Keywords: Serial, SPI, I2C, data rate, interface, comparison, DAC, microcontroller, 3-wire, 2-wire, 1-wire, D/A, Digital to Analog, pullup resistors, noise, master, slave

APPLICATION NOTE 4024

# SPI/I²C Bus Lines Control Multiple Peripherals

Jun 21, 2007

*Abstract: This application note compares two common serial, digital interfaces used by most analog ICs: SPI™ or 3-wire, and I²C or 2-wire. Each serial interface offers advantages or disadvantages for many designs, depending on criteria such like needed data rate, space availability, and noise considerations. This application note discusses the differences between these two serial interfaces and presents examples to demonstrate each interface in detail.*

## Introduction

Although real-world signals will always be analog, today more and more analog ICs communicate through digital interfaces. Serial interfaces communicate between a master, which provides the serial clock, and a slave/peripheral. Today's SPI (3-wire) and I²C (2-wire) ports found on most microcontrollers are popular means for transmitting and receiving data. Microcontrollers thus communicate over several bus lines to control peripherals including analog-to-digital converters (ADCs), digital-to-analog converters (DACs), smart batteries, port expanders, EEPROMs, and temperature sensors. Unlike data sent through a parallel interface, serial data is transmitted with numerous bits in succession, usually over two, three, or four data/timing lines. Although parallel interfaces offer speed, serial interfaces have the advantage of much fewer control and data lines.

## Serial Interface Basics

Serial interfaces are available in three varieties: 3-wire, 2-wire, and single-wire. This article focuses on the 3- and 2-wire interfaces. The Serial Peripheral Interface (SPI), Queued Serial Peripheral Interface (QSPI™), and MICROWIRE™ (or MICROWIRE PLUS™) standards communicate through 3-wire interfaces. The inter-IC (I²C) and System Management Bus (SMBus™) standards communicate over 2-wire interfaces. Both types of serial interfaces have their advantages and disadvantages. See **Table 1** below.

### The 3-Wire Interface

Three-wire interfaces use a chip-select line (active-low CS or active-low SS), a clock line (SCLK), and a data input/master output line (called DIN or MOSI). These interfaces can also include a data output/master input line (called DOUT or MISO), for which they are sometimes called 4-wire interfaces. For simplicity, this article refers to both 3-wire and 4-wire interfaces as 3-wire.

Three-wire interfaces operate at higher clock frequencies and do not require any pullup resistors. SPI/QSPI and MICROWIRE interfaces also feature full-duplex operation (data can be transmitted and

received at the same time), and are less prone to problems in noisy environments. Three-wire interfaces are edge triggered, rather than level triggered.

The main drawback to a 3-wire interface is that it requires a separate active-low CS line for each slave on the bus unless the slaves are configured in a daisy-chain configuration, as shown in **Figure 1**. (Daisy-chaining is discussed in greater detail below.) The 3-wire interface also offers no acknowledgment that data has been correctly transmitted or received. From a software perspective, 3-wire interfaces are simpler and more efficient than 2-wire interfaces for single-master/single-slave applications.
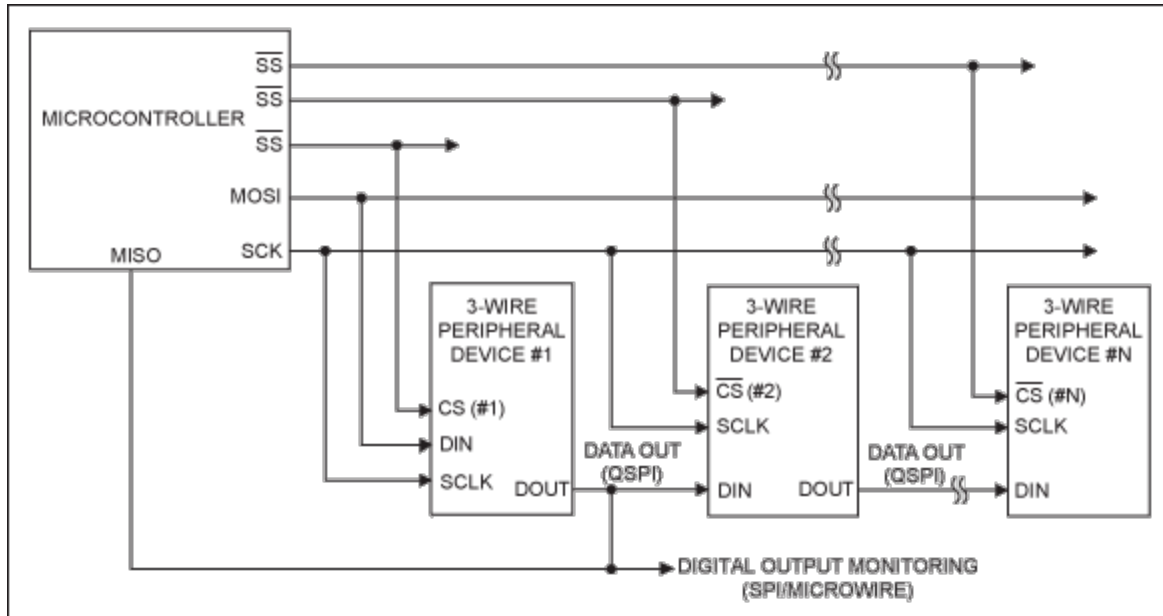


