CMOS Shift Registers Solve I/O Issues in Automotive Applications

Prepared by: Fred Zlotnick
ON Semiconductor

ON Semiconductor®
http://onsemi.com

APPLICATION NOTE

There are a number of very popular shift registers available to solve Input Output problems very economically. The automotive industry, in particular, often needs to send data from one set of switches to a micro−controller that is several meters away. Sending many parallel bit is terribly inefficient. Using a dedicated micro−controller is still an expensive solution. The available industry−standard shift registers make a very economical solution to this problem. This article will try to point out the pitfalls in using these devices properly.

The choice of standard family is generally the first decision. Two families are the most widely used for automotive uses, the Metal Gate CMOS Family also known as the “4000” family and the High Speed Logic or “HC” family. ON Semiconductor offers many part types in these two families, and also in some of the newer, faster lines. These applications usually demand 16 or more bits of information, representing the position of a mechanical switch, and speed is usually of no consequence. Since a high speed serial bus of say 10 MHz might cause RFI[1], a clock rate of 250 kHz or 500 kHz is sufficient and even preferable to high speed data transmission. However, many designers attempting to keep RFI to a minimum unwittingly introduce delays in the system that wreak havoc on the timing and cause unexpected results.

The beauty of the shift register is its simplicity, but there are a few critical parameters that must be taken into account. All shift registers have a “set−up and hold” time. It is assumed that the board is running from 12 V nominal, with an on−board 5 V regulator. The MC74HC165ADR2 is selected as the shift register. This device is rated for automotive temperatures, and is inexpensive and available in a small 14 pin packages. This device is quite flexible, with 8 bits of parallel inputs, Q and Q−Bar outputs, a clock inhibit, and the ability to string several registers together, to create any string of parallel inputs desired. Since the microcontroller is providing the clock signal, there is no need for clock recovery. This design is SPI compatible in the MISO[2] mode. The system designer may choose to use a SPI port or simply use a few I/O pins of the microcontroller, while “bit−banging”.

Since the MCU is polling the shift register, simple RC circuit will suffice for the parallel inputs. If the data is in transition, all that is necessary is for the MCU to come back and poll this data again, before taking action. A polling rate is 10 ms should be sufficient for de−bounce. At a 250 kHz clock rate, it only takes 32 μs, to read 8 bits of data. The capacitors need to have parallel resistors to bleed off the charge after a switch is opened. This combination of filtering and polling is a very lost cost implementation of de−bounce. For reasons of RFI, the maximum data rate will be set to 250kHz. This is well below the 20 MHz rating of the part, even at 125°C. The device has a set−up and hold time specification that requires that the data is present and valid (in proper logic level state) for 22 ns (at 5 V). Fig. 1 shows the use of one “HC165” to expand 8 bits into a 3 wire serial bus. The MCU must “poll” the shift register at a desired rate, say every 10 ms. The act of polling consists of setting pin 1 High, and holding it High, and “clocking” 8 pulses. After each clock pulse the data will be present on pin 9, 45 ns later. All that is necessary to read this data is for the software designer to send out a clock pulse, wait sufficient time for the clock pulse to arrive at the shift register and get transferred to its output, in this case 400 ns and then read the data. He does this 8 times and reads all 8 bits. It is highly recommended that the designer use Schottky diodes and a resistor to form an RC time constant, to limit the noise on the lines. This is all very straight−forward. The only critical point is to wait long enough, to make sure the data read is valid. A preferred solution is to recognize the data, as valid after receiving the same data twice, in sequence.

