Introduction to I2C

By David Kalinsky and Roee Kalinsky
Embedded Systems Design
(07/31/01, 09:35:16 AM EDT)

An Inter-IC bus is often used to communicate across circuit-board distances. Here's a primer on the protocol.

At the low end of the spectrum of communication options for "inside the box" communication is I^2C ("eye-squared-see"). The name I^2C is shorthand for a standard Inter-IC (integrated circuit) bus.

I^2C provides good support for communication with various slow, on-board peripheral devices that are accessed intermittently, while being extremely modest in its hardware resource needs. It is a simple, low-bandwidth, short-distance protocol. Most available I^2C devices operate at speeds up to 400Kbps, with some venturing up into the low megahertz range. I^2C is easy to use to link multiple devices together since it has a built-in addressing scheme.

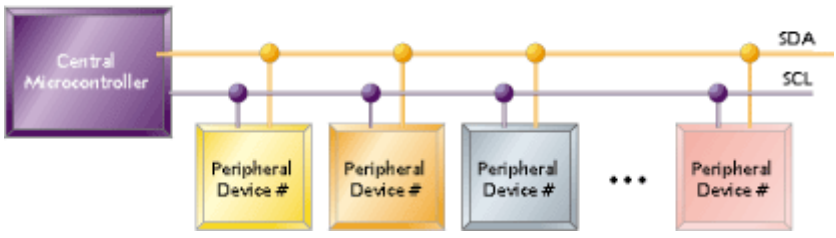
Philips originally developed I^2C for communication between devices inside of a TV set. Examples of simple I^2C-compatible devices found in embedded systems include EEPROMs, thermal sensors, and real-time clocks. I^2C is also used as a control interface to signal processing devices that have separate, application-specific data interfaces. For instance, it's commonly used in multimedia applications, where typical devices include RF tuners, video decoders and encoders, and audio processors. In all, Philips, National Semiconductor, Xicor, Siemens, and other manufacturers offer hundreds of I^2C-compatible devices.

Inside the box

I^2C is appropriate for interfacing to devices on a single board, and can be stretched across multiple boards inside a closed system, but not much further. An example is a host CPU on a main embedded board using I^2C to communicate with user interface devices located on a separate front panel board. A second example is SDRAM DIMMs, which can feature an I^2C EEPROM containing parameters needed to correctly configure a memory controller for that module.

I^2C is a two-wire serial bus, as shown in Figure 1. There's no need for chip select or arbitration logic, making it cheap and simple to implement in hardware.

Figure 1: I^2C has two lines in total



The two I^2C signals are serial data (SDA) and serial clock (SCL). Together, these signals make it possible to support serial transmission of 8-bit bytes of data-7-bit device addresses plus control bits-over the two-wire serial bus. The device that initiates a transaction on the I^2C bus is termed the master. The master normally controls the clock signal. A device being addressed by the master is called a slave.

In a bind, an I2C slave can hold off the master in the middle of a transaction using what's called clock stretching (the slave keeps SCL pulled low until it's ready to continue). Most I2C slave devices don't use this feature, but every master should support it.

The I2C protocol supports multiple masters, but most system designs include only one. There may be one or more slaves on the bus. Both masters and slaves can receive and transmit data bytes.

Each I2C-compatible hardware slave device comes with a predefined device address, the lower bits of which may be configurable at the board level. The master transmits the device address of the intended slave at the beginning of every transaction. Each slave is responsible for monitoring the bus and responding only to its own address. This addressing scheme limits the number of identical slave devices that can exist on an I2C bus without contention, with the limit set by the number of user-configurable address bits (typically two bits, allowing up to four identical devices).

Communication

As you can see in Figure 2, the master begins the communication by issuing the start condition (S). The master continues by sending a unique 7-bit slave device address, with the most significant bit (MSB) first. The eighth bit after the start, read/not-write (), specifies whether the slave is now to receive (0) or to transmit (1). This is followed by an ACK bit issued by the receiver, acknowledging receipt of the previous byte. Then the transmitter (slave or master, as indicated by the bit) transmits a byte of data starting with the MSB. At the end of the byte, the receiver (whether master or slave) issues a new ACK bit. This 9-bit pattern is repeated if more bytes need to be transmitted.

Figure 2: I²C communication

In a write transaction (slave receiving), when the master is done transmitting all of the data bytes it wants to send, it monitors the last ACK and then issues the stop condition (P). In a read transaction (slave transmitting), the master does not acknowledge the final byte it receives. This tells the slave that its transmission is done. The master then issues the stop condition.

A simple bus

As we've seen, the I2C signaling protocol provides device addressing, a read/write flag, and a simple acknowledgement mechanism. There are a few more elements to the I2C protocol, such as general call (broadcast) and 10-bit extended addressing. Beyond that, each device defines its own command interface or address-indexing scheme.

Standard I2C devices operate up to 100Kbps, while fast-mode devices operate at up to 400Kbps. A 1998 revision of the I2C specification (v. 2.0) added a high-speed mode running at up to 3.4Mbps. Most of the I2C devices available today support 400Kbps operation. Higher-speed operation may allow I2C to keep up with the rising demand for bandwidth in multimedia and other applications.

Most often, the I2C master is the CPU or microcontroller in the system. Some microcontrollers

even feature hardware to implement the I^2C protocol. You can also build an all-software implementation using a pair of general-purpose I/O pins (single master implementations only).

Since the I^2C master controls transaction timing, the bus protocol doesn't impose any real-time constraints on the CPU beyond those of the application. (This is in contrast with other serial buses that are timeslot-based and, therefore, take their service overhead even when no real communication is taking place.)

The elegance of I^2C

I^2C offers good support for communication with on-board devices that are accessed on an occasional basis. I^2C's competitive advantage over other low-speed short-distance communication schemes is that its cost and complexity don't scale up with the number of devices on the bus. On the other hand, the complexity of the supporting I^2C software components can be significantly higher than that of several competing schemes (SPI and MicroWire, to name two) in a very simple configuration. With its built-in addressing scheme and straightforward means to transfer strings of bytes, I^2C is an elegant, minimalist solution for modest, "inside the box" communication needs.

David Kalinsky is director of customer education at OSE Systems. Earlier in his career, he was involved in the design of many medical and aerospace systems. David holds a PhD in nuclear physics from Yale and can be reached by e-mail at david@enea.com.

Roee Kalinsky is a senior design engineer at QuantumThink Group. He has designed numerous embedded systems including networking, multimedia, and hand-held products. Roee holds a BSEE from the University of California at San Diego. He can be reached by e-mail at roeek@qthink.com.

Resources

For more recent information on the Inter IC bus, go to Inter IC (I2C) bus design and test for embedded systems

1. Willey, H. Michael. "One Cheap Network Topology," *Embedded Systems Programming,* January 2001, p. 59.

2. Sarns, Steven and Jack Woehr. "Exploring I^2C," Embedded Systems Programming, September 1991, p. 46.

3. Philips' web site for I^2C: www-us.semiconductors.philips.com/I2C/

Return to August 2001 Table of Contents