Data Communication/MIDI

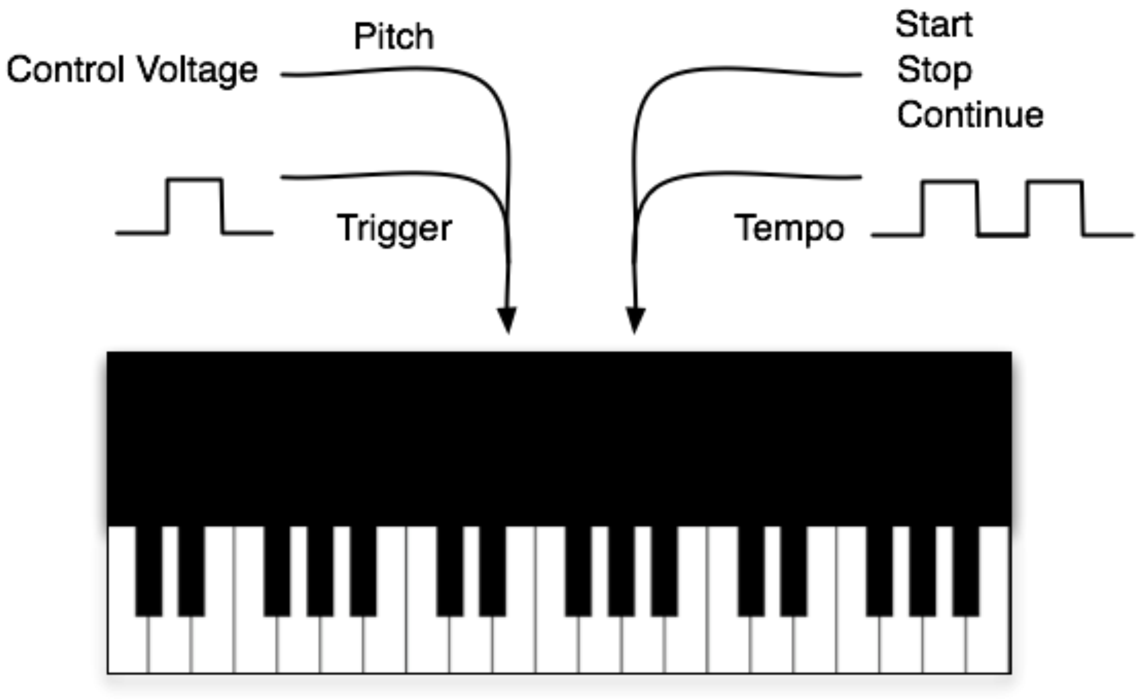
Juan P Bello

MIDI

- The Musical Instrument Digital Interface (MIDI) is a standard for communication between electronic musical instruments, which has also been applied to a larger range of equipment (Fx units, mixers, light control, etc)
- Main functions: transmission of performance/control data around a digitally-controlled music systems and of other data such as timing info, set-up parameters, samples, etc.
- The MIDI standard was formulated by agreement between the manufacturers in the early 80s
- While use of the hardware interconnection scheme is in steady decline, the communications protocol is still widely used.

Background (1)

- Electronic musical instruments, and the need to control them remotely, predated MIDI
- Old instruments used analog voltage control instead of microprocessors



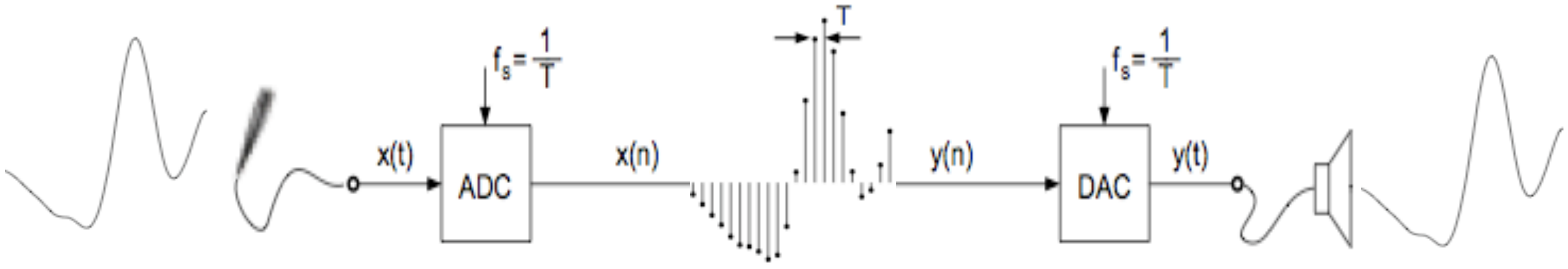
- These often used one port for timing and another for note triggering and pitch info (as a DC control voltage)

