MIDI Standard for Dummies - Implementing MIDI Out

Qiushi Jiang
ECE 492 Group 7 Winter 2015

Purpose of This Tutorial:
MIDI short for Musical Instrument Digital Interface is the most widely used standard for storing and communicating digital music data. If you are working on any music related projects, it very likely you will be running into MIDI. This tutorial will show you the basic specification of MIDI standard and guide you through a simple implementation of a MIDI output jack.

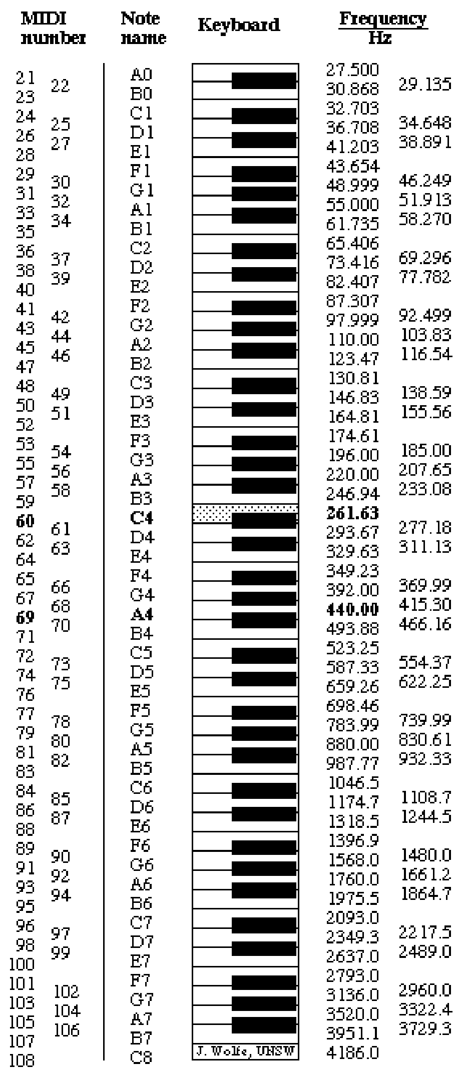
MIDI Standard Crash Course:
<u>Overview:</u>
MIDI is the digital representation of traditional music notation. It records the essential information about each music note such as duration, pitch and velocity.

<u>Data Format</u>:
A single MIDI message contains three bytes: status byte, pitch byte and velocity byte. Additionally there are two separator bits wrapping around each byte, 0 at the front and 1 at the back.

0 [status byte] 1   0 [pitch byte] 1   0 [velocity byte] 1

[Status byte]:
The left 4 bits of the status byte defines the note type, and the lower 4 bit defines which MIDI channel is used. Two most important types are: “note on” (1001) and “note off” (1001). There are total 16 channels you can select, from “channel 1” (0000) to “channel 16” (1111). For example: “10010000” means note on signal on channel 1.

[Pitch byte]:
Pitch byte defines the pitch of the note. Figure 1 shows the mapping between the pitch byte and piano keyboard. For example: C4 is represented as 60 (00111100) inside the pitch byte.


