Application Note

SPI Daisy Chaining with TXE81xx-Q1 GPIO Expander

Tyler Townsend

ABSTRACT

This application note walks through an example use case of SPI daisy chaining with multiple TXE8124-Q1 devices.

Table of Contents

Trademarks

1 Introduction

Daisy chaining is a feature used by several SPI protocol devices. Daisy chaining is used to save on wiring costs and helps to reduce the total number of required IOs from the MCU and processor making for a more concise and less complex PCB.

2 What is SPI Daisy Chaining?

SPI daisy chaining is a connection scheme used to communicate with multiple SPI peripheral devices in series. Daisy chaining reduces the amount of wire/trace length needed and saves GPIO on the MCU for multiple chip selects.

In a normal SPI configuration of multiple peripherals, a chip select signal is required for each SPI peripheral device. This means a GPIO from the MCU must be reserved for each SPI peripheral in the system. See the example below of a normal SPI configuration between 4 peripheral devices.

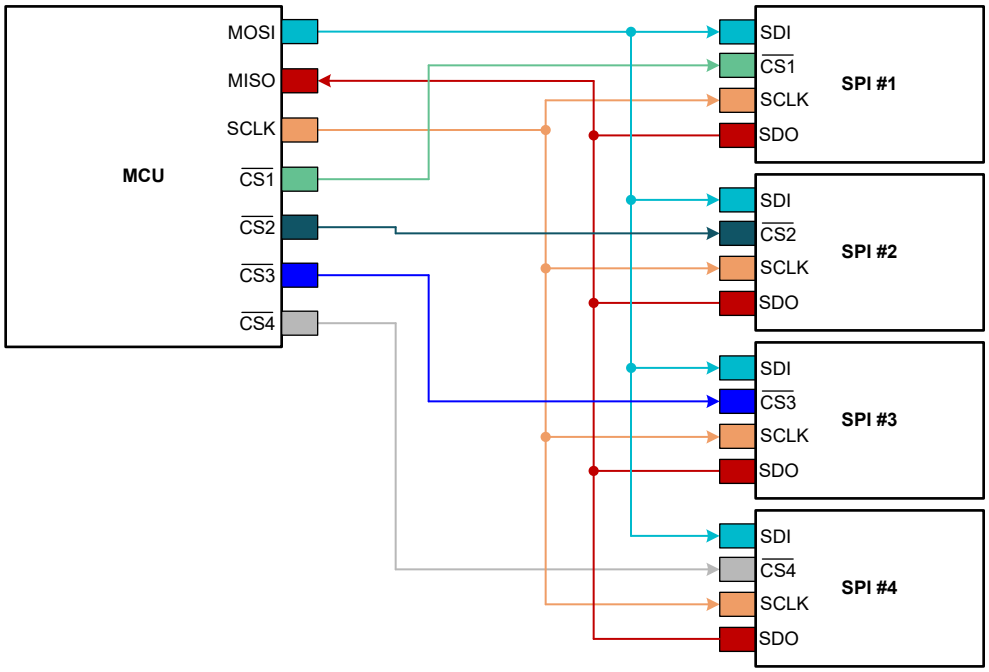

Figure 2-1. Example of the SPI Bus without Daisy Chain

Eight pins from the MCU are needed to control 4 SPI peripheral devices. PICO from the MCU is connected to all serial data inputs (SDI). MISO is connected to the serial data outputs (SDO). The clock pin (SCLK) is shared amongst all devices in the system. An individual chip select must be dedicated to each peripheral in the system.

The normal implementation of SPI requires multiple GPIO pins from the MCU of which can be limited in certain systems. This also means that the system uses more wiring to route each chip select to each peripheral device. This can mean more physical wiring and therefore added weight in a system, or a more cluttered PCB.

To solve these two problems of reducing wiring and the number of chip select lines required, a SPI daisy chain connection scheme can be implemented. SPI daisy chain must be supported by the IC device manufacturer for this to work.

