Application Report

Enabling SPI Communication using TI's Smart High Side Switches

Elizabeth Higgins, Timothy Logan

ABSTRACT

The SPI interface provides synchronous data transfer used for short distance high-speed communication. Frame by frame, the SPI interface exchanges data between master devices and slave devices. In industrial applications, a common example of a SPI master would be a microcontroller where an example of a SPI slave might be a robotic arm or mechanical relay. As SPI is full duplex and can communicate simultaneously in both directions the production speed can be increased allowing for more autonomous communication. In this application note, we discuss the advantages of this configuration and how this communication method can be integrated into a design using TI's Smart High Side Switches. A reference hardware design and various software snippets are provided in this application note to demonstrate a simple and practical example of how to enable the SPI interface on a TI high side switch with minimal external components. This application note is provided with a supplementary support package which includes the gerber files of the referenced SPI expander implementation as well as an example software implementation for the CC2652R microcontroller.

The software package contains all of the collateral listed in this application note.

Table of Contents

List of Figures

Trademarks

All trademarks are the property of their respective owners.

1 Reference Design with TPS272C45

By combining SPI Communication with the existing protection features of TI's Smart High Side Switches, a designer can have the confidence that their system will be protected while quickly controlling a device's power rail with the fast data rate associated with SPI. For more information on High side switch protection features such as adjustable current limit and high accuracy current sense, refer to the following app notes on the subjects:

- High Accuracy Current Sense of Smart High Side Switches – SLVAE08
- Adjustable Current Limit of Smart Power Switches – SLVA859

SPI integration is a key care-about in industrial applications. With the emergence of increasingly complex industrial designs and the digitization of manufacturing, industrial applications are becoming more efficient with less waste. To optimize a system through this growth of technology, it is essential for industries to identify where they can improve their processes and leverage new devices and features that support integration of different devices, predictive maintenance, and fast speed.

SPI allows a user to communicate directly with the processor on the back end side of a system thus limiting the need for numerous separate I/O lines. In the case of the TPS272C45, a minimum of three I/O lines are required: EN1, EN2, and FLT. In common industrial applications such as digital I/O modules it is a common requirement to drive up to sixteen individual loads. With each TPS272C45 dual channel device responsible for two loads this would translate to eight TPS272C45 devices and twenty four I/O lines on the microcontroller. Instead of requiring twenty four individually controlled and isolated I/O lines a SPI interface can be used to simplify the design. The system block diagram below in Figure 1-1 uses a single SPI command to trigger sixteen inputs while producing an output signal representing the eight FLTx lines.

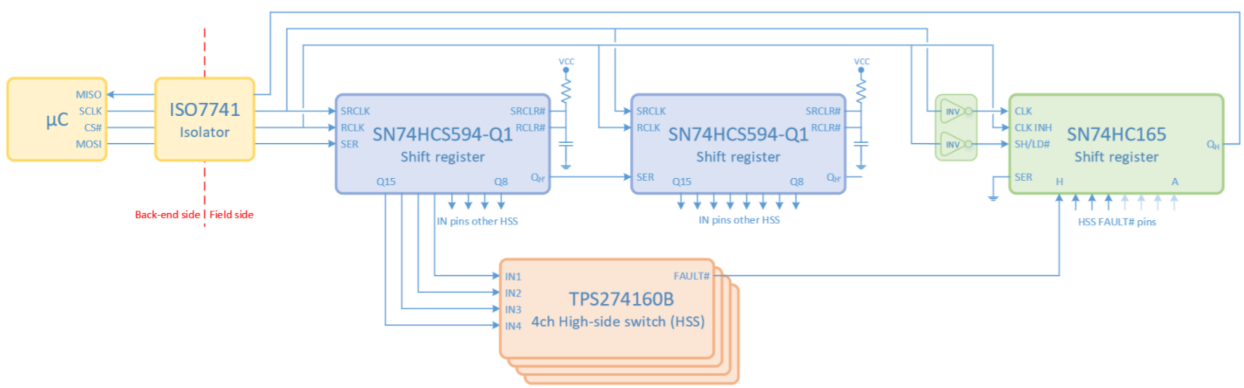

Figure 1-1. Block Diagram for SPI Expander

In this system the microcontroller acts as SPI master and writes a 16-bit word into the two SN74HC594-Q1's shift registers via its SPI peripheral. This in turn updates the sixteen ENx signals of the connected TPS272C45s simultaneously with the rising CS# edge of the SPI signal. Additionally the SPI master microcontroller can read the FLTx data byte of the SN74HC165 that correlates to the eight individual fault lines of the attached TPS272C45s. A schematic of the overall "SPI expander" can be seen below in Figure 1-2.

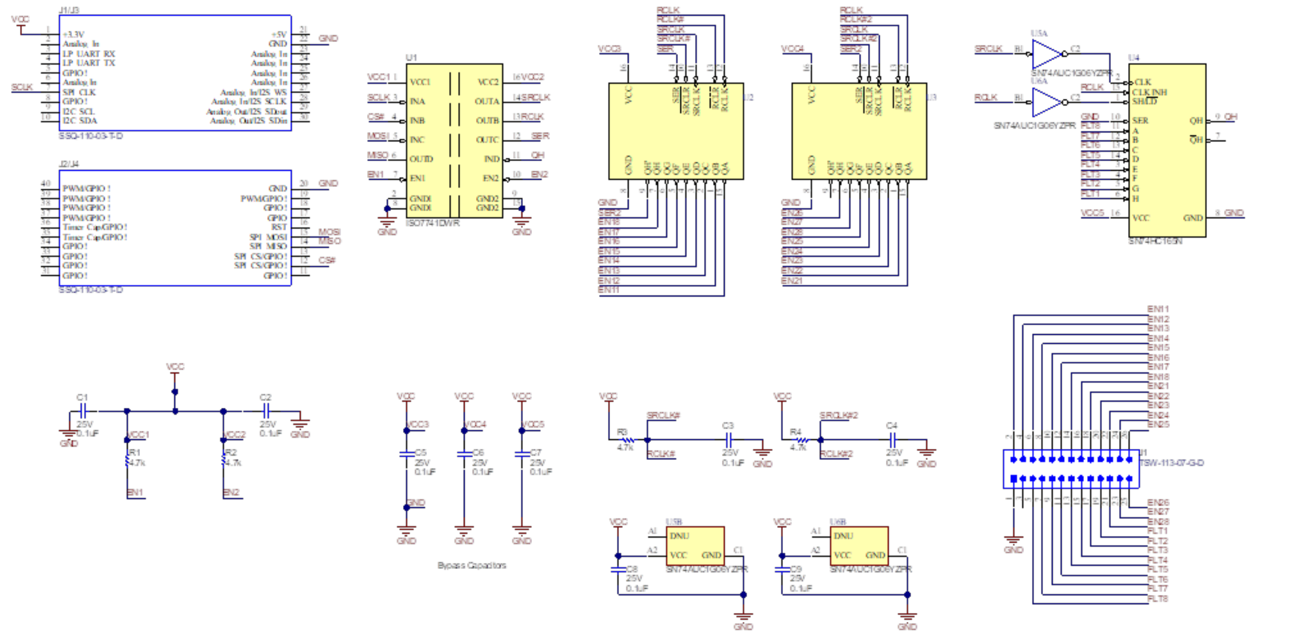

Figure 1-2. SPI Expander Components

The ISO7741 isolator provides high performance quad-channel digital isolation preventing noisy and potentially harmful currents on the device data lines. This device has the advantage of small die space compared to larger optocouplers which require multiple external components. The ISO7741 only requires two external bypass capacitors for a complete design.

