Reading An SPI Interface with CompactRIO

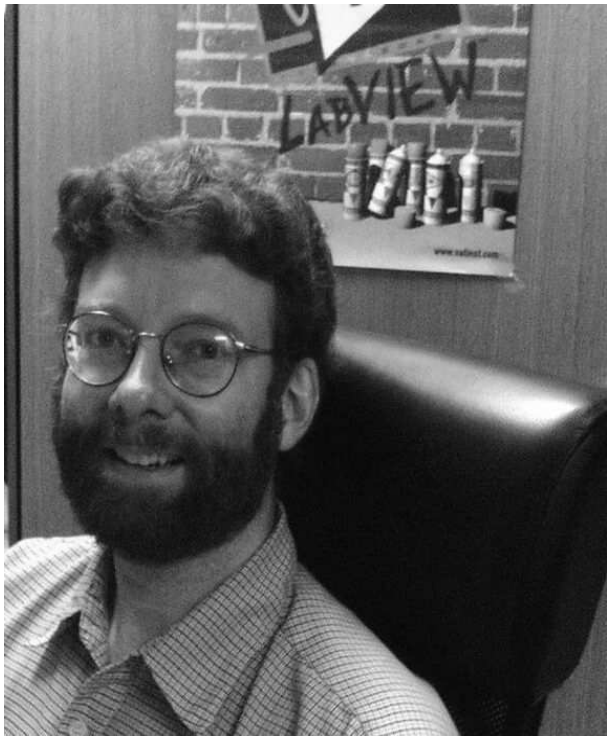
Back to Document

The SPI (Serial Peripheral Interface) bus is a fairly common, low cost serial data bus that can be easily implemented with a few integrated circuits. It can be found in many micro-processor based electronic devices, where it may be used to communicate between the processor and A/D (Analog-to-Digital) or D/A (Digital-to-Analog) converters or other devices. In this example, a commercial device is used to decode the current positions of 16 digital switches which had been transmitted over a remote control radio system. The signals are transferred to CompactRIO using SPI bus communication and are intercepted and decoded using CompactRIO hardware and LabVIEW software. By using SPI communication, this solution used only three digital input bits, and could easily be expanded to handle 32 or more switches while still only using the three digital input lines.

Table of Contents:

About the Author

David Thomson works at the NOAA Aeronomy Lab, where he develops atmospheric chemistry instrumentation for use on high-altitude research aircraft. He is also the principal of Original Code consulting, an NI Alliance Member which provides LabVIEW development and system integration for laboratory, R&D, and manufacturing clients



Switching Multiplexing/Demultiplexing

Our system involves using a Futaba hobby-grade radio remote control to operate a remote vehicle. In order to provide additional control options, we added a third-party switch system, the KeyCoder from Vantec. The KeyCoder adds anywhere from 12 to 32 on/off switches to the Futaba radio and multiplexes the states of these switches into one of the 8 radio channels, leaving 7 channels for normal use. A demultiplexor attached to the radio receiver decodes the switch states and activates high-current FETS for driving various loads.

For our vehicle, we utilize 16 KeyCoder switches to send commands and triggers from the radio to the remote device. In the first version of our system, which used a PXI-based data acquisition and control system, we utilized two bytes of digital inputs to directly read all 16 of our KeyCoder switches. Since the maximum channel count of the cRIO modules is less than the digital port on a PXI system, we were interested in finding a more efficient method of reading the KeyCoder states. Investigation of the demultiplexor module revealed that an internal microprocessor decodes the proprietary radio transmission codes and then sends the switch states to the FETS through an SPI-based shift register chip. It then became apparent that we could take advantage of the reconfigurable FPGA in the CompactRIO

system to read this SPI bus directly and decode the state of the switches based on the data being transmitted to the shift register.