(Note: SDO is connected to SDI, only one chip select signal is used)

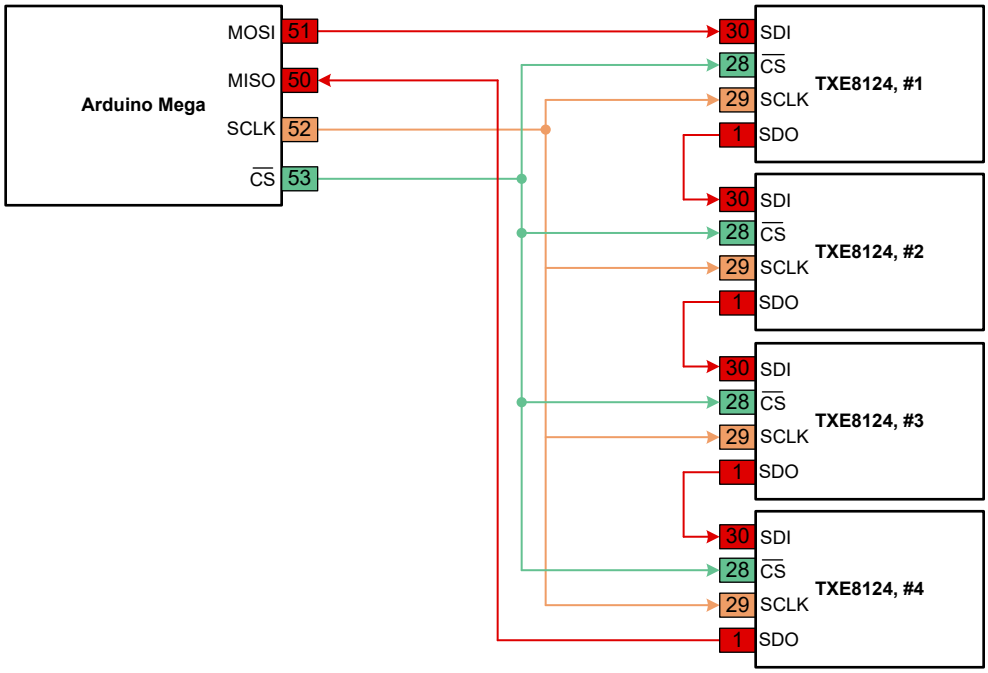

Figure 2-2. Example SPI Bus with Daisy Chaining

The daisy chain implementation streamlines the serial data by connecting the output from one peripheral to the input of another. In this case, the wiring can be reduced drastically since the routing does not need to come from the MCU, but can be "chained" to each peripheral in the sequence.

3 SPI Daisy Chain Example with TXE81xxEVM

The following example use case daisy chains 4 x TXE81XXEVM together by connecting the SDO and SDI lines in a chain. The SCLK pin is shared amongst all devices including the MCU as well as the chip select signal.

(Boards are labeled 1 - 4 starting from right to left)

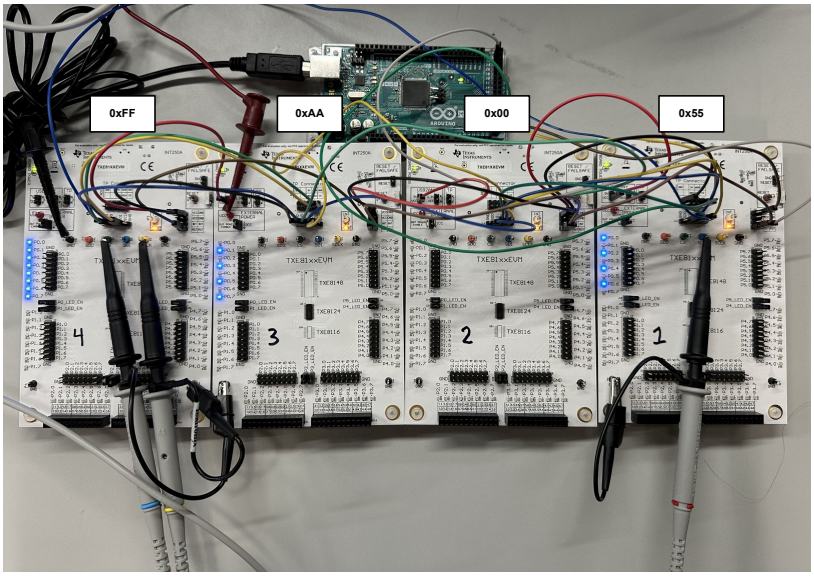

Figure 3-1. 4 x TXE81XXEVM's Connected in Daisy Chain Configuration

Data is written to each direction configuration register (0x04) of each TXE8124-Q1 device. Only Port 0 is written to. A *1* in the direction configuration register sets the GPIO to an OUTPUT. A *0* in the direction configuration register sets the GPIO to an INPUT. See the Board Configurations table for the exact data written in the daisy chain example.