The SN74HCS594 8-bit serial-in parallel-out shift register feeds an 8-bit D-type storage register. This register is communicated with the MOSI line of the microcontroller to individually enable and disable the attached EN lines of the TPS272C45. Equipped with Schmitt-Triggered inputs this shift register will eliminate erroneous data outputs due to noisy input signals, and reduces the overall current drawn by the system. The storage and shift register clocks (RCLK and SRCLK) are tied together to cause the shift register to be one count pulse ahead of the storage register. The SN74HC165: 8-bit parallel-in serial-out shift register takes the FLT signals from the connected high-side switches and outputs a serial QH digital signal back to the microcontroller. When SH/LD! Is low, the device loads all the 8 bits simultaneously through the parallel load input and shifts the data when CLK toggles on the rising edge.

It is important to note that while this reference implementation uses the TPS272C45 any Texas Instruments high side switch could use the same design to enable SPI interaction. The ENx and FLTx signals that get relayed in between the shift registers and high side switches are simple digital signals and can be used on devices such as the TPS1H100-Q1, TPS27S100, or any other high side switch from Texas Instruments. It is also important to note that the design described in this application note is a starter reference solution. The design is by no means meant to be comprehensive and many aspect of interfacing with the high side switch are omitted. While sensing load current through the analog sense pin is not supported in this implementation it can easily be added in design via a simple analog power multiplexer.

2 Connecting the SPI Expander to the TPS272C45EVM

The reference implementation provided with this application note contains two different hardware components: a SPI expander board and an attachable daughter card. The expander board is designed to be connected directly to the SPI master (normally a microcontroller) and contains all of the necessary shift registers and isolators required for SPI communication. Additionally the SPI expander board is populated with BoosterPack headers allowing for easy interfacing with a Texas Instruments LaunchPad based microcontroller. A picture of the SPI expander board can be seen below in Figure 2-1 and a picture of the SPI expander connected to a LAUNCHXL-CC26X2R1 can be seen in Figure 2-2.

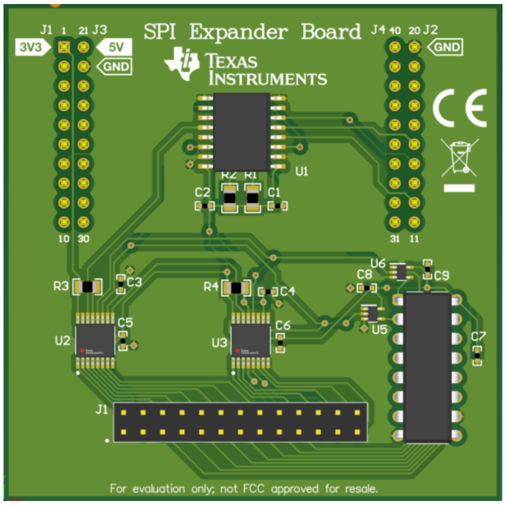

Figure 2-1. SPI Expander Board

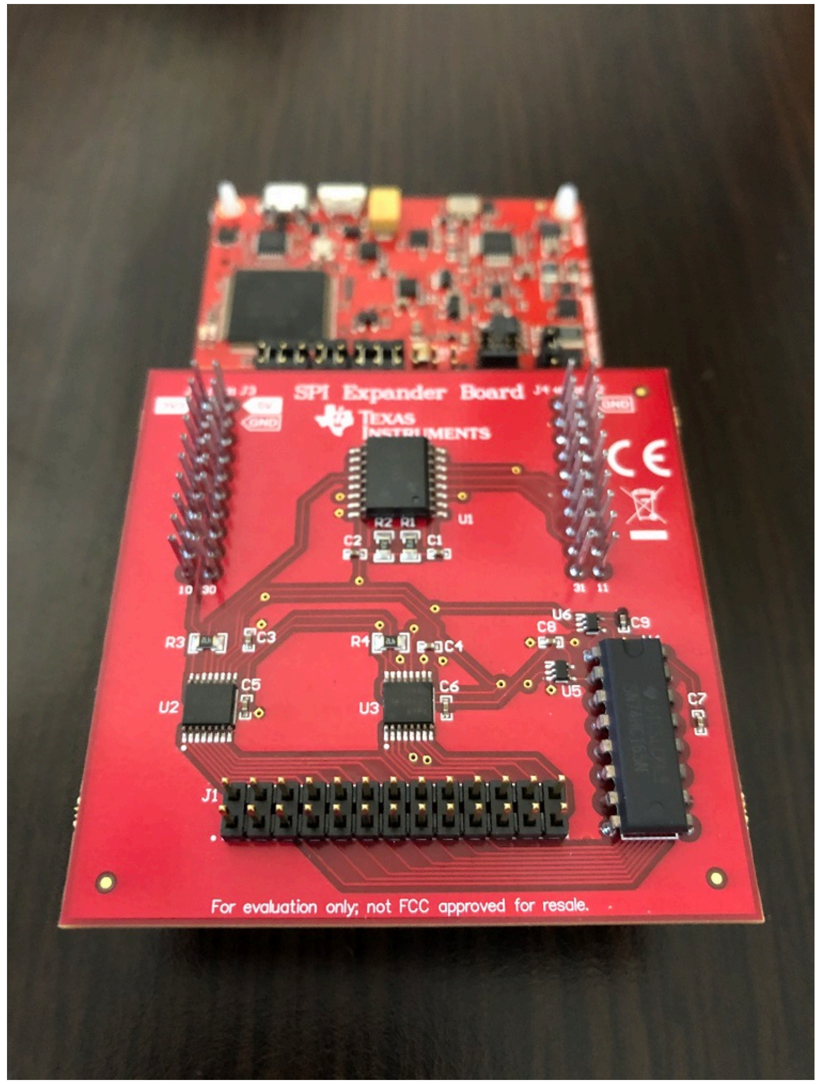

Figure 2-2. SPI Expander and LAUNCHXL-CC26X2R1

The daughter card implementation attached is a breakout board designed to be connected to the SPI expander board via standard 2x13 ribbon cable. The breakout board has four separate BoosterPack headers used to connect four separate TPS272C45EVMs. A picture of the completed SPI expander + daughter card + 4 x TPS272C45EVM solutions can be seen below in Figure 2-3.