Background (2)

- With the advent of microprocessor-based control in musical instruments a number of digital control interfaces appeared
- Incompatibility amongst them created the need for standardization and agreement between major manufacturers
- This resulted on the MIDI 1.0 specification released in 1982/83.
- The standard core functionalities remain largely unchanged, although several others have been added over time, e.g.: MIDI files, general MIDI, sample dump, MIDI timecode, etc.
- The MIDI Manufacturers Association (MMA), is the governing body regulating modifications to the standard (http://www.midi.org/).

MIDI vs Audio (1)

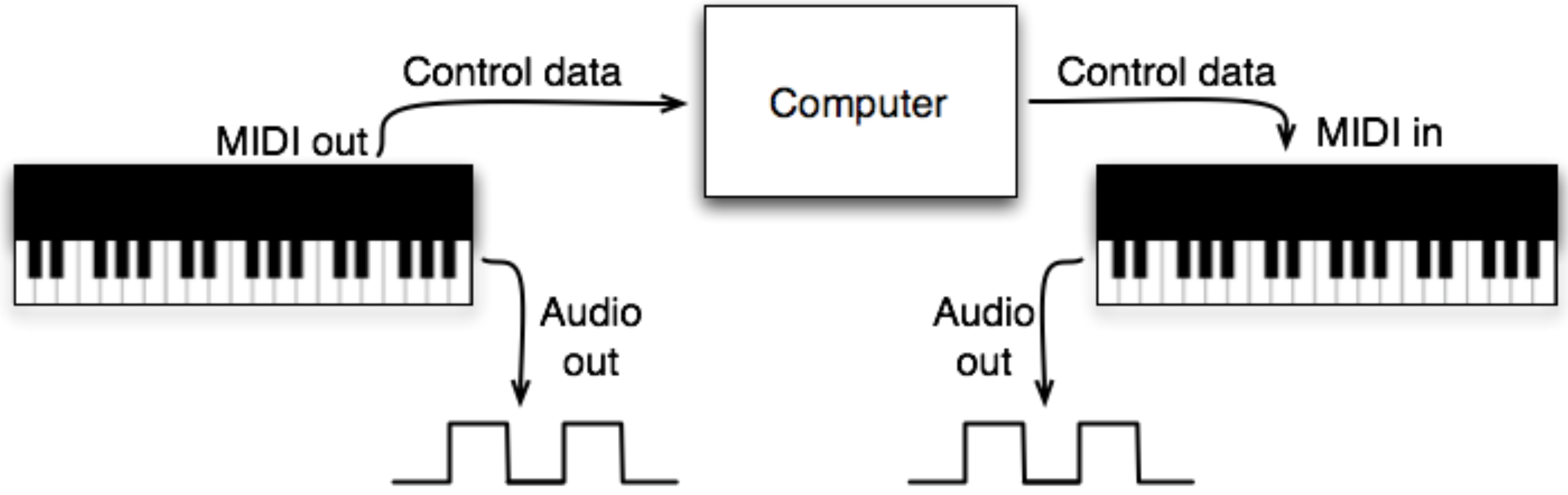
- In digital audio, the waveform is converted to the digital domain, where it is processed and stored before being converted back to analog



- Sounds are stored and replayed precisely, but we have no access to control data (the parameters that generated that sound)
- Since we need high temporal and amplitude precision to properly represent an audio waveform, digital audio uses a lot of memory space.