Table 3-1. Board Configurations

Board Number	Register Address	Port	Data	Input / Outputs?
1	0x04	0	0x55	Inputs = P0.1, P0.3, P0.5, P0.7 Outputs = P0.0, P0.2, P0.4, P0.6
2	0x04	0	0x00	Inputs = P0.0 - P0.7 Outputs = none
3	0x04	0	0xAA	Inputs = P0.0, P0.2, P0.4, P0.6 Outputs = P0.1, P0.3, P0.5, P0.7
4	0x04	0	0xFF	Inputs = none Outputs = P0.0 - P0.7

In TXE81xx, there are 4 types of SPI segments: Status, Header, Address, and Data. The following table describes the bit-by-bit description of each segment to be sent in the daisy chain.

Table 3-2. SPI Segment Description

SPI Segment Type	Bit Assignments
Status	Bit [15:14] = 1, indicates status segment Bit[13:8] = Bit 5 to 0 of the Fault Status Register (0x1900) Bit[7:0] = 0, by default
Header	Bit [15:14] = 0 and 1 respectively, indicates header segment Bit [13] = reserved Bit[12:0] = determine the number of devices in the daisy chain
Address (register address)	Bit [15] = indicates SPI mode of operation (1 = read, 0 = write) Bit [14:13] = Don't care (X) Bit[12:8] = Feature Address Bit[7] = Don't care (X) Bit[6:4] = Port Selection Bit[3:1] = Don't Care (X) Bit[0] = Multi-Port
Data	Bit[7:0] = Data to write to register

To begin sending data through the chain, a header segment is sent first, followed by the register address of the furthest board in the chain. If there are four devices in the chain, register address of the 4th device is sent first, followed by the 3rd and so on. Data bytes follow after the register address bytes. The first data byte applies to the furthest device in the chain. If there are four devices in the chain, the first data byte applies to the 4th device, then 3rd, and so on. See *Sequence of Each Byte in the Chain* for a more detailed byte-by-byte example of how SPI data is sent.