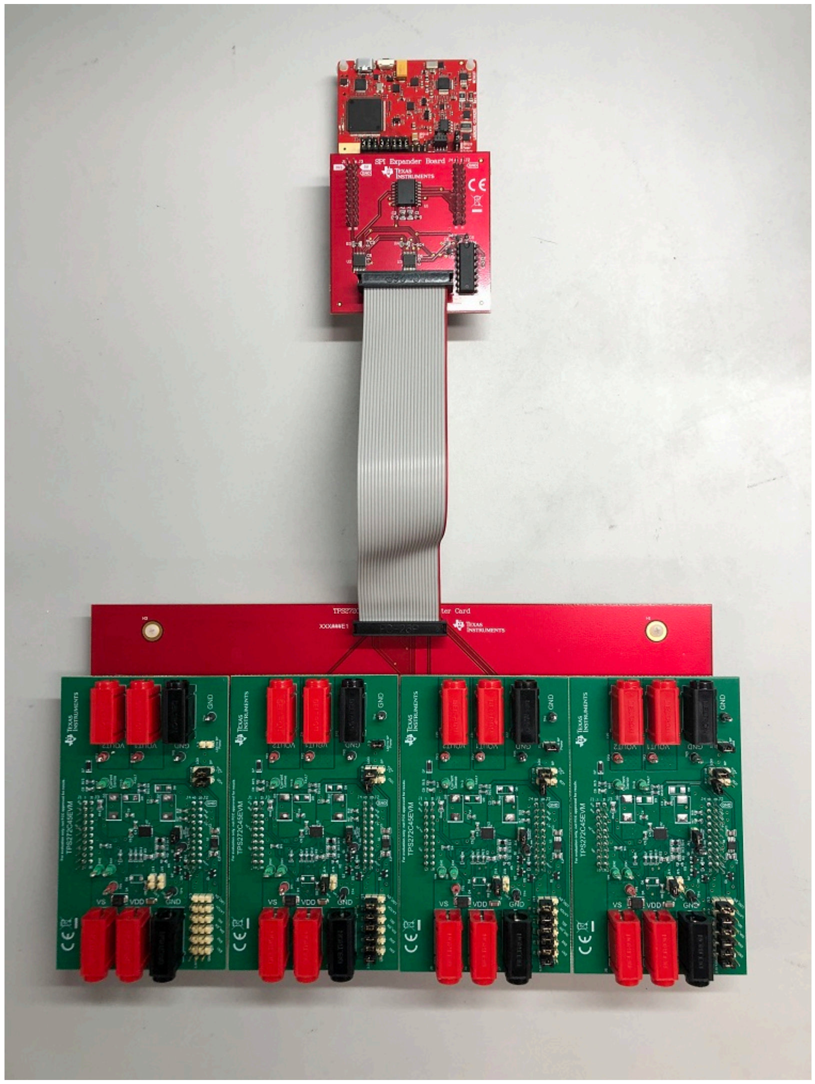

Figure 2-3. Connected SPI Expander

Note that the function of the 2x13 ribbon cable is to relay the FLTx and ENx signals from the TPS272C45EVMs on the daughter card to the SPI expander board. Take care to make sure that the relevant PIN1 of the ribbon cable on the SPI expander matches that of the PIN1 header on the daughter card as shown in the picture above. The pinout for the ribbon cable header on the main SPI expander board can be seen below in Figure 2-4.

4	3	2	1	8	7	6	5	C	B	A	9	E
GND	VCC 3.3V	FLT8	FLT7	FLT6	FLT5	FLT4	FLT3	FLT2	FLT1	G	F	D

If you write 1234 5678 9ABC DEFG
You've receive this following signals in this order above
If you read this outputs, you'll receive FLT12345678

Figure 2-4. Pinout for SPI Expander Board

3 Using the SPI Expander with the TPS272C45EVM

This application note comes with a referenced ZIP file that contains the hardware gerber files and BOM as well as an example software implementation of the SPI expander for the SimpleLink CC2652 wireless microcontroller.. The gerber files and BOM are located in the gerbers directory of the attached ZIP file and are separated into two different folders for the SPI expander and daughter card attachment. The software folder contains a sample software implementation that can control the SPI expander board via the CC2652R's SPI peripheral with a serial port terminal connection.

The sample application provided is a simple application that lets the user control the SPI expander board from the onboard serial port attached to the LAUNCHXL-CC26X2R1 LaunchPad. It is recommended to use a serial port application such as PuTTY or TeraTerm for serial port communication. The serial port settings can be seen below in Figure 3-1.

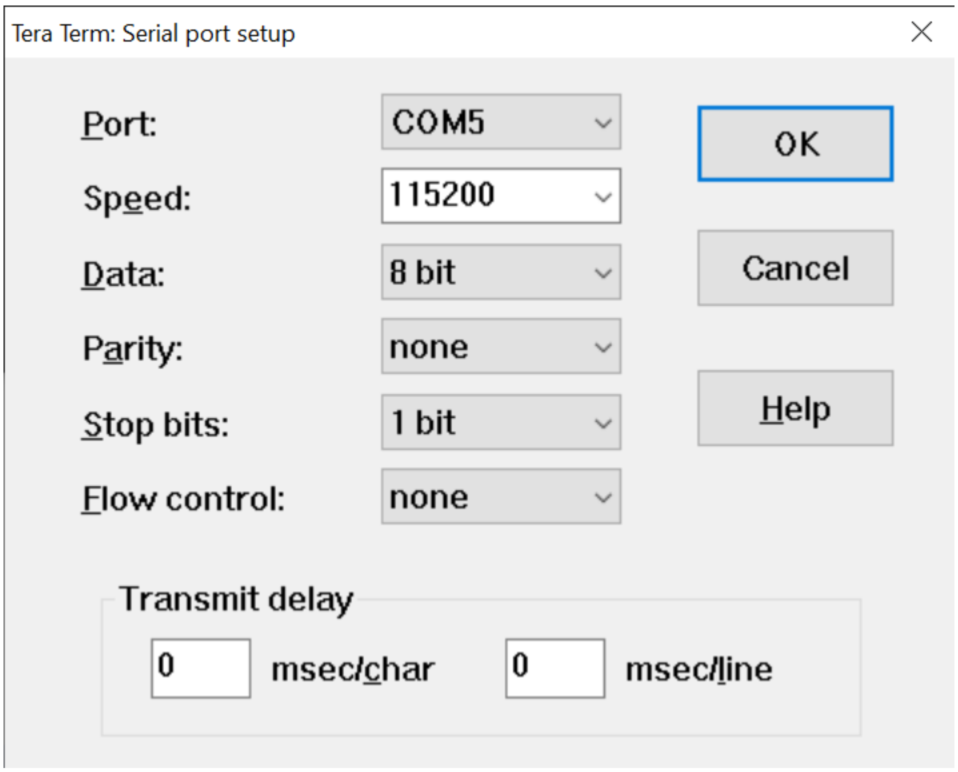

Figure 3-1. Serial Port Settings

When the ***spi_expander_test.out*** binary is flashed onto the LaunchPad using Uniflash, open the serial port program of your choice. Make sure to open the appropriate serial port terminal marked as the Application/User serial port in your serial terminal program such as shown in Figure 3-2.

