University of Massachusetts Amherst
Department of Electrical and Computer Engineering

ECE 353, Computer Systems Lab I
Fall 2005

Lab 2: Design of two MIDI receivers on an 8-bit microcontroller.

1. Introduction

In this lab you will design and implement two serial input ports designed to receive MIDI (Musical Instrument Digital Interface) data. Both of your serial ports will operate on a popular 8-bit microcontroller: the AVR ATmega32 from the Atmel Corporation of San Jose, California.

The first serial port in this lab will be a "soft" serial port and it will be written in assembly language. The second serial port will also be written in assembly language but it will make use of an on-board programmable serial port hardware subsystem.

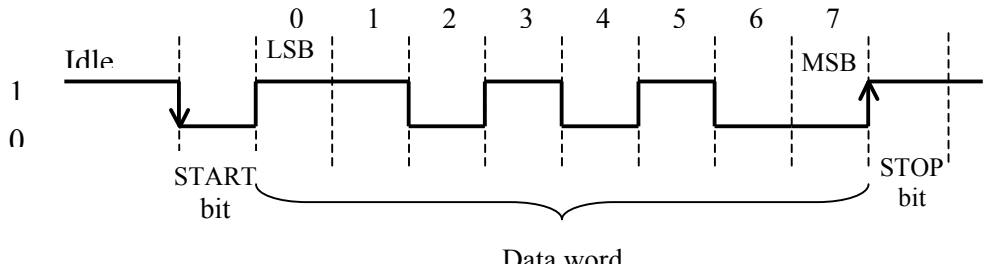

Figure 1. MIDI message format

2. Project Description

As in lab 1, your system will be designed to receive MIDI (Musical Instrument Digital Interface) data. The MIDI data will generated in the same manner as in lab 1. Your board will make use of the same optical-isolation circuit from lab 1. However, instead of sending the serial data stream from the opto-isolator output to a CPLD input you will route it to a microcontroller, the Atmel AVR ATmega32.

In Lab 2 lab you will design and implement two serial input ports. Both of your serial ports will operate on a popular 8-bit microcontroller: the AVR ATmega32 and will be programmed in AVR assembly language.

The first Lab 2 serial port will be a "soft" serial port. It is called a "soft" serial port because it is almost a software-only solution. (The only on-chip AVR I/O (input/output) subsystem it will use is one line of a generic 8-bit parallel port.)

The second Lab 2 serial port will make use of a rather powerful on-chip AVR I/O (input/output) subsystem: the USART (Universal Synchronous/Asynchronous Receiver/Transmitter). You will initialize the USART to meet the MIDI spec and receive the data via the USART.

As in lab 1, your systems will display the MIDI Note Number in binary form on 7 LEDs.

For details regarding the MIDI specification please refer to the lab 1 handout. As a reminder, Figure 1 is a diagram of a single byte of a MIDI message.

Even though Lab 2 will not make any use of the Altera CPLD of lab 1, leave the Altera circuit on your breadboard. The only modifications should be to remove the 5V power lines to the CPLD, remove the wire between the opto-isolator and the CPLD and disconnect the line from the 4 MHz clock and the CPLD.

3. Design Specification

The MIDI standard specifies a uni-directional serial interface running at 31,250 bits/s ±1%, a convenient division of the typical 4MHz clock rate to be used in this lab. Fig. 2 illustrates the principle of decoding the MIDI message. It follows the general design of the Universal Synchronous and Asynchronous Receiver Transmitter (USART) (or Universal Asynchronous Receiver Transmitter (UART)) (with the exception for the baud rate). Each 8-bit byte is framed by a START bit and a STOP bit. For transmission at 31,250 bits/s, each of these bits take 32 μs, called bit time (BT). The addition of the start bit and stop bits means that each byte of the MIDI message takes 10 bit periods to transmit, taking a total of 320 μs.

