ANALOG DEVICES

AN-686
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106 • Tel: 781/329-4700 • Fax: 781/326-8703 • www.analog.com

Implementing an I²C® Reset

By Jim Greene

The I²C bus is a high integrity, robust serial bus used for control purposes in many systems. The primary components that make up a system are at least one master and one slave. Under normal conditions, everything works fine; however, it is the abnormal conditions that generate problems. Two questions present themselves when a problem arises: Is the problem device or system related, or some combination of both? What, if anything, can be done about it?

Hard device failures are relatively easy to isolate. Perhaps a function does not work, proper power cycling does not resolve the issue, pins are stuck high or low, and so on. System related problems sometimes disguise themselves as device failures, or worse, are intermittent. It is the latter area that this application note examines because it represents the majority of bus fault conditions.

Perhaps a brief description of the I²C bus is in order. The I²C (inter integrated circuit) bus was developed and patented by Philips. It allows devices to communicate over an open-drain (or open-collector) 2-wire serial bus. Interfacing is simple; serial data (SDA) and serial clock (SCL) are the only signals that traverse the circuit board. Due to the low speed (literally dc to 400 kB/s) problems associated with routing, transmission line effects and matching are nonexistent. The limiting factor is bus capacitance, which is limited to 400 pF.

The following terms are used to describe the I²C bus:

Master—the device that initiates a message, and defines the direction of the I²C bus. The master is also responsible for the generation of the clock (SCL). (9 clocks per byte: 8 for data and 1 for the acknowledge.)

Slave—a device with an address that is addressed by a master.

Start—a bus condition in which the SCL line is high and the SDA line transitions from a high to a low. It is the first operation on the bus and is always followed by an address. The least significant bit determines the direction of the bus. A high tells the slave that the bus will read, while an LSB = 0 identifies a write to the specified address.

Stop—the condition opposite Start, under which the SCL line is high while the SDA line goes from a low to a high state. It is the only method of ending a transmission after the reception of a byte.

Byte Width—all bytes are 8 bits wide, with no exceptions.

Message Length—technically there is no maximum length for a message; a minimum message consists of 2 bytes (an address and a data byte).

Wait State—this condition is rarely used, but is worth understanding. Once the SCL line is low, a device may continue to hold it low to identify a wait state. The wait state permits slow devices to not lose synchronization with the transmitting device. An example is writing many bytes to an E²PROM; another is a processor holding off data from a slave to handle an interrupt.

Acknowledge—The "ACK" is the condition under which the master generates a 9th clock pulse on the SCL line (for each byte) while the receiving device pulls the SDA line low in order to signify that the last byte was received. A "NAK" is only generated by the master; it signals the slave that no additional data need be sent. A NAK is used prior to a STOP to prevent the slave from driving the bus with additional data when the master is about to terminate the communication.

Frequently the master, which is usually a microcontroller or a gate array, will be interrupted in the middle of its communication with an I²C slave and, upon return, find a stuck bus. Initially this looks like a device problem, but it is not. The slave is still waiting to send the remainder of the data requested by the master. The problem is that the master has forgotten where it was when it was interrupted or reset. An extraneous reset on the processor will generally create this condition, especially if the processor cannot save its status. At this point, the slave will have put the next bit out on the SDA line (because the SCL line may have dropped to a low on reset) and awaits the next clock on SCL. Of course the processor does not send it, and as a result this slave just waits and waits. If the bit the slave puts on the SDA line is a 0, the newly awakened processor sees what appears to be a hung bus. The bus is in a nonoperational mode; however, it is not due to the slave. It is the processor's fault for not finishing the message it started. Generating graceful resets is not within the scope of this application note.

What should you do? The slave must be permitted to finish sending this last byte or be reset externally.

Solution 1: Clocking Through the Problem

The first solution (letting the slave finish) requires no additional hardware because it is implemented in software. Note that while this method is very effective, it may not be possible to clear a hung bus on every manufacturer's device all the time. (The design of the I^2C state machine will determine the effectiveness of the clocking approach.)

The method is quite simple. It is the master's job to recover the bus and restore control to the main program. When the master detects the SDA line stuck in the low state, it merely needs to send some additional clocks and generate a STOP condition. How many clocks will be needed? The number will vary with the number of bits that remain to be sent by the slave. The maximum would be 9. This number is derived from the worst-case scenario, the case where the processor was reset just after sending an ACK to the slave. Now the slave is ready to send 8 data bits and receive 1 ACK (or NAK in the case of a bus recovery).

The procedure is as follows:

1) Master tries to assert a Logic 1 on the SDA line
2) Master still sees a Logic 0 and then generates a clock pulse on SCL (1-0-1 transition)
3) Master examines SDA. If SDA = 0, go to Step 2; if SDA = 1, go to Step 4
4) Generate a STOP condition

Note that this process may need to be repeated because the cleared SDA line may have been cleared for the next bit, which was a 1. There may be some concern about the effect this additional clocking and STOPping has on other peripherals. There is no adverse effect; other slaves are not paying attention due to the fact that they have not been addressed. Only the slave that had the interrupted message will respond to the clocks.

This procedure is useful in the system code to help restore the bus in the event that an SDA = 0 bus fault is encountered, regardless of the reason.

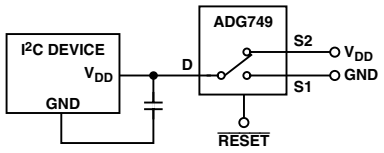
Solution 2: Adding a Reset Pin to an I^2C Slave

Another method will reset the I^2C slave. One function never seen on an I^2C slave is a reset pin. To remedy this type of problem, a reset function is added via additional hardware: an analog switch. The analog switch needs several attributes to perform the reset function properly. The ADG749 fills the requirements:

- Small package: the SC70 requires less than 5 square mm of board space
- SPDT switch with break-before-make action
- Very low on resistance: 3.5 Ω at 5 V and 4.5 Ω at 3 V
- Excellent on resistance flatness (allows repeatable resets in digital devices)
- At 1 μA of supply current, the power budget is not affected

The diagram below shows how the ADG749 can provide a reset to an I^2C slave device. When a reset to the slave must occur, the processor sends a logic low to the control pin on the analog switch now labeled $\overline{\text{RESET}}$ (see diagram). The low going reset pulse must be of sufficient width to permit the switch to discharge the decoupling capacitors and internal circuitry. The ADG749 is capable of generating a reset to many I^2C devices with their associated decoupling capacitors. Testing has shown that a 15 μs reset pulse will switch the V_{DD} line of 2 slaves and 1 μF of capacitance to within 0.1 V of ground in <10 μs. The turn on time is equally impressive at <5 μs, which means that the I^2C state machine will reset itself on power up.

With an operational voltage range of 1.8 V to 5.5 V, the ADG749 permits literally any I^2C device to be reset by the processor. Analog Devices has other analog switches if level translation functions are required.



NOTE: SWITCH SHOWN WITH $\overline{\text{RESET}}$ = LOGIC 1

Figure 1. Simple Interface Resets I^2C Bus