*Figure 1. Three-wire interfaces utilize data input, data output, clock, and chip-select lines.*

## The 2-Wire Interface

Two-wire interfaces use only a data line (SDA or SMBDATA) and a clock line (SCL or SMBCLK). This use of one or two fewer lines is a particularly useful advantage for compact designs such as cell phones and fiber-optic applications. Two-wire interfaces also allow you to connect multiple slaves on the same bus without needing chip-select signals. This design is possible because each slave has its own unique address. Two-wire interfaces also transmit an acknowledge bit after a successful read has been completed. Because 2-wire interfaces have only one data line, they can operate in half-duplex mode only (data can only be transmitted or received on a given cycle, but not both). Two-wire interfaces are level-triggered, which could create problems in a noisy environment if a data bit is incorrectly identified.

**Table 1. Advantages and Disadvantages of 3-/2-Wire Interfaces**

| Interface | Advantages | Disadvantages |
|---|---|---|
| 3-Wire: SPI, QSPI, and MICROWIRE PLUS | 1. Speed<br>2. No pullup resistors required<br>3. Full-duplex operation<br>4. Noise immunity | 1. Larger number of bus line connections<br>2. Individual chip-select lines required to communicate with more than one slave at a time<br>3. No acknowledgment of received data |
| | 1. Fewer bus line connections<br>2. Multiple devices | 1. Speed: SMBus limited to 100kHz; I²C limited to 3.4MHz |

| 2-Wire: I²C and SMBus | share the same bus 3. Received data is acknowledged | 2. Half-duplex operation 3. Open-drain bus lines require pullup resistors 4. Reduced noise immunity |
| --- | --- | --- |

A master and a slave communicate through multiple bus lines over a serial interface. During a write cycle, the master uses its own clock and data signals to transmit data to the slave. During a read cycle, the slave transmits data to the master.

## SPI, QSPI, MICROWIRE Designs

The SPI interface, established by Motorola®, is available on popular processors and microcontrollers such as the MAXQ2000. As noted above, SPI designs require two control lines (active-low CS and SCLK) and two data lines (DIN/SDI and DOUT/SDO). The Motorola SPI/QSPI standards call the DIN/SDI data line the MOSI (master-out, slave-in); the DOUT/SDO data line is the MISO (master-in, slave-out), and the active-low CS line is SS (slave-select). For simplicity and clarity, this article refers to the 3-wire data lines from the slave's perspective: DIN is the slave's data input, and DOUT is the slave's data output. This article also refers to the 3-wire bus lines as active-low CS, SCLK, DIN, and DOUT because the Maxim peripherals use these pin names.

Most SPI interfaces have two configuration bits, clock polarity (CPOL) and clock phase (CPHA), that determine when the slave will sample the data. CPOL determines whether SCLK idles high (CPOL = 1) or low (CPOL = 0) when it is not switching. CPHA determines on which SCLK edge data is shifted in and out. With CPOL = 0, setting CPHA to 0 shifts data into the slave on the SCLK rising edge. Setting CPHA to 1 shifts data into the slave on the SCLK falling edge. The two CPOL and CPHA states allow four different combinations of clock polarity and phase; each setting is incompatible with the other three. Both the master and the slave must be set to the same CPOL and CPHA states to communicate with each other.