Before the first frame is transmitted, the signal line idles high (1). The receiver monitors the line, waiting for the signal to drop to 0. Once the negative-going transition is detected, the receiver synchronizes on this transition and starts sampling the message. Conceptually, the receiver reads the 8 bits of the serial data by sampling the input "in the middle" of each bit, i.e., at 1.5 BT (bit 0), 2.5 BT (bit1), …, 8.5 BT (bit7), as shown in Figure 2. Finally, the STOP bit is sampled at 9.5 BT, and the procedure is repeated for the next message byte, synchronizing on the start bit.

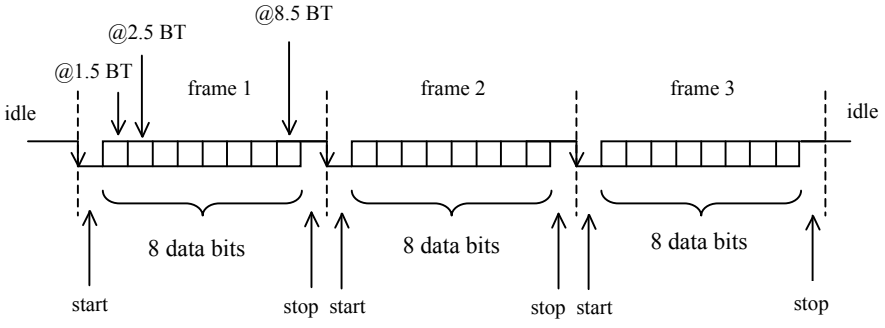

Figure 2. Serial protocol and sampling of MIDI message

There are several methods that can be used for such a sampling. Traditional UART module samples the data at a rate of 16x or 8x of the transmitted message. If the sampling at 9.5 BT of the STOP bit does not produce the expected high value, the receiver sets a flag to indicate a *framing error*.

Furthermore, several samples are made for each bit and the system uses "voting" to make sure that consistent data and the right bits are detected. In the soft serial port portion of this lab you may sample each bit once, provided that you use the correct sampling frequency. You are not required to implement any framing error detection.

4. Implementation

In this experiment you will design two software versions of a serial MIDI receiver. They are to be implemented in the Atmel AVR ATmega32 8-bit microcontroller. Input to the AVR is a single-bit input line from the output of the opto-isolator circuit.

The signal your system will receive and display originates in the PC. The MIDI OX software transforms each key of the PC's keyboard into an electronic music keyboard. MIDI OX will send a MIDI Note On message out serially via the MIDI cable. The cable is terminated with an opto-isolator. The AVR chip will be clocked by a 4MHz crystal oscillator. The internal AVR system clock will also be at 4 MHz. Output of the AVR will drive seven LEDs to display the note number of the note played on computer keyboard in binary. In the future labs we will construct an electro-optical-mechanical device that will produce an audible signal associated with that note.

You will use Altera's AVR Studio software to design, simulate and program your design on an AVR ATmega32. AVR Studio can be downloaded from Atmel web site:
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

You will use AVR assembly language for your design, and will simulate it portions of it using AVR Studio. Each student should register and make use of the AVR resources at avrfreaks.net.

You will program the AVR using Olimex USB JTAG programmers. DO NOT USE the Altera CPLD programmers on the AVRs.

Finally, you should use a logic analyzer to store analyzer traces, to demonstrate the correctness of your implementation.

5. Demonstration and Report

You will demonstrate the functioning of your design to a course instructor by the designated due date, Tuesday, 11 Oct 2005 (Refer to the schedule on the course website for all other due dates.). In the report, provide a complete description of your design and its components, including: AVR code, block/schematic diagram of the design and simulation results. You should also include the logic analyzer printouts with your comments, clearly indicating that you analyzed the results and whether the design functions correctly or not.

Finally, you should include brief explanation of the debugging and any problems encountered. Please refer to the link on the class website regarding the required report format. The designated deadline for the report, regardless of the actual demo date, is Thursday, 20 Oct 2005. This first report will cover both labs 1 and 2.