

SPI-LIN Slave Bridge

Author: Valeriy Kyrynyuk
Associated Project: Yes
Associated Part Family: CY8C27443
PSoC Designer Version: 4.2
Associated Application Notes: No

Abstract

This Application Note demonstrates an SPI-LIN slave bridge using a PSoC™ device. Demonstration projects are included.

Introduction

Nowadays LIN bus is widely used in vehicle applications. LIN bus is implemented as a low-cost alternative to CAN bus for low-speed device control. But microcontrollers, which are used for such devices, very often do not have the communication means or spare resources for LIN protocol implementation (i.e., timers, UARTs, or program or RAM space).

To solve this problem, the SPI-LIN bridge for LIN slave devices was designed. The bridge allows simple integration of vehicle devices to LIN bus, and renders communication with LIN master and other slaves according to LIN Bus 2.0 Specifications (<http://www.lin-subbus.org/>). In this design, the bridge was created using a standard SPI interface, but I2C or UART interfaces can easily be implemented.

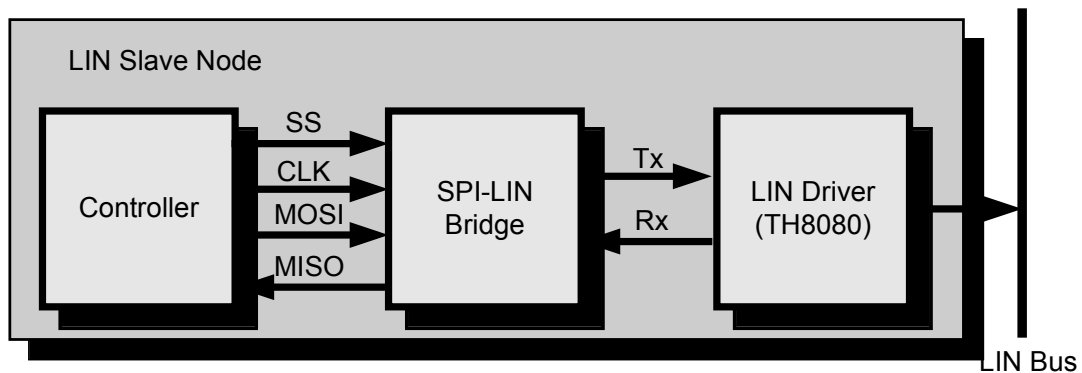


Figure 1. SPI-LIN Bridge Integration into LIN Slave Device

The main purpose of this Application Note is to simplify the process of integrating devices into LIN bus. Users need not have deep knowledge of LIN communication or how to integrate attached devices.

Description of the LIN-SPI Bridge

The bridge uses the PSoC CY8C27443 device and only five digital blocks. It uses dynamic reconfiguration with three configurations – base, synchronization receipt, and LIN frame receive/transmit.

The LIN bus program code works in the background through interrupt handler functions. The SPI communication control is realized in the main program. The functional diagram of the bridge is shown in Figure 2.