In its most basic form, the SPI interface transmits data eight bits (one byte) at a time, though some microcontrollers transmit two or more bytes at a time. The MAXQ2000 microcontroller, for example, can transmit 8 or 16 bits at a time. With CPOL = 0 and CPHA = 0, an active-low CS transition from high to low begins a transmission from the master to a slave. Active-low CS must be held low while SCLK is pulsed high and low for eight complete cycles. DIN data is latched on the rising SCLK edge. The data byte is loaded into the slave after active-low CS transitions low to high. Data is available from the slave's DOUT line on SCLK's falling edge during the same 8-bit cycle. **Figure 2a** illustrates 3-wire SPI timing with CPHA = 1. **Figure 2b** shows 3-wire SPI timing with CPHA = 0.
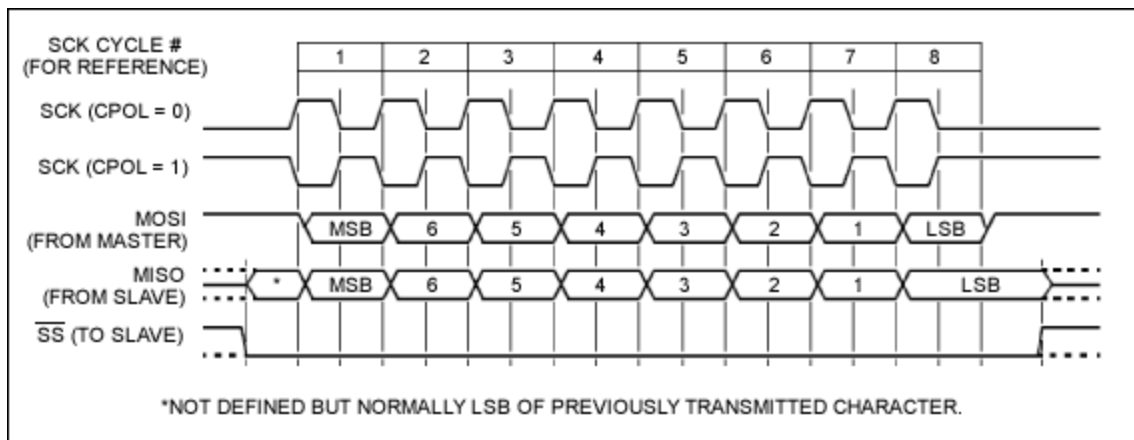


*NOT DEFINED BUT NORMALLY LSB OF PREVIOUSLY TRANSMITTED CHARACTER.

*Figure 2a. 3-Wire interface timing (CPHA = 1). With CPHA = 1 and CPOL = 1, the 3-wire interface*

*clocks data into the peripheral device on the clock's rising edge and data out of the peripheral on the clock's falling edge.*
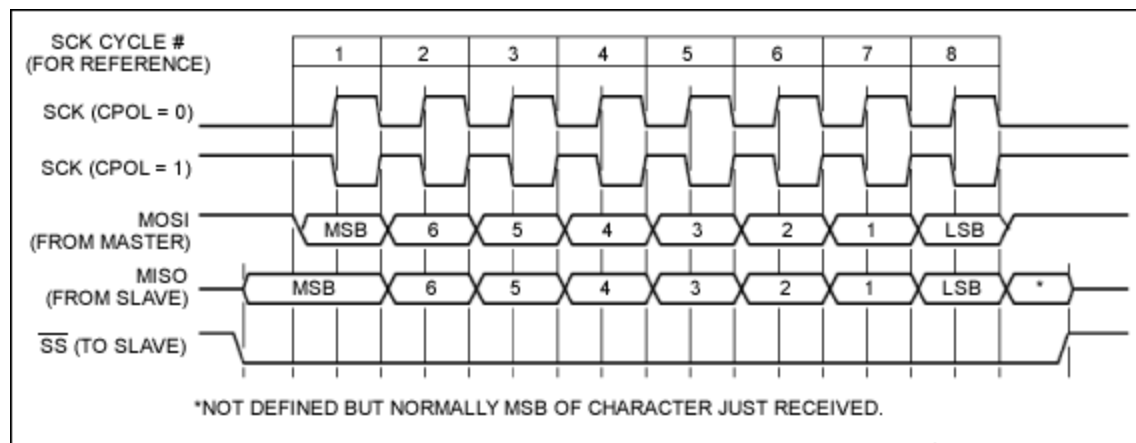