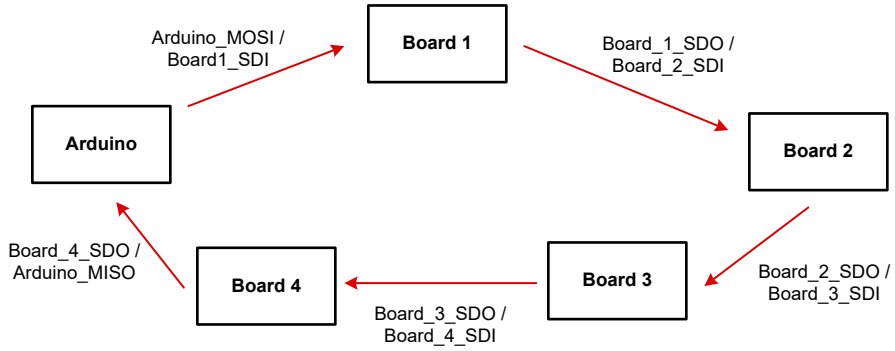

Figure 3-2. Daisy Chain Block Diagram

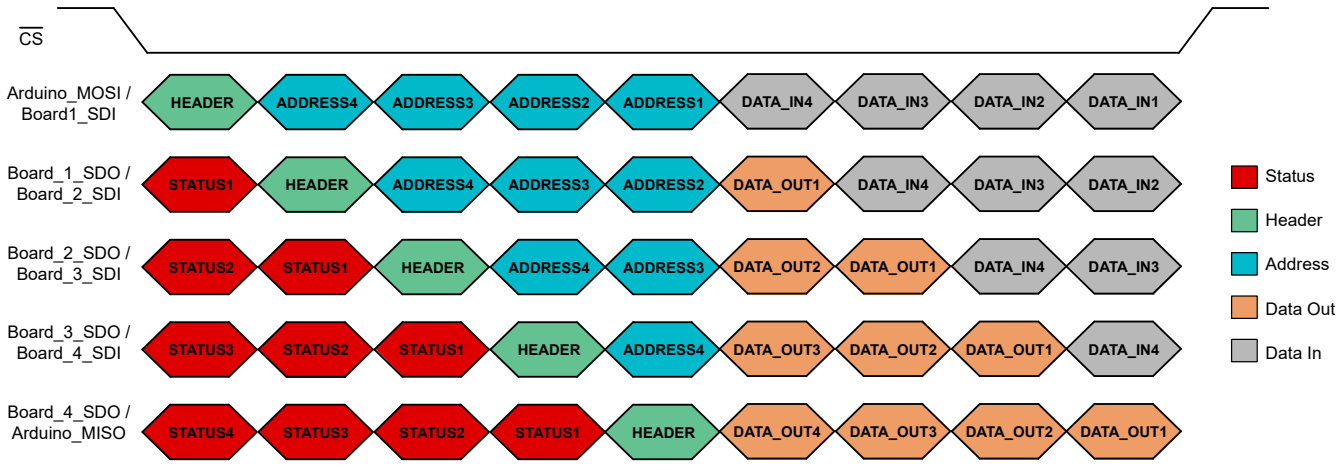

Figure 3-3. Sequence of Each Byte in the Chain

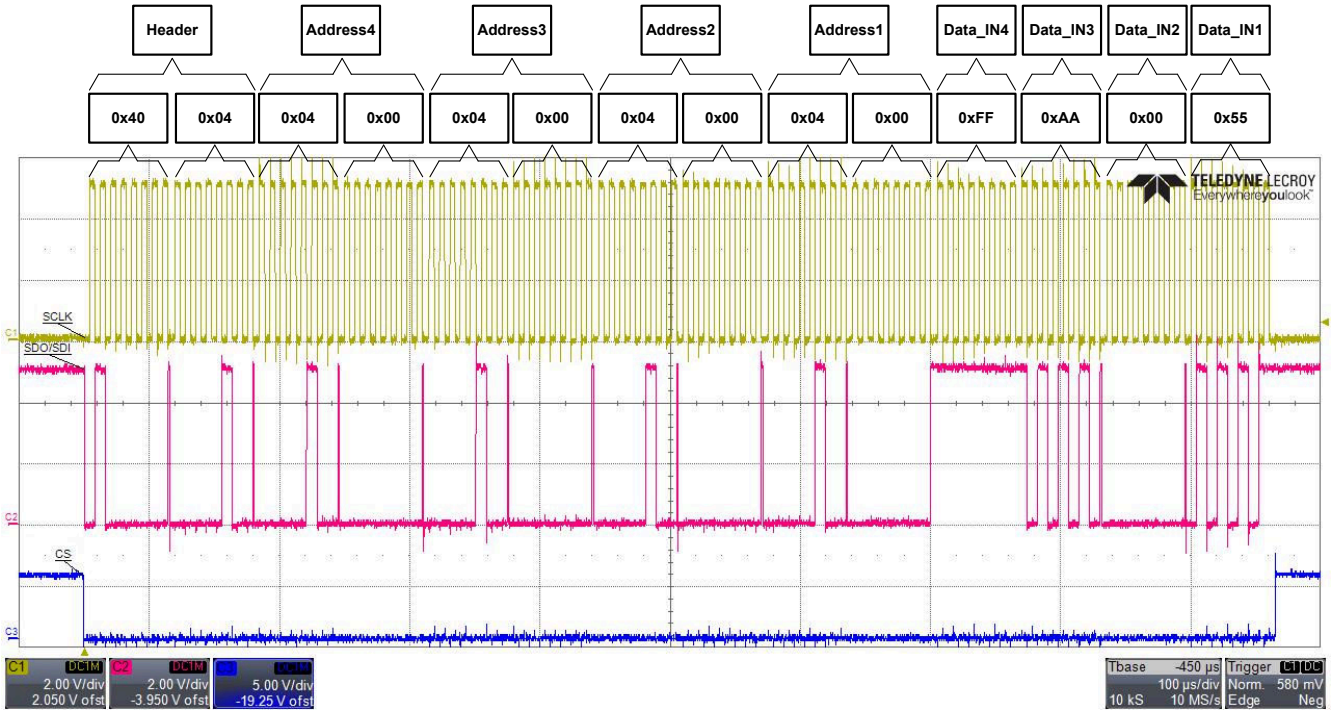

Figure 3-4. Annotated Waveform of the Daisy Chain SPI Transfers Probed Between Arduino_MOSI and Board 1 SDI

