AN98538

Serial Buses Comparison: JTAG, SPI, and I2C

Author: Russell Hanabusa

AN98538 introduces three serial buses: JTAG, SPI, and I2C. It discusses features of these three buses including pinout definition, connection method, and bus protocol.

1 Introduction

This paper discusses three popular serial buses:

- JTAG - Joint Test Action Group
- SPI - Serial Peripheral Interface
- I2C - Inter-Integrated Circuit

A typical electronic product today has one or more serial buses in them.

Using a serial bus in a system has many advantages over a parallel bus.

1. Lower component cost
2. Smaller printed circuit board
3. Simplified design
4. Lower power consumption

The following table compares the JTAG, SPI, and I2C buses:

Table 1. Overview of SPI, I2C, and JTAG Serial Buses

Name	Architecture	Feature	Multi-Master	Data Rate	Flyby Data Transfer	Full Duplex
SPI	Two shared unidirectional data signals and a shared clock	Bidirectional communication on share bus	Possible, but not standard	132 Mbps @133 MHz	No	Yes
I2C	Shared data signal, and a shared clock signal	Bidirectional communication on shared bus	Yes	0.1 Mbps, 0.4 Mbps, 1.0 Mbps, 3.5 Mbps, 5.0 Mbps	Yes	No
JTAG	Daisy Chain data signal	Can verify device connectivity, in-circuit emulation, device configuration, and internal device functional testing	No	10-100 MHz typical	N/A	N/A

These three serial buses are described in the following sections.

2 SPI

SPI is a de facto standard that allows for full-duplex data transfers. Typically, there is only one bus master; all other SPI devices are slaves. Some vendors like Xilinx™ provide a means for SPI bus arbitration, which illustrates one of the many nonstandard features that are available with this interface.

SPI has a shared data bus. The master transmits data on the SO signal line and receives data on the SI signal line; this allows the bus master to simultaneously transmit and receive data. All data transfers must take place between the Bus Master and slaves. Data transfers directly between two slave devices (Flyby transfers) are not allowed. The SPI Bus Master signals are shown below.

SCK SPI Bus Clock output by SPI Bus Master

SI Input Data to Bus Master

SO Output Data from Bus Master

Figure 1 shows an example of an SPI system with one bus master and three slave devices. The SPI Bus Master outputs an individual device select signal for each SPI device slave.

Figure 1. Typical SPI Bus System

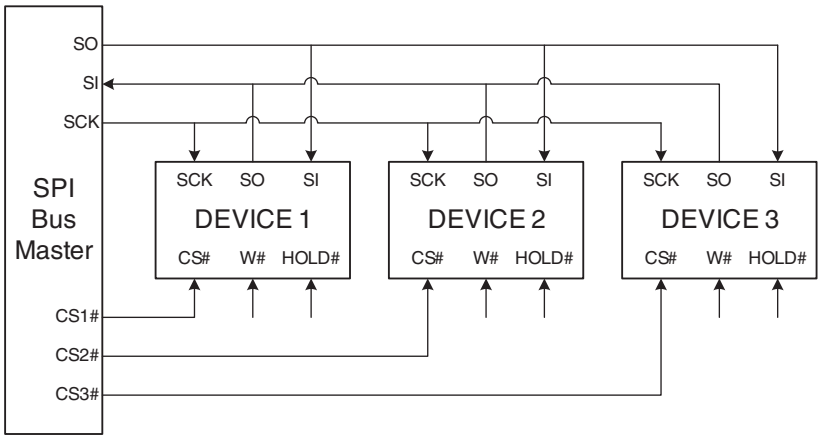
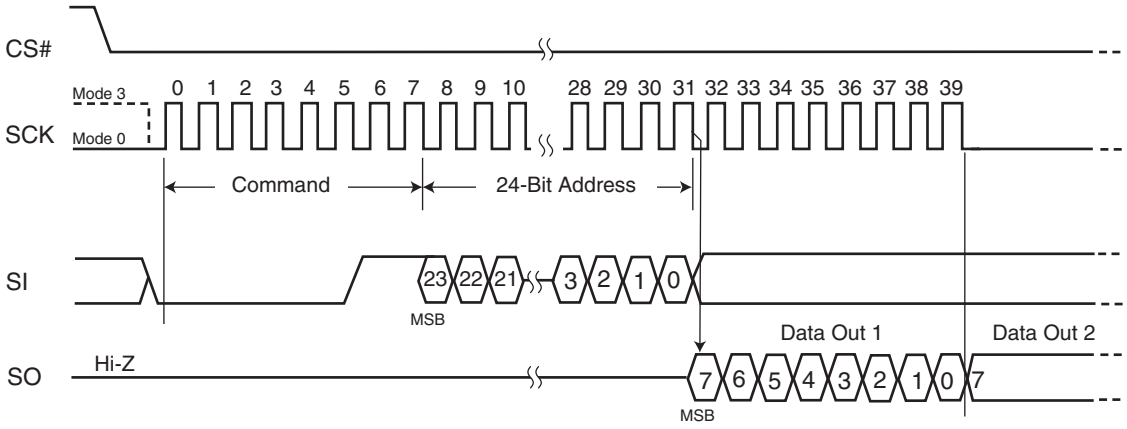


