project can be used with both CY8CKIT-042 and CY8CKIT-044.

Figure 1 shows the PSoC Creator schematic design of the code example.

Figure 1. LIN Slave Design with PSoC 4



Note: NSLP is the output pin used to keep the LIN transceiver either in sleep mode or active mode.

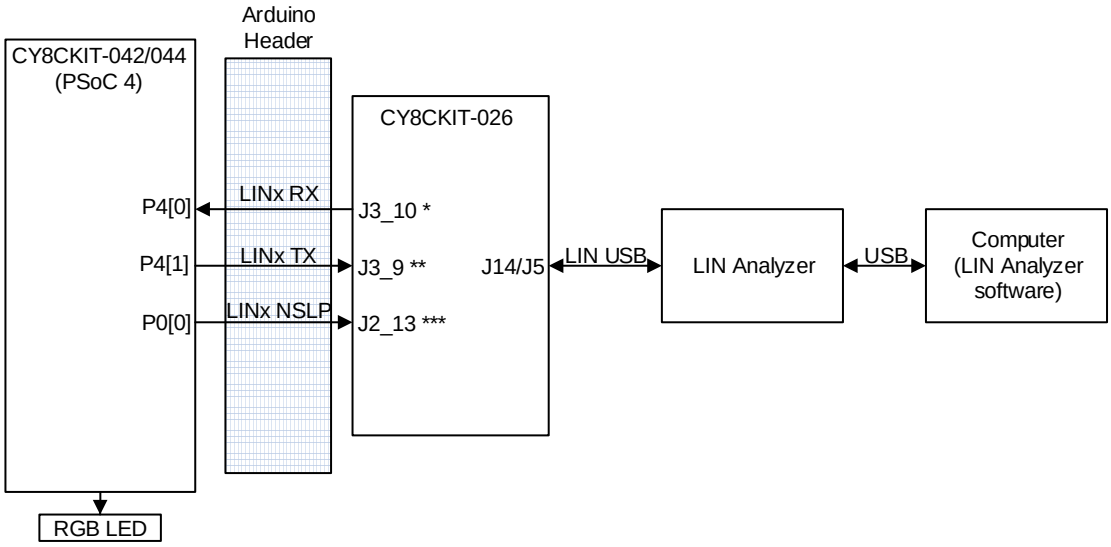
Design Considerations

- This code example has been designed for both the CY8CKIT-042 PSoC 4 Pioneer Kit and CY8CKIT-044 PSoC 4 M-Series Pioneer Kit.
- The maximum baud rate that LIN protocol supports is 20 kbps. The LIN Component has four different baud rates in the configuration UI: 19.2 kbps, 10.4 kbps, 9.6 kbps, and 2.4 kbps.

Hardware Setup

The hardware setup block diagram and connections are shown in Figure 2; detailed hardware connections are explained below.

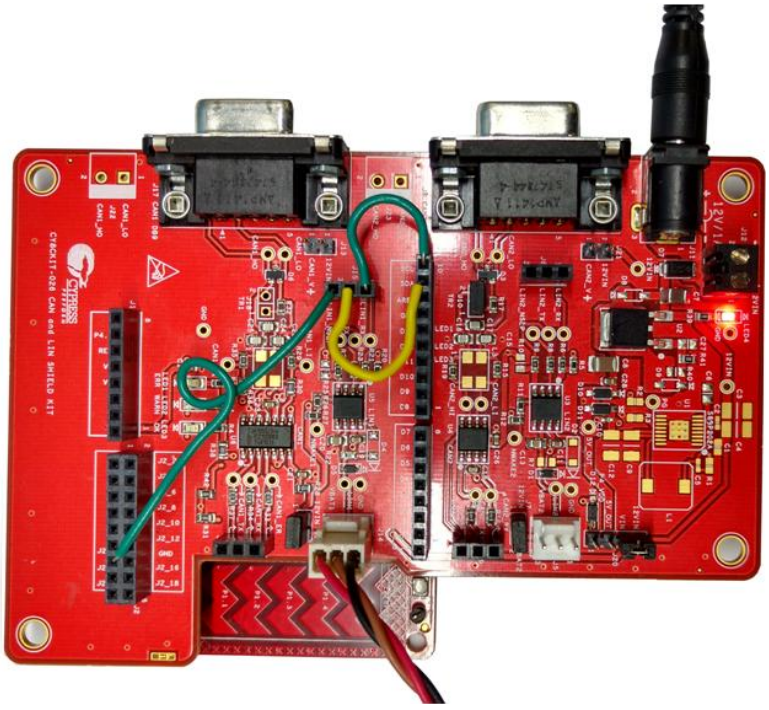
Figure 2. LIN Slave Communication Hardware Setup



* Connect J3_10 to J15_1 (LIN1_RX) or J6_1 (LIN2_RX)

** Connect J3_9 to J15_2 (LIN1_TX) or J6_2 (LIN2_TX)

*** Connect J2_13 to J15_3 (LIN1_NSLP) or J6_3 (LIN2_NSLP)



Follow these instructions to set up the hardware:

1. Because there are two LIN transceivers on CY8CKIT-026, choose either of the transceivers to use (U5 – LIN1 transceiver or U3 – LIN2 transceiver). Connect the Arduino header pins (which are connected to the base board controller) to the appropriate LIN transceiver using jumper wires as shown in Table 1.

Table 1. Pin Connection on CY8CKIT-026

Arduino Header Pins	CY8CKIT-026 Pins	
	LIN1 Transceiver	LIN2 Transceiver
J3_10	J15_1 (LIN1_RX)	J6_1 (LIN2_RX)
J3_9	J15_2 (LIN1_TX)	J6_2 (LIN2_TX)
J2_13	J15_3 (LIN1_NSLP)	J6_3 (LIN2_NSLP)

2. Make sure that jumper J16 (if the LIN1 transceiver is being used) is shorted or jumper J7 (if the LIN2 transceiver is being used) is shorted.

   Note: This is to provide a 12-V supply to the Silicon Engines LIN-USB analyzer; if any other analyzer is being used, then follow the instructions for power supply requirements and short the jumper only if it requires a 12-V supply.

3. The baseboard can be powered using USB or it can be powered from the Shield kit by selecting jumper J20 appropriately as shown in Table 2. See the CY8CKIT-026 user guide for more details.

Table 2. Powering Options with Jumper (J20)

J20 Connection	Power Option	Baseboard (CY8CKIT-042/44) Requirement
Short pin 2 and 3	Power baseboard using Shield Kit with 5 V	Baseboard power selection jumper (J9 on CY8CKIT-042/ 044) should be at 3.3 V (only if USB is not connected to the baseboard).
Short pin 3 and 4	Power baseboard using Shield Kit with 12 V	Baseboard power selection jumper (J9 on CY8CKIT-042/ 044) should be at 3.3 V (only if USB is not connected to the baseboard).

   Note: If the baseboard is powered through USB, and a 12-V supply is connected to the Shield Kit, the position of the power jumper selection (J20) is not significant.

4. Plug in CY8CKIT-026 to CY8CKIT-042/044 through the Arduino-compatible connectors.

5. Connect the LIN analyzer to the J14 connector to use LIN1 transceiver or J5 connector to use LIN2 transceiver.

   Warning: Be careful to not power up the Shield Kit with multiple supplies.