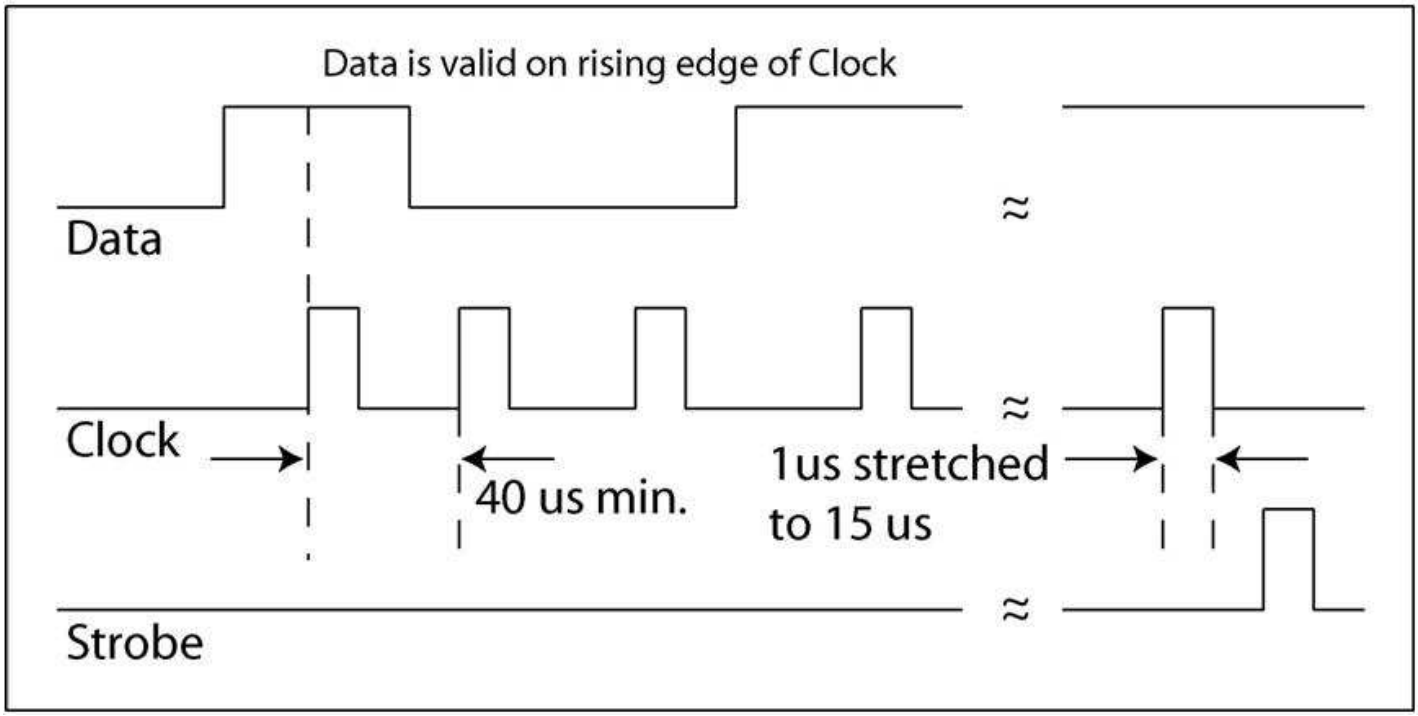


Figure 1. Typical timing diagram for the SPI bus signals. Note that the unmodified Clock and Strobe signals are 1 microsecond long, but never occur less than 40 microseconds apart. External signal conditioning is used to stretch these pulses to 15 microseconds. Clock pulses arrive at variable intervals since the SPI bus is software driven by a microprocessor

LabVIEW FPGA code for the SPI Bus

The LabVIEW code for reading the SPI bus turned out to be quite simple. As shown in Figure 1, the SPI bus consists of three lines: Data, Clock, and Strobe. The state of the Data line is read whenever a rising edge is present on the clock line. A rising edge on the Strobe line then indicates when the complete data packet has been sent. In order to decode this on the FPGA, we simulated the shift register that was implemented in the demultiplexor. As shown in Figure 2, a shift register initialized with a 16 element Boolean array is used to store the pattern of bits read from the Data line as Clock pulses are detected. Each new bit read causes the pattern in the shift register to be shifted down one element, with the new bit replacing element 0. The Strobe line is then used to determine when the pattern is valid, at which point the pattern is displayed on an indicator. The Strobe pulse not only ensures that the data is synchronized to the shift register (so that Bit 0 data is in element 0 of the array), but also discriminates against data from other devices that share the SPI bus, since the Data and Clock lines are shared for such devices.

