**July 28th, 2020.**

**Footnote Program:**

1. Written in C or C++. Executable file that runs in the Win10 command line.
2. Using a LabJack U3-LV interface the program reads 16 inputs from a series of sensors and outputs MIDI commands over USB in response to the input states.
3. We will use the screen for testing and development. The developed program has no screen output.

The LabJack U3-LV in stream mode can scan its inputs at 50K scans per second. We will use the U3 unit in stream mode and run it at that maximum speed. I’m hoping that we can scan all 16 inputs, calculate the MIDI bytes and output the MIDI bytes in 10ms for each iteration of the scanning loop. That means that the latency, the amount of time between hitting a key and hearing a sound, would be 10ms. We could tolerate up to 20ms if we must. The Python script which we wrote that was not complete was taking 75ms for each iteration of the scanning loop, far too long.   
  
We are using one computer and one program to scan two footnote interfaces: one for the right foot and the other for the left foot. Each footnote generates one MIDI note at a time (monophonic). Footnote One generates MIDI on MIDI channel #1, Footnote Two generates MIDI on MIDI channel #2.

Footnote One has the following signals:

1. Slide1, determines MIDI note number, selects one of 49 MIDI notes by slide position
2. Slope1, outputs MIDI CC #102, a 7-bit value proportional to rotation
3. Pressure Pad 1A, outputs MIDI CC #103, a 7-bit value proportional to pressure
4. Pressure Pad 1B, outputs MIDI CC#104, a 7-bit value proportional to pressure
5. Toe Switch 1, outputs either a value of 0 or a value of 127 to MIDI CC#105
6. Keyboard Gate 1A
7. Keyboard Gate 1B
8. Timer1 measuring time interval between Gates 1A and 1B going high which generates a 7-bit value used as the note’s velocity number which determines volume.

Footnote Two has the following signals:

1. Slide2 , determines MIDI note number, selects one of 49 MIDI notes by slide position
2. Slope2 , outputs MIDI CC #106, a 7-bit value proportional to rotation
3. Pressure Pad 2A, outputs MIDI CC #107, a 7-bit value proportional to pressure
4. Pressure Pad 2B, outputs MIDI CC#108, a 7-bit value proportional to pressure
5. Toe Switch 2, outputs either a value of 0 or a value of 127 to MIDI CC#109
6. Keyboard Gate 2A
7. Keyboard Gate 2B
   1. Timer2 measuring time interval between Gates 2A and 2B going high which generates a 7-bit value used as the note’s velocity number which determines volume

In the LabJack U3-LV unit we initialize the device with the following parameters set:

Pins FIO-0 through FIO-7 are set as analog inputs. Pins EIO-0, 1, 2, 4, 5, 6 are set as digital inputs, normally LOW. Pins EIO-3 and EIO-7 are set as Timer1 and Timer2 respectively. DAC0 is set to output 2.4 volts. This is the maximum voltage input to the analog inputs so this is very useful to send to the sensors so that they scale correctly as they are scanned.

**Signal Chart:**

|  |  |  |  |
| --- | --- | --- | --- |
| Signal Name | U3 Pin Name | Program Variable Name | U3 Pin Function |
|  | Footnote One |  |  |
| Slide1 | FIO-0 | V1 | Analog input AIO-0 |
| Slope1 | FIO-1 | V2 | Analog input AIO-1 |
| Pressure Pad 1A | FIO-2 | V3 | Analog input AIO-2 |
| Pressure Pad 1B | FIO-3 | V4 | Analog input AIO-3 |
| Toe Switch1  Keyboard Gate 1A  Keyboard Gate 1B Timer 1 | EIO-0  EIO-1 EIO-2 EIO-3 | TS1 KG1A KG1B T1 | Digital input EIO-0  Digital input EIO-1 Digital input EIO-2 Digital timer EIO-3 |
|  | Footnote Two |  |  |
| Slide 2 | FIO-4 | V5 | Analog input AIO-4 |
| Slope 2 | FIO-5 | V6 | Analog input AIO-5 |
| Pressure Pad 2A | FIO-6 | V7 | Analog input AIO-6 |
| Pressure Pad 2B | FIO-7 | V8 | Analog input AIO-7 |
| Toe Switch2  Keyboard Gate 2A Keyboard Gate 2B  Timer 2 | EIO-4 EIO-5 EIO-6 EIO-7 | TS2 KG2A KG2B T2 | Digital input EIO-4 Digital input EIO-5 Digital input EIO-6 Digital timer EIO-7 |
|  |  |  |  |

The Slide signals, the Keyboard Gate signals, and the Timers will require some calculations. The Slide signal determines the pitch of the note, the Keyboard Gate signals control when the note sounds and the time determines the loudness of the note.

