Keywords: interface, DAC, data converter, I²C, SPI, GPIO, SPI error, noise

APPLICATION NOTE 5139

Interface, Happy or Sad Face?

By: Bill Laumeister, Strategic Applications Engineer
Apr 19, 2012

Abstract: How can an interface change a happy face to a sad face? Engineers have happy faces when an interface works properly. Sad faces indicate failure somewhere. Because interfaces between microprocessors and ICs are simple—even easy—they are often ignored until interface failure causes sad faces all around. In this article, we discuss a common SPI error that can be almost impossible to find in a large system. Links to interface tutorial information are provided for complete information. Noise as a system issue and ICs to minimize its effects are also described.

A similar version of this article was published in the December 2011 issue of *Semiconductor Network*.

Introduction

Common interfaces between system microprocessors and integrated circuits (ICs) include SPI, I²C, and GPIO. This article discusses a common SPI error that can be almost impossible to find in a large system. Noise as a system issue is discussed and ICs that minimize its effects are mentioned. The trade-offs in an I²C system implementation are explained, and the basic operation of GPIO communication from a microprocessor is described. Links to tutorial information are provided for further reading.

How Can an Interface Make a Happy Face a Sad Face?

Common interfaces for ICs include SPI, I²C, and GPIO. Determining how an IC will interface to the microcontroller or CPU is quite important for any successful design. Yet interfaces between microprocessors and ICs are simple, even easy, and thus all too often ignored in many designs. The time and effort seemingly saved early on in a project can make design engineers happy...for the moment. This situation brings to mind the happy face or ubiquitous cute "smiley face"emoticon that has become a part of everyday life[1] and is our metaphor here.

The engineer's "sad"face usually emerges late in the project when Murphy's law[2] that "anything that can go wrong, will go wrong at the worst possible moment"looms over a designer's bent shoulders. This article tells a sad SPI interface story, surprisingly not that uncommon. It is the story of how the operation of an interface can make a happy face sad.

"My Computer Hates Me and My SPI Bus Randomly Does What It Wants, When It Wants."

Basic SPI Operation

A serial peripheral interface bus, or SPI (pronounced "spy") bus named by Motorola, is a synchronous serial data bus that operates in full-duplex mode.[3] Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave-select (i.e., chip-select) lines. Sometimes an SPI interface is called a "three-wire,"(i.e., no read back from the slave) or "four-wire" serial bus.[4] An SPI interface with unidirectional signaling offers easy galvanic isolation to reduce ground loops in the plant. Such an SPI interface is called unidirectional because each of the four wires only passes information in one direction. Galvanic isolation can be accomplished with optical, capacitor, or transformer coupling.

The first mistake that some designers make is to assume that an SPI bus has intelligence. It does not. **Figure 1** explains how a SPI slave works; for receiving data it is just a serial-in, parallel-out shift register.

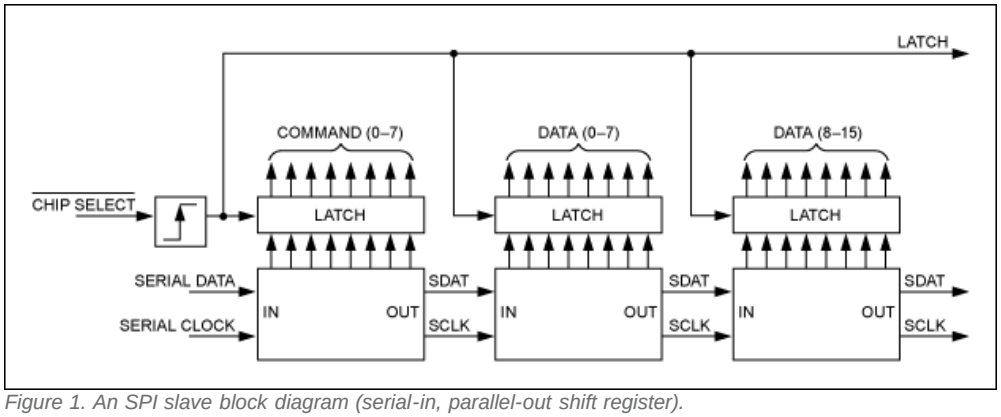

Figure 1. An SPI slave block diagram (serial-in, parallel-out shift register).