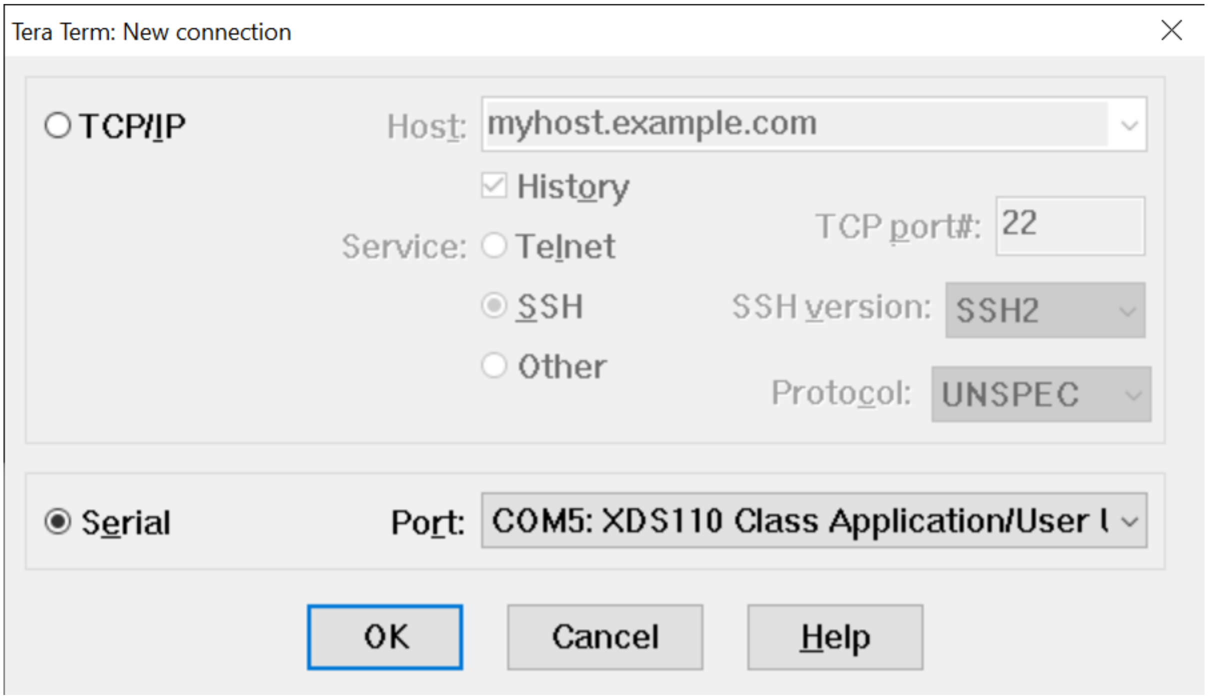

Figure 3-2. User/Application UART

Once connected to your serial port and the program has been flashed onto the CC2652R press the RESET button on top of the LaunchPad. The following menu print on the serial terminal appears as shown in Figure 3-3.

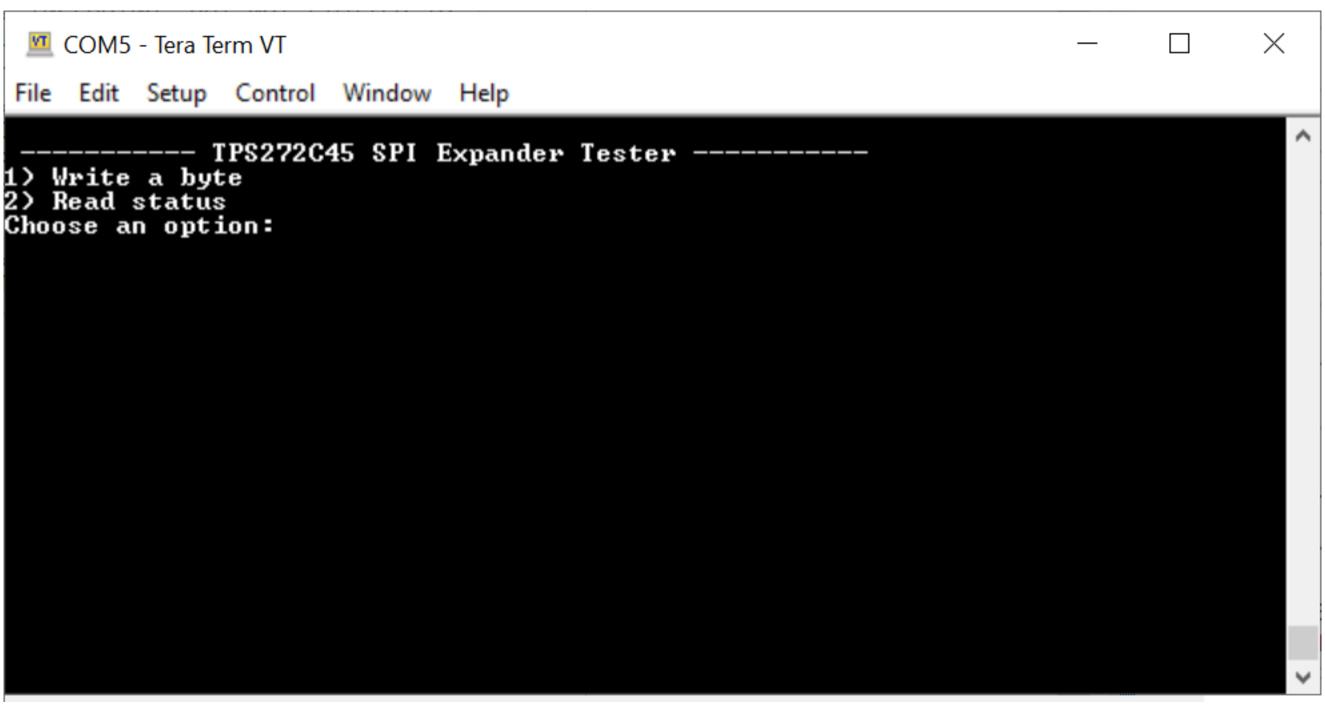

Figure 3-3. Main Menu

The serial port in this application has the ability to take in user input as well as display the status of the SPI expander. From the main menu enter **1** to write a word of data. You will then be prompted to enter four hexadecimal characters that represent the 16-bit word of the ENx lines that you want to toggle. In Figure 3-4, we enter **CAFE** to write the hexadecimal value 0xCAFE to the SPI expander.