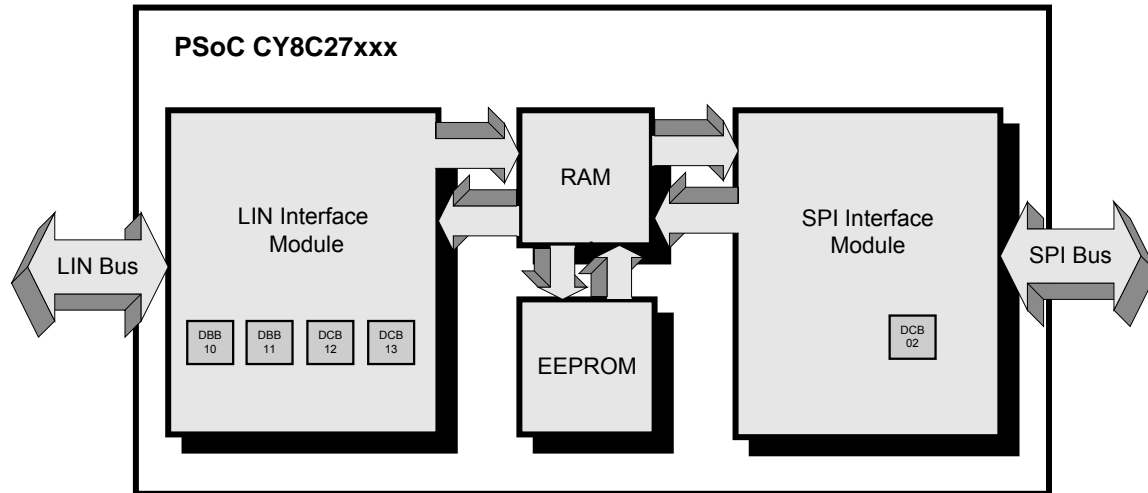


Figure 2. Functional Diagram of the LIN-SPI Bridge

The bridge translates messages from the LIN bus and exchanges message data with the RAM buffer. The master controller of the LIN slave node reads and writes data from this buffer. It also can carry out the configurations in the Message Configuration Record table (Table 1). These configurations are written into the internal EEPROM of the bridge. The message PIDs (Protected IDs) are also saved to EEPROM. This action is initiated by the LIN interface module. The detailed descriptions of configuration parameters, data exchange, and order of configurations are ahead.

A description of each pinout is as follows:

- MISO – SPI Data Output
- MOSI – SPI Data Input
- CLK – SPI Clock Input
- SS – SPI Chip Select Input
- TX – LIN Data Output
- RX – LIN Data Input
- A0..A3 – NAD Select Pins

All other pins are unused and can be used for other tasks.

Node Configuration

The LIN node (the device that supports the bus interface) can enter into the LIN bus when the configuration procedure is complete. For this bridge, the main controller of the device must create the message table and configure each message. These messages will be used later for data exchange between the other LIN bus nodes. This bridge allows the user to implement up to 16 messages with total data size of up to 100 bytes. Implementation of messages is as simple as writing proper data into the message table. The data includes the following parameters: Message ID, which is unique per LIN cluster (for all nodes in the LIN bus), direction of message data stream, location of message data, status bytes into bridge RAM buffer, and length of data (see Table 1).

Figure 3 shows the LIN-SPI bridge pinouts.

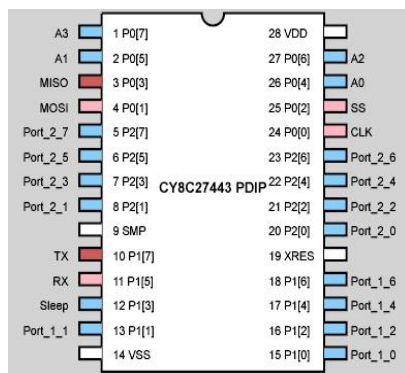


Figure 3. Bridge Pinouts

Table 1. Message Configuration Record

Parameter	Size in bytes	Description
Message ID	2	Must be unique in LIN cluster
PID (Protected Identifier)	1	Must be set to undefined state (0xFF) by SPI master. This parameter will be defined later by LIN master.
Type of Transaction	1	01 – from (LIN) master to slave 02 – from (LIN) slave to master 04 – event triggered
Location of Message Buffer	1	Pointer to message buffer (data+status) into 100-byte RAM buffer
Length Of Message Data in Bytes	1	LIN messages can be 1, 2, 4 and 8 bytes in length

Regarding RAM buffer allotment, it is necessary to reserve one byte more than the length of the message. This extra byte is used for status. The status byte is the first byte into the buffer. The bytes following are data messages. The status byte is needed to indicate any data errors (Table 2).

Table 2. Status Byte Bit-Mask Descriptions

Status Bit-Mask	Description
0x01	Transaction was successful
0x02	Transaction bit error
0x04	Timeout error during receiving
0x08	Checksum error
0x10	New data is available
0x20	Response error bit

The Message Table (Table 3) can consist of up to 16 message records. Used records must be sequential, one by one. Unused records (meaning message is absent) must have Message ID = 0xFFFF and be placed at the end of the table (after the used records). The message table for the slave node is usually written to once and can be done during assembly.