We cannot overemphasize that an SPI bus is simple, even dumb without any error checking. It is essentially a string of registers written serially. When chip select is raised, the registers unload a parallel word into the chip. The SPI bits have no protocol or meaning on their own. There is not even a set number of bits in a message; the number of bits varies depending on what the slave needs. To complicate matters in a design, multiple devices of different types and from various manufacturers can be daisy-chained together. Of course, they must all have the same clock and data relationship (one out of the four possible formats[3]).

Now picture three devices in a daisy chain. Chip 1 first sees chip 3's commands go through, followed by chip 2's commands. Only after its own command is shifted in does the chip-select line go high. Then, and only then, does any command make sense to any of the individual chips.

As a system gets larger and busier, the system may send partial SPI commands, which is fatal. The fatality occurs when bits are loaded into the beginning of the SPI shift register. Then before the bits are in the proper position in the shift register, the chip select goes high. Whatever was in the register is parallel shifted into the chip. It is, therefore, *mandatory that interrupts are masked during the SPI command* to ensure that a partial SPI command is not sent.

To illustrate the proper workings of an SPI interface, we use a MAX5312 12-bit, digital-to-analog converter (DAC).

Table 1 shows the most significant bit (MSB) loaded first and the least significant bit (LSB) last. Figure 2 shows the 4-bit command loaded first. Note that only six of the possible 16 control words are used and there is a warning not to use any unlisted commands. If the unlisted commands are used, the device may or may not react. In any case, there is no assurance that anything good will happen. In fact, the probability is that something bad will happen, so do not use any unspecified commands.[5] The MAX5312 also has Schmitt trigger inputs on the digital pins to minimize noise effects.

Table 1. Serial-Data Format

Control Bits				Data Bits											
MSB															LSB
C3	C2	C1	C0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Table 2. Serial-Interface Programming Commands

Control Bits*				Input Data	Function
C3	C2	C1	C0	D11–D0	
0	0	0	0	XXXXXXXXXXXX	No operation; command is ignored
0	0	1	0	12-bit DAC data	Load input register from shift register; DAC output unchanged.
0	1	0	0	12-bit DAC data	Load input and DAC registers from shift register; DAC ouput updated.
0	1	1	0	XXXXXXXXXXXX	Load DAC register from input register; DAC output updated; input register unchanged.
1	0	0	0	XXXXXXXXXXXX	Enter shutdown; input and DAC registers unchanged.
1	1	0	0	XXXXXXXXXXXX	Exit shutdown; input and DAC registers unchanged.

X = Don't care.

*All unlisted commands are reserved commands. Do not use.

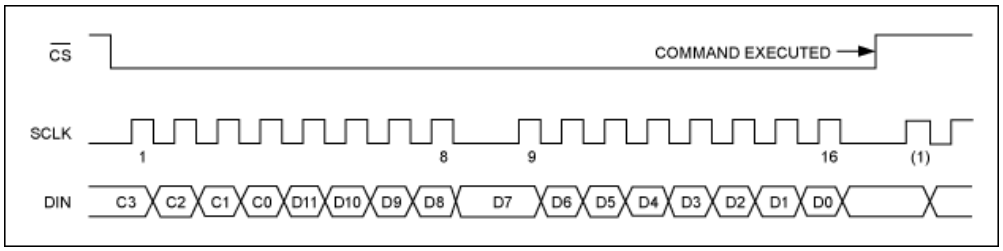

Figure 2. SPI interface example from the MAX5312 data sheet.

