E85.-2603 Digitally-Controlled Music Systems

Juan P Bello

Juan P. Bello

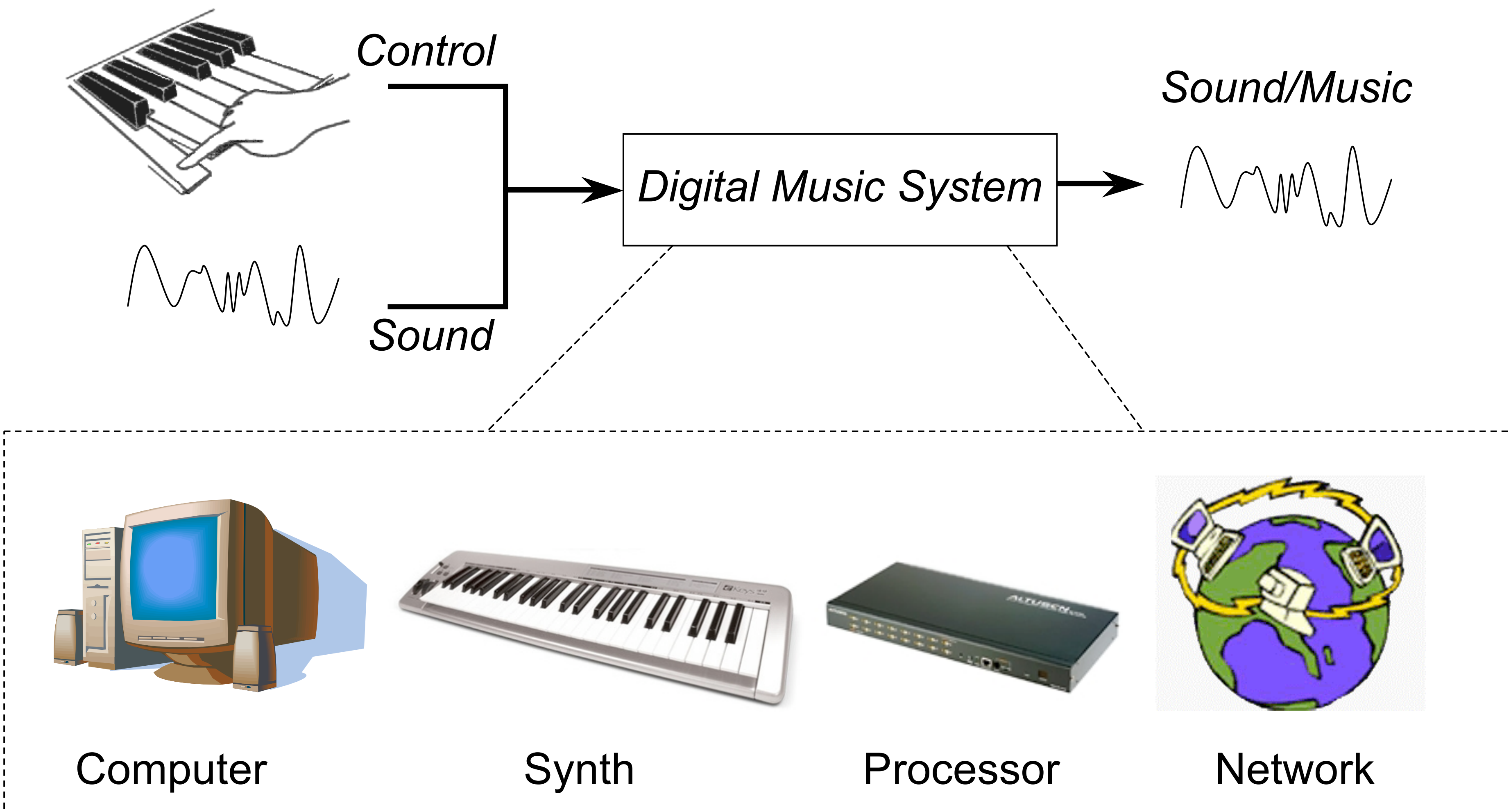
- Office: 409, 383 LaFayette Street (ext. 85736)

- Office Hours: Wednesdays 2-4pm

- Email: jpbello@nyu.edu

- Course-info: Mondays 6.45-8.25pm (Studio F)
  E85.2603: Digitally-Controlled Music Systems
  http://www.nyu.edu/classes/bello/DCMS.html

Digitally-controlled music systems



Lectures tentative schedule

- 01/28 & 02/04: Basics of digital systems / MIDI / MAX
- 02/11: DSP Fundamentals / MSP
- 02/18: PRESIDENT'S DAY ---------------
- 02/25: DSP Fundamentals / MSP Part II -- Interactive Arts Performance Series (8pm) --
- 03/03-10: Sound Synthesis
- 03/17: SPRING BREAK -------------
- 03/24: Mid-term examination
- 03/31: Invited talk (TBA)
- 04/07: The Internet
- 04/14: Audio/Music Compression
- 04/21: Music Information Retrieval
- 04/28 & 05/05: Project Presentations