*Figure 2b. 3-Wire interface timing (CPHA = 0). With CHPA = 0 and CPOL = 1, the 3-wire interface clocks data into the peripheral device on the clock's falling edge and data out of the peripheral on the clock's rising edge.*

The active-low CS bus line is used as an enable signal for each slave because every IC on the bus needs its own chip-select line. If four slaves are on the same bus, four chip-select lines are needed to select the appropriate slave. If the slave's active-low CS line is high (inactive), the slave ignores SCLK transitions and holds the DOUT line in a high-impedance state.

Some 3-wire interface peripherals can be programmed with a method called daisy-chaining. Rather than connecting a separate active-low CS line to each peripheral, daisy-chaining allows a single active-low CS and SCLK line to control multiple peripherals connected in series. To daisy-chain peripherals in this manner, the 3-wire interface must include a DOUT line. As shown in Figure 1, Peripheral Device #1's DOUT line serves as the DIN line to Peripheral Device #2, etc.

The SPI standard does not specify a maximum data rate. Instead, the peripherals specify their own maximum data rates, with most in the MHz range. Microcontrollers can accommodate a wide range of SPI speeds. When communicating over an SPI bus, however, a slave is unable to slow down the master or acknowledge a proper data transfer.

The QSPI standard is nearly identical to the SPI standard. In fact, peripherals cannot distinguish between a QSPI bus and an SPI bus. Unlike SPI masters, however, QSPI masters allow data transfers through programmable chip selects. Furthermore, these QSPI masters can transfer between 8 bits and 16 bits at a time, while SPI devices typically transfer only 8 bits. You can configure QSPI devices to transfer up to 16 data words in succession (256 bits maximum). This transfer is handled entirely by the QSPI interface and requires no intervention by the microcontroller. Like SPI, the QSPI standard does not specify a maximum data rate.

The older MICROWIRE standard, established by National Semiconductor, is quite similar to the SPI. However, MICROWIRE has a fixed clock polarity and fixed clock phase (CPOL = 0 and CPHA = 0). The data at DIN is always latched into the slave on SCLK's rising edge. Data shifts from the slave's DOUT pin on SCLK's falling edge. Like SPI, the MICROWIRE standard does not specify a maximum data rate.

