EXPERIMENT NUMBER 8
Interfacing the 8051 with an External Sensor I

INTRODUCTION:

External sensors play an important role in embedded systems. Most digital applications require some kind of sensor. There are a wide variety of sensors available: from a simple keypad of buttons to ultrasonic sensors measuring distance. The ability to communicate with these sensors is a key part of developing a successful product. In this lab, you will program the 8051 to read the temperature from a Dallas 1-wire digital thermometer (DS1822) and will print it through a serial port connection using the Universal Asynchronous Receiver Transmitter (UART) feature of the 8051. All communication between the microcontroller and the external sensor will take place over a single wire using the Dallas 1-Wire communication protocol.

OBJECTIVES:

1. Develop a C program for the 8051 that communicates with an external sensor and that utilizes the on-board UART to report results.
2. Learn how to display serial data from the 8051 on the computer's monitor using HyperTerminal software.
3. Exposure to fundamentals of network communication protocols.
4. Reinforce concepts of cosimulation and coverification.

REFERENCES:

1. XS40 Schematic: *http://www.ece.umr.edu/courses/cpe214/schematics.pdf*
2. Datasheet for the VHDL model of the DS1822: *http://www.ece.umr.edu/courses/cpe214/ds1822-datasheet.pdf*
3. Dallas Semiconductor DS1822 datasheet: *http://pdfserv.maxim-ic.com/arpdf/DS1822.pdf*

MATERIAL REQUIRED:

1. Keil µVision 2
2. Simulation models: *http://www.ece.umr.edu/courses/cpe214/lab8.tar*
3. Unix-based computer
4. Mentor Graphics software
5. Ftp program
6. Windows-based computer with an unused parallel port and serial port
7. XS40 board
8. DS1822 from Dallas Semiconductor

9. MAX233 from Maxim
10. Null-modem cable
11. 1-Wire communication C-library: *http://www.ece.umr.edu/courses/cpe214/lab8.c*

BACKGROUND:

In this lab you will create hardware and software to allow the 8051 to read temperature from a Dallas 1-wire digital thermometer and will report that temperature through the serial port. Dallas Semiconductor (now part of Maxim) makes a line of simple sensors, which require only 1 wire for both communication and power. Several devices can even be placed on the same wire (Figure 1). This allows many sensors to be placed in a system with a minimal number of wires, connections, or additional components.

Each 1-wire device has a unique 64-bit serial code, which allows multiple devices to function on the same 1-wire bus. When there is only one 1-wire device on the bus, the system is referred to as a "single-drop" system; if there are multiple 1-wire devices on the bus the system is referred to as "multi-drop". The block diagram in Figure 1 illustrates the simplicity of the hardware configuration when using multiple 1-wire devices. A single-wire bus provides both communication access and power to all devices. Power to the bus is provided through a weak pull-up resistor from a 3 to 5.5 V supply rail. The idle state of the bus is a logical high. Devices communicate with one another by pulling the bus low. The microcontroller always initiates a communication sequence; hence, the microcontroller is the bus master and other devices on the bus (the DS1822s) are bus slaves.