Evaluation

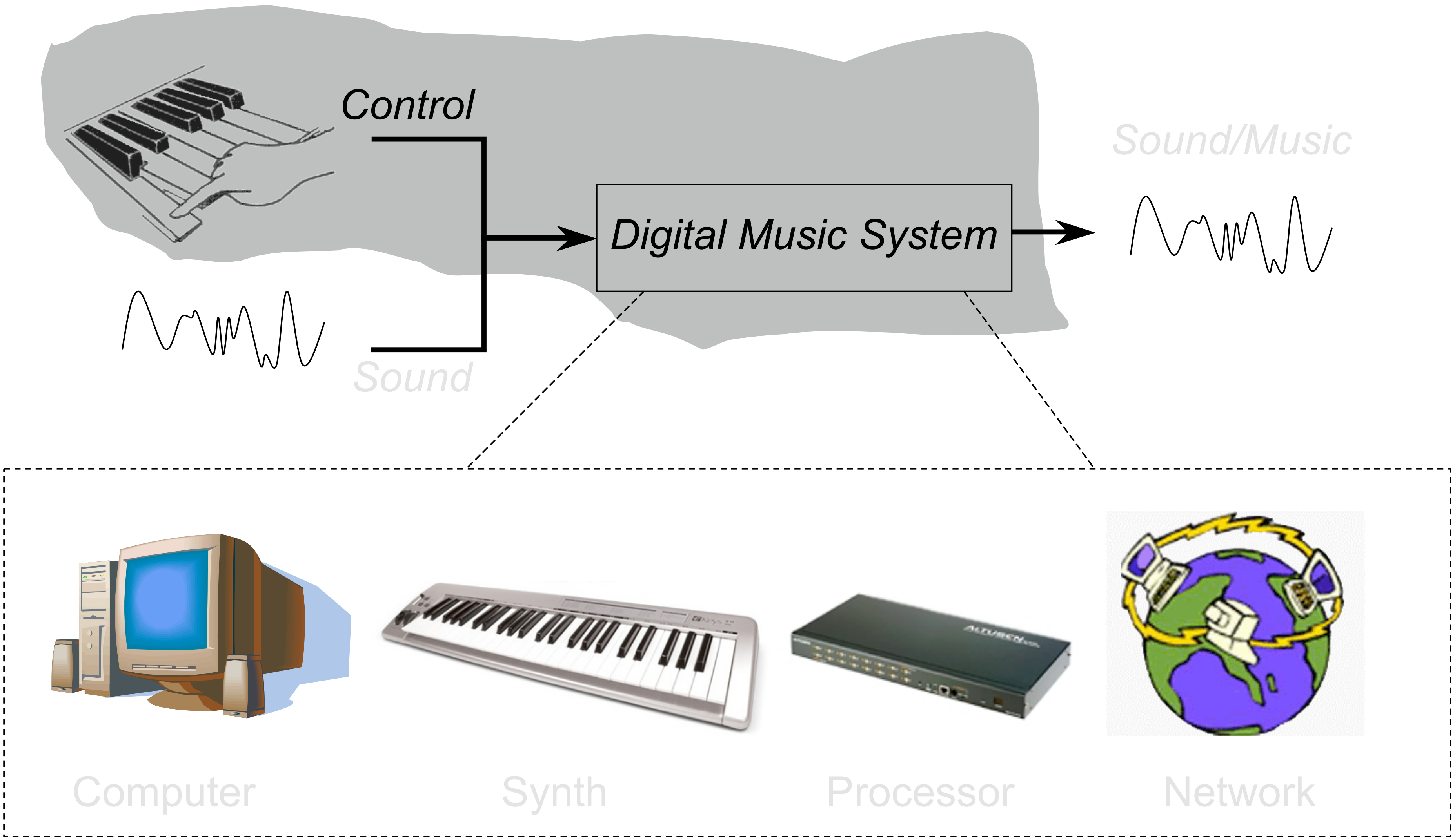
- Mid-term examination (30%): 03/24

- Assignments (30%)
  - Assignment # 1 (Due date Mon. 02/25): 5%
  - Assignment # 2 (Due date Mon. 03/10): 5%
  - Assignment # 3 (Due date Mon. 03/24): 10%
  - Assignment # 4 (Due date Mon. 04/07): 10%

- Group project (40%)
  - 03/31: Project and group definition
  - 04/28 & 05/05: Project Presentations

- Attendance and Participation (?%)
- Interactive Arts Performance Series: 02/25 8pm Loewe Theater

Basics of digital systems / MIDI



Binary system

- Digital systems represent information using a binary system, where data can assume one of only two possible values: zero or one.

- Appropriate for implementation in electronic circuitry, where values are characterized by the absence/presence of an electrical current flow.

- The binary system represents numbers using *bi*nary digi*ts* (*bits*) where each digit corresponds to a power of two.

binary	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**
Power of two	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
decimal	128	64	32	16	8	4	2	1

- The total (in decimal) is *128* + *64* + *32* + *4* + *1* = *229*

- Since we begin counting from zero, n bits can represent 2^n values: from 0 to 2^n-1 inclusive (e.g. 256 values, from 0 to 255, for 8 bits).

Binary system

MSB x 2^7

LSB x 2^0

binary	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**
Power of two	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
decimal	128	64	32	16	8	4	2	1

4 bits = 1 *nybble*

4 bits = 1 *nybble*

8 bits = 1 *byte (from by eight)*

byte	*byte*	*byte*
MSbyte		LSbyte

Hexadecimal numbers

- To avoid writing down long binary words, it is often easier to use hexadecimal (base 16) notation (aka Hex).
- Values in MIDI implementation charts are often expressed as hexadecimal numbers. Each symbol representing 1 nybble, that can take one of 16 different values.
- These values are represented as follows: the values 0-9 are represented by the digits 0-9, and the values 10-15 are represented by the capital letters A-F.