# The Inter-Integrated Circuit (I²C) Interface

Unlike the full-duplex, 3-wire serial interface, the I²C standard established by Philips communicates in half-duplex mode over one data line (SDA) and one control line (SCL). The I²C standard defines a simple master/slave bidirectional interface. In this scheme, the microcontroller designates whether it will operate as the master (write mode) or as the slave (receive mode). Each slave has its own unique address, allowing the master to communicate with many different slaves on the same bus without chip-select signals. See **Figure 3**. The number of slaves is limited only by the maximum allowable bus line capacitance (400pF). The I²C protocol is based on a 7-bit or a 10-bit address, though 7-bit addresses are more common. With a 7-bit protocol, you can connect up to 127 different peripherals to the bus. SCL and SDA are open-drain lines that must idle in a high state for proper operation. Connect a 1kΩ or greater pullup resistor to these lines when using a 3V supply, and a 1.6kΩ or greater pullup resistor when using a 5V supply.
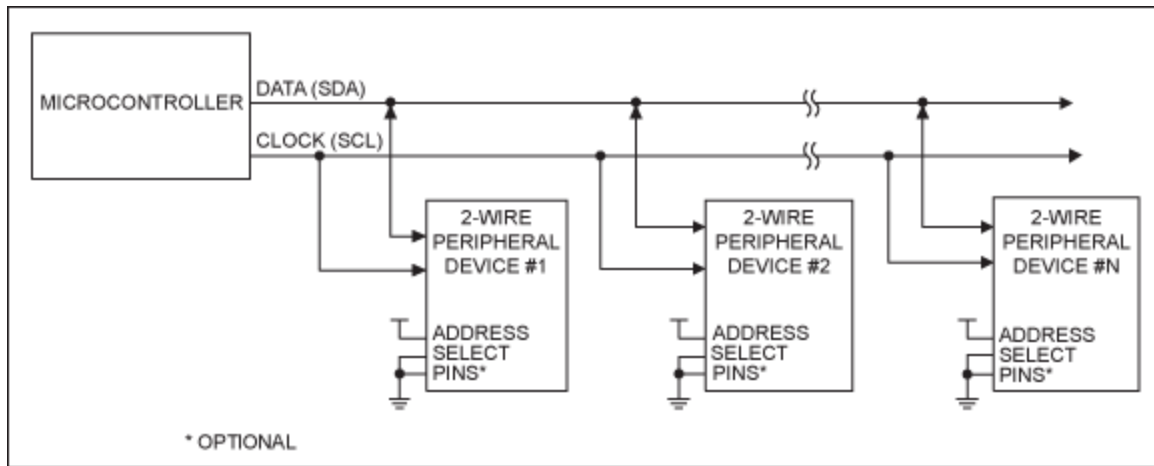


*Figure 3. Two-wire interfaces provide data input/output and clock lines.*

I²C communications begin with a start command, which occurs when SDA transitions high to low with SCL high. See **Figure 4a**. One data bit transfers during each SCL clock cycle; a minimum of nine bits are required to transfer a byte in or out of the slave. The write cycle includes eight data bits followed by an acknowledge (ACK) or not-acknowledge (NACK) signal. See **Figure 4b**. When data is transferred over the I²C bus, it is latched into the slave on SCL's rising edge and read out of the slave on SCL's falling edge. The data on SDA must remain stable during the high period of the SCL clock pulse. A transmission is complete following a stop or repeated start command, at which point SDA transitions low to high with SCL high. Both SDA and SCL remain high when the bus is not busy.
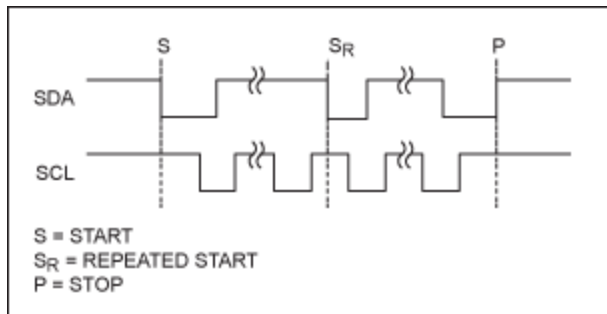


*Figure 4a. Start and stop conditions. The 2-wire interface uses start, repeated start, and stop commands to transfer data between the master and the slave.*
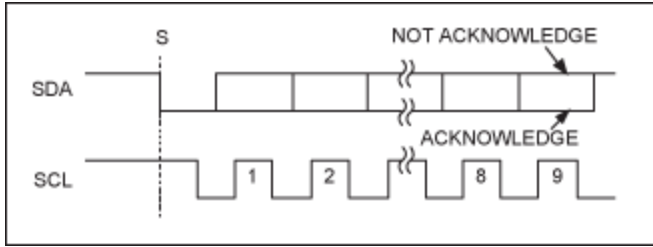
*Figure 4b. I²C acknowledge bits. The 2-wire interface pulls the SDA line low when data is acknowledged.*

An I²C write cycle begins with a start command, followed by writing the 7-bit slave address and an eighth bit that signals a write or read command. Set the eighth bit low for a write command or high for a read command. The master releases the bus line after the eighth clock cycle. The slave holds the SDA line low on the ninth clock cycle, if the slave acknowledges a proper transmission. The slave releases the SDA line (which is then held high by the pullup resistor) if the slave does not acknowledge a proper write command.

The master then writes an 8-bit command byte, which is followed by a second ACK/NACK bit. Next, the master writes an 8-bit data byte, which is followed by a third ACK/NACK bit. The data byte(s)' final acknowledge bit completes the read/write cycle, and the peripheral's outputs are updated. **Figure 5a** illustrates an example of a write cycle.

