***Microcontroller-Based Eight-Channel Sequencer***

Team Members:

Ryan Jensen – [ryan.jensen-2@mnsu.edu](mailto:ryan.jensen-2@mnsu.edu)

Josh Roal – [joshua.roal@gmail.com](mailto:joshua.roal@gmail.com)

EE336: Principals of Engineering Design I

Project Report

Fall 2013

Minnesota State University, Mankato

**Project Introduction:**

The Microcontroller Based Eight-Channel Sequencer was designed to create tones according to a programmable sequence in order to create basic music. The eight channels will each have a tone and the sequence that they will play in will be programmed into the microcontroller. The sound signals will be generated by the microcontroller as well.

The sequence that the tones will be played is the song data and will be programmed inside the microcontroller and they user will be able to alter it from that access point. The channels and sequence will be displayed using a 8x8 LED display which will scroll the sequence across the matrix with the last column being the output.

**Completed Project Details:**

**Table 1: Calculations for the timer1 prescaler for tone generation.**



In table 1, some calculations are done to select the proper prescaler for timer1 on the ATmega328P. timer1 is used for tone generation because it is the largest of the three timers provided on the ATmega328P. The other two timers are 8-bit timers. Timer1 is a 16-bit register. The extra bit depth is necessary to get a proper range of frequencies.

Timer0 is used to precisely time the beats of the song. The timer starts at 0 and counts up to the value stored in the output-compare register. Once this happens, timer0 resets itself and the song is advanced one beat. So each time timer0 makes a full cycle, the song will advance to the next set of notes.

The 8x8 LED matrix shows the song data. When a note is being played on a particular beat, the LED corresponding to that note is activated on the matrix.

**Project Expectations:**

\*Unsatisfied requirements are in **bold**.

|  |  |
| --- | --- |
| **Requirement ID** | **Requirement Description** |
| MBS-01 | The MBS (Microcontroller Based Sequencer) will have eight output channels. Each channel may be either on/off (high/low) output signals, or audio signals. |
| MBS-02 | Of the eight output channels, the channels that are digital outputs will be fed into circuits/devices that will produce sounds/tones. |
| MBS-03 | All of the sounds generated by the microcontroller or generated by the peripheral circuits that the microprocessor controls will be mixed together before reaching the output. |
| MBS-04 | The device must be able to output the synthesized audio to a 1/8” audio jack. |
| MBS-05 | The MBS will have an 8x8 LED matrix that will display the song data to the user. |
| MBS-06 | Each column of the 8x8 LED matrix (MBS-01) will represent an eighth note in each measure. Each row of the matrix will be associated with one of the eight channels. |
| MBS-07 | The song data will be stored in measures. Each measure will be comprised of eight beats. For each beat in the song, there will be a byte of data stored. Each of the eight bits in the byte will correspond to one of the eight sound channels. If a bit it set (1), the sound on that channel will be played. If the bit is unset (0), the sound on that channel will not be played. |
| MBS-08 | **The user must be able to manually navigate the song data while visually seeing the song data displayed on the 8x8 LED matrix.** |
| MBS-09 | **The user must be able to copy and paste entire measures of song data with simple user input.** |
| MBS-10 | **The user must be able to erase entire measures of song data with simple user input.** |
| MBS-11 | The device will be in an enclosure that will be no larger than 18cm long, 22cm wide, and 8cm deep. |
| MBS-12 | The device should weigh no more than 1.5 kg |
| MBS-13 | The device will operate in the temperature range of 0°C to 40°C |
| MBS-14 | The device will be able to produce an assortment of different waveforms at different frequencies. Sine waves, triangle waves, square waves, and ramp waves are all signals that should be able to be produced at a myriad of different frequencies inside human hearing. |

**Project Budget:**

The budget for the Microcontroller Based Eight-Channel Sequencer is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Cost of Item ($)** | **Quantity** | **Total Cost Of Items ($)** |
| Atmel ATmega328P microcontroller | 2.24 | 1 | 2.24 |
| Atmel ATtiny24A microcontroller | 0.70 | ~4 | 2.80 |
| Atmel AVR Dragon | 52.44 | 1 | 52.44\* |
| LEDs for the 8x8 LED matrix | 0.04 | 64 | 2.56 |
| User interface hardware: buttons, potentiometers, misc LEDs, etc… | -- | -- | 7.50 |
| Audio synth circuit hardware | -- | -- | 7.50 |
| Audio processing hardware | -- | -- | 5.00 |

|  |  |
| --- | --- |
| **COST (minus AVR Dragon programmer): $27.60** | **27.60** |
| **COST: $80.04** | **80.04** |

\*Note: The Atmel AVR Dragon was already owned by a team member so did not contribute to the cost of the project.

**Project Conclusion:**

In the end the team produced a product that satisfied most all of the requirements and remained within the expected budget. However there were some features that the team intended to implement however were not able to produce in the time that the team had to work on the project. These features were the user input interface and some of the more complex sounds for the eight channels.

The user input was intended to be an external interface that allowed the user to navigate the 8x8 matrix and change the note sequence for the eight channels. The team intended to be able to generate more complex sounds like bass notes and high hats so that more complex music could be created using the device. However after research and working on other aspects of the project the team decided they did not have the time required implement these features. The device works in such a way that more complex sound generation circuits can and will be added in the future.

While the more complex sound generation circuits were not built, a simpler noise generator was implemented using a peripheral microcontroller (ATtiny24A) outputting random bytes to an 8-bit DAC. The main microcontroller has the ability to control the noise generator. So the ATmega328P can enable/disable the audio output of the noise signal.

Tone generation was also added to the device’s features. This was implemented using timer1 on the ATmega328P.