6. If the LIN analyzer DOES NOT provides 12 V to the VBAT pin, connect a 12-V supply to CY8CKIT-026 through the J11 power jack or J12 screw terminal connector.

7. If the LIN analyzer DOES provide 12 V to the VBAT pin, the Shield kit can be powered from the LIN analyzer if desired. For that configuration, the J16 jumper (for LIN1 transceiver) or the J7 jumper (LIN2 transceiver) should be placed.

   Note: The Silicon Engines LIN-USB analyzer requires a 12-V input supply, so either J16 (for LIN1) or J7 (for LIN2) must be placed and a 12-V supply must be provided.

Components

Table 3 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each.

Table 3. List of PSoC Creator Components

Component	Hardware Resources
LIN	SCB
Pin	3 pins for LEDs, 1 pin for LIN transceiver sleep functionality 2 pins for LIN Tx/Rx (part of the LIN Component)

Parameter Settings

Figure 3 and Figure 4 shows the parameter settings for the PSoC Creator LIN Component used in the code example. Only the parameters that vary from their default values are shown.

LIN Component

Figure 3 shows the Frame tab settings for the LIN Component. Use this tab to add a new frame or delete an existing frame from the LIN slave. Click the 'Add' button to add a new frame. For this example, two frames named "InFrame" and "OutFrame" are used as shown in Figure 3.

- **InFrame:** The frame ID value is 0x10, the direction of the frame is 'Subscribe' and the frame type is 'Unconditional'.
- **OutFrame:** The frame ID value is 0x11, the direction of the frame is 'Publish' and the frame type is 'Unconditional'.

Note: If the direction (publish/subscribe) is set as 'Publish', the slave responds to master; if the direction is set as 'Subscribe', the slave uses the received data from the master for its application.

Figure 3. LIN Component Frames Tab Settings

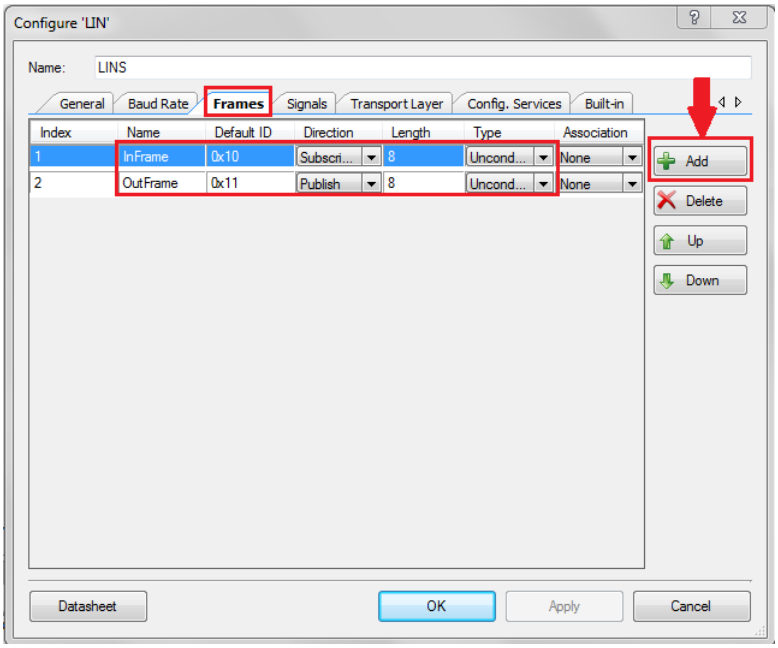
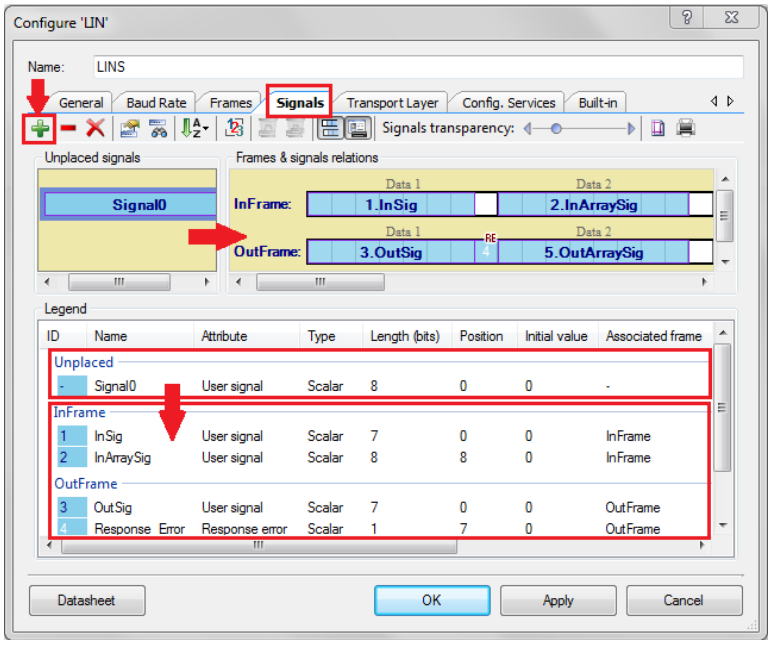


Figure 4 shows the Signals tab settings for the LIN Component. Click the '+' button to add new signals and select the signal properties such as signal name, type, length, and initial value. Place the signals to the corresponding frames as shown the Figure 4.

Figure 4. LIN Component Signals Tab Settings



All other configuration settings are left at their default.

Design-Wide Resources

Figure 5 and Figure 6 shows the pin assignments for this code example.

Figure 5. Pin Assignments for PSoC 4200 (CY8CKIT-042) Example

Name	Port	Pin	Lock
\LINS:SCB:rx\	P4[0]	20	☑
\LINS:SCB:tx\	P4[1]	21	☑
LED_BLUE	P0[3]	27	☑
LED_GREEN	P0[2]	26	☑
LED_RED	P1[6]	43	☑
NSLP	P0[0]	24	☑

Figure 6. Pin Assignments for PSoC 4200M (CY8CKIT-044) Example

Name	Port	Pin	Lock
\LINS:SCB:rx\	P4[0]	27	☑
\LINS:SCB:tx\	P4[1]	28	☑
LED_BLUE	P6[5]	16	☑
LED_GREEN	P2[6]	8	☑
LED_RED	P0[6]	45	☑
NSLP	P0[0]	39	☑

All other design-wide resources are left at their default values.