MIDI vs Audio (2)

- In MIDI, processing and storage also occurs in the digital domain, but the information being processed is not the audio signal but the control data used to generate it.
- An electronic instrument is needed to reproduce the sound, which means that unless we use the exact same synthesis engine, MIDI-generated sounds are never the same.



- Because it comprises control data only, MIDI uses significantly less memory space than digital audio

Data communication (1)

- Serial vs Parallel: 1 bit per clock vs *n* bits per clock

Serial:

Transmitter	→	Receiver
101001	→	11

Parallel:

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

- Parallel communication can be extremely fast, but also bulky and expensive for use over long distances (common within computers)
- Serial interface, although slower than parallel, allows for simple connectors and cabling (simple implementation and low cost).
- Speed of communication is measured in # data symbols per second (bauds). This is commonly equivalent to the information (data) rate, but not always.

Data communication (2)

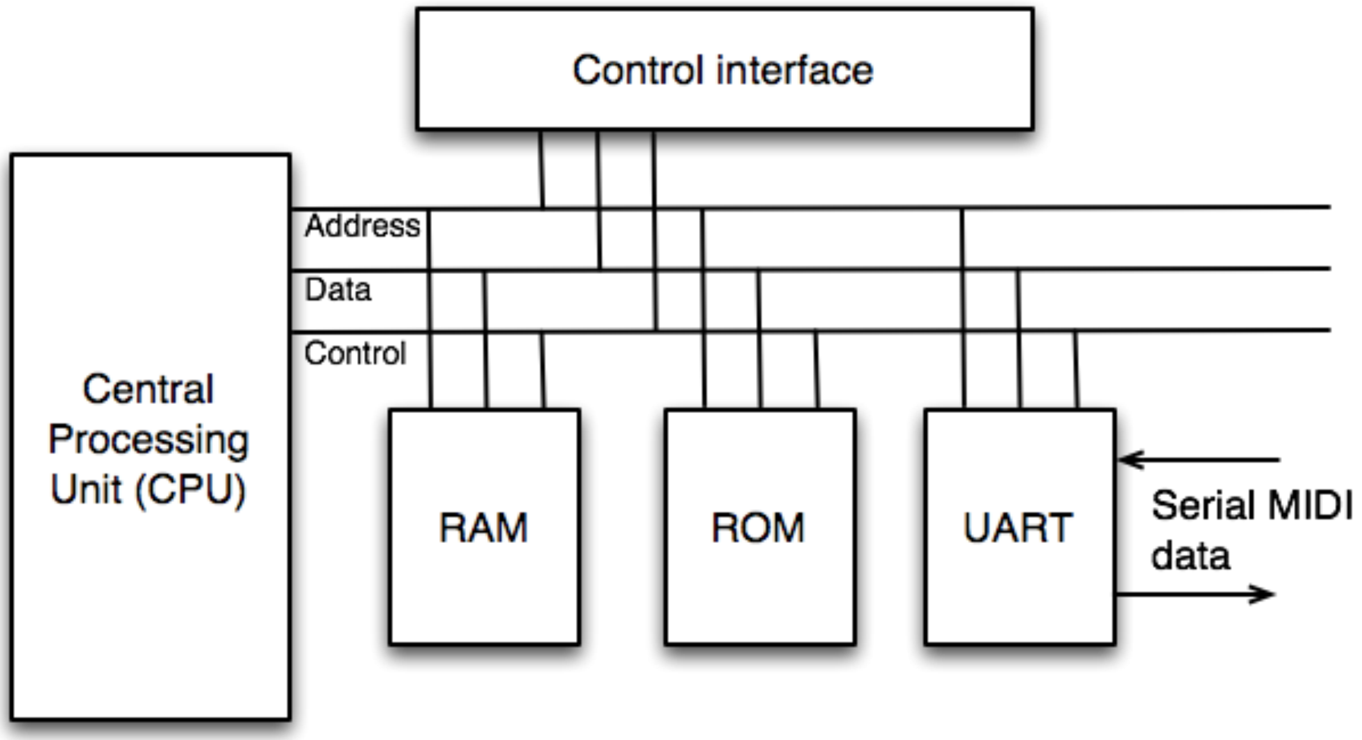
- In synchronous communication, the data is accompanied by a clock signal (on a separate wire or modulated with the data)
- This is used to tell the receiver the time slot in which it should register each arriving data bit.

