Musical Instrument Digital Interface

- Here we look at another way of dealing with sound on a computer the use of MIDI files.
- The MIDI file differs from the “wav” file, because it **does not** usually **contain any information about the sounds** which are played, except for their name and possible allocation.

Why MIDI?

How MIDI changed the world of music?
http://www.bbc.co.uk/news/technology-20425376

Online Sequencer

MIDI

- Usually the **sound card** in a computer **contains** a **sound synthesiser**. This can be played by an external keyboard or a MIDI file.
- Before we look at how a MIDI file is made up, let's look at how MIDI is used to connect and play musical instruments.

What is MIDI ?

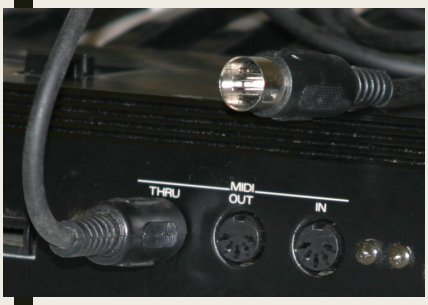
- **M**usical **I**nstrument **D**igital **I**nterface
- Originally devised to allow musical instruments to talk to each other.
- Predominately keyboard, but also wind, guitar, drums.
- Operates in real time along a serial interface similar to RS232.

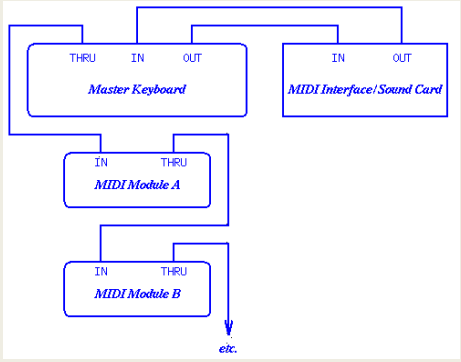
Typical MIDI setup

Hardware Aspects of MIDI

MIDI connectors: - three 5-pin ports on the back of every MIDI unit

MIDI IN: the connector via which the device receives all MIDI data.
MIDI OUT: the connector through which the device transmits all the MIDI data it generates itself.
MIDI THROUGH: the connector by which the device echoes the data receives from MIDI IN.





MIDI & Channels

- Although the MIDI standard allows transfer of audio sample data, MIDI usually relies on the receiving instrument having some inbuilt sound generation ability.
- To play a tune in real time, MIDI sends "note on" and "note off" code followed by the note number.
- A single MIDI connection allows a note to be played on one (or more) of sixteen different channels.
- The channel information is sent with the "note on" and the "note off" information.

Velocity

- Earlier we said that “note on” and “note off” messages are sent. This is a simplification.
- There is a data parameter called **velocity** indicating how forcefully the note is played. In other words **how loud** it is.
  - *Note on 9<n> <kk> <vv>*
  - *Note off 8<n> <kk> <vv >*
    - n *is channel number*
    - kk *is the key number*
    - vv *is the velocity or value*

Example MIDI note sequences

90 3C 7F

- **Key on (9)** channel 1 (n=0) middle C (kk=3C) and **full volume** (vv = **7F** (127))

80 3C 7F

- **Key off (8)** channel 1 (n=0) middle C (kk=3C) and **full volume** (vv = **7F** (127))

***90 3C* 00**

- **Key on (9)** channel 1 (n=0) middle C (kk=3C) and **zero volume** (vv = **00**) is same as **note off**.

Running status

- Do not need to keep sending status byte (9n) if it has not changed.
- So we could send
  - ***90*** *3C 7F 3C 00*
  - ➢ This would turn on middle C with full volume and then
  - ➢ later turn it on with zero volume (that is, turn it off).

Patch changes

- MIDI allows us to **tell the sound** device to **change sounds**
- The code to do this takes the form:
  - *C<n> <pp>*
  - ➢ n is the channel number.
  - ➢ pp is the number of the new program (**sound**).

Controller

- MIDI allows for other controls apart from the keyboard
- Pitch bend, modulation wheel.
- Controller codes take the form :
  - *B<n> <cc> <vv>*
  - n is channel number
  - cc is controller number.
  - vv is value.

Other MIDI messages

- **Channel messages:** messages that are transmitted on individual channels rather that globally to all devices in the MIDI network.
- Examples: Note on, Note off, Pitch bend, etc.
- **System Messages:** messages that are system-wide and not channel specific.
- Examples: Timing signal for synchronization, positioning information in pre-recorded MIDI sequences, and detailed setup information for the destination device.