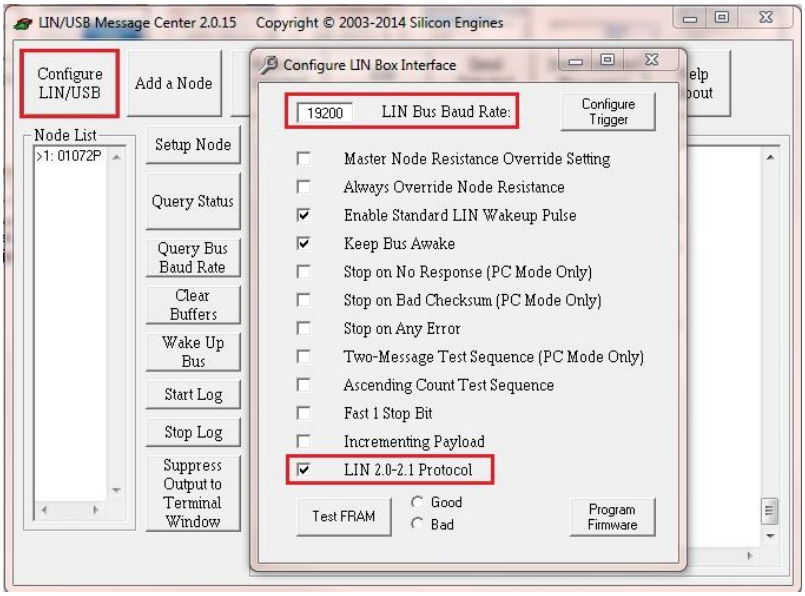
Operation

To test the project a “Silicon Engines LIN-USB Converter – Model 9011” is used as LIN analyzer. Other LIN analyzers may also be used.

Do the following to complete the testing:

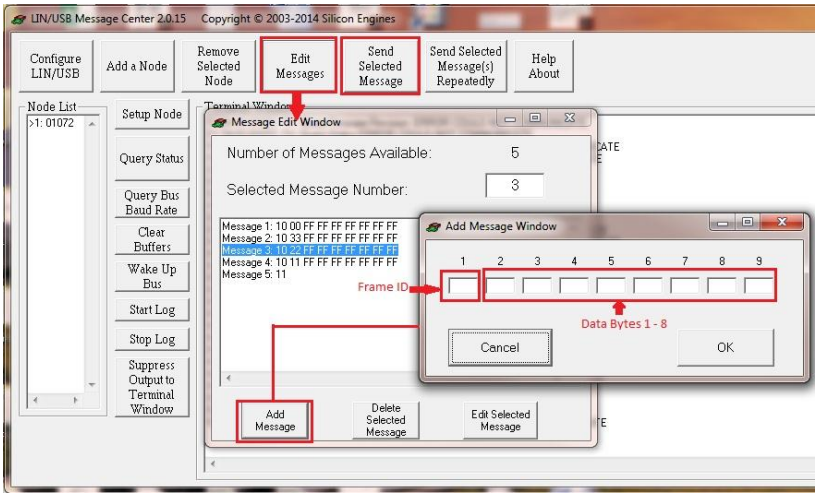
- Make the hardware connections as described in Hardware Setup section on page 3.
- Program the hex file on to the baseboard (CY8CKIT-044/042) using PSoC programmer or PSoC Creator.
- Install the ‘SE9004 and 9011 LINUSB Message Center’ software (if a different analyzer is being used, install the appropriate software) on your PC.
- Connect the LIN analyzer to a PC through a USB cable and open the LIN analyzer software in PC.
- On the LIN analyzer software, go to **Configure LIN/USB**. The Configure LIN Box Interface window is displayed. Set the **LIN Bus Baud Rate** to 19200 bps, and then select the **LIN 2.0-2.1 Protocol** checkbox, as shown in Figure 7.

Figure 7. LIN Analyzer Configurations



- If you are using any other analyzer, make sure that the checksum setting is selected as enhanced checksum since LIN v2.1/2.2 specification supports only enhanced checksum (this option is not required in Silicon Engines LIN-USB converter software).
- Add the message in the analyzer software that should be transmitted to the slave and send it through the analyzer, as shown in Figure 8, Note that the message must start with the ID.

Figure 8. Adding and Sending Message Using LIN Analyzer



- If a frame with ID = 0x10 is received from the master (analyzer), the slave controls the RGB LED based on the received data command from the master as shown in Table 4. Note that the message must start with the ID and must contain eight data bytes even though only the first byte is used to control the LEDs. The other seven data bytes can be any value.

Table 4. Slave Response per Commands from Master

Command	Slave Response
0x11	Turns on Red LED
0x22	Turns on Green LED
0x33	Turns on Blue LED
0x00	Turns off RGB LED

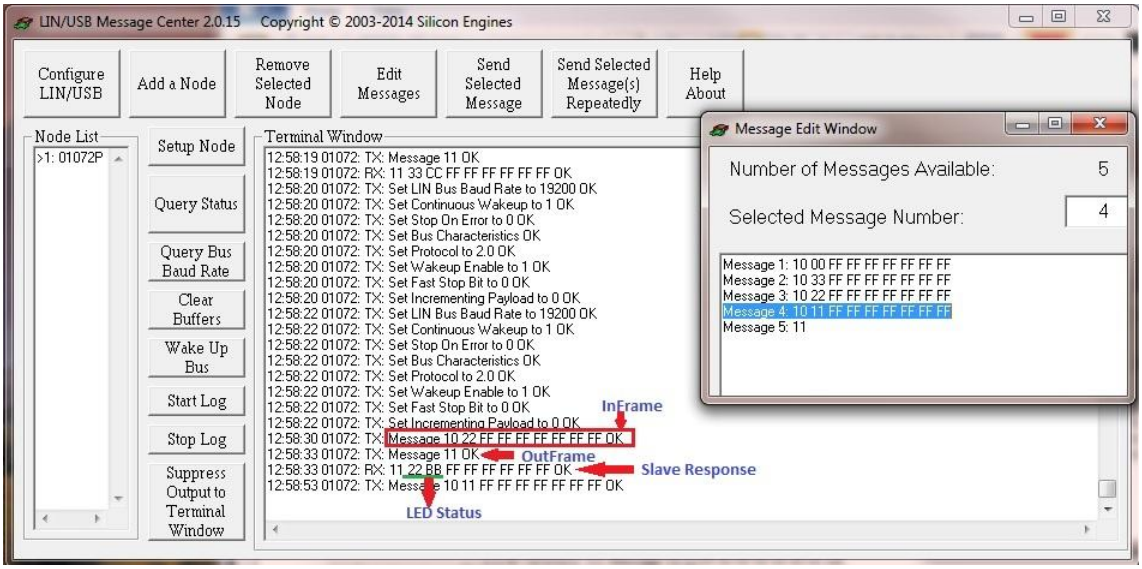
- If a frame with ID = 0x11 is received from the master (analyzer), then the slave sends the RGB LED status back to the master as shown in Table 5. The message in this case only needs the message ID. No data bytes are required.

Table 5. RGB LED Status

RGB LED status	Data byte
Red LED on	0xAA
Green LED on	0xBB
Blue LED on	0xCC
RGB LED off	0xDD

- The result of transmitted and received data at the LIN analyzer is shown in Figure 9.

Figure 9. Results at LIN Analyzer



In this figure, InFrame refers to “10 22 FF FF FF FF FF FF FF” where 10 is the frame ID and 22, FF, FF, FF, FF, FF, FF, FF are the eight data bytes. Because the 'InFrame' in LIN slave is configured with only two bytes named as ‘InSig’ and ‘InArraySig’, the rest of the data bytes (3 to 8) are ignored by the slave. When the 'OutFrame' is received from the analyzer as marked in this figure, the slave responds to the frame with the RGB LED status along with frame ID as “11 22 BB” where '11' is frame ID, '22' is the previous 'InFrame' data byte (command) and 'BB' is the RGB LED status (i.e., Green LED is ON).