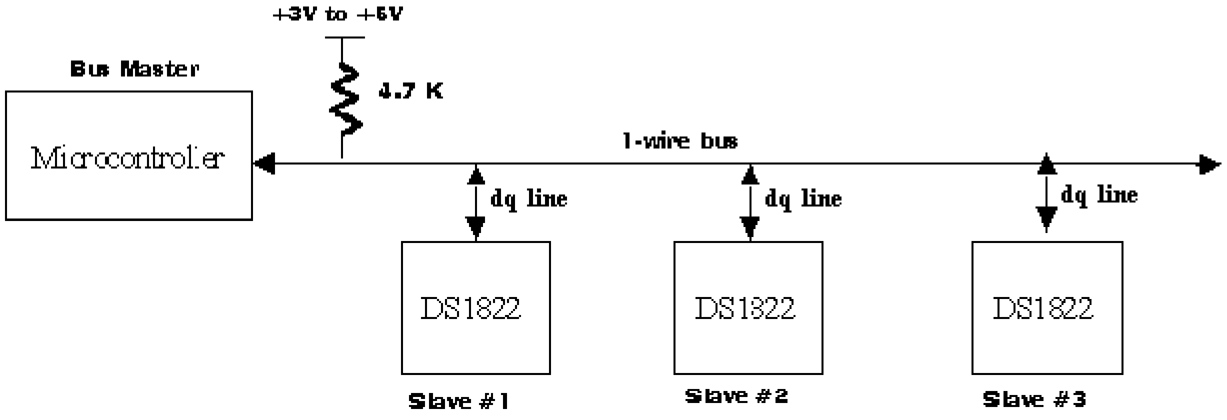

Figure 1. Microcontroller interface to multiple DS1822 temperature sensors

Communication between the microcontroller and a 1-wire device begins with a reset pulse from the microcontroller, which is followed, by a presence pulse from the 1-wire device (Figure 2). When the 1-wire device sends the presence pulse in response to the reset, it is indicating to the master that it is on the bus and ready to operate. The bus master transmits a reset pulse by pulling the 1-wire bus low for a minimum of 480 μs. It then releases the bus and goes into receive mode. When the bus is released, the pull-up resistor pulls the 1-wire bus high. The 1-wire device responds with a presence pulse by pulling the bus low for 60-240 μs.
Data is read and written to the bus using read- and write-slots. One bit of data is transmitted for each slot. To write a '0', the master pulls the 1-wire bus low for 60 μs – 120 μs. To write a '1', the bus

master holds the 1-wire bus low for about 15 μs then lets it go high for about 45 μs. A read time slot is initiated by the master. To tell a 1-wire device to write a bit to the bus, the master pulls the bus low for about 1 μs. When the 1-wire device sees this 1 μs low pulse (and is ready to write something to the bus), it will either pull the bus low to write a 0 to the bus or will let the bus go back high, thus writing a 1. Any data transmitted on the 1-wire bus starts with the least-significant-bit first. The figure below shows part of a communication sequence between the master and the DS1822. The master issues a reset pulse and slave responds with a presence pulse. The master then issues a Read ROM (0x33) command, followed by read time slots, so that the DS1822 can transmit its serial code back.

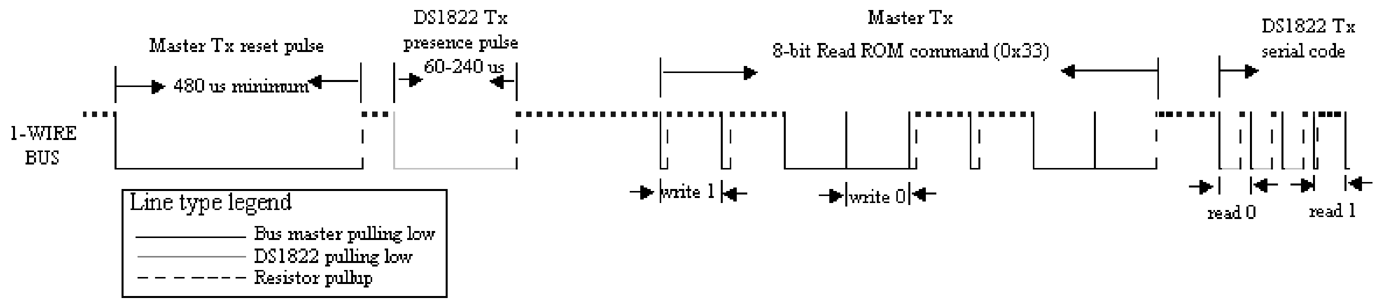

Figure 2. 1-Wire Communication Sequence. Note that on a "read 1", the bus is pulled low by the master but allowed to immediately go high again by the slave.

The bus master has to follow a specific transaction sequence to access the DS1822:

1. Initialization (reset pulse followed by presence pulse)
2. ROM Command (followed by any required data exchange)
3. Function Command (followed by any required data exchange)

All transactions between the bus master and the DS1822 begin with an initialization sequence in which the bus master sends a reset pulse and receives a presence pulse(s) transmitted by the DS1822(s). After the bus master has detected a presence pulse, it can issue a ROM command.

There are five ROM commands[1], but in this lab, you will deal with only one: the Skip ROM command (0xCC). Normally the master uses the ROM command to identify a particular 1-Wire device it wants to talk to. Remember that each device has a unique identifier. If there is only 1 device on the bus, though, you can skip this part entirely. As the name "skip ROM" indicates, the master is skipping the identifier and going straight to a function command. A function command can be issued immediately after a skip ROM command.