Bin	Hex	Dec	Bin	Hex	Dec	Bin	Hex	Dec	Bin	Hex	Dec
0000	0	0	0100	4	4	1000	8	8	1100	C	12
0001	1	1	0101	5	5	1001	9	9	1101	D	13
0010	2	2	0110	6	6	1010	A	10	1110	E	14
0011	3	3	0111	7	7	1011	B	11	1111	F	15

MIDI

- The Musical Instrument Digital Interface (MIDI) is a standard for communication between electronic musical instruments, although it is now applied to a larger range of equipment (Fx units, mixers, light control, etc)
- Main functions: transmitting performance/control data around a DCMS. Also used to transmit other data such as timing info, set-up parameters, samples, etc.
- The MIDI standard has two parts: the hardware physical interconnection scheme, and the communications code to be transmitted.
- The MIDI 1.0 standard was formulated by agreement between the manufacturers. Its core remains largely unchanged, although several functionalities have been added over time.
- MIDI was designed to be simple and easy to install in economic equipment, and widely available to as many users as possible.

MIDI communication

- MIDI is a uni-directional ***serial*** interface that uses ***asynchronous*** communication
- Serial vs Parallel: 1 bit per clock vs *n* bits per clock

Serial:

Transmitter → Receiver
101001 → 11

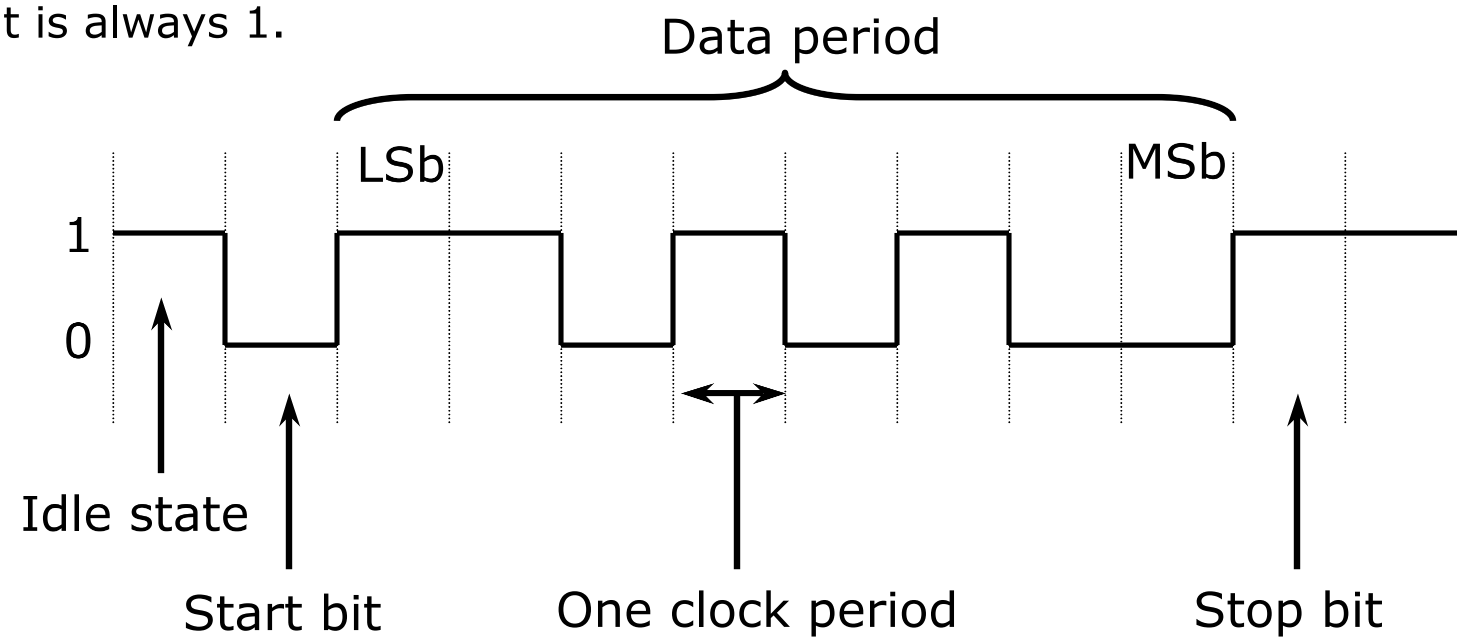
Paralell:

Transmitter	→	Receiver
1	→	1
1	→	1
1	→	1
0	→	0
0	→	0
1	→	1
0	→	0
1	→	1

- Serial interface, although slower than parallel, allows for simple connectors and cabling (simple implementation and low cost).
- Speed of communication: 31.25K bits/s (bauds). As a reference USB 1.1 low-speed transmit at 1.5Mbauds, USB-2 at 480Mbauds