1. RFI − Radio Frequency Interference
2. MISO − Master In Slave Out

January, 2004 − Rev. 0
Publication Order Number:
AND8144/D

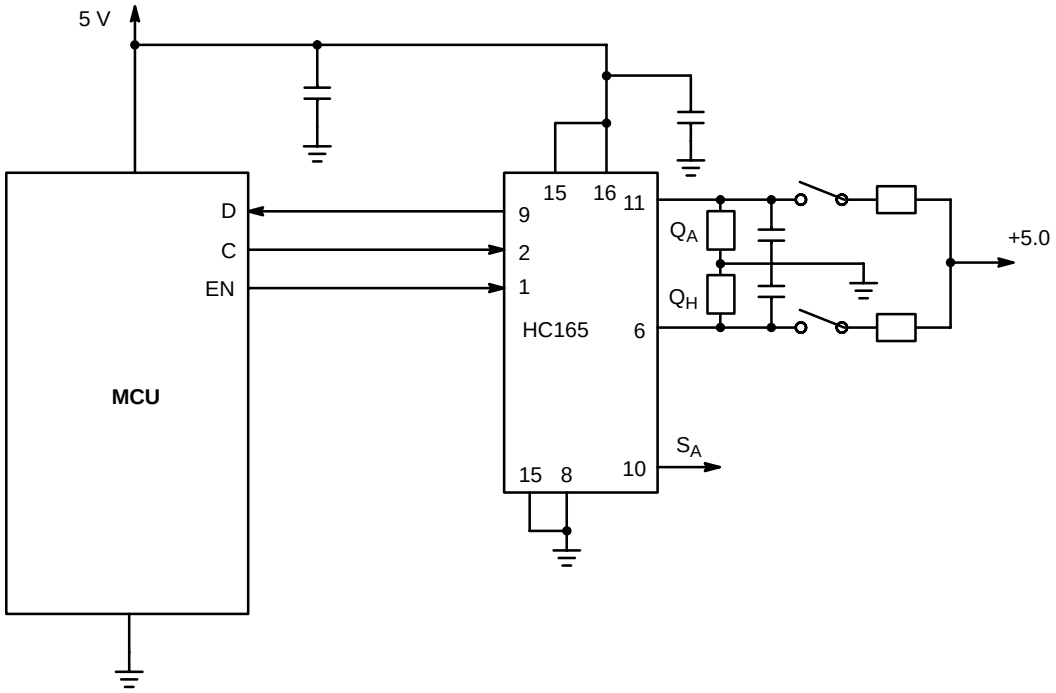

Figure 1.

Cascading Devices

The “HC165” has pins that make it convenient to cascade, however, if the designer is not careful, this circuit can result in data that is incorrect, producing strange results in the data. If the designer is not careful, he can inadvertently be reading the wrong data. What’s worse, this may not show up under normal conditions, and so may get missed in a production environment, and the result is units failing in Minnesota or Arizona.

For 5.0 V applications, a logic level high (V_{IH}) is specified at 3.5 V and logic level low (V_{IL}) is specified as 1.5 V. The actual transition is somewhere between these two voltages, and the actual trigger point is **NOT** specified. A designer must not make any assumptions about this point. The “165” device supports a maximum rise time of 400 ns (at 6.0 V) for clock signals. A rise time of $\approx$400 nsecs, has a slew rate of 100 ns/volt. If every device switched at exactly 50% of Vcc, the slow rise time would not matter. The only safe assumption however, is that the trip point for the clock is between 1.5 and 3.5 Volts, this puts 200 nsecs of ambiguity as to when the clock strobes the data, this ambiguity must be taken into account.

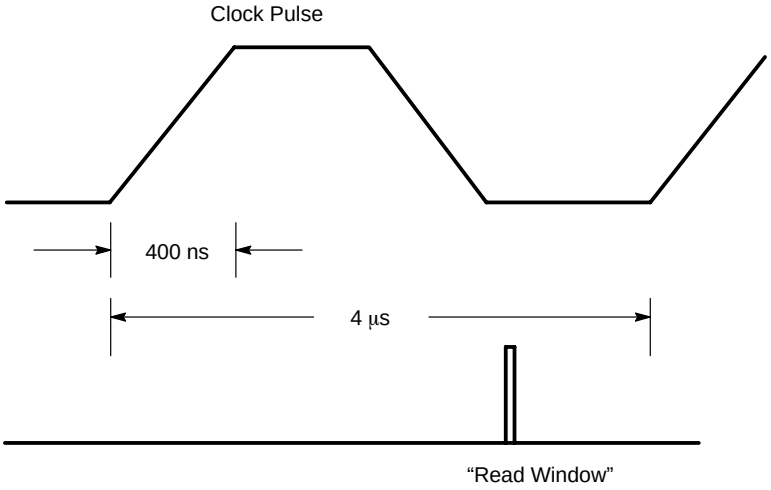

Figure 2.