The DS1822 supports six function commands, but in this lab, you will only use two: Convert T and Read Scratchpad. The Convert T command (0x44) tells the DS1822 to measure the temperature and convert it to a digital number. The resulting measurement is stored in a 2-byte temperature register in the scratchpad memory of the DS1822. Each DS1822 has a 9-byte scratchpad memory. Byte 0 and Byte 1 of the scratchpad memory contain the LSB and MSB of the temperature register, respectively. The temperature is stored in degree Celsius as a 16-bit sign-extended two's

[1] To find out more about all the DS1822 command, please see the DS1822 datasheet: *http://pdfserv.maxim-ic.com/arpdf/DS1822.pdf*

complement number as shown below. For example, if the MSB and LSB contain 01h and 98h, respectively, then the temperature is 25.5 degrees Celsius (i.e. $2^4 + 2^3 + 2^0 + 2^{-1}$).

MSB (Byte 1 of Scratchpad)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
S	S	S	S	S	2^6	2^5	2^4

LSB (Byte 0 of Scratchpad)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}

The master uses the Read Scratchpad command (0xBE) to read the contents of the scratchpad. The data transfer starts with the least significant bit of byte 0 and continues through the scratchpad until all bytes are read. The master has to provide read time slots to read each bit. It can issue a reset pulse to terminate reading at any time.

In this lab you will create hardware and software to allow the 8051 to communicate with a DS1822. You will create an instance of the VHDL model of the DS1822 in the XS40 schematic and will develop a C program to read temperature from it. The temperature "sensed" by the VHDL model of the DS1822 can be changed directly in the schematic. After simulation, you will implement your design in hardware using an actual DS1822 from Dallas Semiconductors. Temperature readings will be written out of the serial port of the 8051.

To allow you to "see" what is written out of the serial port during simulation in Mentor Graphics, you'll connect the serial unit from lab 7 to the serial port of the 8051. In the actual hardware, you will display data from the serial port on the computer's monitor using the HyperTerminal software. To connect the 8051 to the PC will require the use of a MAX233 part, which shifts the 0 to +5V TTL levels used by the microcontroller to the –10V to +10V levels used by the computer's RS232 serial port.

PRELIMINARY:

Write a C program to read the temperature from a DS1822. The 8051 will have to first issue a "Convert T" command, then a "Read Scratchpad" command, and will then have to read the temperature (the scratchpad) written back to the 8051 by the DS1822. The program will look like:

1. Set the 8051 serial port to transmit data in mode 1 with a baud rate of 2400 bps
2. Issue Convert T command
    a. Send reset pulse to the DS1822.
    b. Detect presence pulse
    c. Send Skip Rom command (0xCC)
    d. Send Convert T command (0x44)
3. Wait till completion of temperature conversion
4. Issue Read Scratchpad command
    a. Send reset pulse
    b. Detect presence pulse

    c. Send Skip Rom command
    d. Send Read Scratchpad command (0xBE)
5. Read value send back by DS1822
6. Convert the temperature from 16-bit sign extended two's complement format to real
7. Write the converted temperature in degrees Celsius out the serial port

To make your life easier, functions are provided at *http://www.ece.umr.edu/courses/cpe214/lab8.c*. These functions allow you to send a reset pulse and detect a presence pulse, to write a command to the DS1822, to read single bits from the DS1822, and to provide delay. You will need to write a function to read one byte or multiple bytes at a time and you will need to integrate all these functions to make your code work.

The functions provided above assume a couple of constants and variables have been declared globally in your program. They are:

- sfr OWDQ = P1^5;
- #define CLKPERIC.

Variable OWDQ defines the port bit the DS1822 is connected to – in this lab Port 1 pin 5. The constant CLKPERIC holds the number of clock cycles per machine cycle used by your 8051. Most 8051s use 12 clocks per machine cycle, though some use 6. Refer to the sample code in the datasheet of the DS1822 for help.[2]

The temperature conversion time for the VHDL model of the DS1822 used in simulation is 750 μs, and for the actual DS1822 used in hardware, it is 750 ms. The conversion time of the VHDL model was changed by the model developers to reduce simulation time. When developing your code for simulation, use a conversion time delay of 750 μs. Before implementing in hardware, though, change the conversion time to 750 ms.

