Introduction to SPI Communication

Materials:

- 2 Series Mixed Series Oscilloscope (MSO)
- Arduino UNO
- Resistors (3)
- Breadboard
- Jumper wires

Procedure:

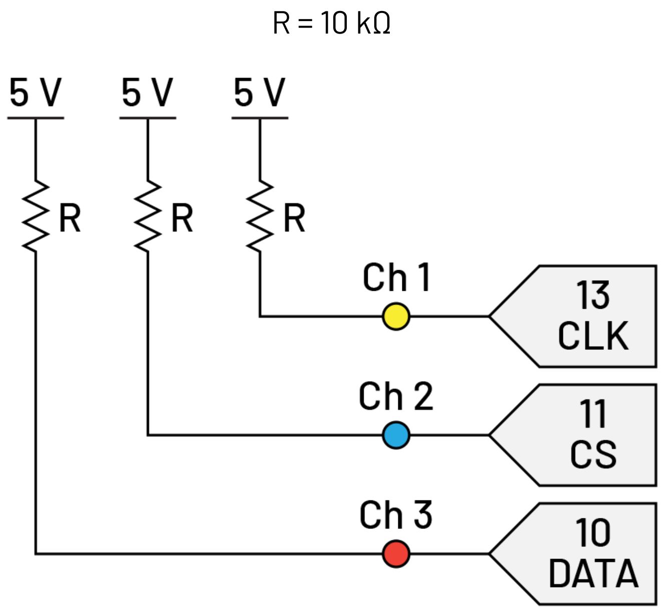


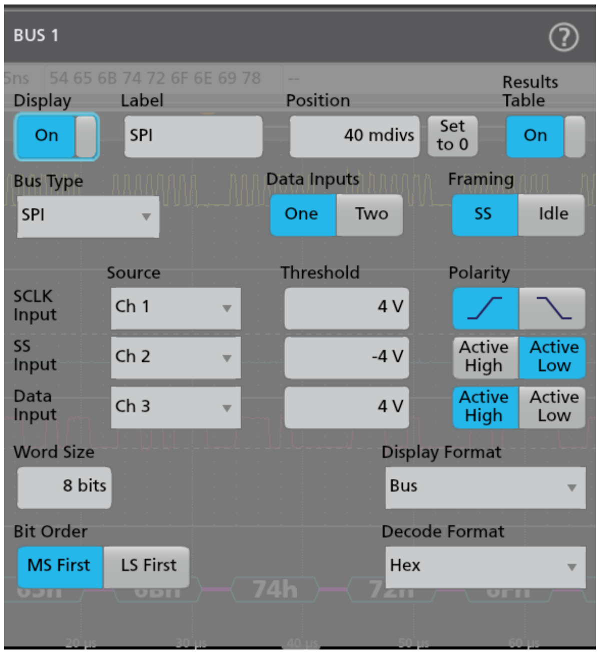
Figure 1. Pullup resistor wiring.

1. Serial Peripheral Interface (SPI) is a synchronous serial communication protocol typically used for short-distance communication in embedded systems. It involves a controller or master device that initiates communication and one or more peripheral or slave devices. The controller generates a clock signal (SCLK) and uses a single data line for bi-directional data transfer (sometimes called SDI/SDO, or simply data line). The controller uses the slave select (SS) or chip select (CS) line to select the peripheral device by activating the signal low. For this lab, the controller device will be an Arduino UNO. The 2 Series MSO will decode the SPI signals sent by the Arduino by using three channel probes.

2. Create the wiring diagram in Figure 1. with the corresponding pull up resistors. Use the 5V output on the Arduino UNO to supply voltage to the resistors. Connect the digital pins 10, 11 and 13 to the resistors. To observe and decode the message, connect channel 1 to the clock (pin 13), channel 2 to the chip select (pin 11) and channel 3 to the data line (pin 10). Make sure to ground the probes to one of the GND pins on the Arduino.

3. Write a script that sends a message with your name using SPI. To start, include the <SPI.h> library to use the SPI communication on the Arduino. Make sure the CS pin is set high to deactivate it. Once ready for transmission, set the CS pin low. Send over the message character by character. Once the entire message is transmitted, set the CS pin back high to stop the transmission. Send this message in a loop with a delay to decode easily in the next step.

4. Activate channels 1-3 by pressing the colored channel buttons on the right panel on the 2 Series MSO. Add a bus channel by pressing the "Bus" button. On the bus menu, select SPI under "Bus Type". Follow the rest of the configuration settings shown in Figure 2. to set up the bus. Exit the menu and double tap on the screen to bring up the "WAVEFORM VIEW" menu. Select "Stacked" under "Display Mode" to see all three channels and the bus.


BUS 1

Display: On
Label: SPI
Position: 40 mdivs
Results Table: On
Bus Type: SPI
Data Inputs: [✓ One] Two
Framing: [✓ SS] Idle

	Source	Threshold	Polarity
SCLK Input	Ch 1	4 V	rising edge
SS Input	Ch 2	-4 V	Active High [✓ Active Low]
Data Input	Ch 3	4 V	[✓ Active High] Active Low

Word Size: 8 bits
Display Format: Bus
Bit Order: [✓ MS First] LS First
Decode Format: Hex

Figure 2. Bus configuration for SPI input.

5. Run the message script onto the Arduino and begin observing the waveforms on the oscilloscope. To help catch the message, set the "Trigger" (located at the bottom right corner of the screen) to "Trigger Type" and select "Bus". Make sure the SPI bus is selected as the source bus and select "SS Active" under the "Trigger On" option. This will have the oscilloscope trigger when the Arduino sets the chip select low.