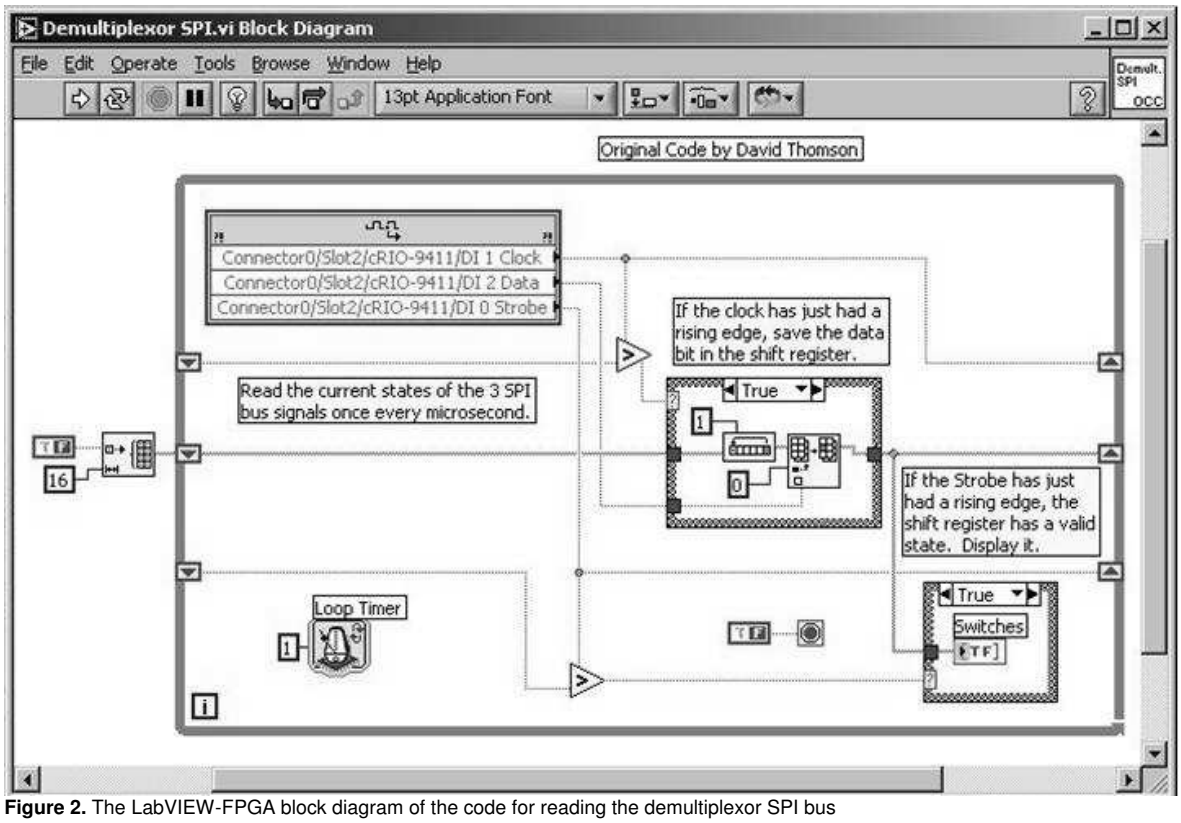


Figure 2. The LabVIEW-FPGA block diagram of the code for reading the demultiplexor SPI bus

Conclusions

Our solution for reading the demultiplexor information resulted in an efficient, simple LabVIEW program that requires only three digital input lines to read 16 switch states. In addition, this solution could be extended to a 32-switch KeyCoder without requiring any additional digital input lines. It should be noted that minimal additional signal conditioning circuitry was required. For the CompactRIO digital input module, we used a cRIO-9411, which has a response time of 500 nanoseconds.

CompactRIO Example Code

See Also:

Example CompactRIO Program: Reading An SPI Interface with CompactRIO