MIDI communication

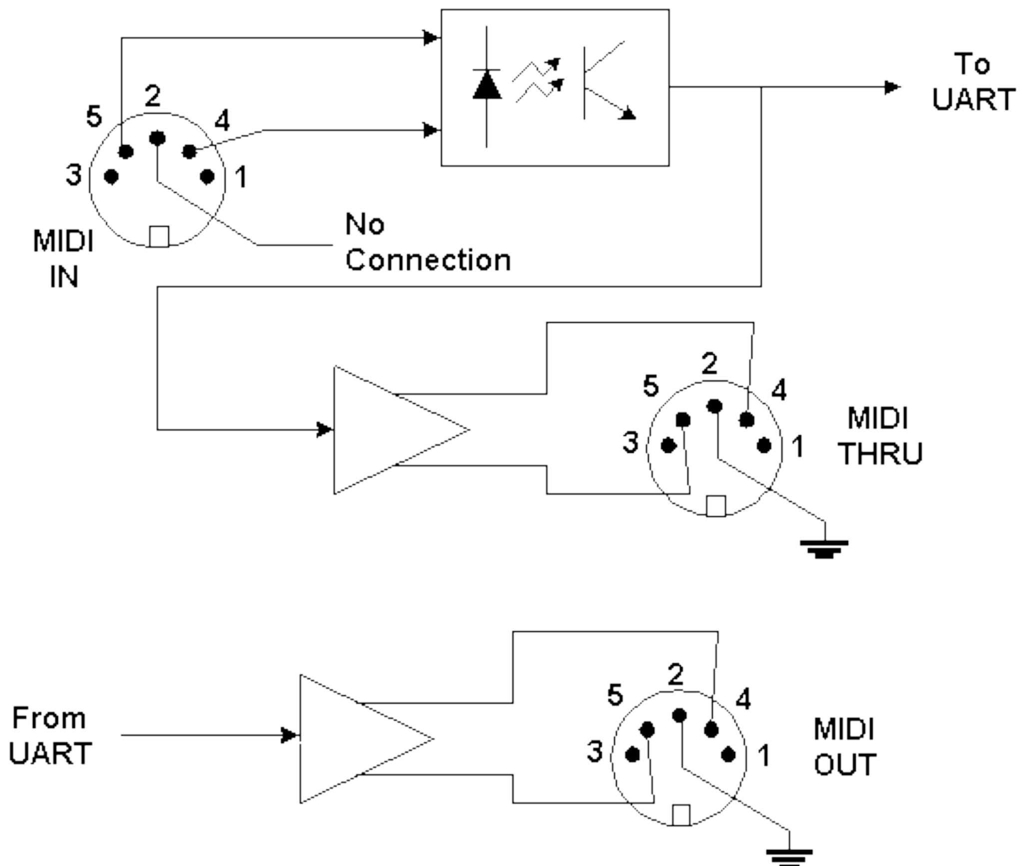
- In asynchronous serial communications the transmitter sends nothing but a serial data stream and (usually) needs no acknowledgement.
- Uses start and stop bits to define the data boundaries.
- In MIDI a binary 0/1 is defined by current flowing/not flowing on the current loop. Thus the idle state is binary 1, start bit is always 0 and stop bit is always 1.



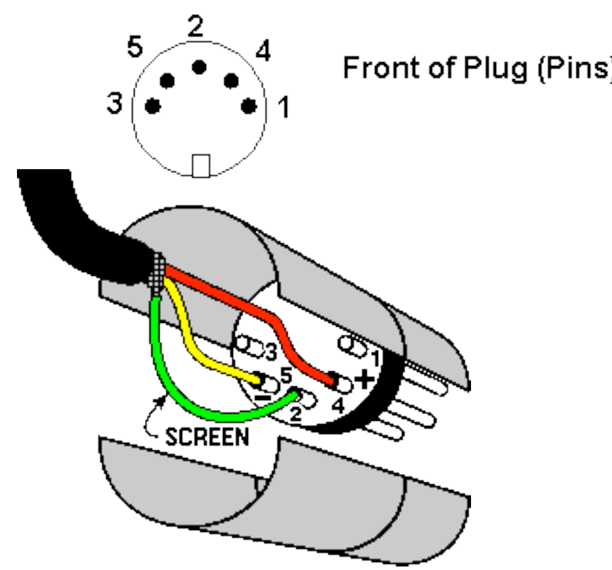
- Every MIDI byte transmitted/received is coded as a “10-bit byte”.
- Clocks must run at exactly the same rate (1% tolerance in MIDI)

MIDI hardware

- The hardware uses cables terminated in 180-degree 5-pin DIN connectors, of which only three pins are used (5, 4 and 2).
- There are 3 kinds of MIDI ports: IN, THRU, and OUT. The IN port accepts input to a device, the THRU port passes an amplified copy of the input signal along, and the OUT port is used to transmit the device’s output.