J. Wolfe, UNSW

MIDI number	Note name	Frequency Hz
21	A0	27.500
22		29.135
23	B0	30.868
24	C1	32.703
25		34.648
26	D1	36.708
27		38.891
28	E1	41.203
29	F1	43.654
30		46.249
31	G1	48.999
32		51.913
33	A1	55.000
34		58.270
35	B1	61.735
36	C2	65.406
37		69.296
38	D2	73.416
39		77.782
40	E2	82.407
41	F2	87.307
42		92.499
43	G2	97.999
44		103.83
45	A2	110.00
46		116.54
47	B2	123.47
48	C3	130.81
49		138.59
50	D3	146.83
51		155.56
52	E3	164.81
53	F3	174.61
54		185.00
55	G3	196.00
56		207.65
57	A3	220.00
58		233.08
59	B3	246.94
60	**C4**	**261.63**
61		277.18
62	D4	293.67
63		311.13
64	E4	329.63
65	F4	349.23
66		369.99
67	G4	392.00
68		415.30
69	**A4**	**440.00**
70		466.16
71	B4	493.88
72	C5	523.25
73		554.37
74	D5	587.33
75		622.25
76	E5	659.26
77	F5	698.46
78		739.99
79	G5	783.99
80		830.61
81	A5	880.00
82		932.33
83	B5	987.77
84	C6	1046.5
85		1108.7
86	D6	1174.7
87		1244.5
88	E6	1318.5
89	F6	1396.9
90		1480.0
91	G6	1568.0
92		1661.2
93	A6	1760.0
94		1864.7
95	B6	1975.5
96	C7	2093.0
97		2217.5
98	D7	2349.3
99		2489.0
100	E7	2637.0
101	F7	2793.0
102		2960.0
103	G7	3136.0
104		3322.4
105	A7	3520.0
106		3729.3
107	B7	3951.1
108	C8	4186.0

Figure 1: MIDI Pitch Byte Mapping

[velocity byte]:
velocity byte defines how hard the note is played. With 8 bits the velocity can range from 0 to 127.

<u>Transmitting MIDI signal</u>

- MIDI operates on 31250 Hz clock.
- MIDI signal uses 5V as digital 1 and 0V as digital 0.
- When sending MIDI signal, the bytes' least significant bit are sent first.

  For example: When transmitting "note on /channel 1/ C4/ 64 velocity" (0 <u>10010000</u> 1 0 <u>00111100</u> 1 0 <u>10000000</u> 1) the actual signal sent to the MIDI jack will be (0 <u>00001001</u> 1 0 <u>00111100</u> 1 0 <u>00000001</u> 1).

- MIDI out is fixed at 5V when there is no signal.

<u>Connecting to MIDI Out Jack:</u>

A standard MIDI Out Jack has 5 pins as shown in Figure 2.

- PIN 1: Unused
- PIN 2: Connects to ground
- PIN 3: Unused
- PIN 4: Connects to 5V voltage supply
- PIN 5: Connects to MIDI signal