An I²C read cycle begins with a start command, followed by writing the slave address with the eighth bit pulled high to signal a read command. Following an ACK/NACK bit, the master writes the command byte to access a new slave register. After the second ACK/NACK bit, the master rewrites the slave address. Then following the third ACK/NACK bit, the slave takes control of the bus and writes out eight data bits at a time. See **Figure 5b**. When reading from the same slave register as the previous reads, the master needs only write a slave's address before reading the data from that slave.
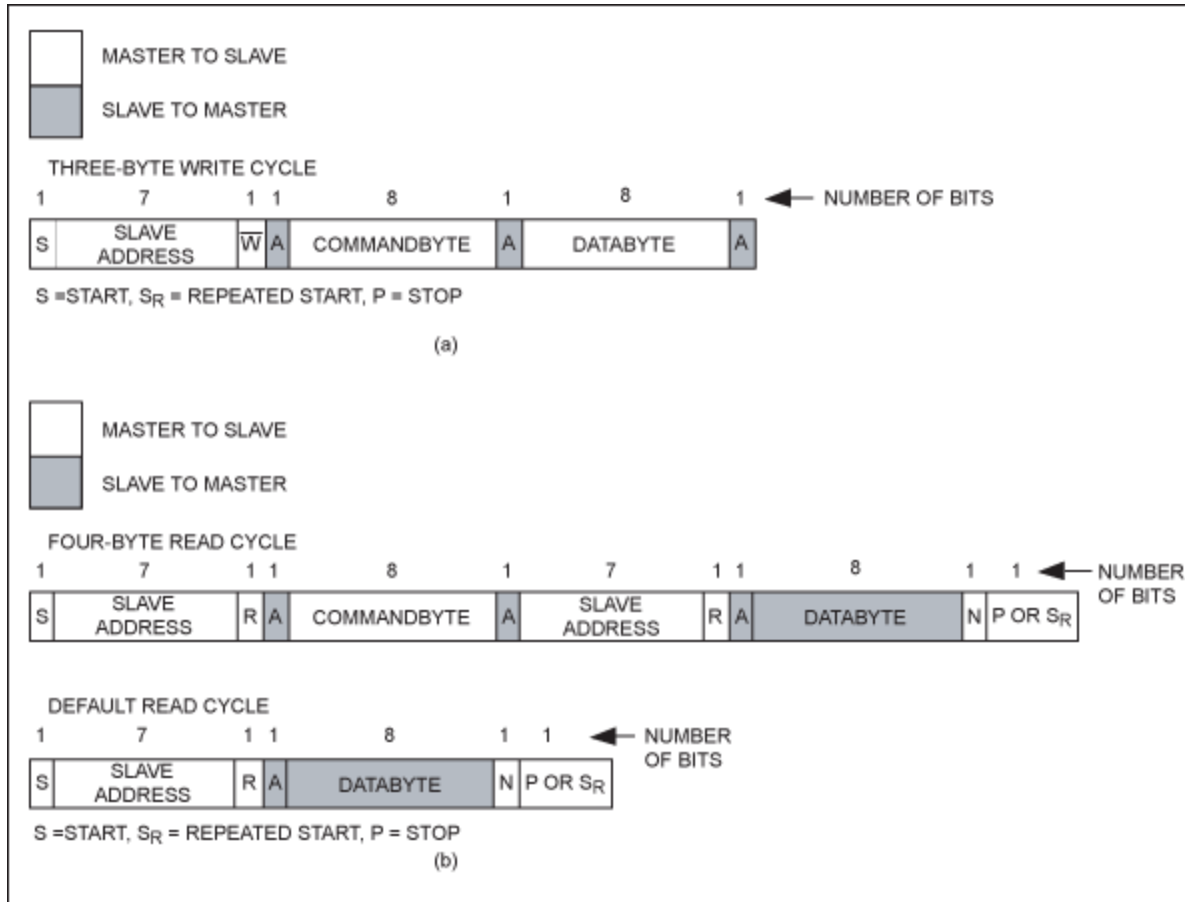
*Figure 5. The 2-wire interface transfers data eight bits at a time. Figure 5a is an I²C write-cycle example. Figure 5b shows I²C read-cycle examples.*

The I²C interface supports slow (up to 100kHz), fast (up to 400kHz), and high-speed (up to 3.4MHz) protocols. I²C recognizes high and low signals based on CMOS voltage levels: a low signal is less than 0.3 x supply voltage; a high signal is greater than 0.7 x supply voltage.