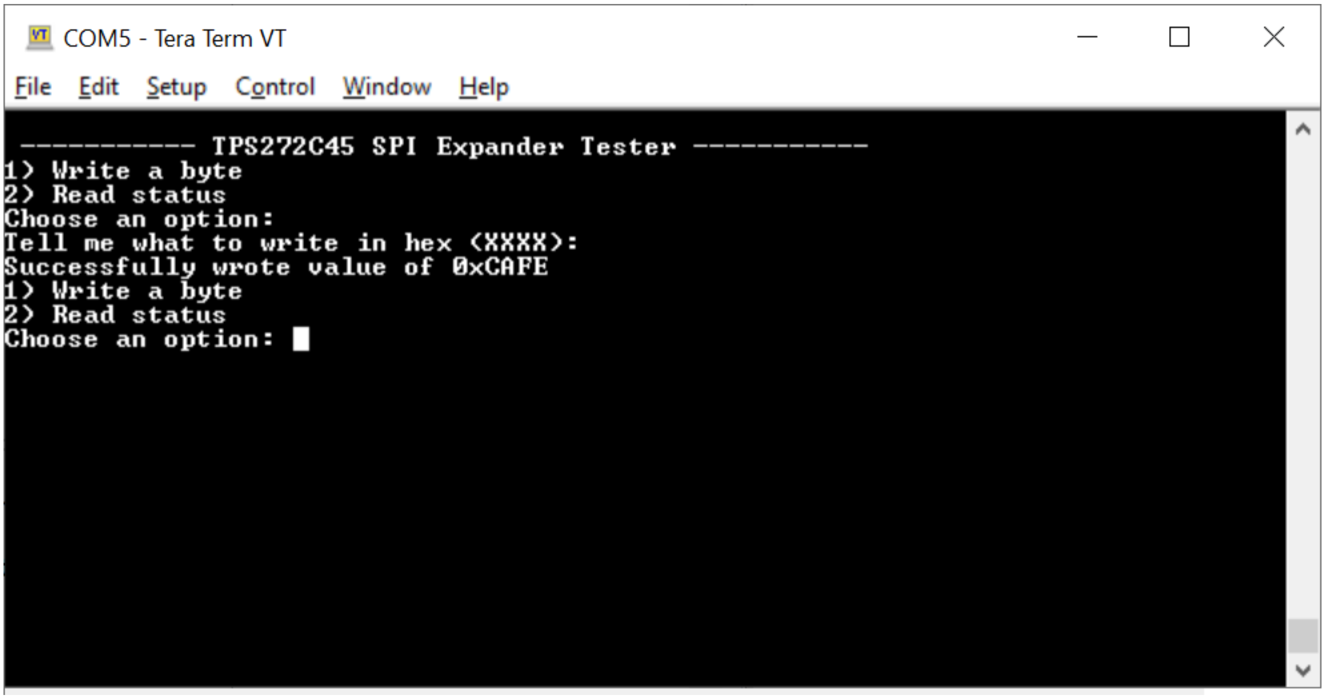

Figure 3-4. Writing 0xCAFE

From an attached logic analyzer in Figure 3-5, you can see the relevant bytes being written over the SPI bus on the CC2652R LaunchPad as well as the relevant ENx bits being enabled/disabled.

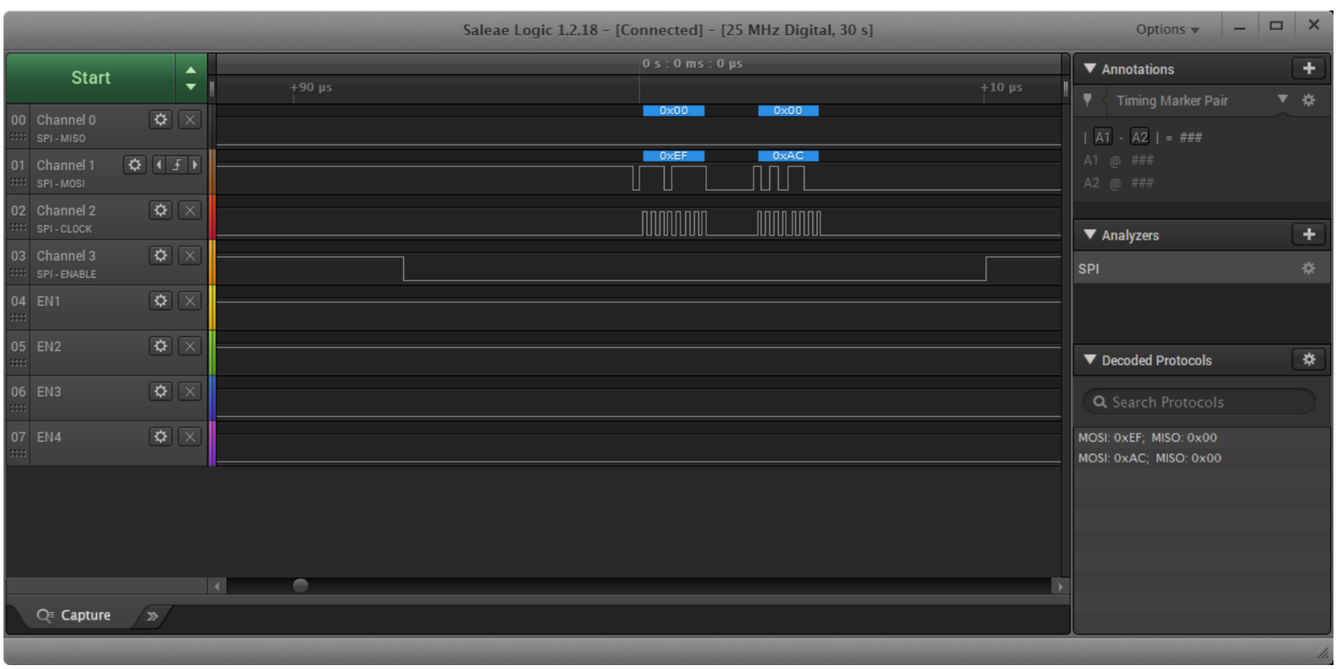

Figure 3-5. SPI Transaction - Writing to ENx

With the TPS272C45EVMs attached to the daughter card this toggles on and off the load connected to the corresponding VOUT1 and VOUT2 channels. Note that the bit order of the write has been reversed to match the pin out of the daughter card as well as the bit order of the attached shift registers. Also note that as SPI is full duplex that both the ENx and FLTx lines are written/read at the same time. When performing a polling transaction to read the FLTx lines it is necessary to also write the appropriate ENx lines so that those bits do not get reset. This can be prevented by having separate SPI CS lines for both the FLTx and ENx shift registers on the schematic (at the expense of an additional GPIO on the microcontroller). The code block for reading the FLTx lines (and by effect writing the ENx lines) can be found below from ***spi_expander_test.c***.

```
/* Reading/Writing the SPI registers */
transaction.count = 2;
transaction.txBuf = &writeByteReversed;
transaction.rxBuf = &readByte;

GPIO_write(SPI_EXPANDER_CS, 0);
retCode = SPI_transfer(masterSpi, &transaction);
GPIO_write(SPI_EXPANDER_CS, 1);

if(retCode == true)
{
    terminalWrite("\r\nSuccessfully read value of ");
    sprintf(readByteString, "0x%X", (readByte & 0xFF));
    terminalWrite(readByteString);
    terminalWrite("\r\n");
}
else
{
    terminalWrite("\r\nError reading byte value!!!\r\n");
}
```

For illustration, a fault was artificially introduced into the SPI expander by tying the FLT1 and FLT2 pins to the 3.3 V rail on the LaunchPad. When the SPI transaction above is now performed, a value of 0x03 can be read from the FLT shift register representing a fault on channel 1 and channel 2 of the first TPS272C45EVM.