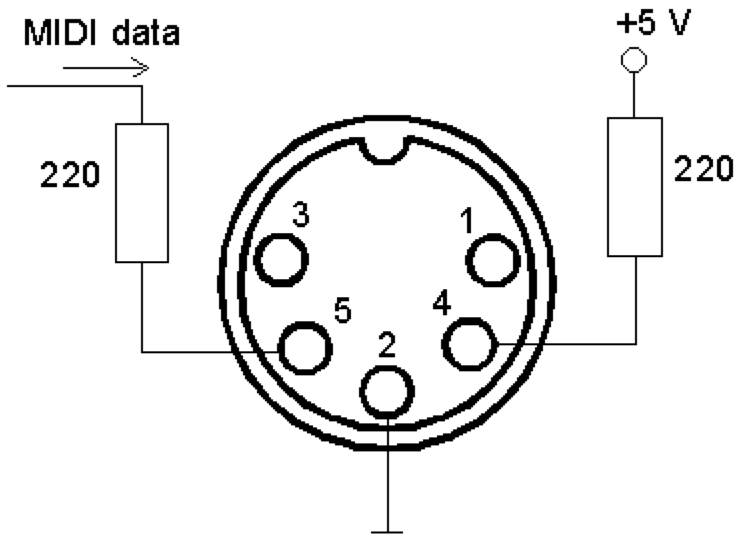
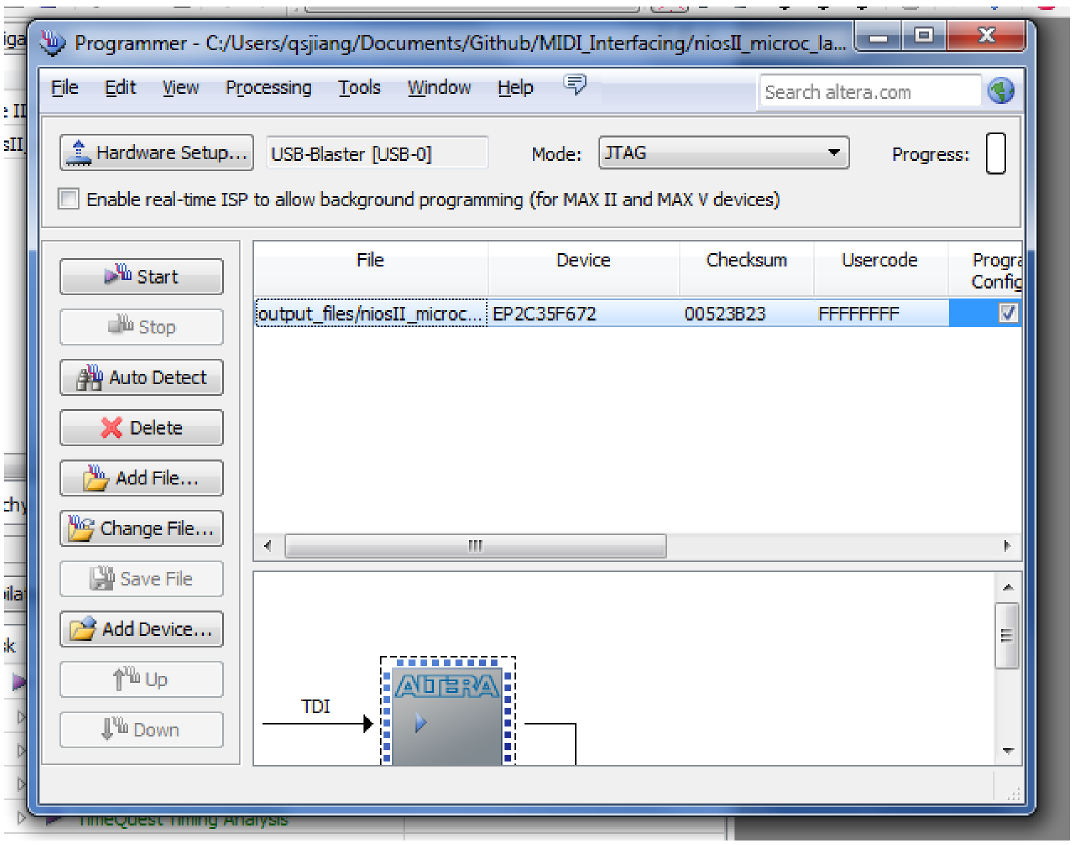


Figure 2. MIDI Out Pins

Setting Up The MIDI Out Demo Project

The demo project servers as an example of what we have introduced above. It has a MIDI Out custom VHDL components receiving MIDI data from Avalon MM, and a software component constructing MIDI data and sending it down to the MIDI Out Avalon MM interface.

1. Download and extract the project zip file.
2. Connect DE2 Board to the computer using USB.
3. Double click “niosII_microc_lab1.qpf” to open “Quartus”.
4. Once inside “Quartus” click on “Program Device”.
5. Select “output_files/niosII_microc...” and click Start.