Figure 2 is an example of an SPI read bus cycle of a Cypress SPI Flash. For Cypress SPI devices, data is sampled on the rising edge of SCK and changes on the falling edge of SCK.

Figure 2. SPI Bus Cycle



The bus cycle events are as follows:

1. CS# is asserted LOW to select the flash.
2. The SPI read command is sent.
3. The internal 24-bit flash memory address is sent.
4. The first flash data byte is read out.

In Figure 2 at the left of the SCK signal are shown the options of *Mode 0* and *Mode 3*. Depending on the system SCK should be HIGH (Mode3) when idle; in other systems, SCK should be LOW (Mode0) when idle. In both mode 0 and 3, data is clocked in on the rising edge of the clock and out on the falling edge of the clock. Some devices can have a reversed SCK polarity.

Note that some SPI devices can be connected in a daisy-chain manner where the data output of the first SPI device connects to the data input of the second SPI, and so on. This paper does not discuss this topic. Like JTAG, none of the daisy-chained devices has independent access to the bus master.

3 I2C

I2C allows for multiple bus masters and flyby data transfers. Standard I2C can support data transfer up to 100 kbps. There is also Fast-mode I2C at 400 kbps and High speed mode I2C at 3.4 Mbps, Fast-mode plus at 1 Mbps, and Ultra Fast mode at 5 Mbps.

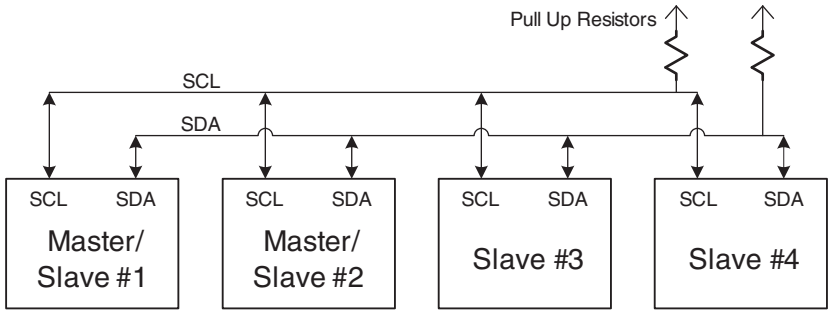
The I2C Bus Master signals are shown below:

- SCL: SPI Bus Clock
- SDA: Data

I2C bus has only two signals: SCL and SDA. They are both bidirectional and open-collector. Pull-up resistors on SCL and SDA are required. SCL is used for clock and wait. SDA is used for address and data since there is only one data line full-duplex cannot be supported.

Figure 3 shows an example I2C system with two Master/Slave devices and two Slave-only devices. The I2C bus does not have device select signals, but selects an I2C device by sending a device select byte. Therefore, all I2C devices must be preprogrammed with a unique I2C bus address before they are used on the I2C bus. The I2C protocol supports up to 127 addressable devices.

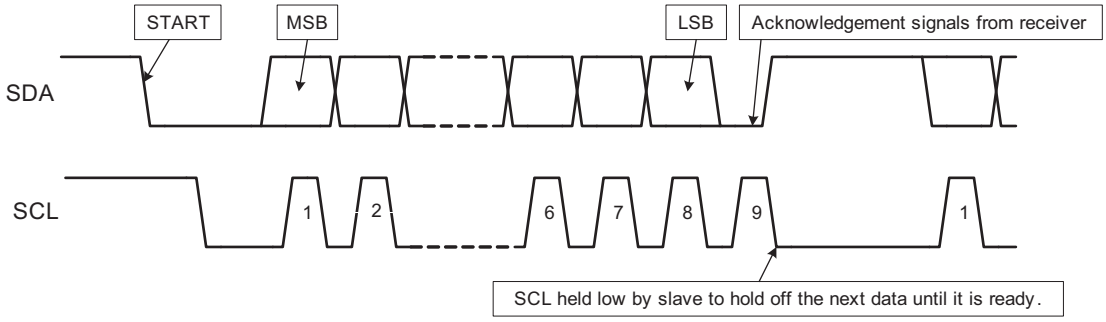
Figure 3. Typical I2C Bus System



3.1 I2C Bus Cycle

The I2C clock signal (SCK) is generated by the current master, but all the slaves can hold the SCK signal LOW until they are ready to allow it to go HIGH. In this way, a rising SCK is delayed until the slowest I2C device is ready. Figure 4 shows the transfer of a byte to a slave: the slave acknowledges the transfer, and then holds down SCL until it is ready for the next byte.