General MIDI (GM)

- MIDI 1.0 specification consists of communications messaging protocol and a physical interface standard.
- GM imposes several requirements beyond the more abstract MIDI 1.0. It requires that all GM-compatible synthesizers meet a certain minimal set of features:
- GM synthesizers are required to be able to:
- Allow 24 voices to be active simultaneously (including at least 16 melodic and 8 percussive voices)
- Respond to note velocity
- Support all 16 channels simultaneously (with channel 10 reserved for percussion)
- Support polyphony (multiple simultaneous notes) on each channel

MIDI files

- When read by a suitable device the MIDI file can play tunes on MIDI equipped musical instruments through a MIDI connection.
- On personal computers a MIDI file may be used to play the internal sound synthesiser on the sound card without a physical MIDI connection.

MIDI files-How?

- Since the MIDI file is intended to play MIDI equipped instruments, it contains all the control codes mentioned above. That is:
  - *"note on/off",*
  - *controller,*
  - *patch change information etc..*
- However, the file's organisation takes a prescribed form.

MIDI files

- It works with chunks as the "wav" format did.
- It starts with a header chunk and is followed by one or more track chunks:
  - *Header chunk "MThd"*
  - *Track chunk "MTrk"*
- A header chunk provides information pertaining to the entire MIDI file. A track chunk contains a sequential stream of MIDI data which may contain information of multiple channels.

end of notes

(continue if you are interested in more details of MIDI)

Chunks

- All chunks have the following format:

Type	Length	Data
4 bytes ASCII	in bytes	“Length” bytes

Header chunks

- The file starts with a header chuck which contains the following:

Type	Length	Data		
MThd	6	format	tracks	division

- "format" may be 0, 1, or 2.
- "tracks" is the total number of tracks to follow.
- "division" is timing information.
  - $78_{16}=120_{10}$ pulses per quarter note. PPQN: the smallest unit of time used for sequencing note and automation events

Formats

- Format 0
  - *Has only one track*
- Format 1
  - *May contain more than one track. The tracks play simultaneously.*
- Format 2
  - *Can contain many tracks which can play independently.*
- **We will consider Format 1 mainly**.

Format 1 track 1

- Track 1 of a Format 1 MIDI file is also called “tempo map”
- Most important data in the “tempo map” are the meta events:
  - Time Signature, and Set Tempo.
- Meta-events:
  - Sequence/Track Name, Sequence Number, Marker, and SMTPE Offset should also be included.

Timing

- *Time signature.*
  - Beats in a bar
- *“set tempo”*
  - Sets how long in microseconds a crotchet is.
- *“division” information (in header chunk).*
  - Number of pulses or “ticks” per quarter note (crotchet).
- *The music timing is therefore dependant on both “division” and “set tempo”.*

Time signature

- *Time signature.*
  - FF 58 04 *nn dd cc bb*
    nn/2^*dd*
- *cc*
  - MIDI Clocks per metronome tick
- *bb*
  - Number of 1/32 notes per 24 MIDI clocks (8 is standard)

"Set tempo"

- ***Set Tempo***
  - FF 51 03 *tt tt tt*

- *In this case*
  - 09 27 $C0_{16}$ =600000_{10} microseconds
  - = 0.6 seconds per crotchet.

Or 100 crotchets per minute.

Track chunks

- Start with the chunk type in Ascii “MTrk”

Type	Length	Data	
MTrk	length	delta_time	event
		delta_time	event …

- Data consists of (delta_time, event) pairs.
- length is the number of bytes in the data.

Track chunks

- delta_time is the time from the last event to this event measured in “ticks”.
- Variable length quantities
  - *Way to avoid allocating too much space for variables.*
  - *A delta time could be 120 or 100000*
  - *So it may take 1 byte or 4 bytes.*
  - *But we do not want to confuse it with other codes.*
  - *7 bits of a byte are used for data, but the MSB determines where the start of a variable length quantity is. It is set to the value one for the preceding bytes and is set to zero for the last byte.*

Events

- Channel events,
  - *the (notes on and off) and controller changes.*
- Meta events
  - *Information about the file like text, tempo, copyright, time signature.*
- System exclusive events (sysex)
  - *Information specific to manufactures machines.*
  - *Not considered here.*

Some channel events

- Channel events,
  - *the (notes on and off) and controller changes.*
- Meta events
  - *Information about the file like text, tempo, copyright, time signature.*
- System exclusive events (sysex)
  - *Information specific to manufactures machines.*
  - *Not considered here.*

Some meta Events

- FF 01 <*len*> <*text*>
  - ***Text Event***
- FF 03 <*len*> <*text*>
  - ***Sequence/Track Name***
- FF 04 <*len*> <*text*>
  - ***Instrument Name***
- FF 2F 00
  - ***End of Track***
- FF 51 03 *tt tt tt*
  - ***Set Tempo***
- FF 58 04 *nn dd cc bb*
  - ***Time Signature***

Exercises for you

- Determine:
  - *The time signature of demo.mid.*
  - *The tempo of demo.mid*
- Change the piano in MIDI file demo to a guitar.
- Double the tempo by:
  - *Altering the "division" value*
  - *Altering the "set tempo" value.*
- Restore the tempo by altering the division value, but adjusting the "set tempo" value to compensate.
- Equally space the bass notes in time.