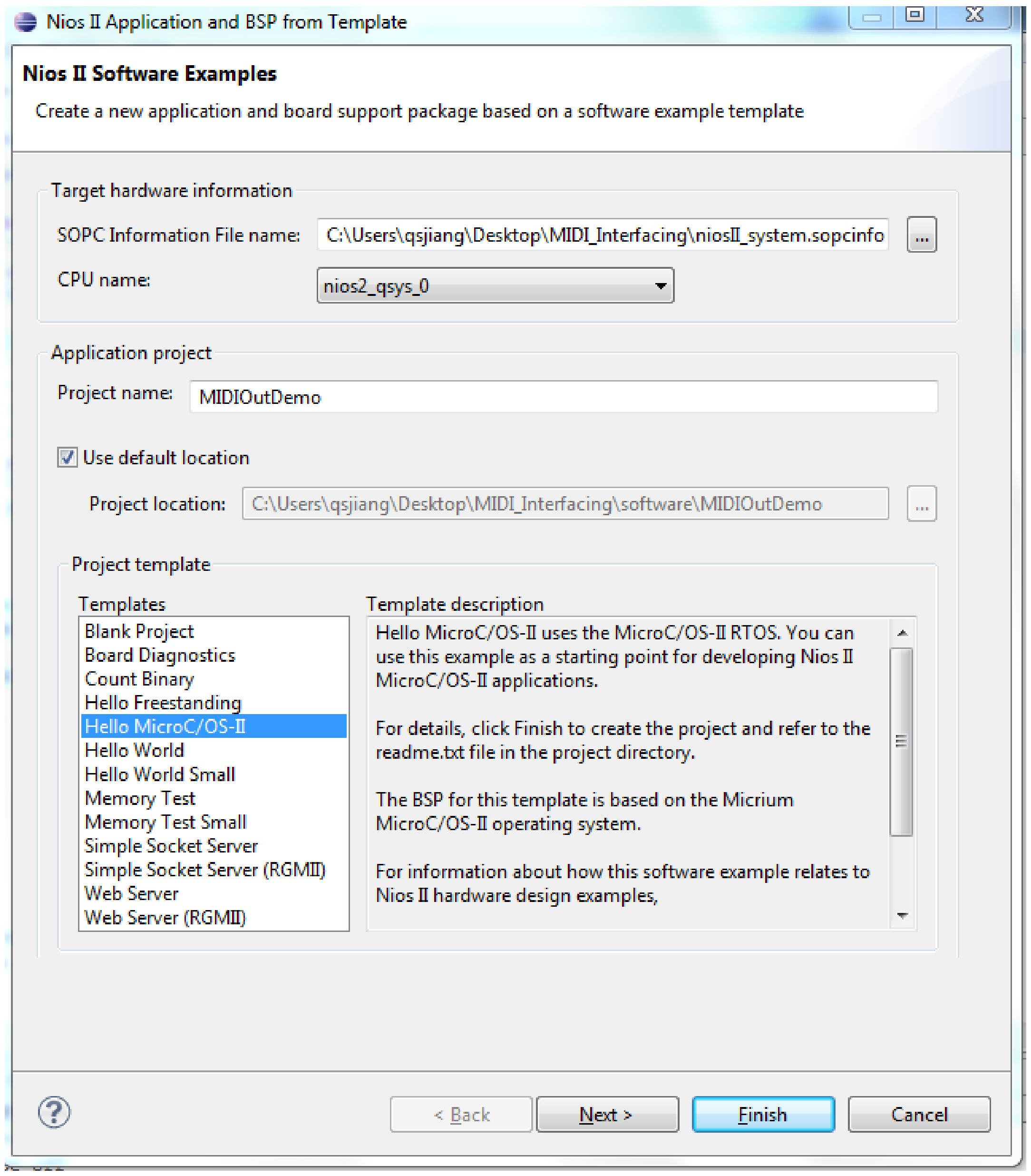

Programmer - C:/Users/qsjiang/Documents/Github/MIDI_Interfacing/niosII_microc_la...

Hardware Setup... USB-Blaster [USB-0] Mode: JTAG Progress:

Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/Configure
output_files/niosII_microc...	EP2C35F672	00523B23	FFFFFFFF	✓

6. After the board is flashed, click on Tools>Nios II Software Build for Eclipse. Set the Eclipse workspace to “software” folder under the project root folder.
7. Create a new project inside Eclipse. File>New>Nios II Application and BSP from Template.
8. Set the SOPC Information File to “niosII_system.sopcinfo” under the project root folder, select “Hello MicroC/OS-II” as our template. Follow the screenshot below and click finish.