- In asynchronous communication, transmitter and receiver clocks are not connected but should oscillate at nearly the same rate.
- Start and stop bits are used in the communications protocol to allow phase adjustment

- Data communication may be unidirectional or bidirectional.
- Bidirectional interfaces can be further divided in half duplex (only one direction at a time) and full duplex (capable of simultaneous transmission/reception).

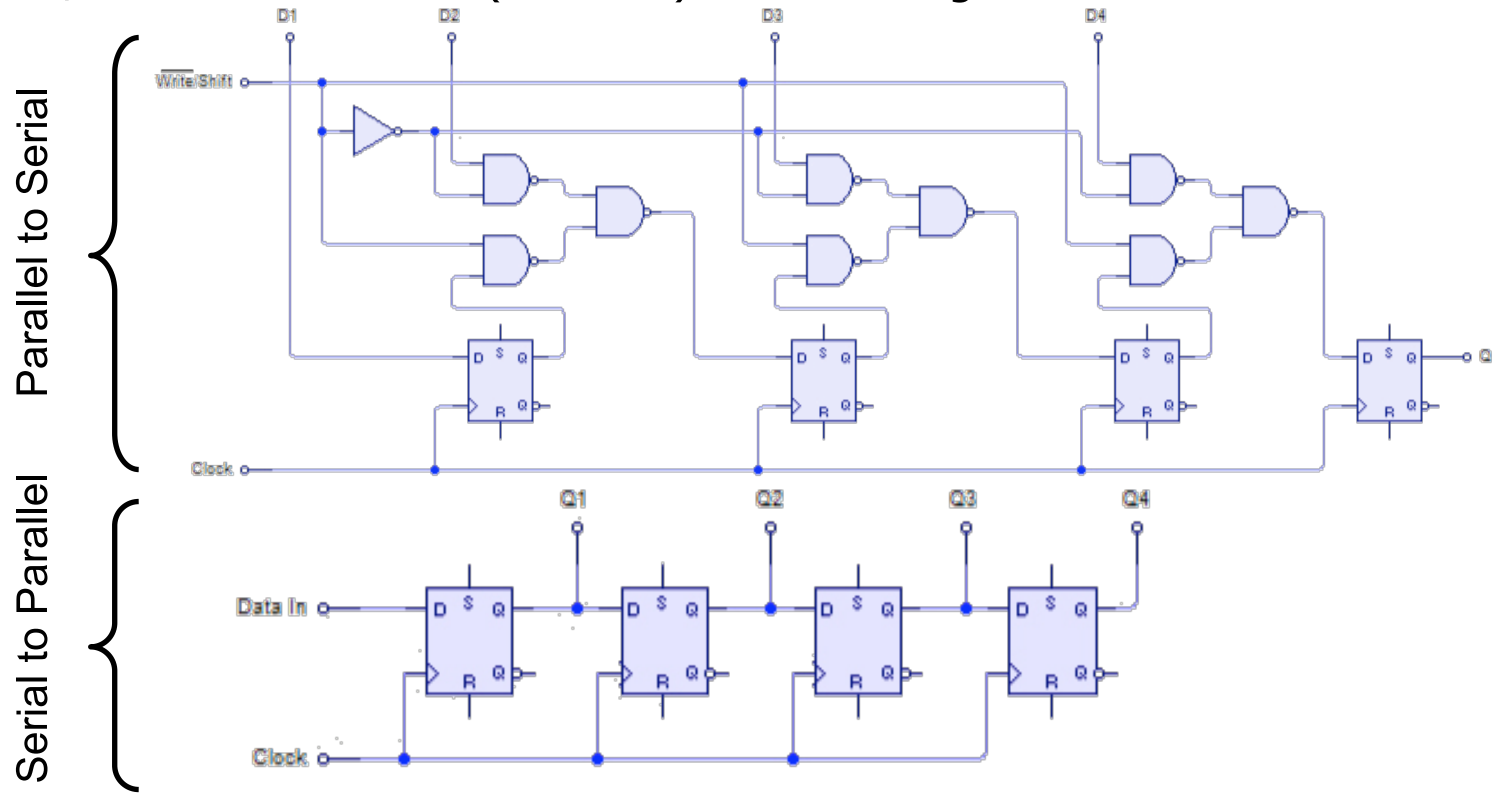
MIDI hardware (1)