Mysterious Noise

Sometimes a customer is concerned that strange things are happening in their system. Many times they blame noise as the cause. That is not necessarily so.

We think that noise is unlikely. Here is why. If an SPI transmission was not happening, the chip select (active-low CS) would have to stay at zero (no noise) while the clock and data chattered through the 30% and 70% voltage data thresholds. If chip select was low during an SPI message, noise on data and clock would possibly intrude. Experience shows that, if the noise is really that bad, nothing ever gets through or the wanted data is trashed with the DAC output jumping wildly. Having one of three lines clean (i.e., active-low CS) and huge noise on the clock and data would be very unusual.

There is an old adage that tells us to look for the simplest, most probable answer first when troubleshooting an issue. So in our example, it is far more likely that this "strange activity" is a partial or incomplete SPI message. But, where did it come from? During system development, one typically works with a small system or subset of the final large system. With the processor not very busy, interrupts seldom truncate an SPI command. As the system grows and the processor increases the number of commands to more devices, the probability of an interrupt truncating an SPI command increases. This is exactly why *it is mandatory to mask interrupts during the SPI transmission*.

How Can Noise Affect ICs?

Huge noise sources can overwhelm a digital transmission[6, 7]. Consequently, the largest potential circuit errors, like lightning strikes, must be addressed at the system level, while smaller issues can be handled at the chip level.

An example of a large system error that cannot be fixed by a normal IC is an unstable power supply. A 3V supply that chatters quickly between 1V and 5V will cause failures that an IC cannot manage. This is, admittedly, an extreme example, but not that unique in the world around us. Every day there are powerline glitches caused by large motors, arc welders, lightning-induced surges, and power substation load switching. Large power transmission grids must react to changing demand by adding and removing power-generation equipment during the day. This equipment switching can cause voltage changes, which can ripple throughout the grid. Other potentially large noise sources are radio frequency interference (RFI), electromagnetic interference (EMI), and electrostatic discharge (ESD). Many regions experience almost daily thunder storms in the summer and low humidity with constant ESD generation in the winter. All of these large glitches and errors must be anticipated and solved at the system level by safety grounds, shielding, and surge suppression.

The small errors that are managed at the chip level using proper power and ground star connections; separate PCB power and ground planes for clean (analog) and dirty (digital) voltages;[6] and power decoupling with series resistors, inductors and ferrite beads, and parallel capacitance.[7]

These major noise sources can disrupt a system. Maxim has, and there are many devices available to protect power, data, and interfaces. In extreme cases, one would also need to provide a watchdog timer circuit to restart the system microprocessor.

I²C in a Sea of Interfaces

There are a plethora of interface systems: RS-232, RS-422/RS-485, USB, Ethernet IEEE® 802 and its alphabet soup, IO-Link®, LIN bus, 1-Wire®, I²C, SMBus, SPI, MICROWIRE®/MICROWIRE PLUS™, M-Bus (EN1434), and CAN (ISO11898), to mention just a few. (Many are trademarked by their originators.) There are so many competing systems, protocols, standards, and partial standards that it is difficult to choose.[8]

The inter-integrated circuit (I²C) is a multimaster, serial, single-ended communication bus used to attach low-speed peripherals to

television sets, consumer equipment, cell phones, or other electronics devices. It is a good choice for communications in a small area, chassis, or PC board where galvanic isolation is not an issue. It has also been expanded for use in a cabled system.

As with most systems, an I²C interface has its trade-offs. Its basic limitations are resistive pullup and a maximum capacitance of 400pF. Ideally, the bus would have lower power consumption, especially for battery-operated equipment. That lower power would also translate to slower bus speeds. But some applications require faster communications. Typical I²C clock speeds are between 100kHz and 3.4MHz; for faster speeds SPI clocks can be in the tens of Megahertz region. I²C is also called a "two-wire" serial bus,[9] and an I²C two-line digital interface is ideal for slower systems. I²C uses bidirectional signaling; a single data wire will transmit data from the master to the slave and acknowledge from the slave to the master. The system can have multiple masters, and since the clock comes from the currently active master, the clock can also be bidirectional. This bidirectional communications complicates galvanic isolation, which is why I²C is typically used only in small areas of a system.