Nios II Application and BSP from Template

Nios II Software Examples

Create a new application and board support package based on a software example template

Target hardware information

SOPC Information File name: C:\Users\qsjiang\Desktop\MIDI_Interfacing\niosII_system.sopcinfo

CPU name: nios2_qsys_0

Application project

Project name: MIDIOutDemo

☑ Use default location

Project location: C:\Users\qsjiang\Desktop\MIDI_Interfacing\software\MIDIOutDemo

Project template

Templates
- Blank Project
- Board Diagnostics
- Count Binary
- Hello Freestanding
- Hello MicroC/OS-II
- Hello World
- Hello World Small
- Memory Test
- Memory Test Small
- Simple Socket Server
- Simple Socket Server (RGMII)
- Web Server
- Web Server (RGMII)

Template description

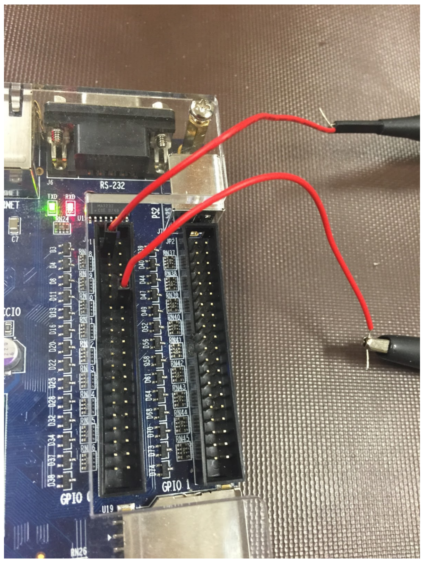
Hello MicroC/OS-II uses the MicroC/OS-II RTOS. You can use this example as a starting point for developing Nios II MicroC/OS-II applications.

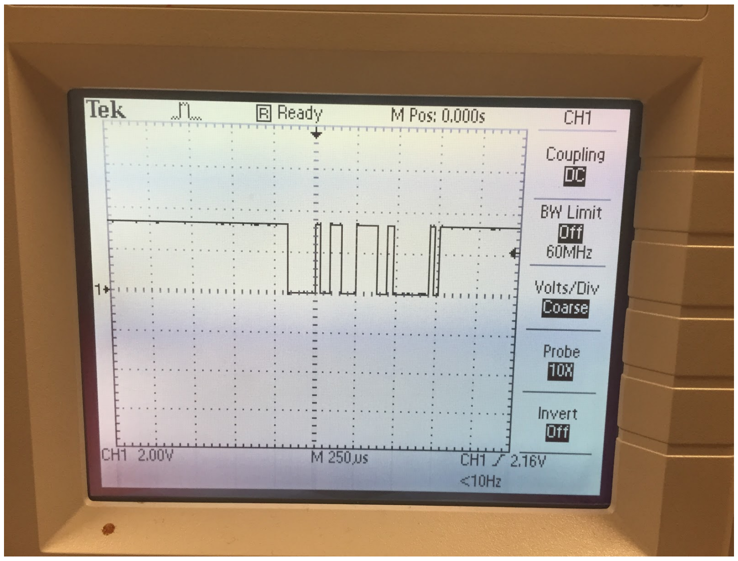
For details, click Finish to create the project and refer to the readme.txt file in the project directory.

The BSP for this template is based on the Micrium MicroC/OS-II operating system.

For information about how this software example relates to Nios II hardware design examples,

9. Copy and paste the code from “/MIDI_Interfacing/software/demo.c” to “hello_ucosii.c” under the newly created project.
10. Right click on the “MIDIOutDemo” inside the navigator and select “Run As> Nios II Hardware”.
11. After the software starts, it will send out MIDI signals on GPIO_0[0] PIN every 1 second.
12. To see the actual MIDI signal, connect an oscilloscope probe to GPIO_0[0] and its ground to GPIO_0[12]. Then set the oscilloscope trigger level anywhere between 0V - 3.3V to capture the MIDI signal.