Table 3. Example of Message Table

Record Number	Message ID	PID	Type of Message	Buffer Location	Length	Description
0	0x1001	0xFF	0x01	0x00	4	Message transfers 4 bytes from master to slave
1	0x1002	0xFF	0x02	0x05	2	Message transfers 2 bytes from slave to master
2	0x1003	0xFF	0x06	0x08	8	Event triggered message transfers 8 bytes from slave to master
3	0x1004	0xFF	0x01	0x11	8	Message transfers 8 bytes from master to slave
4	0xFFFF	-	-	-	-	No record
..	0xFFFF	-	-	-	-	..
..	0xFFFF	-	-	-	-	..
15	0xFFFF	-	-	-	-	No record

As the message table is configured, the bridge can enter into the LIN bus as a slave node. Upon entry, the LIN master finds the newly connected slave node and assigns a unique PID for each slave node message. The PID value is saved in the bridge EEPROM into the message table. Later, the PID is used in the LIN bus cluster as a unique identification of the frame. Frame ID values are sources of data such as speedometer point position, the quantity of fuel, or distance to obstacle, etc.

We did not closely examine LIN bus and all aspects of functionality. Those details are considered in the LIN Bus 2.0 Specifications. We did examine the NAD (Node Address for Diagnostic) determination procedure. The NAD is the unique 7-bit address of the LIN slave device in the LIN cluster. It is used only during diagnostic configuration, when the slave node enters into the LIN bus. In this application, the NAD is determined by the hardware – the special 4 addressing pins A0-A1 are used. Each pin has 2 address bits of NAD (Table 4).

Table 4. NAD Pin Description

Pin	Address Bits	Address HEX Mask
A0	0,1	03
A1	2,3	0C
A2	4,5	30
A3	6	40

Every address pin can have 4 states (Table 5).

Table 5. Pin Addressing Description

Pin Condition	Address Bits
Strong GND (Pull-down through <1 kΩ)	00
Weak GND (Pull-down through >50 kΩ)	01
Weak Vcc (Pull-up through >50 kΩ)	10
Strong Vcc (Pull-up through <1 kΩ)	11

Table 6. SPI Interface Commands

Command 0x40..0x4F		Write Status and Data to Message Buffer						
Input Data							Output Data	
Message record number +0x40	Length of write data	Status	Data0	Data1	...	Data last	Message record number +0x40	Length of write data
0x40 ... 0x4F	0..9	xx	xx	xx	xx	xx	xx	xx

Command 00..0x0F		Read Status and Data to Message Buffer						
Input Data		Output Data						
Message record number	Length of read data	Message record number	Length of read data	Status	Data0	Data1	Data last
0x00 ... 0x0F	0..9	xx	xx	xx	xx	xx	xx	xx

When the NAD is read, the bridge turns the address port pins to High-Z drive mode to reduce power consumption.

SPI Data Transfer

Now we have reached the aim of this bridge's task – data exchange into the LIN bus cluster. There are three directions of data frame transmission: from master to slave, from slave to master, and from slave to additional slave. The LIN master node initiates transmission in all cases. The data of each message is held in its own RAM buffer. The first byte into the buffer is always the status byte. The following byte is the data message. The master controller can read and write from this buffer using the SPI bus and simple data protocol.

In addition to the data bytes, 2 extra bytes are provided – command set and length of data. The length of data cannot exceed 9 bytes or be longer than the message buffer size. The command set can be divided into two parts: data reading and data writing.

The descriptions of the SPI interface are shown in the tables below. There are read/write data commands and message configurations.

Command 10..0x1F		Read Configuration Record from Message Table							
Input Data		Output Data							
Message record number +0x10	Length of read data	Message record number +0x10	Length of read data	ID, high byte	ID, low byte	PID	Type of transaction	RAM buffer location	xData length
0x10 ... 0x1F	6	xx	6	xx	xx	xx	xx	xx	xx

Command 0x50..0x5F		Write Configuration Record from Message Table							
Input Data								Output Data	
Message record number +0x50	Length of write data	ID, high byte	ID, low byte	PID	Type of transaction	RAM buffer location	Data length	Message record number +0x50	Length of write data
0x50 ... 0x5F	6	xx	xx	xx	xx	xx	xx	xx	xx

Command 0x20..0x3F Read Other LIN Bridge Parameters. Reserved for Future Implementation.

Command 0x60..0x7F Write Other LIN Bridge Parameters. Reserved for Future Implementation.

When the bridge receives a command it always returns the command and length, in bytes, as confirmation. So, the SPI master (main controller) must always wait for confirmation from the bridge. When the SPI master waits, the bridge always sends bytes with a value of 0xFF (Figure 4).