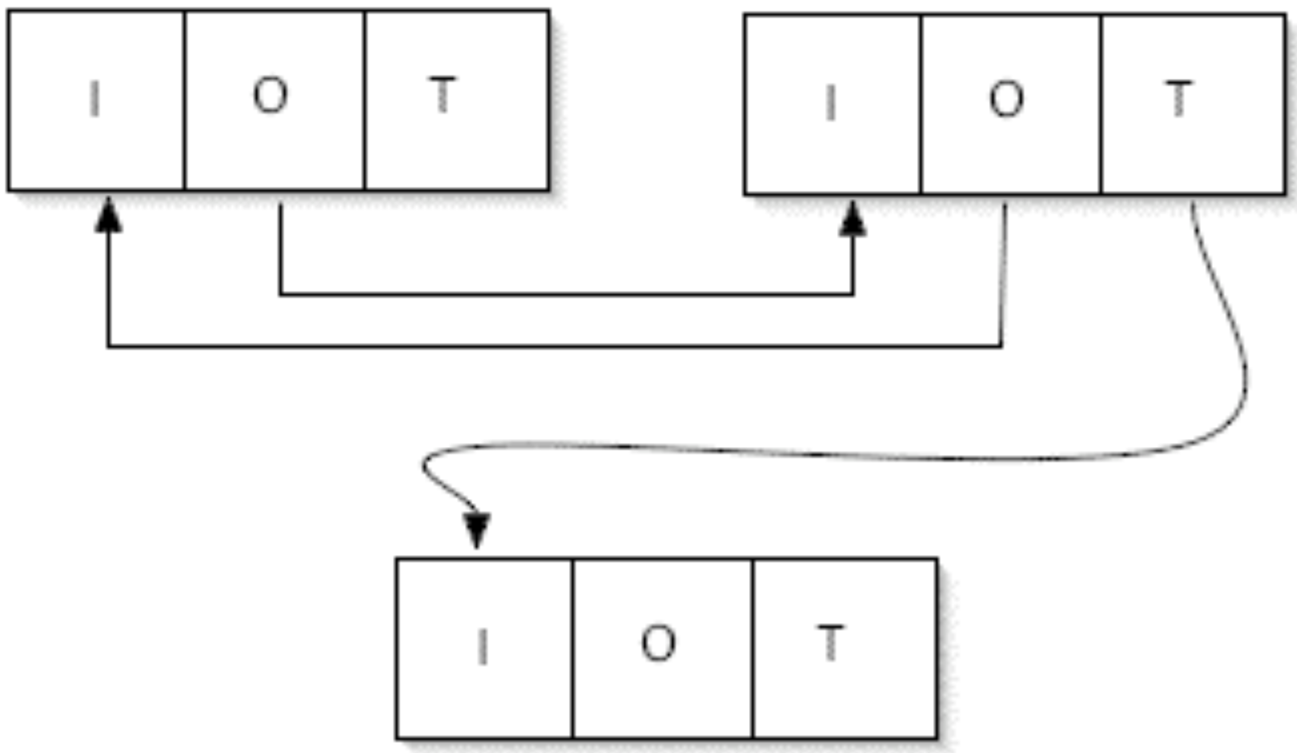


Universal Asynchronous Receiver/Transmitter (UART) parallelizes serial input



MIDI hardware

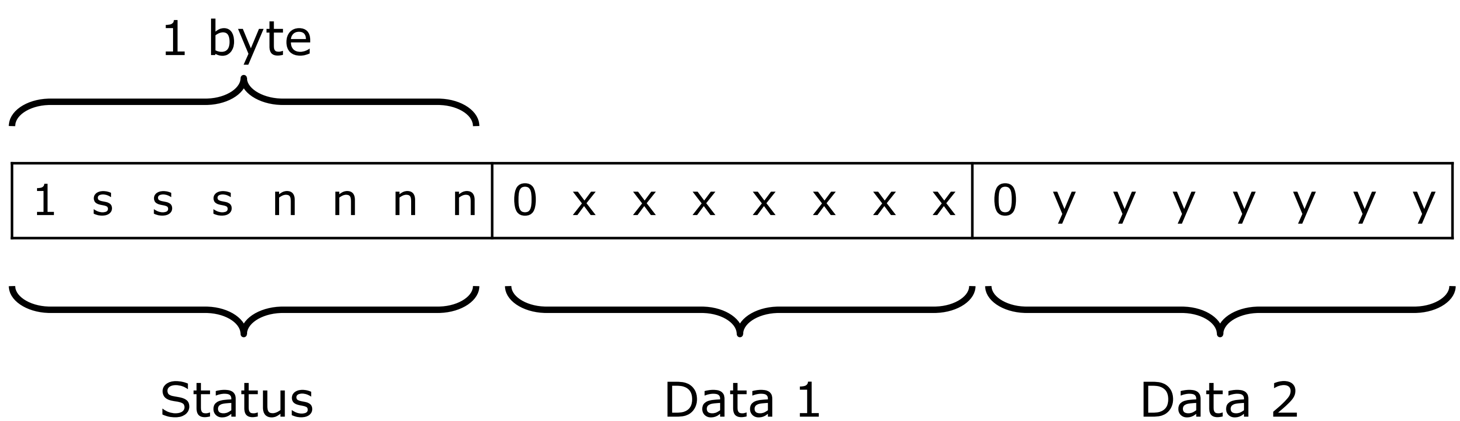
- The IN port accepts input to a device, the THRU port passes an amplified copy of the input signal along, and the OUT port is used to transmit the device’s output.



- Too many instruments in series or too-long cables (> 15m) can cause susceptibility to interference/distortion

MIDI Code: the message format

- 2 types of MIDI message bytes: the status byte and the data byte
- Status bytes always begin with 1, and data bytes with 0. That leaves only 7 bits per byte to represent the message (128 possible values).
- MIDI messages begin with the status byte, where 3 bits (*sss*) are used to denote the type of message, and 4 bits (*nnnn*) to denote the channel number to which the message apply (max. 16 channels).



