1. Introduction

In this lab you will build and program a MIDI (Musical Instrument Digital Interface) controller, the GME (General MIDI Explorer). A MIDI controller is a device which sends MIDI messages to a synthesizer. In this lab, the synthesizer is a program running on the PC.

The MIDI standard was established many years ago by a consortium of electronic musical instrument manufacturers. General MIDI (GM) is an extension of the original MIDI specification (MIDI 1.0). Whereas 1983's MIDI 1.0 did not tie certain musical instrument sounds to certain MIDI codes, GM did just that. Established in 1991, GM assigned 128 musical instrument sounds (timbres) their own 7-bit codes. SoundBlaster-compatible soundcards around the world support GM. [GM Instruments listed at http://www.midi.org/specifications/item/gm-level-1-sound-set]

The GME that you will create will allow the user to "explore" the sounds available in General MIDI. Specifically, the GME will record General MIDI notes coming into the device and will modify and replay the notes on command. The user will interact with the GME to control the modifications by changing the amount of light falling on optical sensors. The light levels on the sensors will modify the sounds that are played back from the GME through the PC.

2. Project Description

The GME has three modes of operation controlled by switches: Record, Playback, Playback with Modification.

Record: When the GME enters recording mode, a new recording is initiated that will overwrite any recording presently stored. To make the new recording the user will send MIDI notes into the GME from the computer. The GME will store to EEPROM the sequence of notes that are received and the timing of the notes. The recording mode should keep count of how many notes have been stored and should stop recording when it reaches its maximum storage capabilities. The recording stops when the GME leaves recording mode, but the stored notes remain in memory until a new recording is made, even if the device is reset.

Playback: When the GME enters playback mode, the notes will be played back from EEPROM memory with approximately the same inter-note timing as in their recording. You may configure the program to enforce a minimum time between playing note on and corresponding note off if the note durations are too short to hear. When the end of the stored recording is reached, the recording will start playing again from the beginning. If the system is reset or turned on when the playback switch set to on, the GME should enter playback mode immediately and begin playing the recorded music from the start.

Playback with Modification: If the playback and modify switches are both on, the playback will be modified depending on the amount of light falling on an optical sensor. The on-chip ADC converts an analog voltage from the photo sensor circuit into a digital value. The digital value controls the speed of playback in a calibrated way. The lab kits contain a second optical sensor, and groups can have fun using the second sensor to modify the instrument, the note played, or another aspect of playback.

3. Design Specification

The system will be built around the ATmega32 AVR microcontroller from Atmel. You will program the device in C using the AVR Studio Integrated Development Environment (IDE). You will use the following AVR on-chip subsystems: Timer1 (a 16-bit timer) with interrupts, EEPROM, Parallel I/O Ports, 10-bit Analog to Digital Converter (ADC), and the USART (Universal Synchronous/Asynchronous Receiver/Transmitter) Serial Port. The most important document you will need to refer to is the 300+ page ATmega32 datasheet. The GME can be based on a loop program structure with timing based on values from 16-bit Timer1 with an appropriate prescaler.
Timing Specification 1: The time between notes during recording can be as long as 4 seconds but will never be longer. You will measure this time based on values of timer1, and therefore timer1 must not overflow within 4 seconds. However, while meeting this requirement, you should also minimize the amount of timing imprecision caused by your choice of prescaler.

Timing Specification 2: Whenever a note is recorded/played, the 7-bit note number is displayed on LEDs. The note remains displayed on the LEDs until the next note is recorded/played, or until 800ms (± 1ms) have elapsed since the start of the note, whichever comes first. The 800ms timer for turning off the LEDs must be based on interrupts triggered by a timer1 event, which is the same timer that counts the inter-note intervals.

Input/Output signals on the ATmega32 AVR microcontroller:

Record Mode – 1 Bit Input Switch – Turns off and on Record Mode.

Play Mode – 1 Bit Input Switch – Turns off and on Playback Mode.

Modify Mode – 1 Bit Input Switch – Turns off and on the playback modifications that are controlled by the analog inputs when the system is in play mode. Switch has no effect when in record mode.

Digital outputs – LED bar shows the last note (i.e. the value of second MIDI byte) recorded or played. Therefore, the LED bar should be changing as the music is being recorded or played back. As mentioned in the specifications, the LEDs turn off after 500ms until additional notes are recorded or played back.

Analog input from optical sensors – analog voltage from a simple voltage divider circuit as shown in slides. Each analog input signal will be connected to one of the eight available Analog to Digital Converter (ADC) inputs on the AVR, say ADC0 (pin 40), ADC1 (pin 39).

MIDI Input – 1 Bit Input Line – MIDI Information. The MIDI input sends the notes into the AVR to store into the AVR’s EEPROM memory. The MIDI input will be received via the USART serial port (pin 14, RXD/PD0).

MIDI Output – 1 Bit Output Line - The MIDI messages will be sent out of the AVR via the USART serial port (pin 15, TXD/PD1).

4. Project Demonstration and Report

At the start of the demo your programmer should be set up to program the board with your program open in AVR studio, MIDI cables should be connected properly, and MIDI-OX should be open and ready to display incoming/outgoing messages from/to your board and to the PC soundcard. Code should be included as a separate file with your report at time of submission. The grading rubric is given in the lab assessment document on the webpage and it provides (in italics) the exact sequence of steps that will be used to evaluate each grade component. Include a self-completed copy of the lab2 assessment form in your Moodle submission to show your understanding of how you’ve done on the lab. A second copy of the assessment will be completed at the demo and used for grading.

The report should answer the questions listed on the scoring rubric in the lab assessment and described here. The writing should be of good quality, but concise.

- List the contributions of each group member.
- A brief explanation of the hardest problem that you had to debug and a description of how you solved it.
- Describe how you’ve satisfied the two timing specifications using timer 1
- Annotated logic analyzer printout/screenshot as described in lab assessment, with indications of how you analyzed the results and determined whether the design functions correctly or not.
- Your report should answer the following questions
  - Report Q1: Given your scheme for measuring timing between notes, what is the maximum imprecision in timing interval between two notes? In other words, if A is the actual time between the start of two notes in a song, and B is the actual time between when you play back the same notes, what is the maximum difference between A and B? How much of that difference is caused by imprecision in the recording timing, and how much is caused by imprecision in the playback timing?
  - Report Q2: Based on the readings from the ADC and basic circuit analysis, calculate the resistance of the photocell under normal conditions, bright conditions, and with the sensor covered. Show your calculations starting from ADC values. After calculating, you can use a multimeter to measure the analog input voltage for purpose of checking your work.