Figure 4. I2C Bus Cycle



Bus arbitration is done with the SDA line. All of the potential bus masters simultaneously try to drive the SDA signal. When any bus master detects that they have driven a '1', but detected a '0' on the SDA signal, it has lost

the bus arbitration. The potential bus master immediately gives give up driving the SDA signal and goes into slave mode in case it is the addressed slave device.

4 JTAG

The JTAG method of connection is defined in IEEE standard #1149; it is also known as “JTAG boundary scan”. JTAG is commonly used for the following applications:

1. Board Assembly Test (verifies the connectivity of device pins to the PCB)
2. Development Tool (in-circuit emulator)
3. System Debug (provides a “back door” into the system)
4. Testing internal device circuitry (not be discussed here)

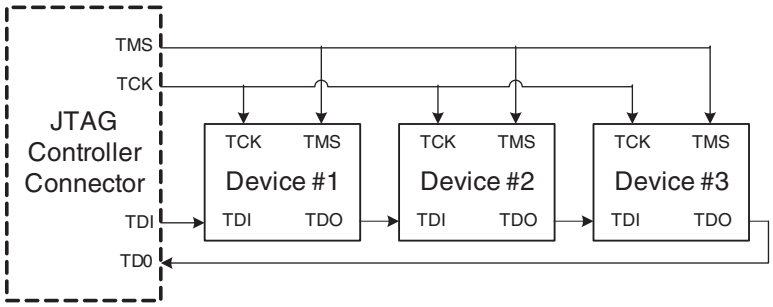
Typically, JTAG is a feature found in relatively high-pin-count devices; it is not typically present in low-pin-count devices. In contrast, I2C and SPI can be found in both high-pin-count devices like microcontrollers and in low- pin-count devices like A/D converters.

JTAG uses four required wires plus an optional reset wire to pass data through devices in a daisy chain. These signals are shown below:

1. TDI (Test Data In) - Daisy Chained
2. TDO (Test Data Out) - Daisy Chained
3. TCK (Test Clock) - Shared
4. TMS (Test Mode Select) - Shared
5. TRST (Test ReSeT) optional

Figure 5 shows an example of a JTAG circuit; there is a JTAG Controller Connection and three JTAG devices.

Figure 5. JTAG Bus System

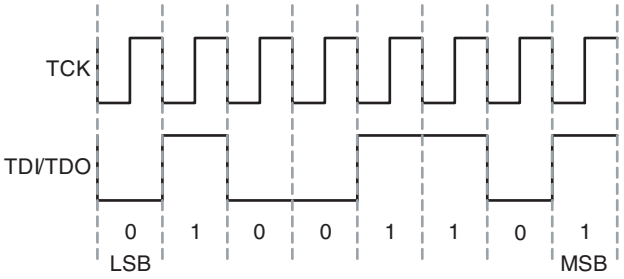


A JTAG Controller is connected at the connector and it drives TCK and TDI into DEVICE #1. The TDI data passes out of the DEVICE #1 TDO pin into DEVICE #2 and so on until the daisy chain is closed back at the JTAG Controller at TDO. The bus signals TMS and TCK control the data transfer.

4.1 JTAG Bus Cycle

Data outputs change on the falling edge of TCK and data is sampled on the rising edge of TCK. TMS and TRST are not shown. Data transfers with JTAG takes more clocks than the other serial buses because in JTAG the data must pass sequentially through all of the devices in the chain. See Figure 6.

Figure 6. JTAG Bus Cycle



5 Summary

Using serial buses instead of parallel buses can cut component costs, space, and power consumption. Complete systems can be built using either SPI or I2C. JTAG is typically used only during product development, manufacturing, or servicing.

Document History Page

Document Title: AN98538 - Serial Buses Comparison: JTAG, SPI, and I2C Document Number: 001-98538				
Rev.	**ECN No.**	**Orig. of Change**	**Submission Date**	**Description of Change**
**			04/13/2007	Initial version
*A	4915272	MSWI	09/10/2015	Updated in Cypress template
*B	5803592	AESATMP8	07/07/2017	Updated logo and Copyright.

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
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Video | Blogs | Training | Components

Technical Support

cypress.com/support

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