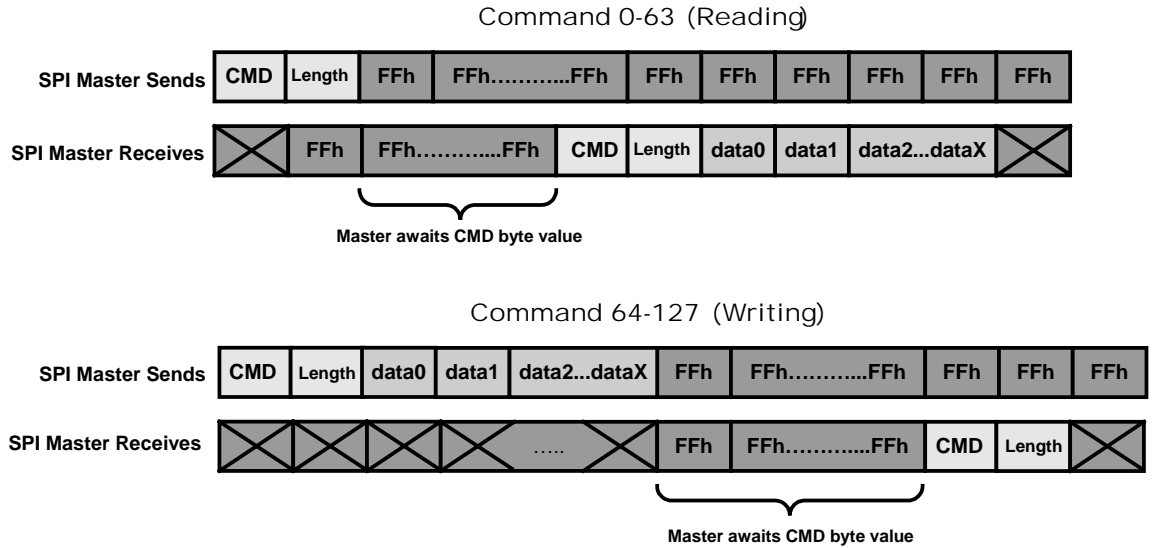


Figure 4. SPI Communication Diagram

When a command of write message configuration is performed, the reading of confirmation bytes must start after 10 ms. This is the EEPROM write time. To initialize SPI bridge interface, the main controller must be reset or SPI communication started. The SPI master must send 16 bytes with value of 0xFF.

Description of Projects

The main purpose of the projects is to demonstrate bridge implementation into customer applications as well as LIN-SPI bridge operation. The demonstration projects include (Figure 5):

- LIN master project, which uses PSoC on the evaluation board.
- SPI-LIN bridge project on the evaluation board.
- The main controller program, which is done on the PC.

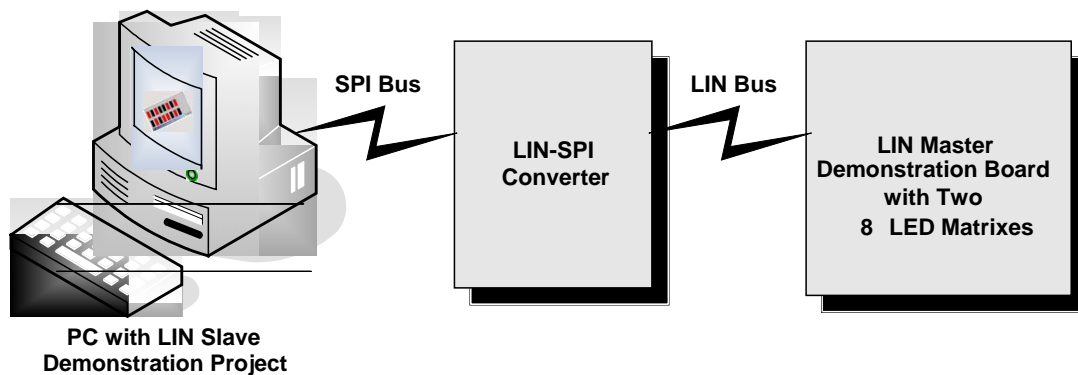


Figure 5. Structure Diagram of Bridge Use in Projects

In bridge operation, the LIN master reads 8 bytes from the slave node, then inverts and transmits them to the LIN slave. So, two messages are used: read data and write data. The master has two rows of LEDs to indicate direct and inverted data: top and bottom. Each row has 8 LEDs. The bottom LEDs indicate data, which the LIN slave transmits, the top LEDs indicate data, which the LIN master transmits.