Writing to the serial port is made easy using the "printf" command. Once the serial port has been set up, simply issuing the command "printf("The temp is %d",42);" will cause the ASCII characters 'T','h','e',' ','t','e','m','p',' ','i','s',' ','4','2' to be transmitted through the serial port at the baud rate previously set up be the user. Data transfer on the serial port starts with the least-significant-bit first. In serial mode one, 10 bits are transmitted for each character, starting with a start bit (always a '0') followed by 8 data-bits (the ASCII code, starting with the least-significant-bit) and a stop bit (always a '1'). Use this command with caution, though, as it uses up quite a bit of execution time and code space.

Develop, compile and simulate your program in software using the Keil software development tools to ensure your program is doing what you expect it to. Come to class with a printout of your code and a hex file ready to run.

PROCEDURE:

Summary: Add VHDL model of DS1822 to the XS40 schematic. Cosimulate your hardware and software together using QuickSim Pro. Check for the correctness of your code and eliminate any errors found. Download and verify your design on the XS40 board with the actual DS1822 from Dallas Semiconductor.

[2] Datasheet located at *http://www.ece.umr.edu/courses/cpe214/ds1822-datasheet.pdf*

1. Download the simulation model of the DS1822 and XS40 board from *http://www.ece.umr.edu/courses/cpe214/lab8.tar* and untar it in your working directory (lab8). This file contains a top-level model of the XS40 schematic and a VHDL model of the DS1822.
2. Set the working directory and user library to directory lab8 using the commands *swd* and *sul* within the lab8 directory.
3. Build the simulation models you will need using the commands:

   build_simulation
   build_ds1822
   build_serial_unit
4. Build your simulation schematic as shown in Figure 3.
   i. Open the xess40_schematic and place the DS1822 symbol within the sheet. Connect the 1-wire bus (dq line) to the pin 5 of Port 1.