Note: If there is an error message such as “BUS STUCK HIGH” or “TX FRAME ERROR”, reset the baseboard (CY8CKIT-042/44) and LIN analyzer.

Example2 – Multiple-Instance LIN Slave Communication

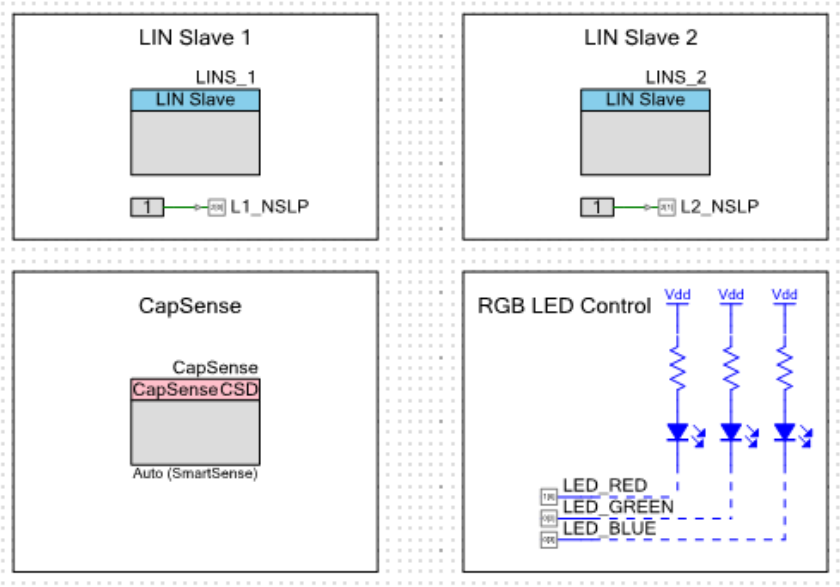
Design

In this example, PSoC 4 acts as two LIN slave nodes; each node is connected to a different LIN network[1]. PSoC 4 continuously scans the CapSense linear slider and then responds to LIN master1 and LIN master2 frames as follows:

- Depending on the frame (Frame ID) received from LIN master1, slave1 (LINS_1) either saves the current CapSense linear slider centroid position or sends the saved linear slider centroid value back to the master.
- Depending on the frame (Frame ID) received from LIN master2, slave2 (LINS_2) either controls the RGB LED color per the data that is available in the received frame or sends the RGB LED status to the master. Note that this is the same functionality as the first example.

Figure 10 shows the PSoC Creator schematic design of the code example2.

Figure 10. Multiple LIN Slaves Design with PSoC 4



Note L1_NSLP, L2_NSLP are the output pins used to keep the LIN transceivers either in sleep mode or active mode.

Note *[1] LIN slave nodes are connected to a LIN master forming a LIN network.

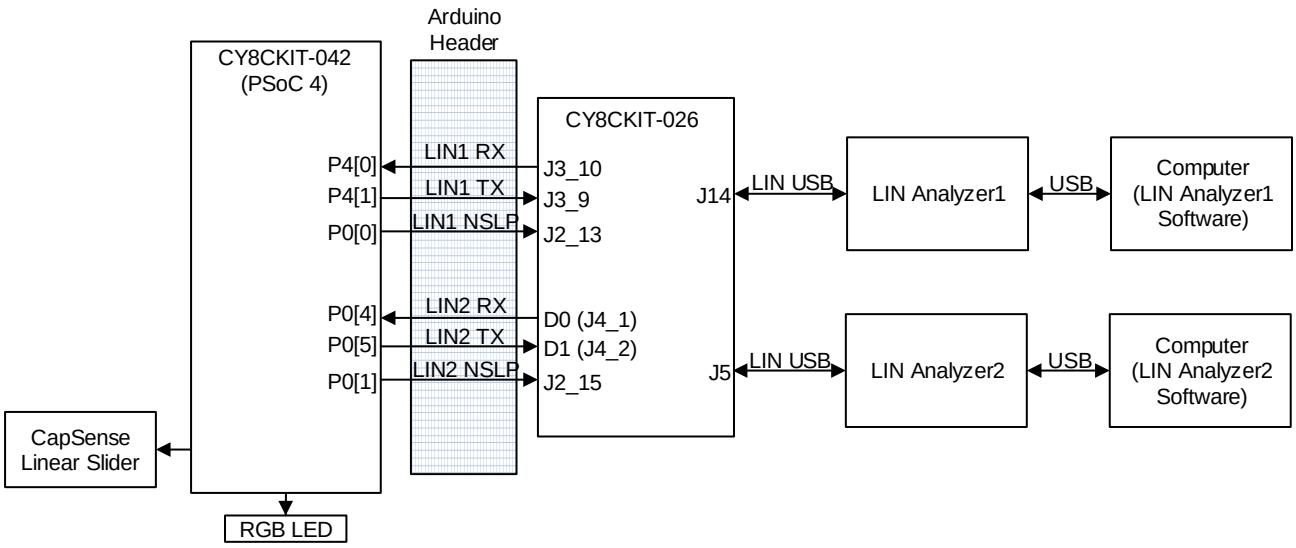
Design Considerations

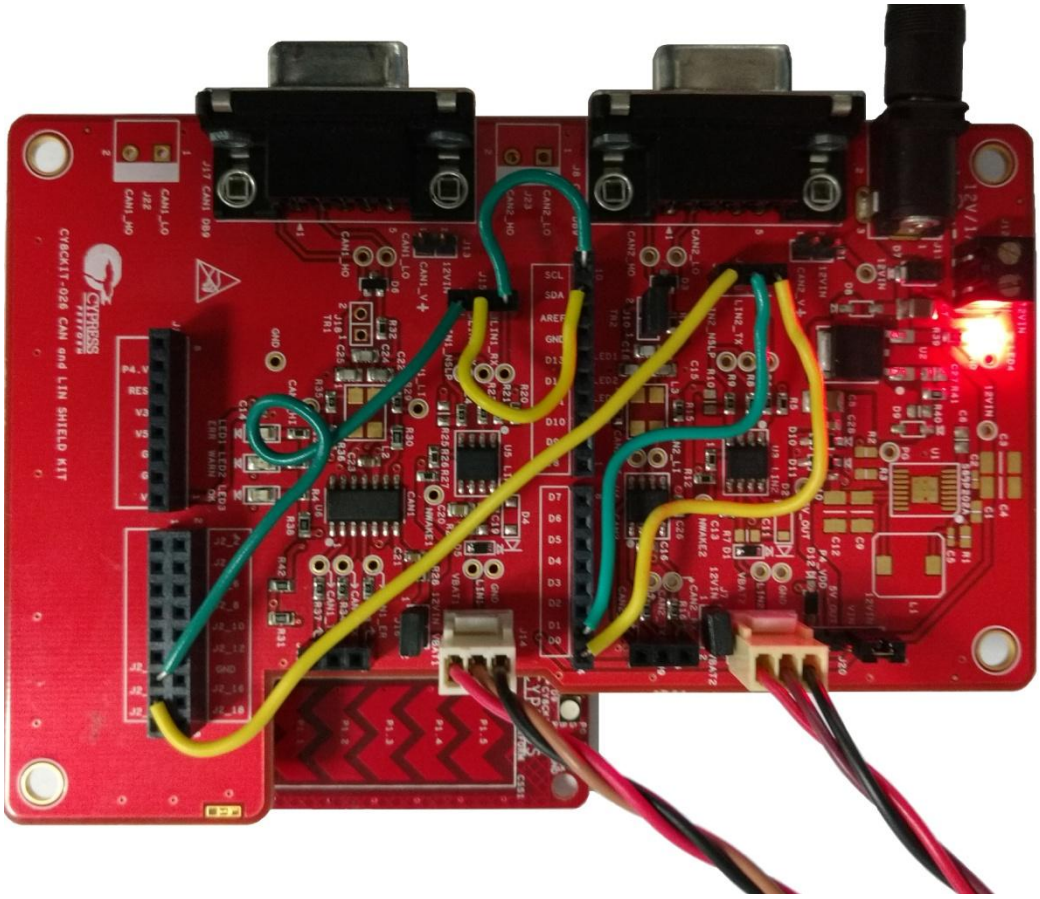
- This code example has been specifically designed for the CY8CKIT-042 PSoC 4 Pioneer Kit.
- The maximum baud rate that LIN protocol supports is 20 kbps. The LIN Component has four different baud rates in the configuration UI: 19.2 kbps, 10.4 kbps, 9.6 kbps, and 2.4 kbps.
- Only two instances of LIN slaves are supported in PSoC 4.