The same two rows of LEDs are depicted in the PC program of LIN slave device emulation. With correct project operation, the LEDs of the both rows will always be lit, inverted from each other.

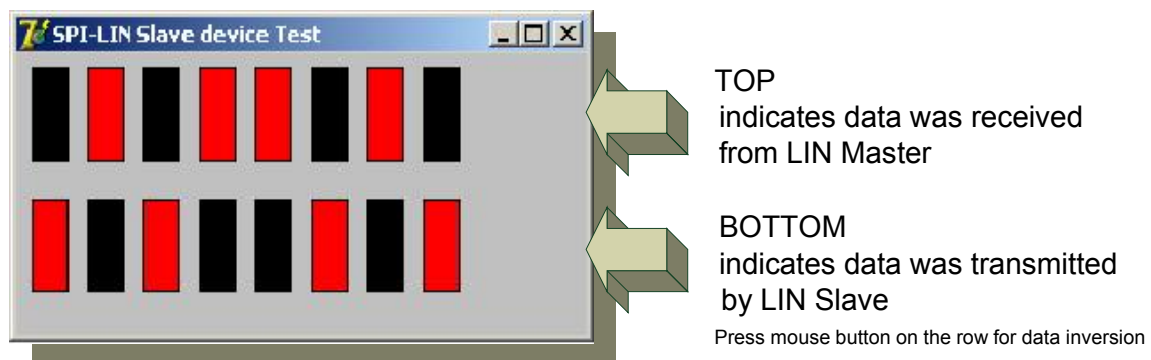


Figure 6. Program Interface for Slave Node Control

The data exchanged between slave and master represents bytes with values of either 0x00 or 0xFF. At 0x00 in the code, the appropriate LED lights will be off, at other code values, they will be lit. The PC program allows users to change the data that the LIN slave transmits to the master. This is done by clicking the mouse button in the LED row of the LIN slave program. The LEDs then invert their state. The same can be done with both LIN master LED arrays.

After the data package exchange, the appropriate LEDs in the PC program invert. The LIN slave controller program performs in a standard Windows environment. First, the program configures the LIN-SPI bridge message table. Next, the LIN master turns on the slave node of the LIN network and the data exchange between LIN master and slave node begins. The data exchange between the bridge and control program via the SPI interface begins as well. The SPI bus support on the PC is via the LPT port (Figure 7). Note that the program uses LPT1 and ECC or EPP parallel port mode.