## System Management Bus (SMBus)

Intel® established the SMBus standard for low-speed communication, and the SMBus interface is similar to I²C. Like I²C, SMBus uses a 2-wire interface that includes a data line (SMBDATA) and a clock line (SMBCLK). The SMBCLK and SMBDATA lines also require pullup resistors. Use a 8.5kΩ or greater pullup resistor with a 3V supply, and 14kΩ or greater pullup resistor with a 5V supply. SMBus operates from a 3V to 5V supply voltage and recognizes a high signal above 2.1V and a low signal below 0.8V.

Timeout and maximum/minimum clock speed are the most significant differences between the I²C and SMBus interfaces. The I²C bus functions down to DC and does not timeout due to bus inactivity. SMBus interfaces can, however, timeout. Timeout occurs when a slave device resets its interface after the clock signal goes low for longer than the timeout period (35ms maximum). The SMBus timeout period dictates a minimum speed of 19kHz for the clock. SMBCLK must be set between 10kHz and 100kHz for proper communication. Either a master or a slave connected to an I²C bus, however, can hold the clock low as long as necessary to process data.

# Peripheral Examples

Microcontrollers often communicate with their peripherals through a serial interface. Using a 3-wire or 2-wire interface, the microcontroller reads and writes to the internal registers of the peripheral devices. The peripherals then bias and control various analog and digital outputs. For example, peripheral devices will: program a battery's charging current and voltage; use a temperature sensor to control a fan; and set a DAC's analog output voltage as well as the bias conditions of various circuits.

**Figure 6** shows a microcontroller communicating with an 8-bit DAC (the MAX5115) through a 2-wire interface. Because this DAC includes four address-select pins that produce 16 unique slave addresses, you can connect up to 16 DACs in parallel. The same two bus lines could also set the bias conditions of an SMBus temperature sensor/fan controller (the MAX6641), because the MAX6641 has a different slave address. This fan controller regulates a MOSFET's gate voltage to turn the fan on and off.
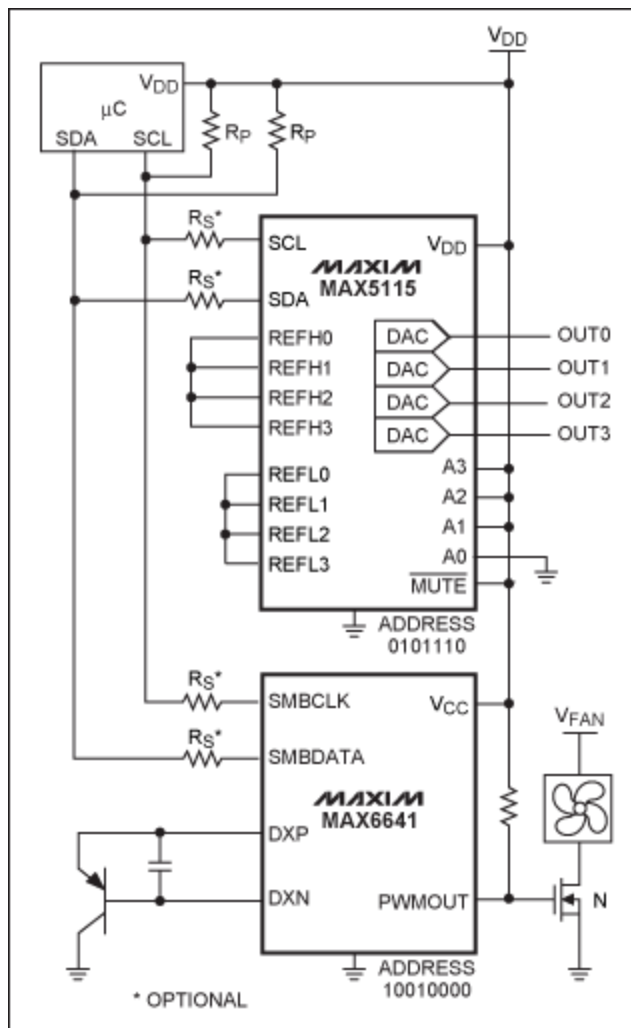