4 MSPM0 Pseudocode Example

```
#include"ti_msp_dl_config.h"//MSP driver library
int main(void)
{
SYSCFG_DL_INT(); //initialize SPI driver
DL_GPIO_setPins(GPIO_LEDS_PORT, GPIO_LEDS_USER_LED_1_PIN | GPIO_LEDS_USER_TEST_PIN); //set /CS HIGH
DL_GPIO_clearPins(GPIO_LEDS_PORT, GPIO_LEDS_USER_LED_1_PIN | GPIO_LEDS_USER_TEST_PIN); //set /CS
LOW
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0b01000000); //header segment
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0b00000100);
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x04); //board 4 address
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x00);
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x04); //board 3 address
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x00);
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x04); //board 2 address
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x00);
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x04); //board 1 address
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x00);
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0xFF); //board 4 data
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0xAA); //board 3 data
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x00); //board 2 data
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x55); //board 1 data
DL_GPIO_setPins(GPIO_LEDS_PORT, GPIO_LEDS_USER_LED_1_PIN | GPIO_LEDS_USER_TEST_PIN); //set /CS HIGH
}
```

5 Arduino Pseudocode Example

Arduino Coding Example

```
#include <SPI.h>
#define CS 53

//MISO = 50
//CS = 53
//MOSI = 51
//SCLK = 52

void setup() {
  Serial.begin(115200);
  pinMode(CS, OUTPUT);

  SPI.begin();
  SPI.beginTransaction(SPISettings(125000, MSBFIRST, SPI_MODE0));
}

void loop() {
  //send SPI in 8-bit words
  uint8_t header_seg1 = 0b01000000;   //default header segment
  uint8_t header_seg2 = 0b00000100;   //4 devices in the chain
  uint8_t address_seg1 = 0b00000100;  //direction configuration register
  uint8_t address_seg2 = 0b00000000;  //port 0 selected, no multi-port

  digitalWrite(CS, LOW);
  SPI.transfer(header_seg1);
  SPI.transfer(header_seg2);
  SPI.transfer(address_seg1);         //board 4 address
  SPI.transfer(address_seg2);
  SPI.transfer(address_seg1);         //board 3 address
  SPI.transfer(address_seg2);
  SPI.transfer(address_seg1);         //board 2 address
  SPI.transfer(address_seg2);
  SPI.transfer(address_seg1);         //board 1 address
  SPI.transfer(address_seg2);
  SPI.transfer(0xFF);                 //board 4 data
  SPI.transfer(0xAA);                 //board 3 data
  SPI.transfer(0x00);                 //board 2 data
  SPI.transfer(0x55);                 //board 1 data
  digitalWrite(CS, HIGH);
}
```

6 Summary

There are several advantages and disadvantages of implementing regular SPI vs. SPI daisy chain in the table below.

Table 6-1. Engineering Tradeoffs for Regular SPI vs. SPI Daisy Chaining

	Regular SPI	SPI Daisy Chaining
Wiring	• Increased wiring due to multiple CS lines • Increases PCB complexity and potential wiring cost • More wire = more weight = more cost	• Reduced wiring due to a single CS line • Less complexity in PCB routing and connection to each SPI peripheral
Device Control	• Individual device control capable	• Difficulty controlling a single device, must talk to the entire chain
Data Transmission	• Data transmission is inherently faster	• Data transmission is inherently slower since data must pass through every device in the chain
Future Designs	• Harder to change future designs, requires additional CS lines for each added peripheral in the system	• Easier to add to the end of the SPI daisy chain for additional peripherals. Still use the same CS line.
Signal Integrity	• Must fan out electrical connection to multiple devices, can result in increased trace length / wiring distance	• Series connections keep trace length short reducing SI. Each peripheral device in the chain redrives SPI data to next device in the chain
Software	• Simpler software implementation	• Streamlined data updates if all peripherals in the chain need configuring • More complex software implementation - longer SPI words
Debug	• Easier to pin-point peripheral failures	• Harder to determine which peripheral in the chain broke • If one peripheral in the chain breaks, multiple peripherals are affected

7 References

•

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265