Hardware Setup

The hardware setup block diagram and connections are shown in Figure 11 and the detailed hardware connections are explained below in detail.

Figure 11. Example2 Hardware Setup





Follow these instructions to set up the hardware:

1. Connect the Arduino-compatible header pins (which are connected to the baseboard controller) to the appropriate LIN transceiver through jumper wires as shown in Table 6.

Table 6. Pin Connection to CY8CKIT-026

Arduino Header Pins	CY8CKIT-026 Pins
J3_10	J15_1 (LIN1_RX)
J3_9	J15_2 (LIN1_TX)
J2_13	J15_3 (LIN1_NSLP)
D0	J6_1 (LIN2_RX)
D1	J6_2 (LIN2_TX)
J2_15	J6_3 (LIN2_NSLP)

2. Short the two pins on the J16 and J7 jumpers.

   Note: This is to provide a 12-V supply to the Silicon Engines LIN-USB analyzers; if any other analyzer is being used, then follow the analyzer instructions on the power supply requirements and short the jumper only if it requires a 12-V supply.

3. The baseboard can be powered using USB or from the Shield kit by selecting the jumper J20 as shown in Table 7. See the CY8CKIT-026 user guide for more details.

Table 7. Powering Options with Jumper (J20)

J20 Connection	Power Option	Baseboard (CY8CKIT-042/44) Requirement
Short pin 2 and 3	Power baseboard using Shield Kit with 5 V	Baseboard power selection jumper (J9 on CY8CKIT-042/ 044) should be at 3.3 V (only if USB is not connected to the baseboard).
Short pin 3 and 4	Power baseboard using Shield Kit with 12 V	Baseboard power selection jumper (J9 on CY8CKIT-042/ 044) should be at 3.3 V (only if USB is not connected to the baseboard).

   Note: If the baseboard is powered through USB and a 12-V supply is connected to the Shield Kit, then the position of the power jumper selection (J20) is irrelevant.

4. Plug in CY8CKIT-026 to CY8CKIT-042 through the Arduino connector.
5. Connect the LIN analyzer1 to the J14 connector and LIN analyzer2 to J5 connector on the CY8CKIT-026 kit.

   Warning: Be careful to not power up the Shield Kit with multiple supplies.

6. If the LIN analyzers DO NOT provide 12 V to the VBAT pin, connect a 12-V supply to CY8CKIT-026 through the J11 power jack or the J12 screw terminal connector.
7. If one or both LIN analyzers DO provide 12 V to VBAT pin, the Shield Kit can be powered from one of the analyzers. For that option, either the J16 jumper (for LIN1 transceiver) or J7 jumper (LIN2 transceiver) should be placed. Be careful not to power up the Shield Kit with multiple supplies.

Components

Table 8 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each.

Table 8. List of PSoC Creator Components

Component	Hardware Resources
LIN (2 instances)	2xSCBs
CapSense	CSD0
Pin	3 pins for LEDs 2 pins for LIN slave sleep functionality 5 pins for CapSense (part of the CapSense Component) 1 pin for Cmod (part of the CapSense Component) 2 pins for LIN1 Tx/Rx (part of the LIN Component) 2 pins for LIN2 Tx/Rx (part of the LIN Component)

Parameter Settings

Figure 12 through Figure 17 show the parameter settings for each of the PSoC Creator LIN Components used in the code example. Only the parameters that vary from their default values are shown.

LIN Component - 1 (LINS_1)

Figure 12 shows the General configuration tab for the first LIN slave (i.e., for LINS_1) Component. Note that the 'Multiple instance support' checkbox is checked. Also note that the instance number is set to 1 because it is the first instance of the LIN Component.

Figure 12. LIN Component General Tab Settings

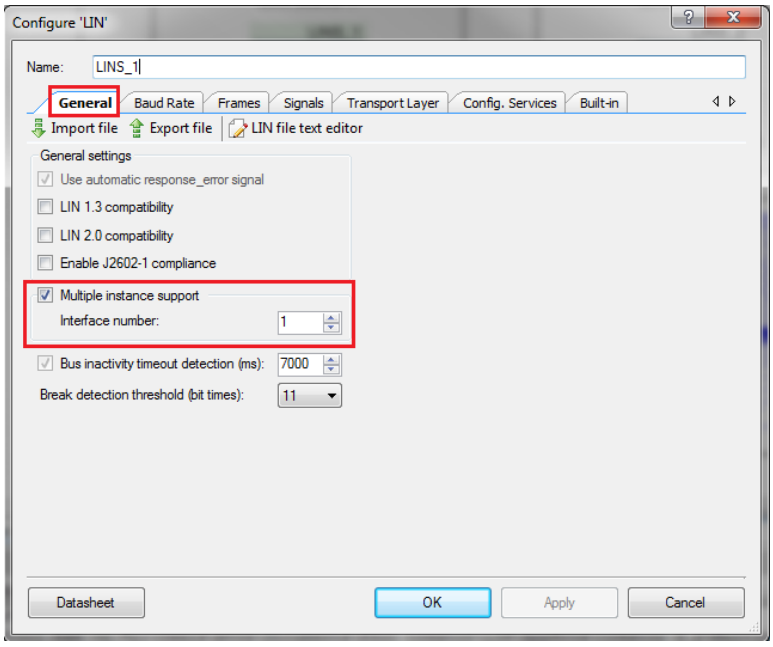


Figure 13 shows the Frames configuration tab for the LIN Component. Use this tab to add a new frame or delete an existing frame from the LIN slave. Click the 'Add' button to add a new frame. For this example, there are two frames named "InFrame1" and "OutFrame1" respectively as shown in Figure 13.

- **InFrame1:** The frame ID value is 0x10, the direction of the frame is 'Subscribe' and the frame type is 'Unconditional'.
- **OutFrame1:** The frame ID value is 0x11, the direction of the frame is 'Publish' and the frame type is 'Unconditional'.