Even though the edge rate meets the specification, a problem now arises when two or more devices are present. One part may “clock” in the wrong data in the serial mode. Data is read by sending out a series of clock pulses out equal to the number of bits in all the shift registers (8, 16, 24 etc.). Figure 2 shows the typical shape of the waveform used. With its slow edge rate it is not likely to create RFI. When the designer connects 2 or more circuits that may have different threshold points, a problem now occurs. All that is known for certain is a logic level “0” is below 1.5 V. and a logic level “1” is above “3.5”. If one device had an actual threshold of 2.1 and the second device had a threshold of 2.9, this is entirely within the specification for this part. Should the higher threshold device be the second device and the lower threshold device the first, strange problems occur. With the first device having the lower threshold it will “clock” the data in 0.8 x 100 ns or 80 ns earlier than the second device. As the MCU is trying to read the data, it has no problems with the first 8 bits however when the 9th clock occurs, it is supposed to be taking the bit “H” of the previous register. The data could easily be incorrect, and if this represents a signal to lower your window, in Minnesota, at temperatures of −40 °F, this is a real problem.

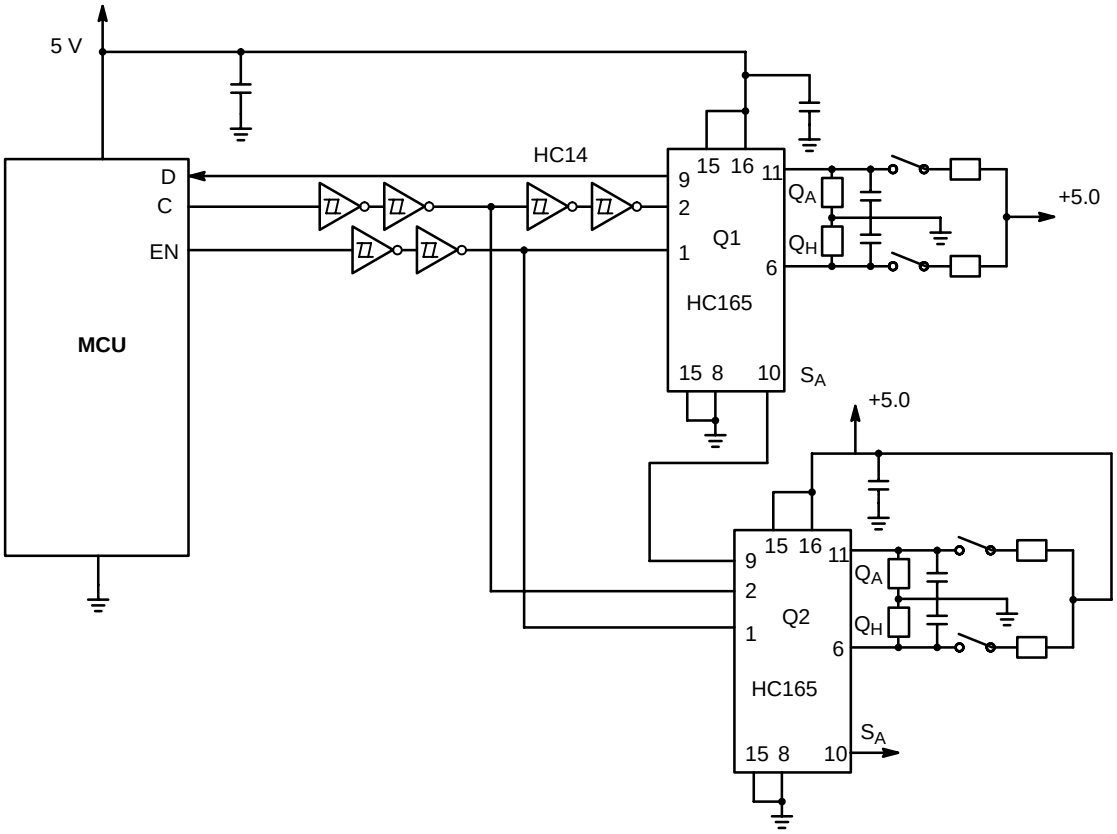

Figure 3.