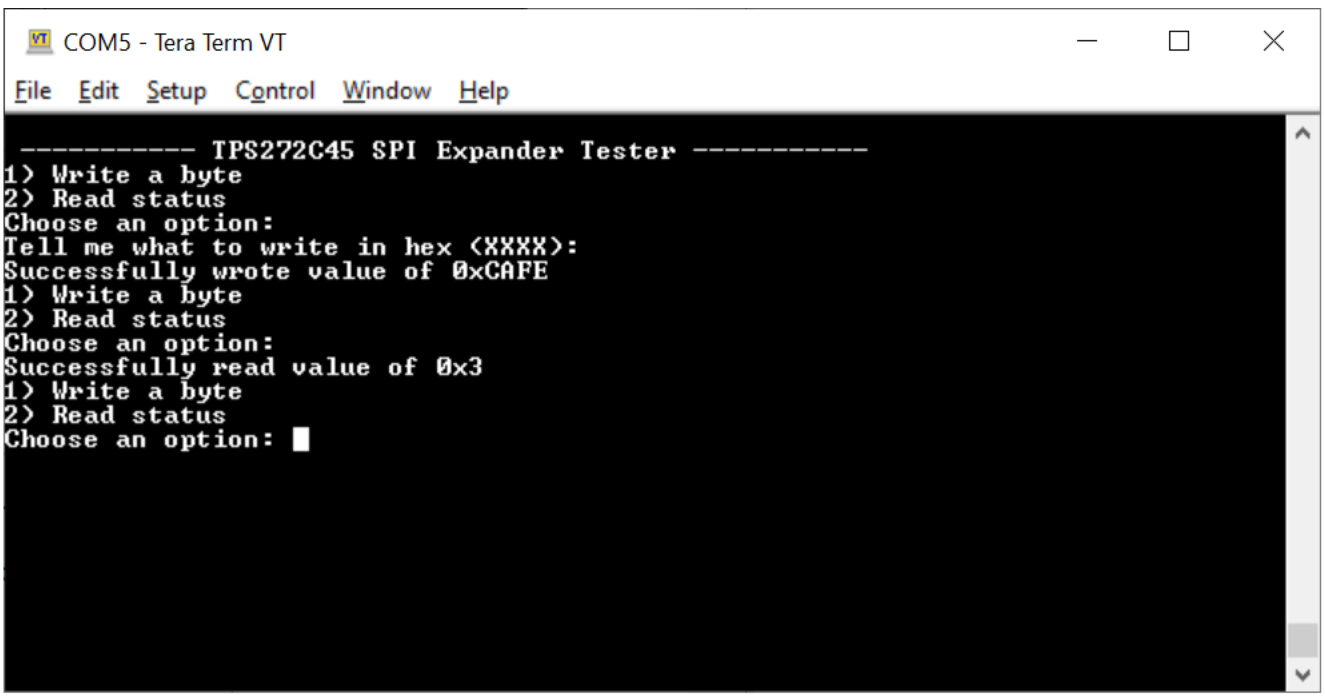

Figure 3-6. Reading a Fault

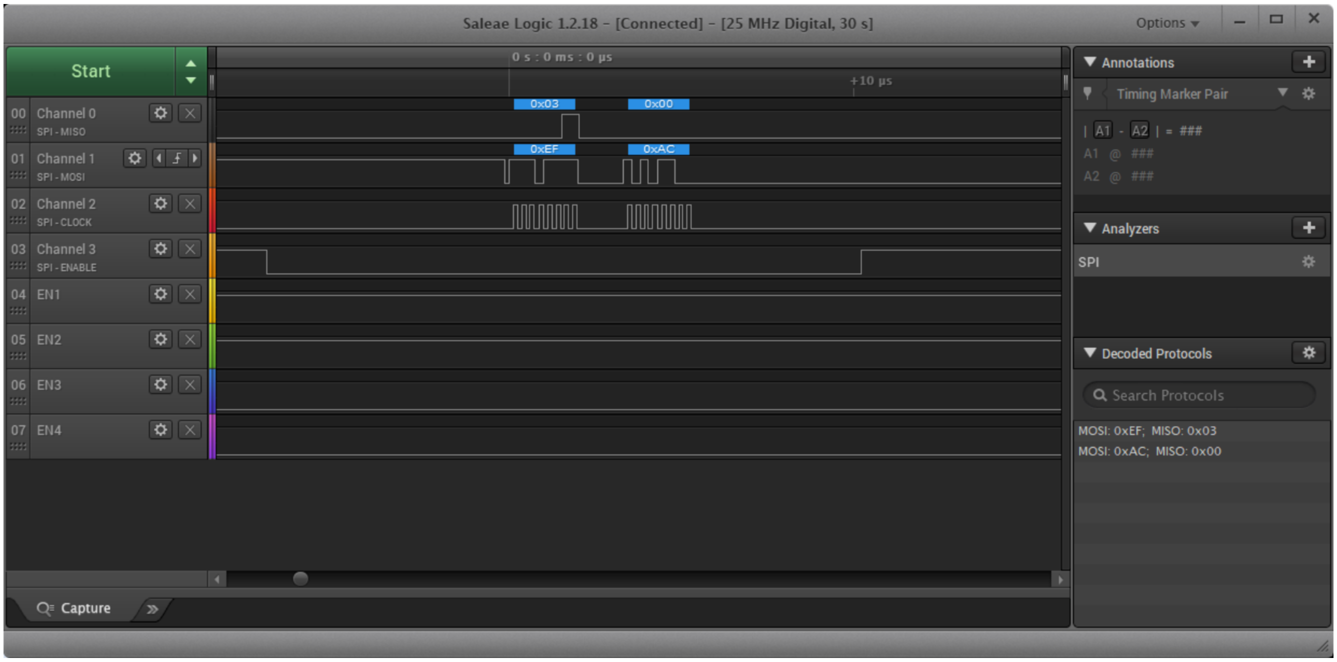

Figure 3-7. Fault on SPI Line

As two bytes are read by the SPI master to match the byte length for the read it is the responsibility of the application to mask out the irrelevant bytes in software. In the example above we take the 16-bit value read by the microcontroller for the FLTx line and bitwise AND it with 0xFF to get the relevant data.