Note: If the direction (publish/subscribe) is set as 'Publish', the slave responds to the master; if the direction is set as 'subscribe', the slave uses the received data for its application.

Figure 13. LIN Component Frames Tab Settings

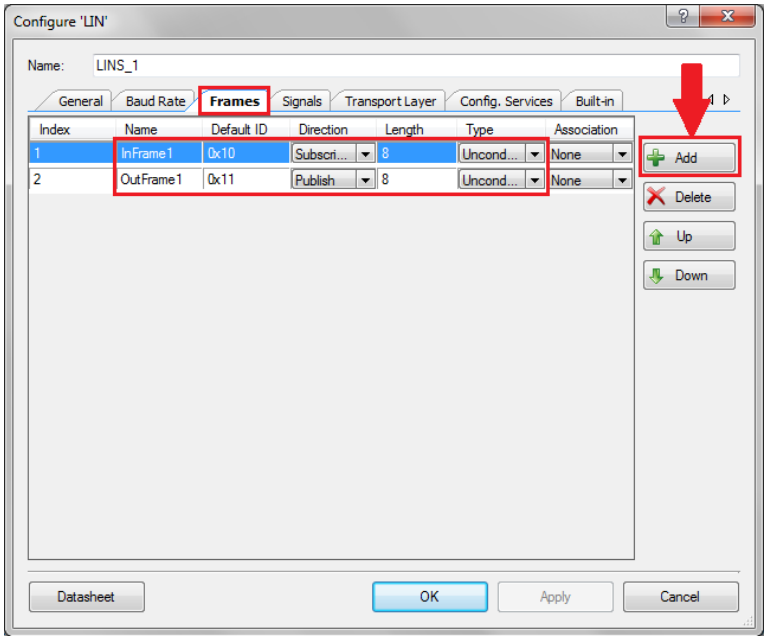
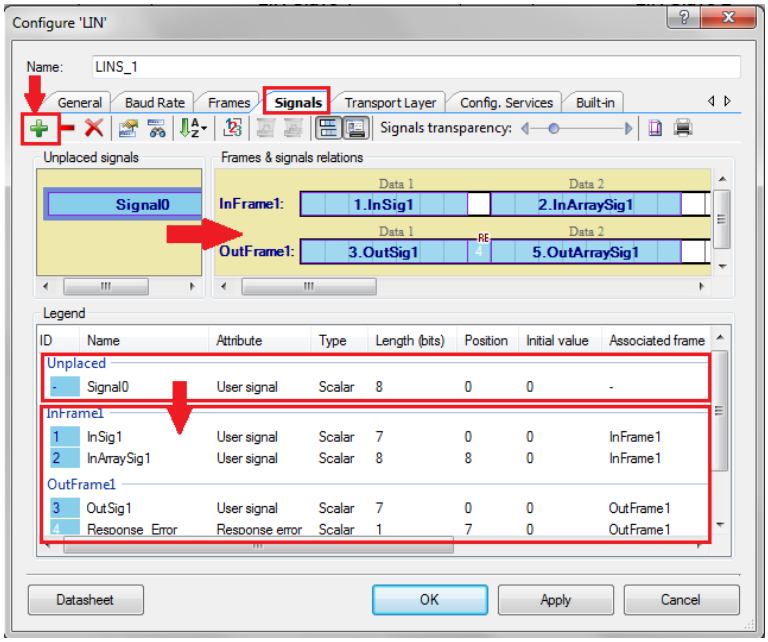


Figure 14 shows the Signals configuration tab for the LIN Component. Click the '+' button to add new signals and select the signal properties such as the signal name, type, length and initial value. After that, place the signals to the corresponding frames as shown the Figure 14.

For LIN v2.x, the Response_Error (1-bit) signal is present by default, which needs to be placed in any of the publish frames.

Figure 14. LIN Component Signals Tab Settings



LIN Component – 2 (LINS_2):

Figure 15 shows the General configuration tab for the second LIN slave (i.e., for LINS_2) Component. Note that the 'LIN 1.3 compatibility' checkbox is selected, which makes the Component compatible with the LIN v1.3 specification. Make sure that the 'Multiple instance support' checkbox is checked, followed by the instance number set as '2' because it is the second instance of the LIN Component.

Figure 15. LIN Component General Tab Settings

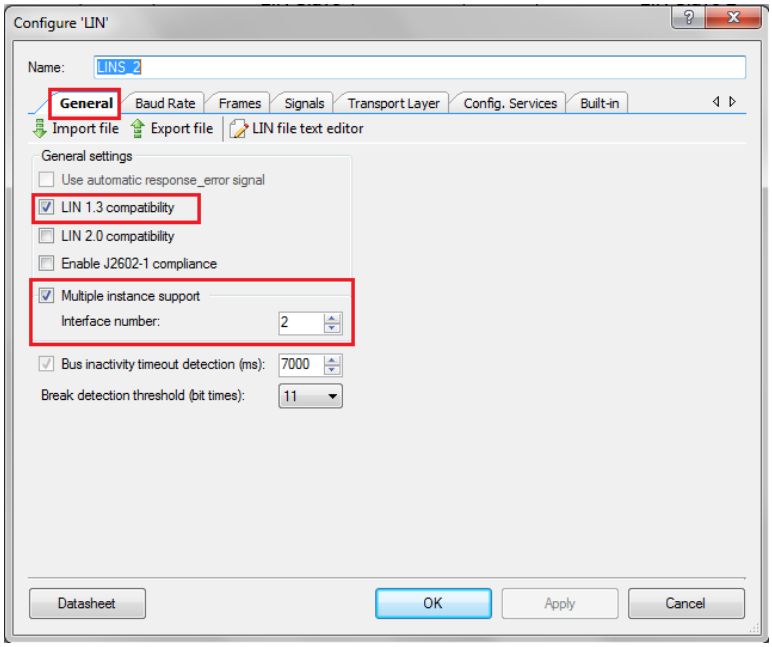


Figure 16 shows the Frames configuration tab for the LIN Component. Use this tab to add a new frame or delete an existing frame from the LIN slave. Click the 'Add' button to add a new frame. For this example, there are two frames named "InFrame2" and "OutFrame2" respectively as shown in Figure 16.

- **InFrame2:** The frame ID value is 0x12, the direction of the frame is 'Subscribe' and the frame type is 'Unconditional'.
- **OutFrame2:** The frame ID value is 0x13, the direction of the frame is 'Publish' and the frame type is 'Unconditional'.