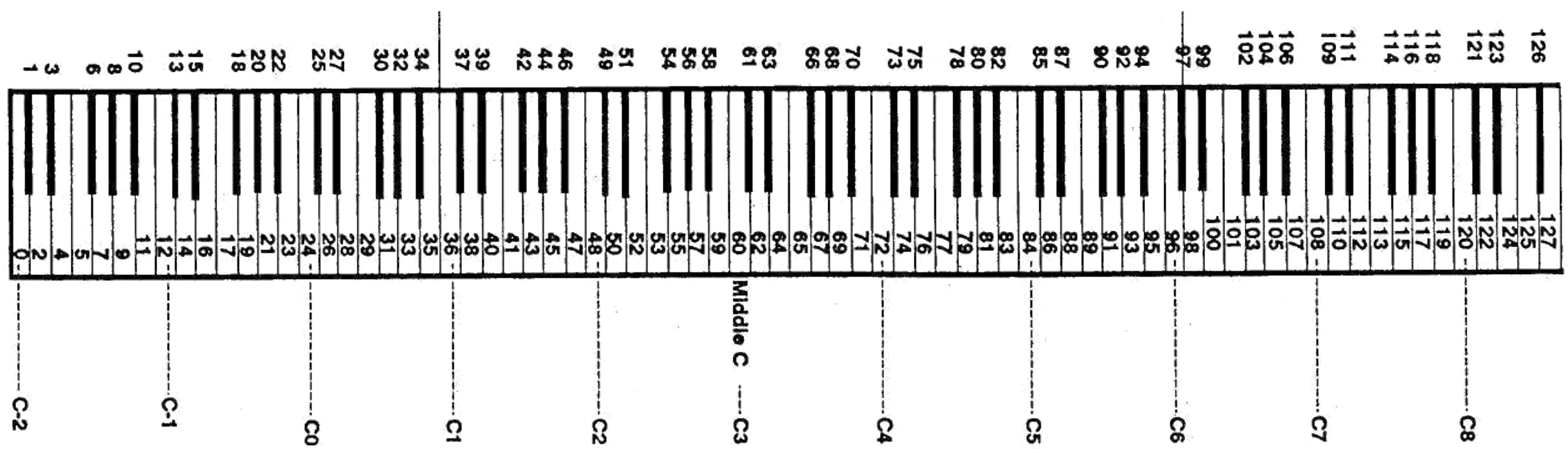
Channel messages

Message	Status	Data 1	Data 2
Note off	8n	Note number	Velocity
Note on	9n	Note number	Velocity
Polyphonic aftertouch	An	Note number	Pressure
Control change	Bn	Controller number	Data
14-bit controllers MSB	Bn	01-20 (controllers)	Data
14-bit controllers (LSB)	Bn	21-3F (same order)	Data
7-bit controllers/switches	Bn	40-78	Data
Channel modes	Bn	79 (reset all controllers)	7F
	Bn	7A (local)	00 off/7F on
	Bn	7B (all notes off)	00
	Bn	7C (omni off)	00
	Bn	7D (omni on)	00
	Bn	7E (mono)	00-0A
	Bn	7F (poly)	00
Program change	Cn	Program number	-
Channel aftertouch	Dn	Pressure	-
Pitch wheel	En	LSbyte	MSbyte

Channel Voice messages

The channel number (*n*) range is *0-F* (e.g. note on ch.5: *&94*)

```
STATUS       DATA BYTES             Description
====================================================
1000nnnn     2 (pitch, velocity)  Note Off event
1001nnnn     2 (pitch, velocity)  Note On event
1010nnnn     2 (pitch, pressure)  Polyphonic aftertouch
1011nnnn     2 (id, value)        Control change
1100nnnn     1 (program)          Program change
1101nnnn     1 (pressure)         Channel aftertouch
1110nnnn     2 (LSB, MSB)         Pitch Bend
```



Channel Voice messages

- *Note on, velocity zero* is equivalent to *note off*.
- It is convenient when large amounts of data are sent to the MIDI bus (e.g. a high-polyphony chord)
- Normally we will need 6 bytes for each note of a chord: *[9n][pitch][velocity]* and *[8n][pitch][velocity]*
- Instead we can clutter *note on* and *off* messages together: *[9n][pitch][vel][pitch][vel]…[pitch][vel]*
- This is known as running status
- For a 4-note chord it means 17 bytes are transmitted rather than 24 bytes (assuming running status remains unchanged).

Channel Mode messages

- Channel Mode Messages are a special case of the Control Change Channel Voice Message (status byte B*n*).
- They set the mode of operation of the instrument receiving on channel *n*.
- A change of channel mode should turn all notes off automatically

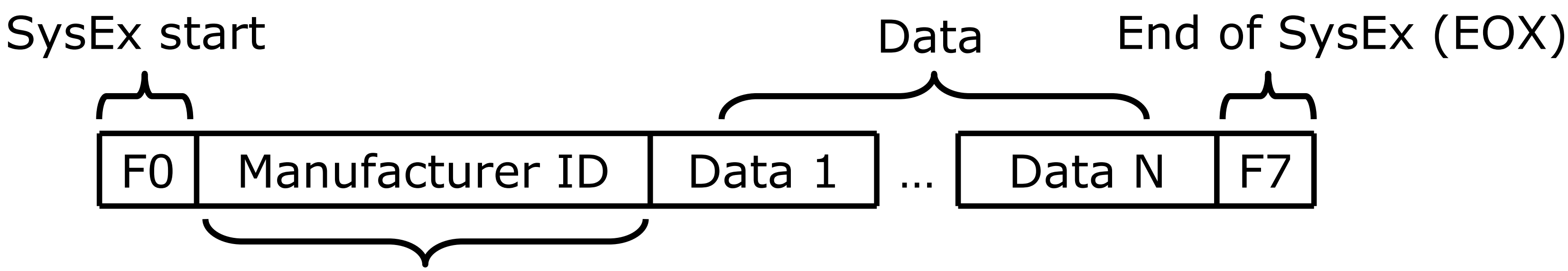
Status	Index	Argument	Description
Bn	79	7F	Reset All Controllers
Bn	7A	7F on/00 off	Local Control On/Off
Bn	7B	00	All Notes Off
Bn	7C	00	Omni Mode Off (All Notes Off)
Bn	7D	00	Omni Mode On (All Notes Off)
Bn	7E	channels	Mono Mode On (Poly Mode Off)
Bn	7F	00	Poly Mode On (Mono Mode Off)

System messages

Message	Status	Data 1	Data 2
System exclusive			
System exclusive start	F0	Manufacturer ID	Data … (Data)
End of system exclusive	F7	-	-
System common			
Quarter frame (MTC)	F1	Data	
Song pointer	F2	LSbyte	MSbyte
Song select	F3	Song number	-
Tune request	F6	-	
System Real-time			
Timing clock	F8	-	-
Start	FA	-	-
Continue	FB	-	-
Stop	FC	-	-
Active Sensing	FE	-	-
Reset	FF	-	-

System Exclusive

- SysEx messages are used for device-specific data transfer.
- Except for SysEx for Universal Info, the standard only defines how messages begin and end.
- A return link is necessary for *Handshaking*.
- Error checking (checksum) is applied to long data dumps



Manufacturers ID		*Universal information*		
Yamaha	43	Universal non-realtime	7E	Sample dumps
Roland	41	Universal realtime	7F	MTC
Akai	47			

System Exclusive

- Manufacturer-specific SysEx messages are mainly used for dumping/loading the settings of a device and for remotely controlling its parameters.