Useful links

http://www.kenschutte.com/MIDI

readMIDI.m	reads MIDI file into a Matlab structure
MIDI2audio.m	convert the Matlab MIDI structure into an audio vector
MIDIInfo.m	display detailed information about MIDI Matlab structure
synth.m	synthesize a single note
getTempoChanges.m	finds times of all tempo changes
MIDI2freq.m	convert MIDI note number (0-127) to frequency in Hz
matrix2MIDI.m	create MIDI Matlab structure from a matrix with information on individual notes
writeMIDI.m	write a MIDI Matlab structure (such as created by readMIDI) to a MIDI file

Reading a MIDI and convert to wav

```
clc
clear all
%% synthesize with FM-synthesis.
% (y = audio samples.  Fs = sample rate.  Here, uses default 44.1k.)
MIDI=readMIDI('jesu.mid');
[y,Fs] = MIDI2audio(MIDI);

%% listen in matlab:
soundsc(y, Fs);  % FM-synth
% There are two other very basic synth methods included:
y = MIDI2audio(MIDI, Fs, 'sine');
soundsc(y,Fs);
y = MIDI2audio(MIDI, Fs, 'saw');
soundsc(y,Fs);
% save to file:
% (normalize so as not clipped in writing to wav)
y = .95.*y./max(abs(y));
wavwrite(y, Fs, 'out.wav');
```

Simple example of creating a chromatic scale

```
%% initialize matrix:
N = 13;  % number of notes
M = zeros(N,6);

M(:,1) = 1;                          % all in track 1
M(:,2) = 1;                          % all in channel 1
M(:,3) = (60:72)';                   % note numbers: one octave starting at middle C
(60)
M(:,4) = round(linspace(80,120,N))'; % lets have volume ramp up 80->120
M(:,5) = (.5:.5:6.5)';       % note on:  notes start every .5 seconds
M(:,6) = M(:,5) + .5;        % note off: each note has duration .5 seconds

MIDI_new = matrix2MIDI(M);
writeMIDI(MIDI_new, 'testout.mid');

%% listen in matlab:
[y,Fs] = MIDI2audio(MIDI_new);
soundsc(y, Fs);  % FM-synth - the default
```

Random Notes

Creating some random music: let's say 200 random notes with random start times and volumes

```
%% initialize matrix:
N = 200;
M = zeros(N,6);

M(:,1) = 1;                              % all in track 1
M(:,2) = 1;                              % all in channel 1
M(:,3) = 30 + round(60*rand(N,1));       % random note numbers
M(:,4) = 60 + round(40*rand(N,1));       % random volumes
M(:,5) = 10 * rand(N,1);
M(:,6) = M(:,5) + .2 + rand(N,1);    % Radom duration .2 -> 1.2 seconds

MIDI_new = matrix2MIDI(M);
writeMIDI(MIDI_new, 'testout2.mid');

%% listen in matlab:
[y,Fs] = MIDI2audio(MIDI_new);
soundsc(y, Fs);  % FM-synth - the default
```

MIDIInfo

```
>> midi = readmidi('jesu.mid');>>
midiInfo(midi);
------------------------------------------------
Track 1
------------------------------------------------

-     0        0:0.000      meta     Set Tempo          microsec per quarter note: 500000
-     0        0:0.000      meta     Time Signature     4/2, clock ticks and notated 32nd notes=24/8
0     751      0:0.978              Note on            nn=43  vel=99
0     171      0:1.201              Note on            nn=67  vel=68
0     133      0:1.374              Note on            nn=67  vel=0
0     29       0:1.411              Note on            nn=69  vel=75
0     91       0:1.530              Note on            nn=69  vel=0
   .
   .
   .
```

MIDIInfo & Notes

MIDIInfo also returns a matrix of size Nx8, where N is the number of notes in the file. The 8 colunms indicate:

1. track number
2. channel number
3. note number (MIDI encoding of pitch)
4. velocity
5. start time (seconds)
6. end time (seconds)
7. message number of note_on
8. message number of note_off

MIDIInfo & Notes

```
>> midi = readmidi('jesu.mid');>>
Notes = midiInfo(midi,0);>>
whos Notes
  Name        Size                    Bytes  Class

  Notes      90x8                     5760  double array

Grand total is 720 elements using 5760 bytes

>> Notes(1:5,:)

ans =

    1.0000         0   43.0000   99.0000    0.9779    1.6120    3.0000    8.0000
    1.0000         0   67.0000   68.0000    1.2005    1.3737    4.0000    5.0000
    1.0000         0   69.0000   75.0000    1.4115    1.5299    6.0000    7.0000
    1.0000         0   71.0000   88.0000    1.6250    1.8268    9.0000   12.0000
    1.0000         0   55.0000   88.0000    1.6354    2.1979   10.0000   16.0000
```