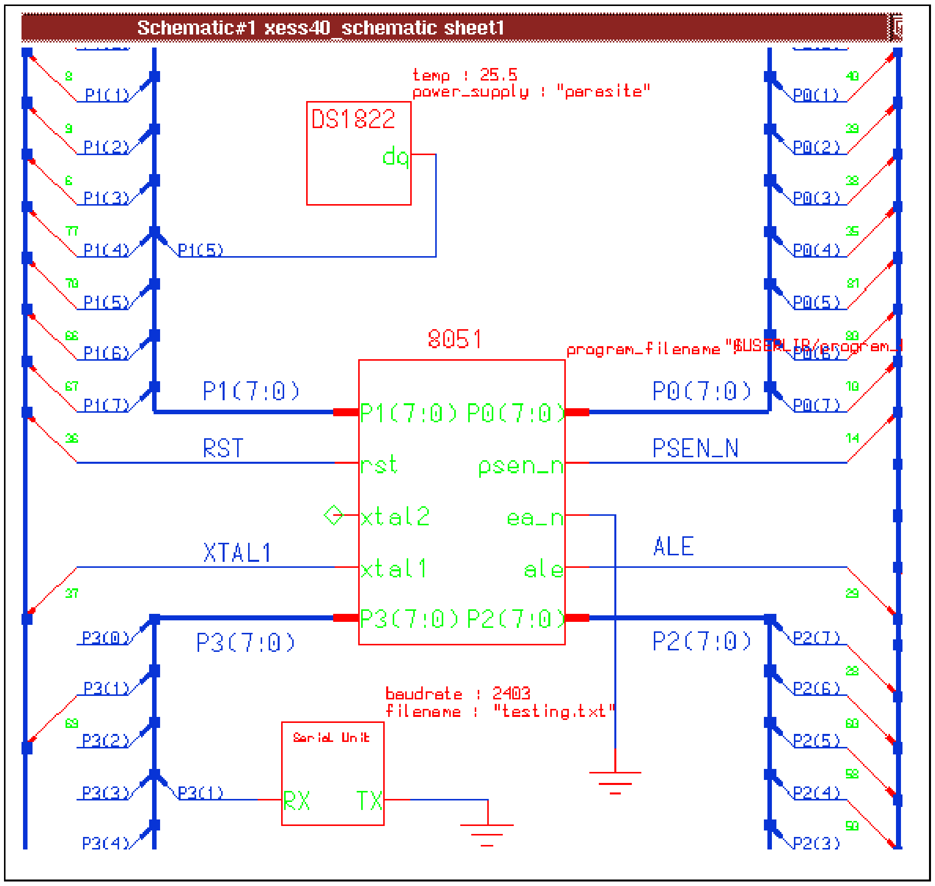

Figure 3. Top-level schematic

   ii. Place the serial unit symbol within the xess40_schematic. Connect receive (Rx) of the serial unit to transmit of the 8051.
   iii. For the serial unit to work properly, the baud rates of the serial port of 8051 and the serial unit should be equal. Change the baud rate of serial unit to 2400 bps.
   iv. Check and save your design.
5. After you have completed the above steps, print out a copy of xess40_schematic and place it in your lab notebook.

6. Cosimulate your hardware with software using QuickSim Pro.
   A. Download the hex file created from your preliminary via an ftp program or floppy disk to the "program_files" directory under your design. Rename your hex file "sram.hex".
   B. Start the QuickSim Pro using the command:
      qspro –as hdl xess40_schematic
   C. Add the GLOBALSETRESET line as an editable waveform and pulse it as in lab 1. Also keep the RST input to the 8051 low by forcing J1-2 high.
   D. Trace the 1-wire bus (pin 5 of port 1) and the transmit pin of the 8051 (TxD), as well as any other signals needed to debug your design.
   E. Simulate your design for 3 milliseconds to verify that the reset pulse has been sent, that the 1-wire device responded with a presence pulse, and that the master then sent an 8-bit Skip Rom and Convert T commands. It may take your computer a while to do this simulation. Check to see if the code is correct. You will probably want to zoom out significantly to see significant portions of the simulation waveform or you may want to look at the report statements given by the slave in the QSPro(HDL) window. For example, if the master issues a reset pulse, the report statement will look like "SLAVE: RESET PULSE DETECTED". If it's not working, you'll need to go back and check your design and 8051 program. Print out the simulation waveform and label the reset pulse, presence pulse, and both commands.
   F. Simulate for 5 more milliseconds to check that the reset pulse has been sent again and that the 1-wire device responded with a presence pulse. This time, after the initialization sequence, the master has to send Skip Rom, Read Scratchpad commands. The slave has to send the contents of Scratchpad (first two bytes) in the read time slots provided by the master. Remember that the data transfer starts with the least significant bit first. Check for the correctness of your code by looking at the report statements in the QSPro(HDL) window.
   G. Continue your simulation until at least part of the temperature is sent out the serial port. The temperature will be displayed one digit at a time in the QSPro(HDL) window. If your code is working properly, it should be equal to the temperature provided as a generic in VHDL model of DS1822. Print this portion of the simulation waveform and show that the number sent out the serial port matches the number you expect on that waveform.
7. Verify your design in hardware using the XS40 board. (Create a bit file, connect up the XS40 board, connect the serial port to the computer, set up the Hyperterm software to display information from the serial port, run program and see your message appear).
   A. Create a bit file of your XC4005 schematic, which will be downloaded to the FPGA on the XS40 board, using the xmake command:
      xmake xc4005 xc4000xl xc4005xl-1-pc84
   B. Modify your program to increase the temperature conversion time from 750 μs to 750 ms (by changing the delay given to the supplied delay function). You may also need to change CLKPERIC, depending on the particular XS40 board you plan to use – some use 12 clocks/per instruction, some use 6 (The Phillips boards use 6 and the Intel use 12). Check with your TA if you are unsure. Create a hex file for your new software.
   C. In the hardware lab, connect pin 5 of port 1 (XCBUS 70) to the DQ line (middle pin) of the DS1822. You may have to use an experimenter's breadboard to make this connection. Ground the ground pin of the DS1822. Do not supply power to the power pin. The pinout of the DS1822 is shown in Figure 4.