- Universal non-realtime messages include Sample/MIDI file dump, General MIDI on/off, Inquiry requests, MIDI Tuning standard.

- Universal realtime messages include MIDI timecode (MTC), MIDI Machine/Show control, Master Volume/Balance control, Bar/Time-signature markers.

- Timecode is a digital code, used in the audiovisual industries, that represents time in terms of hours, minutes, seconds and frames (hh:mm:ss:ff)

SysEx and MIDI timecode

F0	7F	Chan	01	subID	hh	mm	ss	ff	F7

- A SysEx Full-frame message transmits MTC: 7F is the universal realtime identifier, channel number (set to 7F for the whole system), 01 identifies the message as a MTC message, and a subID of 01 refers to full-frame message.
- Hours are coded as [0 qq ppppp], where *qq* represents the time-code frame rate (in fps):

  00 = 24 frame (used in films)

  01 = 25 frame (EBU: PAL and SECAM TV pictures in Europe/Australia)

  10 = 30 drop-frame (SMPTE drop frame: NTSC color TV in the US/Japan)

  11 = 30 non-drop-frame (SMTPE: monochrome US TV pictures)
- *ppppp* represents hours from 00 to 23. Minutes, seconds and frames are represented by a byte each
- SysEx can also transmit MTC origination information (user bits) and MTC cue messages.

System common messages

- Like SysEx Universal messages, SCM are intended for the attention of all systems

Status	Data 1	Data 2	Description
F1	Data	-	Quarter-frame (MTC)
F2	LSbyte	MSbyte	Song Pointer
F3	Song number	-	Song Select
F6	-	-	Tune Request

- Song select determines which pre-recorded sequence of MIDI data (song) from a collection is to be accessed.
- Song pointer directs devices towards a particular location in a song (in beats from the beginning of the song).
- Tune request asks synthesizers to re-tune themselves to a pre-specified reference.

System common messages

- SCM also provide an alternative to transmitting MTC info: the *quarter-frame message*.
- This message consists of 1 status byte (F1) and 1 data byte.
- As timecode info is much longer than this, then we split the message into 8 separate messages transmitting the frames' LS/MSnybbles, seconds' LS/MSnybbles, minutes' LS/MSnybbles and hours' LS/MSnybbles.
- The message type is encoded in the first data nybble (as 0-7)
- Hour information is encoded as in SysEx MTC, including frame rate info.
- Despite the message name, the MTC is updated every two frames (causing a delay at the receiver).
- At 30 fps, and 8 messages to represent a frame, we need to send 120 messages per second.
- If transmitted continuously this takes ~7.7% of the data bandwidth
- SysEx full-frame messages can be used to avoid this.

System realtime messages

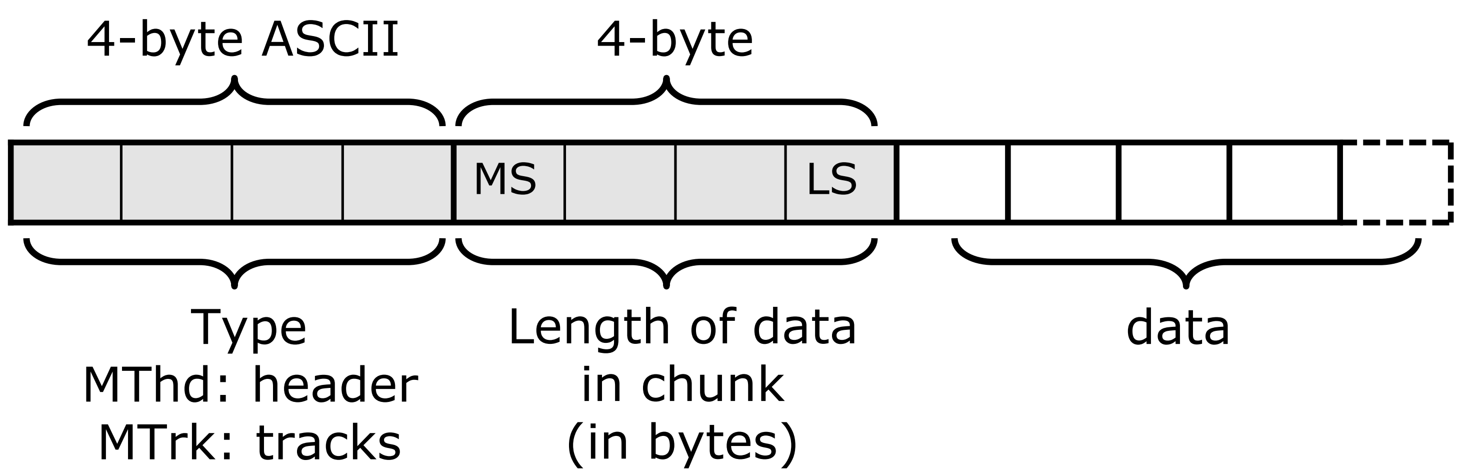
- They control the execution of timed sequences in a MIDI system

```
System Real-Time Messages
STATUS Description           Details
==========================================================
F8     Timing Clock
FA     Start                 from beginning of song
FB     Continue              from stop position
FC     Stop                  stops song’s execution
FE     Active Sensing        flags active connection (3/sec)
FF     System Reset          Reset all devices to init state
```