The Slope, Pressure Pads and Toe Switches are all easy and don’t require any calculation. Those four signals are mapped to Midi Continuous Controller Messages. Since those MIDI CC#s are unassigned the user can assign them to control any parameter of sound they wish using external programs.

**Keyboard Gate Logic:**

The Keyboard Gate is generated by two sequentially thrown switches. The switches are only a few millimeters apart so they are thrown with a few milliseconds of time between their sequential closings. This amount of time represents the velocity of the switch closure. This measure amount of time is mapped to the MIDI velocity parameter, a 7 bit value. The speed with which you strike a key determines the loudness of the sounded note.   
  
In each iteration of the scanning loop the following conditions are found:

KG1B = LOW, FALSE. No MIDI note data is sent. All other parameters are scanned and sent.  
  
First KG1A=HIGH, TRUE. Then KG1B=HIGH, TRUE. The MIDI Note # determined by the value of Slide1 is sent as a MIDI NOTE\_ON message. The Note sounds.

We only want the note to be triggered once. At the next iteration of the scanning loop no further MIDI message is sent. Since we have not sent a MIDI NOTE\_OFF message the note is sustained and held.

First KG1B=LOW, FALSE. Then KG1A=LOW, FALSE. The Midi Note # determined by the value of Slide1 is sent as a MIDI NOTE\_OFF message. The Note ceases to sound.

**Milestones:**

1. Set up the LabJack U3 hardware with the sensor emulation box that I am building. Get the Test5.py python script running and see how it works. Get ideas for writing the C program. Perhaps we can accomplish that on the evening of August 3rd.
2. Write C program to set the initial state of the U3 device. Measure 2.4 volts between ground and the output of DAC0. Verify that FIO-0 to FIO-7 are set as analog inputs, verify the digital inputs and the two timers.
3. Develop the C program to replicate the functionality of the Test5.py Python script. Generate and control the 49 MIDI notes.
4. Add the Keyboard Gate functions so that the MIDI notes are gated.
5. Add the timers to calculate and generate velocity control of the MIDI notes.
6. Add the CC MIDI outputs from the Slopes, Pressure Pads and Toe Switches.
7. Add comments and documentation to the source code. Verify that I can modify and correctly recompile the source code to revise the executable file as needed.

Naturally we don’t know how long this will take and what problems we will encounter. I’m hoping we can get this done in 30 hours of programming time spread over two weeks. I will do everything I can to make this possible.

This will all be quite a bit clearer when you have the hardware in hand and are successfully running the Python program. LabJack is excellent with their support. Problem is that hardly anyone is really working with MIDI. I can certainly help you there.

**August 12th, 2020.**  
  
Next step is to implement the keyboard gate functions. These are controlled by variables KG1B and KG2B which are scanning U3 pins EIO-2 and EIO-6 respectively. We will pull down these digital inputs to ground (value = 0) with some 5K ohm resistors and pull them high with +5V from the Vss pin on the U3.  
I will set this up for you when I visit on the afternoon of August 13th.

We will need some sort of note memory function. We need to remember the state of the Keyboard Gate function from the last scanning loop.

The control flow for Footnote #1 is as follows:  
No note memory value. First scan.  
PIN EIO-2 reads high: Result: MIDI note-on signal is sent, note sounds  
Note memory value is set HIGH.

Next scanning iteration:

PIN EIO-2 reads high: Was this pin high the last time I scanned it? If the answer is YES, then no MIDI signal is sent. The scanning loop returns to scan the inputs again.

PIN EIO-2 reads high: Was this pin high the last time I scanned it? If the answer is NO, then a MIDI note-off signal is sent that corresponds in value to the last MIDI note-on signal sent. The note stops sounding.

The control flow for Footnote #2 is the same except we are scanning pin EIO-6.

When this part of the keyboard gate is working we will only hear a single note when we bring one of the keyboard gates high. The note will not be retriggered as it is now. A single note will sound and hold as long as we hold the keyboard gate high. When we let go of the keyboard gate the note will stop sounding.

After this basic keyboard gate function is achieved the next step will be to add the velocity functions. To do this we will measure the amount of time between the closings of Keyboard Gates 1A and 1B, as well as the amount of time between the closings of Keyboard Gates 2A and 2B. That measured time will be mapped to a seven bit value and inserted into the Note-on MIDI statement to give the note a variable velocity.

Note that all of this was solved 44 years ago in 1976 with the introduction of the first touch sensitive synthesizer keyboards (MIDI wasn’t invented yet). We are reverse engineering all of this so that we can apply it to a physical interface that is quite different from a standard keyboard.