Figure 16. LIN Component Frames Tab Settings

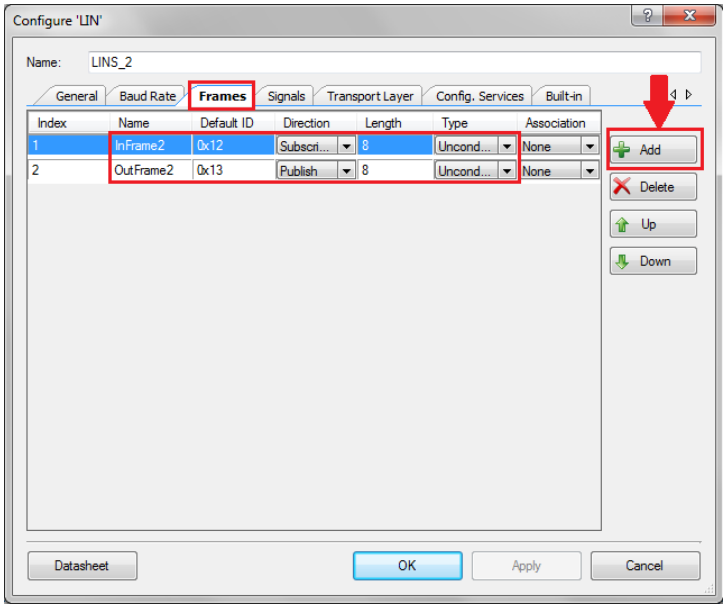
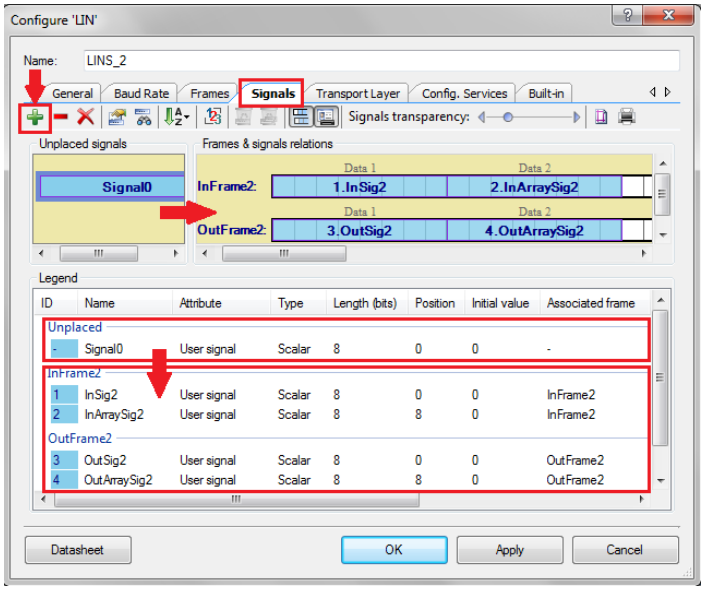


Figure 17 shows the Signals configuration tab for the LIN Component. Click the '+' button to add new signals and select the signal properties such as Signal name, type, length and initial value by double-clicking on the corresponding signal. After that, place the signals to the corresponding frame slot as shown the Figure 17.

Figure 17. LIN Component Signals Tab Settings



All other configuration settings are left at their default values.

Design-Wide Resources

The pin assignments for this code example are shown in Figure 18.

Figure 18. Pin Assignments for PSoC 4200 Example

	Name	Port	Pin	Lock
	\CapSense:Cmod\ (Cmod)	P4[2]	22	☑
	\CapSense:Sns[0]\ (CapSense_Slider_e0__LS)	P1[1]	38	☑
	\CapSense:Sns[1]\ (CapSense_Slider_e1__LS)	P1[2]	39	☑
	\CapSense:Sns[2]\ (CapSense_Slider_e2__LS)	P1[3]	40	☑
	\CapSense:Sns[3]\ (CapSense_Slider_e3__LS)	P1[4]	41	☑
	\CapSense:Sns[4]\ (CapSense_Slider_e4__LS)	P1[5]	42	☑
	\LINS_1:SCB:rx\	P4[0]	20	☑
	\LINS_1:SCB:tx\	P4[1]	21	☑
	\LINS_2:SCB:rx\	P0[4]	28	☑
	\LINS_2:SCB:tx\	P0[5]	29	☑
	L1_NSLP	P0[0]	24	☑
	L2_NSLP	P0[1]	25	☑
	LED_BLUE	P0[3]	27	☑
	LED_GREEN	P0[2]	26	☑
	LED_RED	P1[6]	43	☑

All other design-wide resources are left at their default values.

Operation

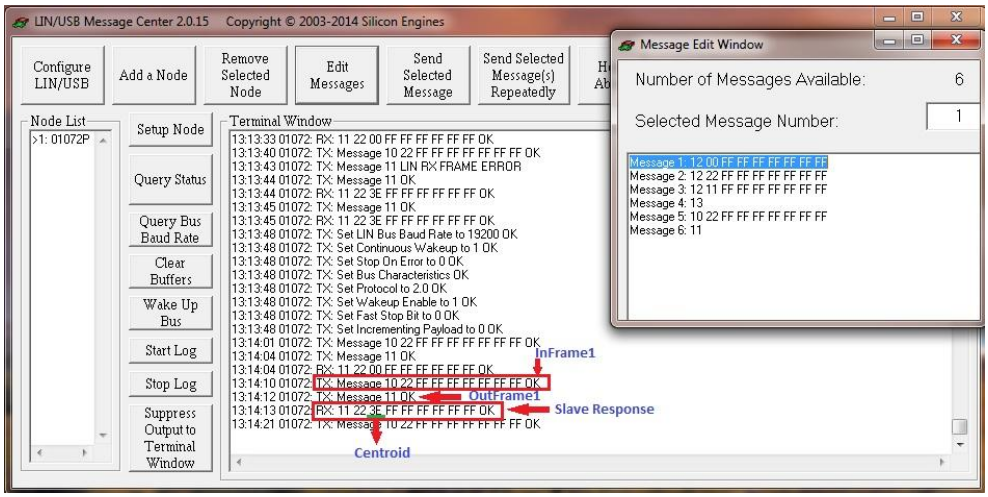
To test the project, a “Silicon Engines LIN-USB Converter – Model 9011” is used as the LIN analyzer; two such analyzers are required for this project. Other LIN analyzers may alternately be used.

Do the following steps to complete the testing:

- Make the necessary hardware connections as shown in Hardware Setup section on page 11
- Program the hex file in to the baseboard (CY8CKIT-042) using PSoC Programmer or PSoC Creator.
- Install the ‘SE9004 and 9011 LINUSB Message Center’ software (if a different analyzer is being used, install the appropriate software) on your PC and connect the two LIN analyzers to different PCs through USB cables.
- Verifying the LIN slave1 output:
  a. Open the LIN analyzer1 software and go to **Configure LIN/USB**, set the **LIN Bus Baud Rate** to 19200 bps, and then select the **LIN 2.0-2.1 Protocol** checkbox as shown in Figure 7.
  b. If you are using any other analyzer, make sure that the checksum setting is selected as ‘enhanced checksum’ because LIN v2.1/2.2 specification supports only enhanced checksum (this option is not required in Silicon Engines LIN-USB converter software).
  c. Add the message/frame in the analyzer software, which needs to be transmitted to the slave and send it through the analyzer shown in Figure 8.
  d. Place your finger on the CapSense linear slider (on CY8CKIT-042) and send a frame with ID = 0x10 and 8 data bytes. The first data byte should be 0x22. The other data bytes can be any value.
  e. If a frame with ID = 0x10 with first data byte = 0x22 is received from LIN analyzer1, the slave stores the current CapSense linear slider centroid position.
  f. If a frame with ID = 0x11 is received from the LIN analyzer1, the slave sends the stored CapSense linear slider centroid position back to the analyzer.

g. The result of transmitted and received data at the LIN analyzer1 is shown in Figure 19.