- The MIDI clock is a single status byte (F8) to be sent 6 times per MIDI beat (defined as a sixteenth note).
- Thus it represents musical tempo rather than real time.
- It is the only MIDI byte that can interrupt the current status
- The internal clock of the receiver is incremented with each clock

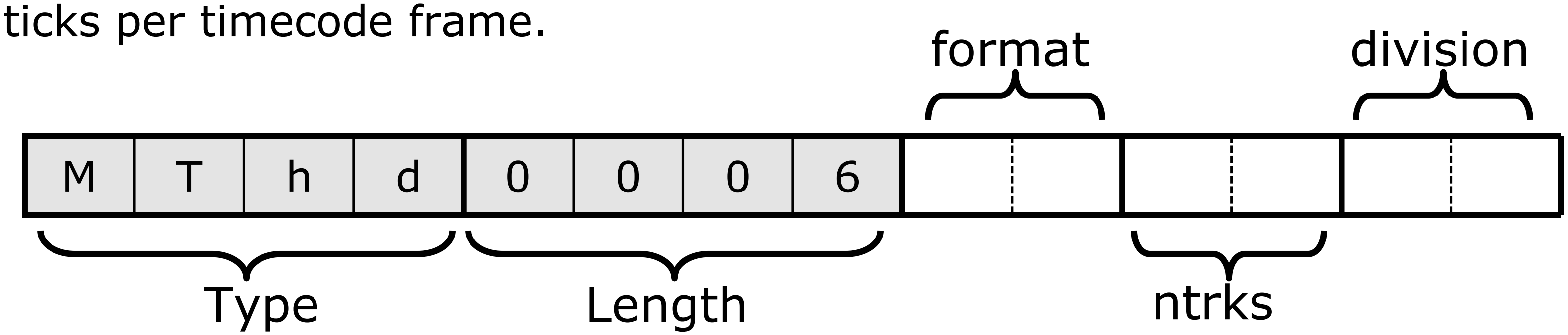
Sequencing MIDI: MIDI files

- Standardized MIDI files were designed to allow exchange of sequenced data between devices / SW sequencers.
- These files represent data as events belonging to individual sequencer tracks, plus info such as track/instrument names and time signatures.
- There are 3 types of MIDI files for representing: single-track data (type 0), synchronous multi-track data (type 1) and asynchronous multi-track data (type 2).
- Data is organized as bytes grouped into *header* and *track chunks*.

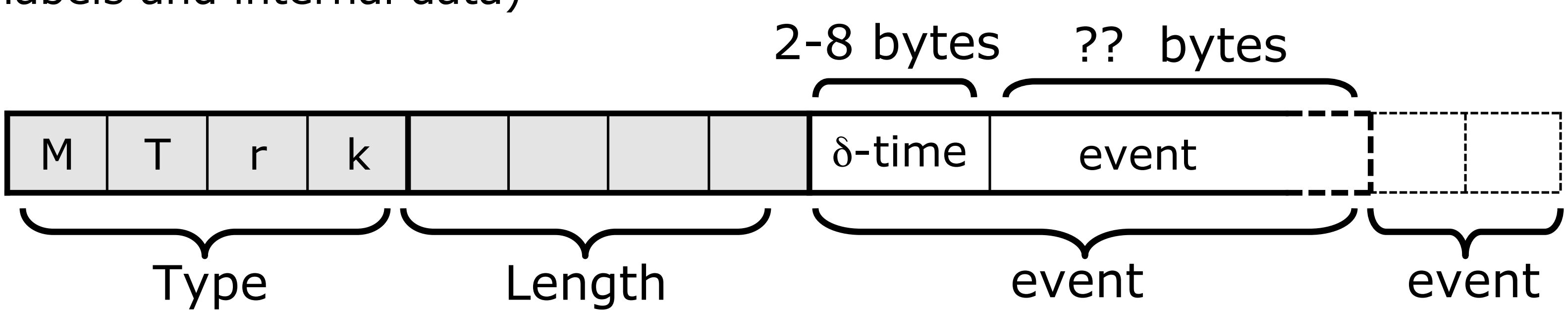


Sequencing MIDI: MIDI files

- Header chunk: *format* defines the MIDI file type (0, 1 or 2), *ntrks* defines the number of track chunks and *division* defines the timing format.
- The timing format is defined by the MSB of the 2-byte *division* word. *0* indicates a division of ticks per quarter note, while *1* indicates a division of ticks per timecode frame.



- Track chunks contains strings of MIDI events, each labeled with a δ-*time* (ticks since the last event) at which the event occurs.
- MIDI events can be channel messages, SysEx and meta-events (containing labels and internal data)



References

- Rumsey, Francis. *MIDI Systems and Control*. Focal Press (1991)
- *MIDI Xplained* Online book: http://linux.tu-varna.acad.bg/~lig/MIDI/xplained/index.htm
- Roads, Curtis. *The Computer Music Tutorial*. MIT Press (1996). Chapter 21: MIDI.
- MIDI Manufacturers Association: http://www.midi.org/
- http://www.harmony-central.com/MIDI/
- http://www.emusician.com/midi/