Additional Voltage Level Shifting Circuit

Since DE2 board's GPIO pin only outputs 3.3V signal, the voltage needs to be shifted to 5V before connecting to MIDI Out Jack. To achieve this you can implement a comparator circuit using op-amp. As shown in Figure 3. Vin is the input voltage, and Vref is the threshold voltage. When Vin > Vref, Vo = VDD. When Vin<Vref, Vo = VEE. For shifting the DE2 GPIO pin to 5V, you can connect VDD to 5V, VEE to ground and Vref to any reasonable threshold between 0 and 3.3V. Then when Vin goes high to 3.3V, the Vo will be 5V, and when Vin in goes low, the Vo will be 0V.

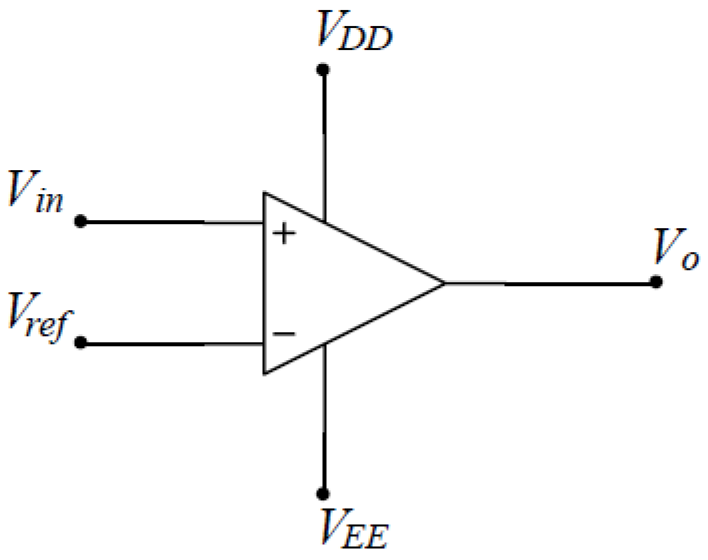


Figure 3. Op-amp comparator circuit

Other Implementation Details

1. MIDI standard actually supports two ways of implementing “note off” signal. One way is setting the top 4 bits of the status byte to “1000”, the other way is setting velocity to 0 and keeping the status byte as “note on”. After reverse engineering several MIDI keyboards, we found that most companies actually use the second approach. That doesn’t mean the first approach is invalid, it just means second approach are probably better supported by other third party devices.

2. Always remember to send “note off” signal after sending the “note on” signal, otherwise the receiving instrument will be stucked on the previous note.

3. MIDI is serial signal, that means technically multiple notes can never be played at the same time, but since each note only takes (1/31250) second/bit * (8 + 2) bit = 0.96ms. You can simply send one right after the other, and human ear would not be able to detect the delay between them.

```
//For example sending a C major Chord
*midiOutPointer = getMidiData(0x90,0x3c,0x40);
 // wait 1 before sending the next note
OSTimeDlyHMSM(0, 0, 0, 1);.
*midiOutPointer = getMidiData(0x90,0x40,0x40);
 // wait 1 before sending the next note
OSTimeDlyHMSM(0, 0, 0, 1);
*midiOutPointer = getMidiData(0x90,0x43,0x40);
```

Reference:

MIDI 1.0 Detailed Specification
http://oktopus.hu/uploaded/Tudastar/MIDI%201.0%20Detailed%20Specification.pdf
Accessed on: April 7, 2015.

Figure 1: MIDI Pitch Byte Mapping retrived from
http://newt.phys.unsw.edu.au/jw/notes.html Accessed on: April 7, 2015.

Figure 2. MIDI Out Pins retrived from http://tryndelka.narod.ru/ Accessed on: April 7, 2015.

Figure 3. Op-amp comparator circuit retrieved from
http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/ Accessed on: April 7, 2015.