4 Schematics

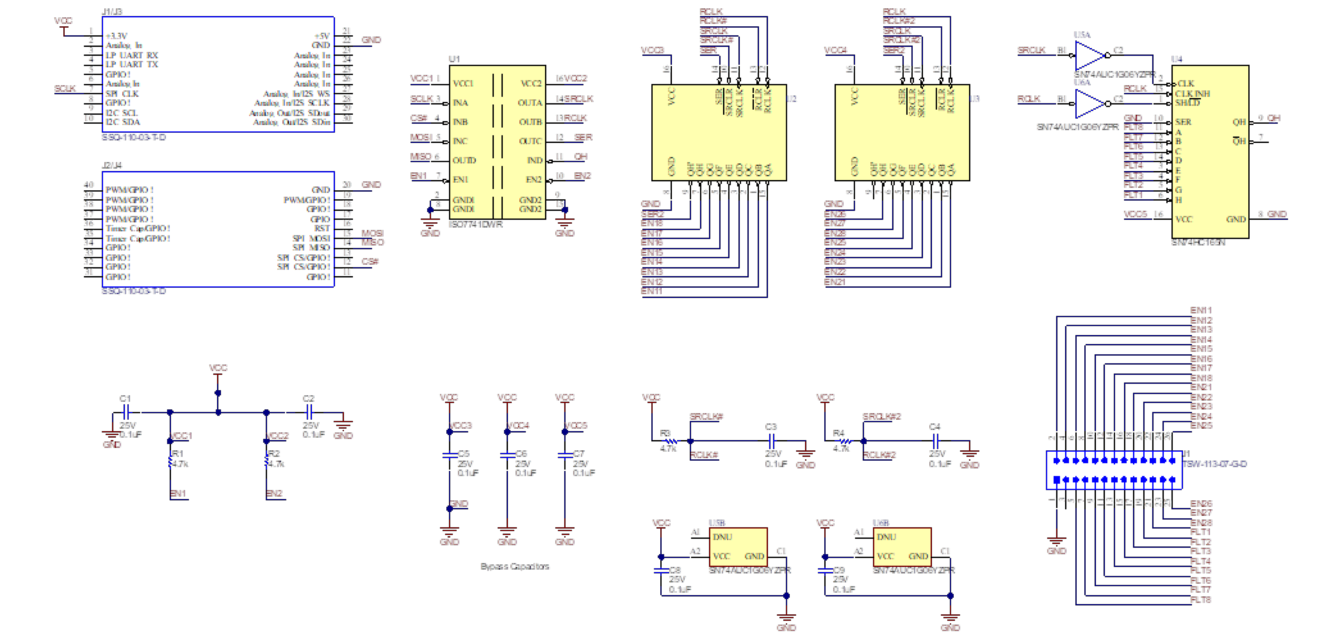

Figure 4-1. SPI Expander Schematic

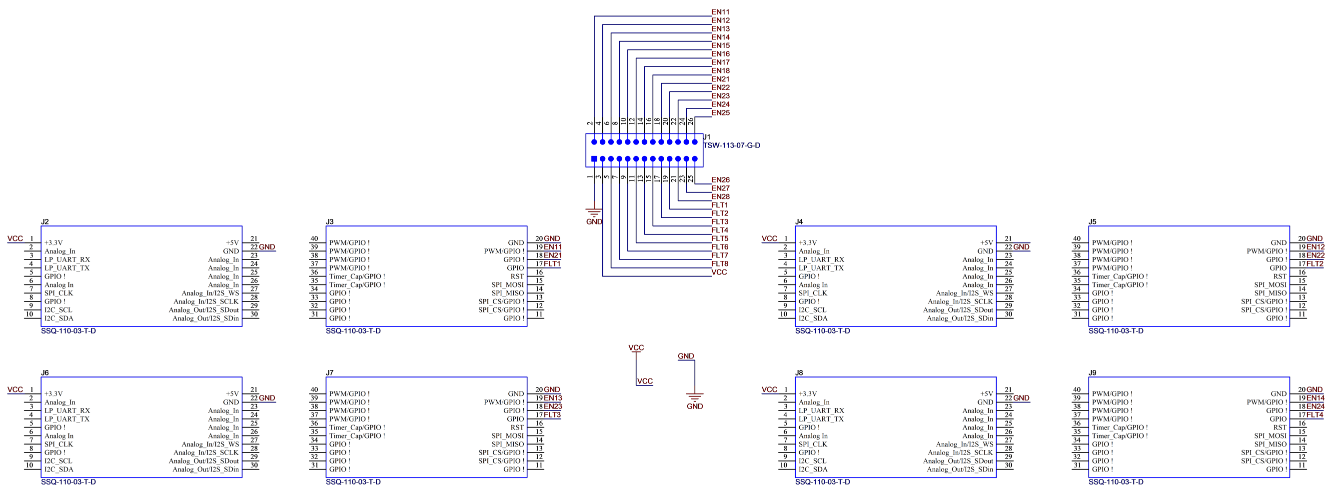

Figure 4-2. Daughter Card Schematic

5 Layouts

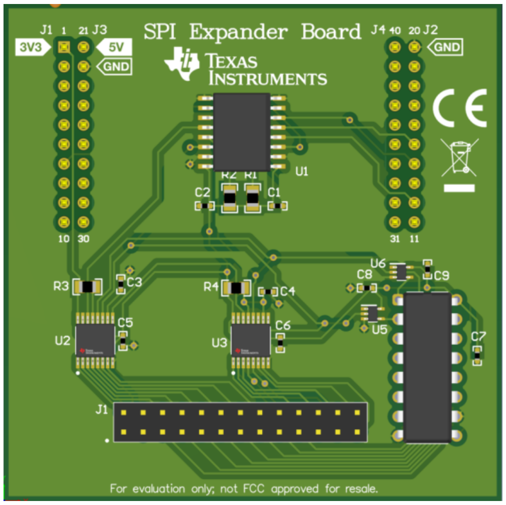

Figure 5-1. SPI Expander 3D Layout

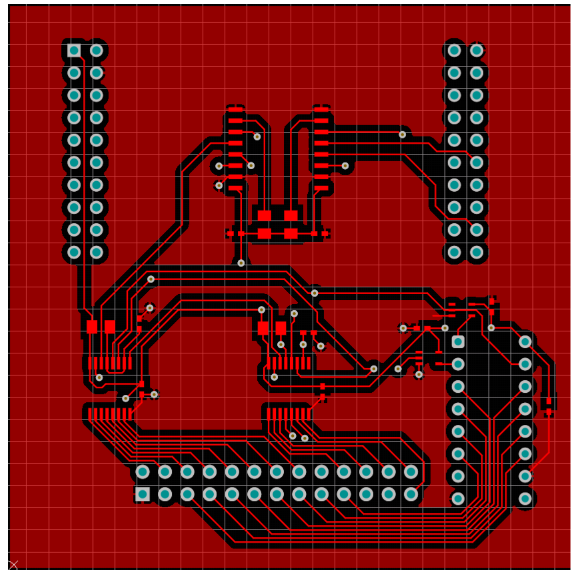

Figure 5-2. SPI Expander Top Layer

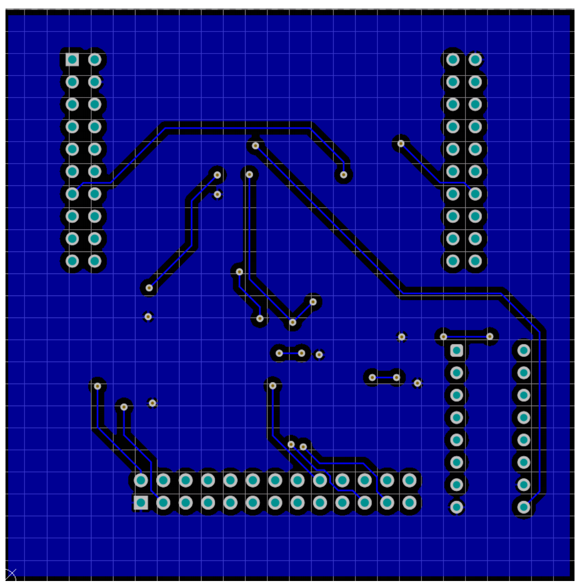

