Stephen Willson

Serial Port Project

We chose to build a serial port. Essentially, it’s a physical interface for either sending data to a computer or receiving data from a computer, 1 bit at a time. The line starts idling on logic high because we’re using negative logic. To start, a data strobe is sent by flipping switch X’ from logic 1 to logic 0 and back again; this initiates the transmission of a character. When data is going to be sent in, there’s a start bit that’s read for 16 clock cycles to make sure it’s not a false start bit due to switch bounce. Then, 8 data bits are sent in. Each one of the switches on our serial port represents one bit, with switch 1 being the most significant bit and switch 8 being the least significant bit. After the data bits are sent, there’s a parity bit that’s sent. If the parity bit is different from the type of parity we used, then somewhere along the line, a bit has changed state. At the end, once the data character is sent, there are one or two stop bits to signal that the transmission is over. 4 of the bits are sent through a 7447 chip and a resistor pack to a 7-segment display and the decimal equivalent of the binary value is shown. The other 4 bits also go through a 7447 and a resistor pack to display the decimal value on another 7-segment display.

There are 6 LEDs that can light during the sending of data. LED A is the Receive Data Available flag. This will light when the entire data character has been received and transferred to the receiver holding register. Once it’s lit, flipping switch Y’ will reset this flag. LED B is the Parity Error flag. It will light if the data character received doesn’t agree with the parity of the sent data character or the specified parity configuration of the UART. LED C is the Framing Error flag. This will light if the received character’s stop bit isn’t valid or if its start bit isn’t valid. LED D is the Overrun Error flag. It will light if data is sent without clearing out the data that’s already on the register (by flipping switch Y’) because it can’t process the data character being sent if the register is full. LED E is the Transmitter Buffer Empty flag. It will light if there may be an extra character in the transmitter buffer. It will turn off if the buffer is full. LED F is the End of Character flag. It will light when no character is being transmitted. It will turn off while the UART is transmitting a character.

The crux of the serial port is the previously mentioned UART chip. Only one is used in our serial port, and it’s a 40-pin chip. Pins 1 through 3 deal with the power requirements for the UART, both powering and grounding the chip. Pins 26 through 33 as well as pin 16 all deal with the transmitting portion. Pins 33 through 26 correspond to switches 1 through 8, respectively. Pin 16 is grounded. If it’s grounded, it essentially enables the status of the data character to be read (in our case, through the LEDs). Since the 5 output LEDs other than the End of Character flag are tri-state, if pin 16 isn’t at logic 0, the outputs act like there’s no connection. The rest of the pins deal with receiving data or making sure the UART will work correctly. Pins 12-9 end up routing the first 4 bits to the top 7-segment display, through the 7447 and resistor pack. Pins 8-5 do the same with the latter 4 bits. Pin 17, as well as 40, are hooked in to the clock. Pin 20 is the serial input to receiver line, and it connects to pin 25, the serial output from transmitter. Pin 4 is grounded, since a logic 0 on it will place the received data onto the output lines (the output lines being the lines that go to the 7-segment displays). Pin 19, 13-15, 22, and 24 correspond to LEDs A through F respectively, all discussed above. Pin 18 goes to switch Y’. When Y’ is flipped, it will reset the Receive Data Available flag. Pin 23 is the data strobe. This is how we initiate the sending of data in the first place. It’s attached to switch X’, discussed above. The rest of the pins deal with setup for the UART. Pin 21 is the master reset. If it’s at logic 1, it will reset the UART. We ground it so the system will actually work. Pin 34 is the control strobe. Basically, it can change states or it can be hard-wired to always be logic 1. Since we have no need to not enter the control bits into the holding register, we wire it to a +5 so the state won’t change. Pin 36 determines the number of stop bits. If it’s a logic 0, there will be one stop bit. Logic 1 will set it for 2 stop bits. We grounded ours so there’s only 1 stop bit. Pins 37 and 38 determine the number of bits per data character. The truth table goes from 2 0’s setting it to 5 bits per character to 2 1’s setting it to 8 bits per character. We went with 8 bits.