D. To display the temperature on the computer screen, you will need to hook up the MAX233 part and the serial communication lines. The pinout of MAX233 along with necessary connections is shown in Figure 4.
   - Connect the transmit pin from the 8051 (XCBUS 69) to the $T1_{IN}$ of MAX233 (pin 2).
   - Connect the $T1_{OUT}$ of MAX233 (pin 5) to the receive pin of the computer's serial port (pin 3 of the 9-pin null modem).
   - Short the following pair of pins of MAX233: 10 and 16, 11 and 15, 12 and17.
   - Ground pin 6 and pin 9 of MAX233.
   - Connect the pin 7 of MAX233 to +5V power supply.
   - Short pin 7 and pin 8 of computer's serial port.

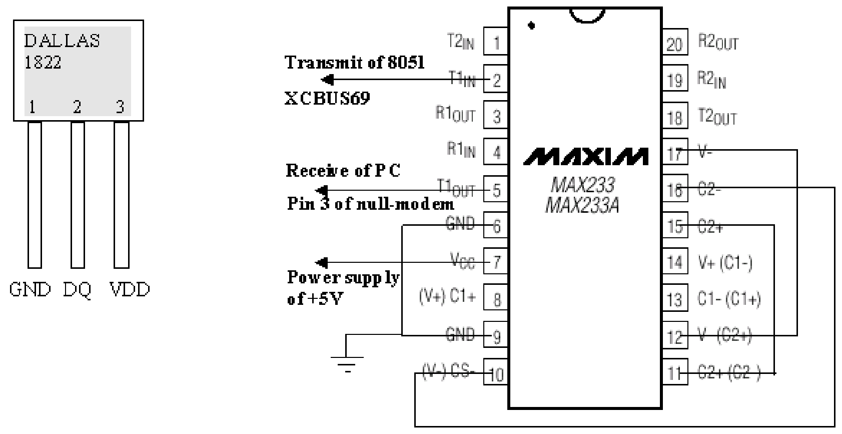

Figure 4. Pinout of DS1822 and MAX233

E. You will need to setup the HyperTerminal to receive data from the serial port of the computer. Open up the HyperTerminal software from Start > Programs > Accessories > Hyperterminal > HyperTerminal.

F. Give a name to your connection and click OK.

G. In the window that follows, you will need to select the serial port that is connected to the 8051. Consult your TA to find out which serial port to use.

H. After you have selected the port, you will need to set its baud rate. Change the baud rate to the same rate you programmed in the 8051 and click OK. The HyperTerminal software is now ready to receive data from the 8051. When HyperTerminal receives a message, it displays it on the screen.

I. Download your hex program file and your bit file to the XS40 board using the GXSLOAD program.

J. Trace the 1-wire bus (DQ-line) and transmit pin (TxD) on the digital oscilloscope. Clear the scope screen with the erase button. Set it to trigger on a rising edge of the 1-wire bus and to capture a single trace. Set the time per division to 200 milliseconds.

K. Start your program by resetting the microcontroller by strobing D0 using the GXSPORT program. Capture a screen full of information on the Digital Oscilloscope. You may want to zoom in, to see the significant portions of the simulation waveform. If your program worked correctly, on the DQ line (the 1-wire bus) you will see a reset pulse, presence pulse, Skip Rom, Convert T, wait period of approximately 750 ms, followed by another reset pulse, presence pulse, Skip Rom, Read Scratchpad, and two bytes of data transferred across the 1-wire bus. You should also see data transfer on the transmit pin of the 8051. If your connections are correct and the HyperTerminal software is set up correctly, then the temperature will be displayed in degrees Celsius in the HyperTerminal window. Does it show something equivalent to room temperature?

L. Demonstrate your working design to your TA and, when finished, delete your files from the lab computer.

QUESTIONS:

1. List a few applications where external sensors like the DS1822 can be used.
2. The approximate time needed to simulate reading the temperature from the DS1822 and displaying it using the serial port is approximately 3 minutes on a Sun Ultra. Imagine if you had a much larger system that you needed to simulate for a long period of time. What could be done to speed up this simulation?
3. In this laboratory, the Master (the 8051) asked the DS1822 for a temperature reading (for the scratchpad which contains the temperature reading) using the command sequence a) Reset pulse (Presence pulse), b) Skip ROM and c) Read Scratchpad (followed by the data). This sequence wouldn't work if there were several DS1822's out on the bus. Why? How might the command sequence be changed to avoid this problem?
4. Why do you think that every command sequence starts with a reset pulse? Why is a reset pulse so long (a minimum of 480 μs)?