Figure 19. Results at LIN Analyzer1

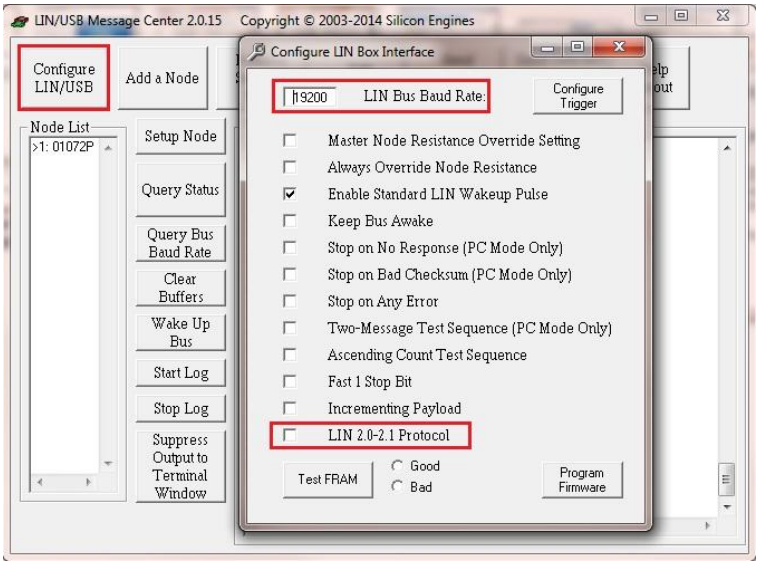


In the above figure, 'InFrame1' refers to “10 22 FF FF FF FF FF FF FF” where 10 is the frame ID and 22, FF, FF, FF, FF, FF, FF, FF are the eight data bytes. Because 'InFrame1' in LIN1 slave is configured with only two bytes named ‘InSig1’ and ‘InArraySig1’, the rest of the data bytes (3 to 8) are ignored by the slave. After receiving this command, the slave stores the current CapSense linear slider centroid position. When 'OutFrame1' is received from the analyzer as shown in the above figure, the slave responds to the frame with the previously stored CapSense linear slider centroid position value along with frame ID as 11 22 3E where '11' is frame ID, '22' is the previous 'InFrame1' data byte (command) and '3E' is the centroid value.

- Verifying the LIN slave2 output:

  a. Open the LIN analyzer2 software and go to **Configure LIN/USB**, set the **LIN Bus Baud Rate** to 19200 bps, and then deselect the **LIN 2.0-2.1 Protocol** checkbox as shown in Figure 20.

Figure 20. LIN Analyzer2 Configuration



  b. If you are using any other analyzer, make sure that the checksum setting is selected as ‘classic checksum’ because the LIN v1.3 specification supports only classic checksum (this option is not required in Silicon Engines LIN-USB converter software).

c. Add the message/frame in the analyzer software which needs to be transmitted to the slave and send it through the analyzer shown in Figure 8.

d. If a frame with ID = 0x12 is received from the LIN analyzer2, the slave controls the RGB LED based on the received data command from master, as provided in Table 9. Note that the frame must contain eight data bytes even though the first data byte is the only one used by the slave. The other seven data bytes may be set to any value.

Table 9. Slave Response as per the Commands from Master

Command	Slave Response
0x11	Turns on Red LED
0x22	Turns on Green LED
0x33	Turns on Blue LED
0x00	Turns off RGB LED

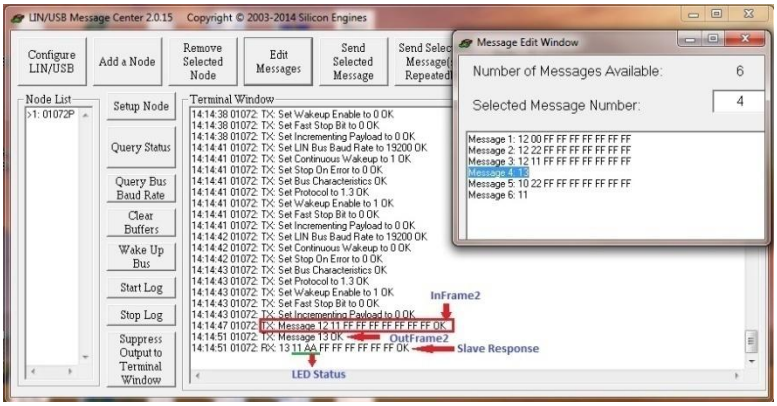
e. If a frame with ID = 0x13 is received from the LIN analyzer2, then the slave will send the RGB LED status back to the master, as provided in Table 10. The message in this case only needs the message ID. No data bytes are required.

Table 10. RGB LED Status

RGB LED status	Data byte
Red LED on	0Xaa
Green LED on	0xBB
Blue LED on	0xCC
RGB LED off	0xDD

f. The result of transmitted and received data at the LIN analyzer2 is shown in Figure 21.

Figure 21. Results at LIN Analyzer2



In the above figure, ‘InFrame2’ refers to “12 11 FF FF FF FF FF FF FF” where 12 is the frame ID and 11, FF, FF, FF, FF, FF, FF, FF are the eight data bytes. Because ‘InFrame2’ in LIN slave2 is configured with only two bytes named ‘InSig2’ and ‘InArraySig2’, the rest of the data bytes (3 to 8) are ignored by the slave. When ‘OutFrame2’ is received from the analyzer as marked in this figure, the slave responds to the frame with the RGB LED status along with frame ID as 13 11 AA where ‘13’ is frame ID, ‘11’ is the previous ‘InFrame2’ data byte (command) and ‘AA’ is the RGB LED status (i.e., Red LED is ON).

Note: If there is an error message such as “BUS STUCK HIGH” or “TX FRAME ERROR”, reset the baseboard (CY8CKIT-042) and LIN analyzer.

Related Documents

Table 11 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component datasheets.

Table 11. Related Documents

PSoC Creator Component Datasheets	
LIN	Implements the industry standard LIN v2.1/v2.2 and LIN v1.3 protocol specifications
CapSense	Controls CapSense CSD block and detects change in capacitance in applications such as touch sense buttons, sliders, touchpad, and proximity detection.
Pins	Controls interface with physical I/O port pins
Device Documentation	
PSoC 4 Datasheets	
PSoC 4 Technical Reference Manuals	
Development Kit (DVK) Documentation	
PSoC 4 Kits	

Document History

Document Title: CE96999 - Basic LIN Slave Implementation in PSoC® 4

Document Number: 001-96999

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4774377	MVRE	05/13/2015	New spec
*A	5168074	MVRE	03/09/2016	Replaced CY8CKIT-017 with CY8CKIT-026 Updated project Operation section Updated the latest Creator version details

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

cypress.com/psoc

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/support

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2015-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you under its copyright rights in the Software, a personal, non-exclusive, nontransferable license (without the right to sublicense) (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units. Cypress also grants you a personal, non-exclusive, nontransferable, license (without the right to sublicense) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely to the minimum extent that is necessary for you to exercise your rights under the copyright license granted in the previous sentence. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and Company shall and hereby does release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. Company shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.