There is a large selection of IC devices that uses many interfaces. When designing, it is best to search for the function required, e.g., ADC, DAC, temperature sensors, or digital potentiometers, and then consider the available interfaces.

GPIO Straightforward and Simple

General-purpose input output (GPIO) control consists of individual parallel control lines from a microprocessor. It is used when simple devices are controlled by just a few wires. GPIO is just using standard microprocessor ports as a parallel interface. Again, today you can find many parallel interface ICs, including ADCs and DACs.

Conclusion

Engineers smile (have happy faces) when an interface works properly. Sad faces all around indicate failure somewhere. What, then, should be the lesson learned? Quite honestly, it is so simple that it is usually overlooked: pay attention to details, especially if you do not appreciate their full importance. It fits well with both happy and sad faces because, you see, it comes from beer talk at a tavern. Our U.K. colleagues tell us to "Mind your Ps and Qs" in a design. When a design is working well, it could mean mind your pints and quarts of beer. Consequently, we celebrate success with beer and we cry in our beer with sadness. That is why many engineers just say, "I'll drink to that," and smile a happy face.

References

1. The "smiley face"is incorporated in the Unicode computer codes in many languages. A quick look at Unicode shows more than 50 faces including, happy, sad, animals, sun, moon, clocks and more. For examples, go to the Unicode computer source, http://unicode.org/.
2. For a general background on Murphy's Law, see http://en.wikipedia.org/wiki/Murphy%27s_law.
3. For a good introduction to SPI and I²C, go to www.byteparadigm.com/kb/article/AA-00255/22/Introduction-to-SPI-and-IC-protocols.html.
4. Application note 802, "Interfacing SPI Peripherals to the MAX7651 Processor."
5. Application note 4429, "Murphy's Law and the Risks of Designing 'Off Data Sheet'," about ESD, RFI, and PCB trace lengths.
6. Application note 4345, "Well Grounded, Digital Is Analog," about ground planes and stars.
7. Application note 4992, "Reduce the Chances of Human Error: Part 1, Power and Ground," on power decoupling, capacitor self-resonance, lowpass filters, power-supply noise, and star grounds.
8. I²C Tutorials and primer www.i2c-bus.org/;
    - I²C Manual www.nxp.com/documents/application_note/AN10216.pdf;
    - The I²C-bus specification www.nxp.com/documents/other/39340011.pdf;
    - The I²C-bus and how to use it www.i2c-bus.org/fileadmin/ftp/i2c_bus_specification_1995.pdf;
    - Effects of Varying I²C Pull-Up Resistors, www.dsscircuits.com/articles/effects-of-varying-i2c-pull-up-resistors.html.
9. Application note 4024, "SPI/I²C Bus Lines Control Multiple Peripherals"; application note 3967, "Selecting a Serial Bus"; and application note 3438, "Serial Digital Data Networks."

1-Wire is a registered trademark of Maxim Integrated Products, Inc.
IEEE is a registered service mark of the Institute of Electrical and Electronics Engineers, Inc.
IO-Link is a registered trademark of ifm electronic GmbH.
MICROWIRE is a registered trademark of National Semiconductor Corporation.
MICROWIRE PLUS is a trademark of National Semiconductor Corporation.

Related Parts		
MAX5312	±10V, 12-Bit, Serial, Voltage-Output DAC	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 5139: http://www.maximintegrated.com/an5139
APPLICATION NOTE 5139, AN5139, AN 5139, APP5139, Appnote5139, Appnote 5139
Copyright © by Maxim Integrated
Additional Legal Notices: http://www.maximintegrated.com/legal