*Figure 6. Because this microcontroller uses an I²C interface, only two bus lines are required to communicate with numerous peripherals, like this DAC and a temperature sensor, connected in parallel.*

While a 3-wire interface requires individual chip-select lines for communication between the microcontroller and multiple ICs connected in parallel, the simpler 2-wire interface uses the same clock and data lines to communicate with each device on the bus. You can place multiple ICs in parallel by setting a different slave address for each peripheral. Most I²C peripherals include address-select pins that allow you to set each peripheral to a different slave address. Previously, the number of slave

addresses with which a peripheral could identify itself was limited to a power of two. If, for example, a peripheral had two address-select pins, it could identify itself on the bus with four unique slave addresses.

New designs provide more slave addresses from fewer address-select pins. The MAX7319 input/output port expander, for example, can be programmed to 16 unique slave addresses from only two address-select pins (AD2 and AD0). These pins can be tied to GND, the supply voltage ($V_{CC}$), SDA, or SCL. **Table 2** shows the 16 slave addresses available. Although bits A6, A5, and A4 must be set to 110, bits A3 through A0 can be programmed by the four different settings for AD2 and AD0.

**Table 2. The MAX7319 Deciphers 16 Individual Addresses with Only Two Address-Select Lines (AD2 and AD0).**

| Pin Connection | | Device Address | | | | | | |
|---|---|---|---|---|---|---|---|---|
| AD2 | AD0 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| SCL | GND | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| SCL | $V_{CC}$ | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| SCL | SCL | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| SCL | SDA | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| SDA | GND | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| SDA | $V_{CC}$ | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| SDA | SCL | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| SDA | SDA | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| **GND** | **GND** | **1** | **1** | **0** | **1** | **0** | **0** | **0** |
| **GND** | **$V_{CC}$** | **1** | **1** | **0** | **1** | **0** | **0** | **1** |
| GND | SCL | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| GND | SDA | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| **$V_{CC}$** | **GND** | **1** | **1** | **0** | **1** | **1** | **0** | **0** |
| **$V_{CC}$** | **$V_{CC}$** | **1** | **1** | **0** | **1** | **1** | **0** | **1** |
| $V_{CC}$ | SCL | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| $V_{CC}$ | SDA | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

## Future Advances

Today's 3-wire interfaces serve different needs than 2-wire interfaces and each offers specific advantages. It is unlikely that any one interface will completely displace the other in the future. I²C devices are advancing more rapidly, as they are starting to incorporate SMBus features like timeout resets, which can be turned off when necessary. New I²C slave addresses are ten bits long, rather than only seven, thus providing users with even more flexibility.

Both 3-wire and 2-wire interfaces will co-exist, but I²C will likely gain market share as more microcontrollers will support 2-wire interfaces. I²C's ease of use and its fewer bus lines will probably drive its growth beyond that of SPI.

Additional information can be found in application notes 3967, "Selecting a Serial Bus," and 3438, "Serial Digital Data Networks."

A similar article was published on the *Portable Design* website in March 2006.

| Related Parts | | |
|---|---|---|
| MAX5115 | Nonvolatile, Quad, 8-Bit DACs with 2-Wire Serial Interface | Free Samples |
| MAX6641 | SMBus-Compatible Temperature Monitor with Automatic PWM Fan-Speed Controller | Free Samples |
| MAX7319 | I²C Port Expander with Eight Inputs and Maskable Transition Detection | Free Samples |
| MAXQ2000 | Low-Power LCD Microcontroller | Free Samples |

**More Information**
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 4024: http://www.maximintegrated.com/an4024
APPLICATION NOTE 4024, AN4024, AN 4024, APP4024, Appnote4024, Appnote 4024
Copyright © by Maxim Integrated Products
Additional Legal Notices: http://www.maximintegrated.com/legal