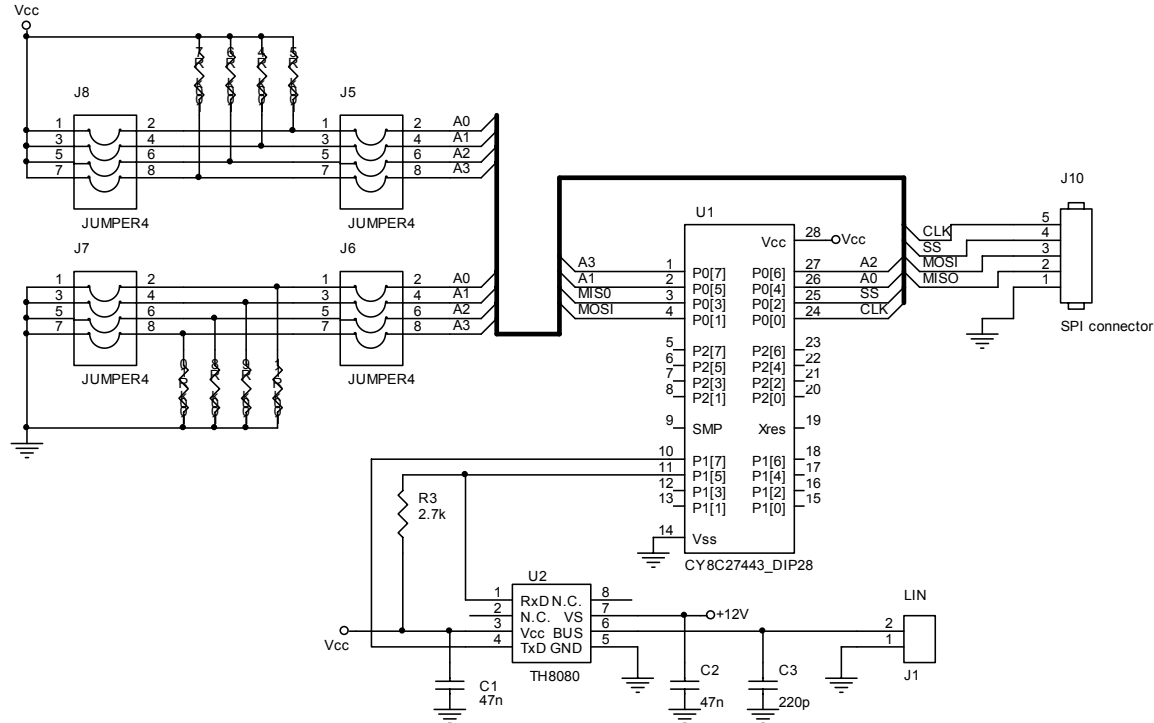


Figure 7. Schematic of Bridge Demonstration Board with LPT Connector

The bridge demonstration board includes the bridge itself (U1), the bridge integration circuit in the LIN Bus (TH8080 (U2)) microcircuit, the NAD (J5-J8) circuit, and the bridge connection SPI-LPT interface (see the schematic in Figure 8).

This schematic is needed for bridge-to-PC connection. It includes the digital buffer MC74VHC541 (U19) for robust CLK and SS signals. The bridge schematic and conditioning scheme were placed on the same evaluation board to minimize SPI line length.

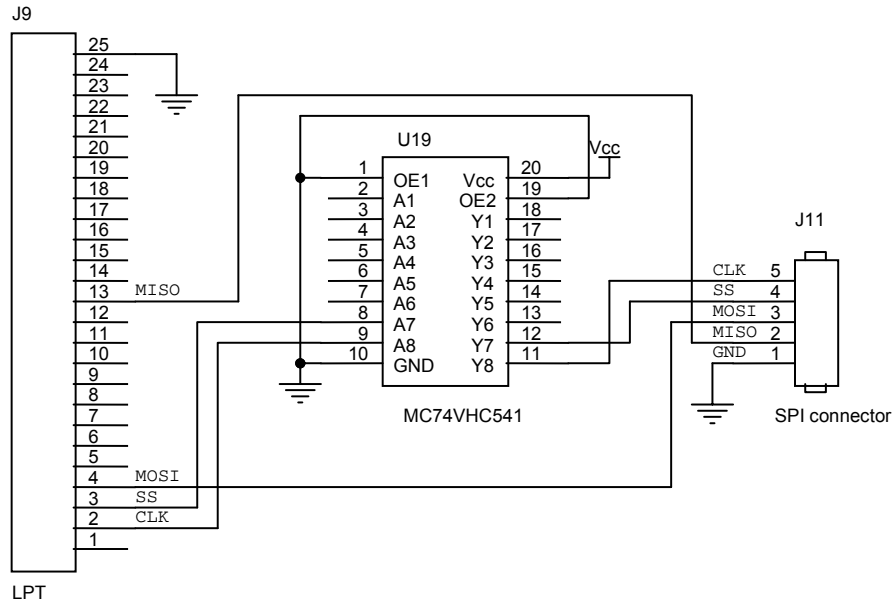


Figure 8. SPI-LPT Conditioning Scheme

The LIN master circuit demonstration board is shown in Figure 9. The LIN master project uses PSoC CY8C27443 (U5). The board also includes 2 arrays of LEDs (8 LEDs in each array). The odd outlet LEDs (left) correspond to the bottom array, and the even outlet LEDs (right) correspond to the top array. Integration of the LIN device into the network circuit uses a TH8082 (U4) (physical layer).

It is recommended that the crystal be included on the LIN master device's circuit. This is done so the LIN master exactly follows the declared speed on the LIN bus (the declared speed in the project is set at 19200). The LIN master project is configured in slave node, search from NAD=0x1B. Users can set this value, which is located on the bridge.