Figure 5-3. SPI Expander Bottom Layer

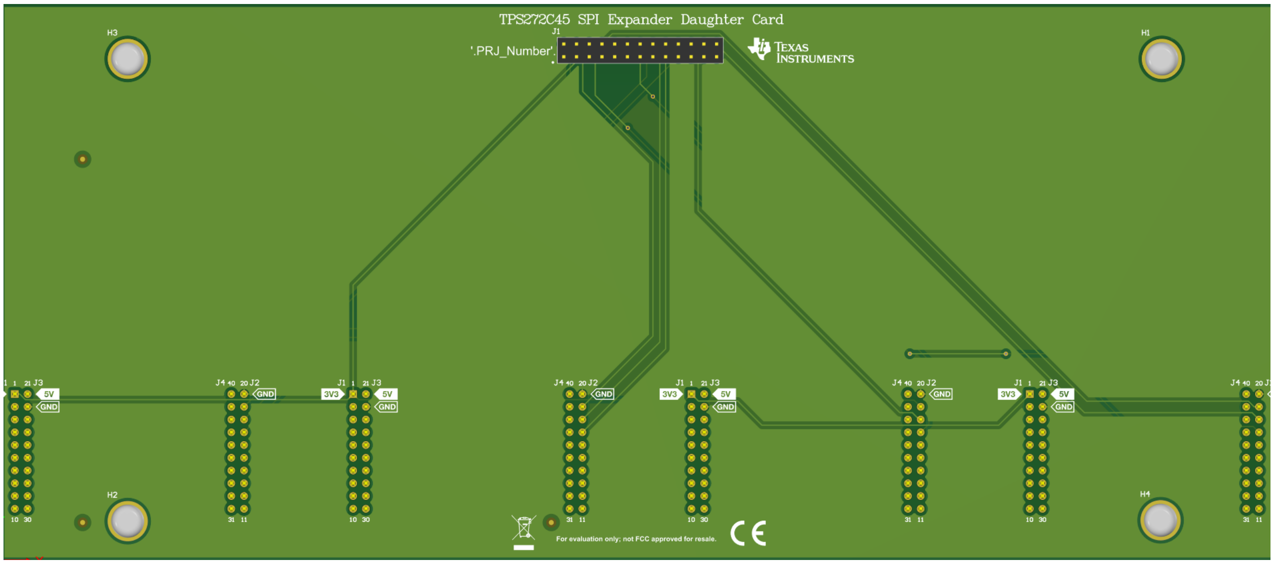

Figure 5-4. Daughter Card 3D Layout

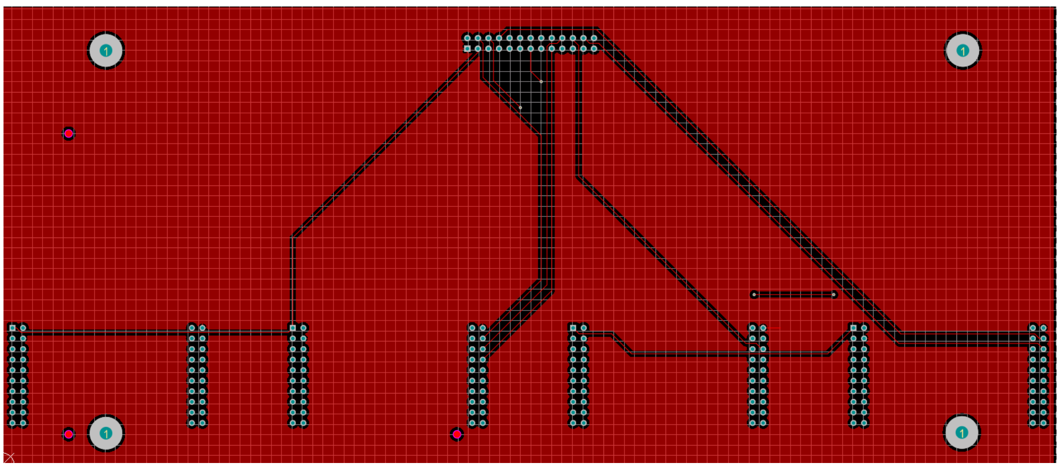

Figure 5-5. Daughter Card Top Layer

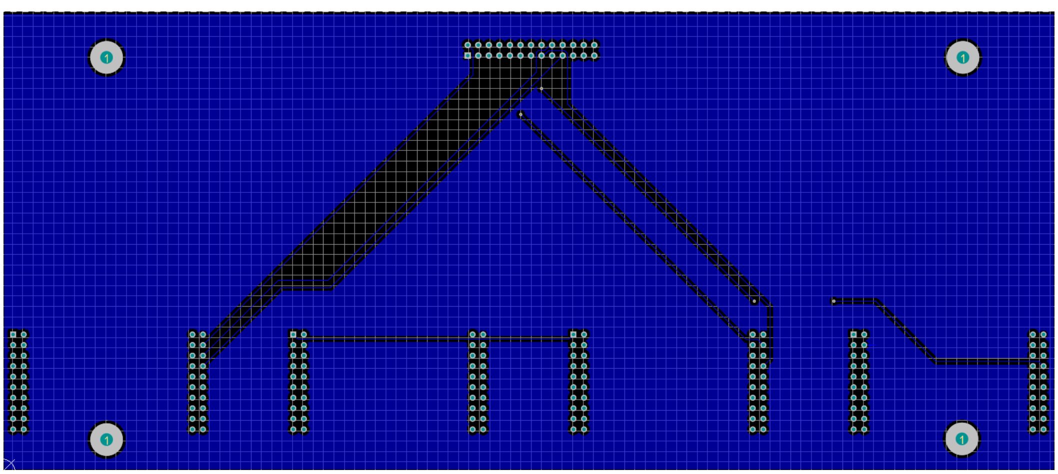

Figure 5-6. Daughter Card Bottom Layer

6 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (January 2021) to Revision A (May 2021)	Page
• Added hyperlink to software	1