6. Record the decoded data in Table 1. The oscilloscope can decode the message in both hexadecimal and binary. To switch to binary, go back to the bus configuration menu and under "Decode Format" select "Binary". Use an ASCII table to decode the message from hexadecimal to characters and verify the message was transmitted correctly.

Data Type	Data
Hexadecimal	
Binary	
Character	

Table 1. Transmitted and decoded SPI data.

Instructor Notes:

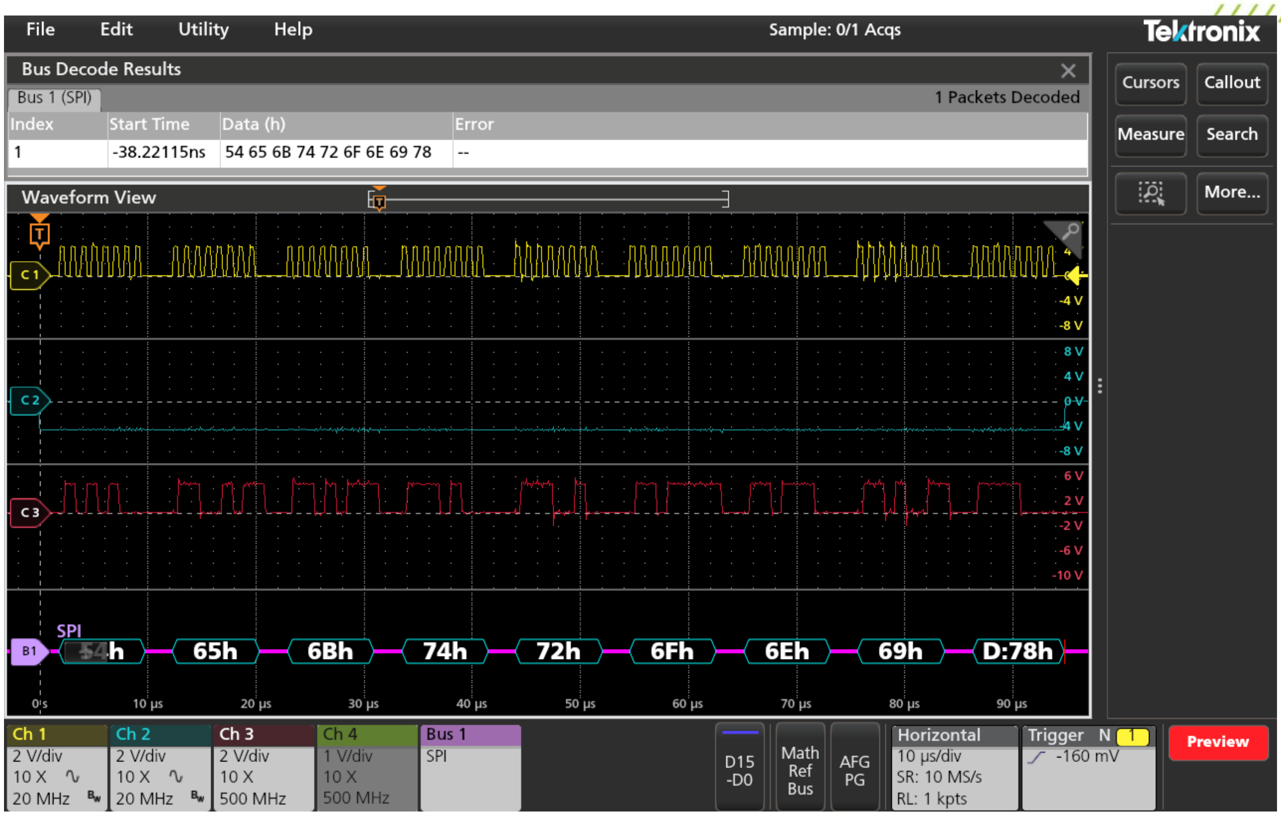
Example code for SPI communication with Arduino UNO:

```
#include <SPI.h>
const int CS_PIN = 10; // Chip Select pin
void setup() {
 // Set the Chip Select pin as an output
 pinMode(CS_PIN, OUTPUT);
 // Ensure the Chip Select pin is high (inactive)
 digitalWrite(CS_PIN, HIGH);
 // Initialize SPI
 SPI.begin();
}
void loop() {
 // Message to send
 const char* message = "Tektronix";

 // Activate the Chip Select pin (set it low)
 digitalWrite(CS_PIN, LOW);

 // Send the message over SPI
 for (int i = 0; message[ i ] != '\0'; i++) {
  SPI.transfer(message[ i ]);
 }
 // Deactivate the CS pin (set it high)
 digitalWrite(CS_PIN, HIGH);


 // Wait for 1 second before sending the next message
 delay(1000);
}
```



Bus Decode Results

Bus 1 (SPI) 1 Packets Decoded

Index	Start Time	Data (h)	Error
1	-38.22115ns	54 65 6B 74 72 6F 6E 69 78	--

Ch 1	Ch 2	Ch 3	Ch 4	Bus 1
2 V/div	2 V/div	2 V/div	1 V/div	SPI
10 X	10 X	10 X	10 X	
20 MHz	20 MHz	500 MHz	500 MHz	

Horizontal
10 µs/div
SR: 10 MS/s
RL: 1 kpts

Trigger N 1
-160 mV

Figure 3. Waveforms results from example code. The scope decodes the data channel into hexadecimal. The message spells out “Tektronix”.

Find more valuable resources at TEK.COM