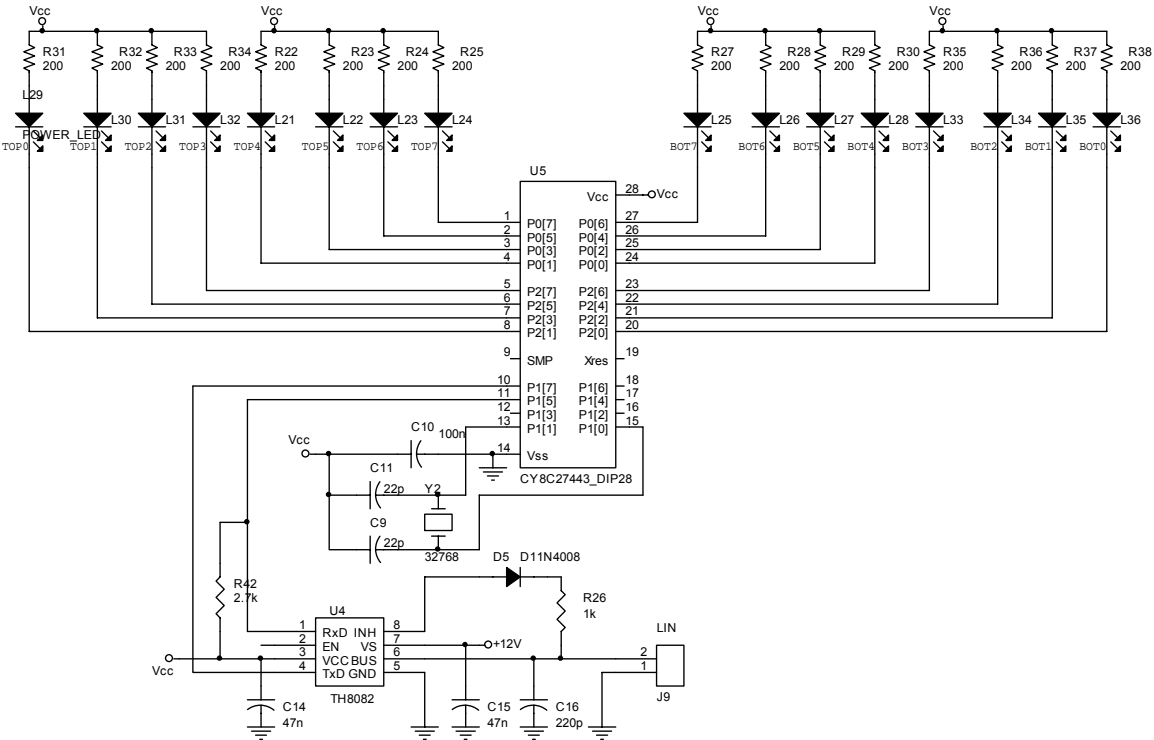


Figure 9. LIN Master Demonstration Board Schematic

Conclusion

The given SPI-LIN bridge project is flexible and can be easily modified to meet user needs. It can be implemented with another interface. For instance, a LIN-I2C bridge can be implemented using a less expensive CY8C21xxx device (the bridge does not use the analog blocks). The bridge can also be implemented on a CY8C24x94 device with USB hardware capability. In the current project, there are further development options including:

- Introduce the ADC/DAC, which is available for use in the LIN slave device circuit. Add the commands (to make ADC/DAC visible to LIN and SPI interfaces at once).
- Add/replace UART, I2C interface.
- Implement EEPROM on the base of unused 8 kbytes (in another words, read/write data into unused PSoC storage area).
- Clone the given project into the PSoC 8-pin SOIC package. This is done by reassigning the pins for LIN and SPI interface and setting the NAD address programmatically via the SPI interface;
- Add support for Sleep Mode and a bridge output signal or command for accessing Sleep Mode.

About the Author

Name: Valeriy Kyrynyuk

Title: Engineer

Background: Seven years experience with fieldbus and communication device design.

Contact: lopik@lviv.farlep.net

Cypress Semiconductor
2700 162nd Street SW, Building D
Lynnwood, WA 98087
Phone: 800.669.0557
Fax: 425.787.4641

<http://www.cypress.com/>

Copyright © 2005 Cypress Semiconductor Corporation. All rights reserved.

"Programmable System-on-Chip," PSoC, PSoC Designer and PSoC Express are trademarks of Cypress Semiconductor Corp.

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

The information contained herein is subject to change without notice. Made in the U.S.A.