Improving the Circuit

If the two clocking actions were to take place simultaneously, the data that the second register sees is that which was there after the previous clock. The problem occurs if the first shift register “clocks” in its data significantly earlier than the second. For more than two registers, the same problem occurs from unit 2 to 3 etc. An excellent solution is shown in Figure 3. This improvement involves adding a Schmitt Trigger Input to create a well defined known edge on the clock. The output rise time is guaranteed to be ≤ 22 ns, 20X faster than the input signal. With strong, well defined signal feeding the 2 or more clocks, the ambiguity of the clock signal trip point, is now reduced by a factor of 20. Since the Schmitt Trigger has 6 individual devices in each package, we will use 2 more devices to create some added delay. This will assure that the design now functions properly. The goal is to make sure that the data presented to the first device Q1, pin 10 is the data that is desired. By making sure that Q2 receives its clock pulse earlier than Q1, we can be sure that the data on Pin 1 of Q1, is correct and ready to be loaded. Figure 3 shows the use of the Schottky diodes, and the Schmitt Trigger devices and the delay circuit all being used.

Summary

The biggest factor in solving the problem is making sure that both (all) registers clock their data at the appropriate time. It is necessary to have fast rising the edge rate clock to reduce the ambiguity of the switch points of multiple devices. Since this action causes RFI, as previously noted, the solution is to add a buffer circuit on the board near the shift registers. The recommended circuit adds an MC74HC14 package with 6 hysteresis inverters. The Output rise time will now be ≈ 20 ns worst case. The ambiguity owing to the different thresholds will now be only 10 ns. Since there are 4 more inverters in the package the design can be further improved using some of the otherwise unused gates. If 2 gates cascaded in series between the shift registers, this will add at least another 20 ns of delay between the circuits, assuring even further that there will be no false clocking.

One further improvement would be to add a package of clamp diodes to the input, to help eliminate large noise pulses and ESD. These Schottky diodes are fast and have low threshold, and protect the HC14 from damage and keep noise to a minimum. If the design calls for 3 devices instead of 2, then instead of using 2 inverters for data and EN, the designer uses one, all that would be needed is to generate an opposite going pulse. Two extra inverters would be available for adding delay. A small value resistor could easily be added to form a small RC time constant, with the input capacitance. For example a 2.2 kΩ resistor would add approximately 20 ns delay, when coupled to the (assumed) 10 pf input of a gate. Although this is not accurate, it is in the right direction and helps to assure the correct timing.

Recommended Input Circuit

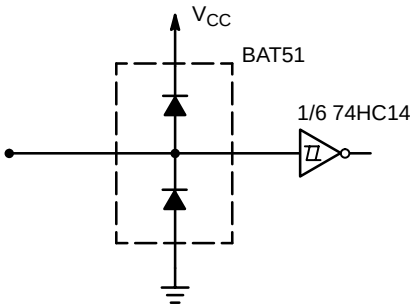

Figure 4.

Conclussions

By using low cost shift registers, a micro−controller can have expanded I/O over a 3 wire bus. Timing of these circuits are important, and it is essential that the relationship between the clocks and the data is appropriate. The circuit will be improved greatly by adding a package of Hex−Schmitt gates to re−establish the clock edge, with a fast rise time. By adding a few more gates between to shift registers, we can be sure that the first device has set up and the second device is now reading data appropriately. Lastly, it is highly recommended to add a package of Schottky diodes to the inputs of all long leads, to prevent any possible issues with ESD and large noise spikes. The Schmitt devices take care of timing and small noise spikes.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
P.O. Box 5163, Denver, Colorado 80217 USA
Phone: 303−675−2175 or 800−344−3860 Toll Free USA/Canada
Fax: 303−675−2176 or 800−344−3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800−282−9855 Toll Free USA/Canada

Japan: ON Semiconductor, Japan Customer Focus Center
2−9−1 Kamimeguro, Meguro−ku, Tokyo, Japan 153−0051
Phone: 81−3−5773−3850

ON Semiconductor Website: http://onsemi.com

Order Literature: http://www.onsemi.com/litorder

For additional information, please contact your local Sales Representative.