- MIDI was designed to be simple and easy to install in economic equipment, and widely available to as many users as possible.
- It uses an uni-directional serial interface and asynchronous communication (no clock is embedded in the transmission)
- Conversion between parallel and serial is performed by an Universal Asynchronous Receiver/Transmitter (UART) unit.



MIDI hardware (2)

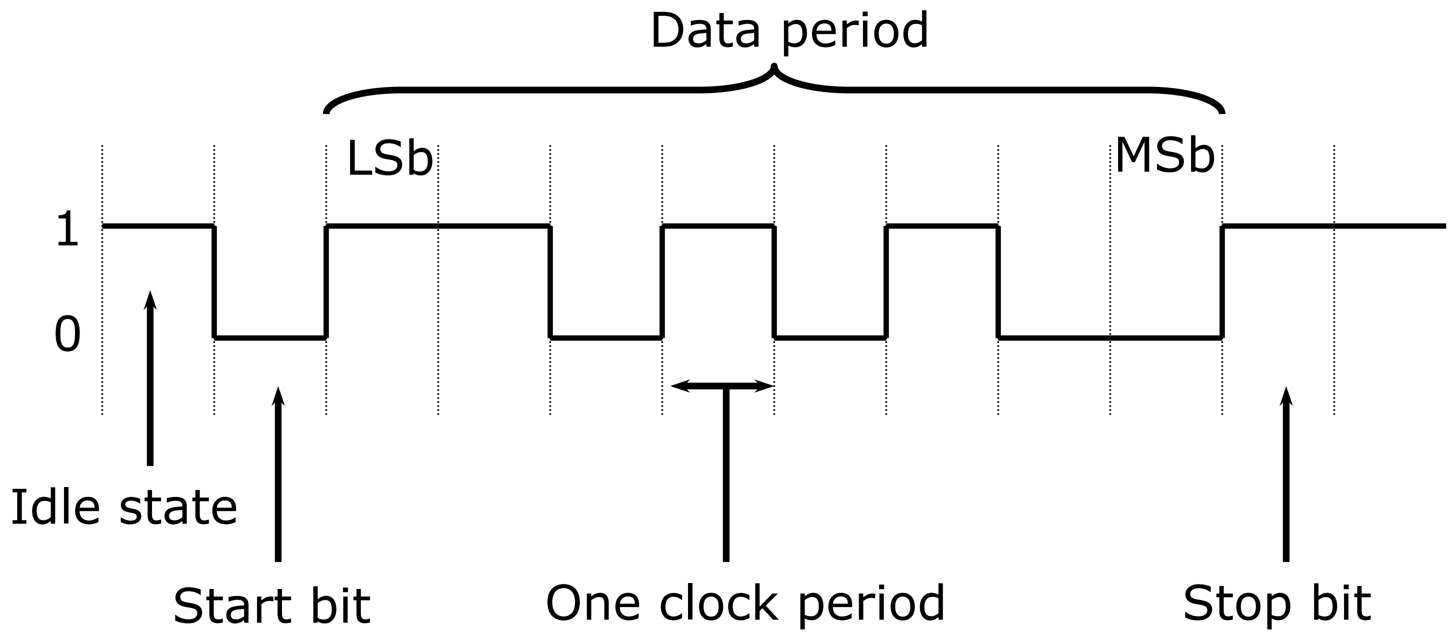
- Parallel/serial conversions (in UART) use shift registers:



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

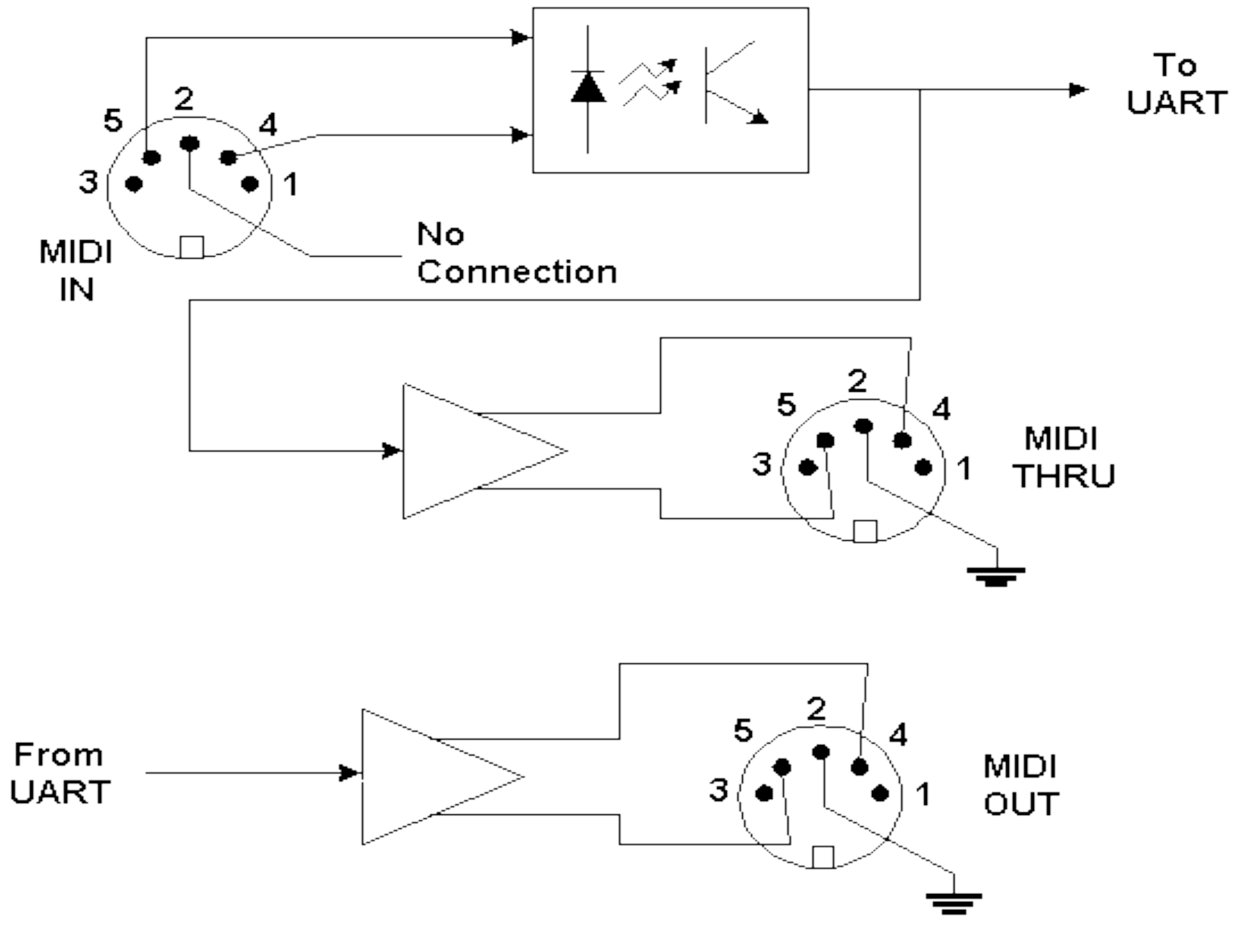
MIDI hardware (3)

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



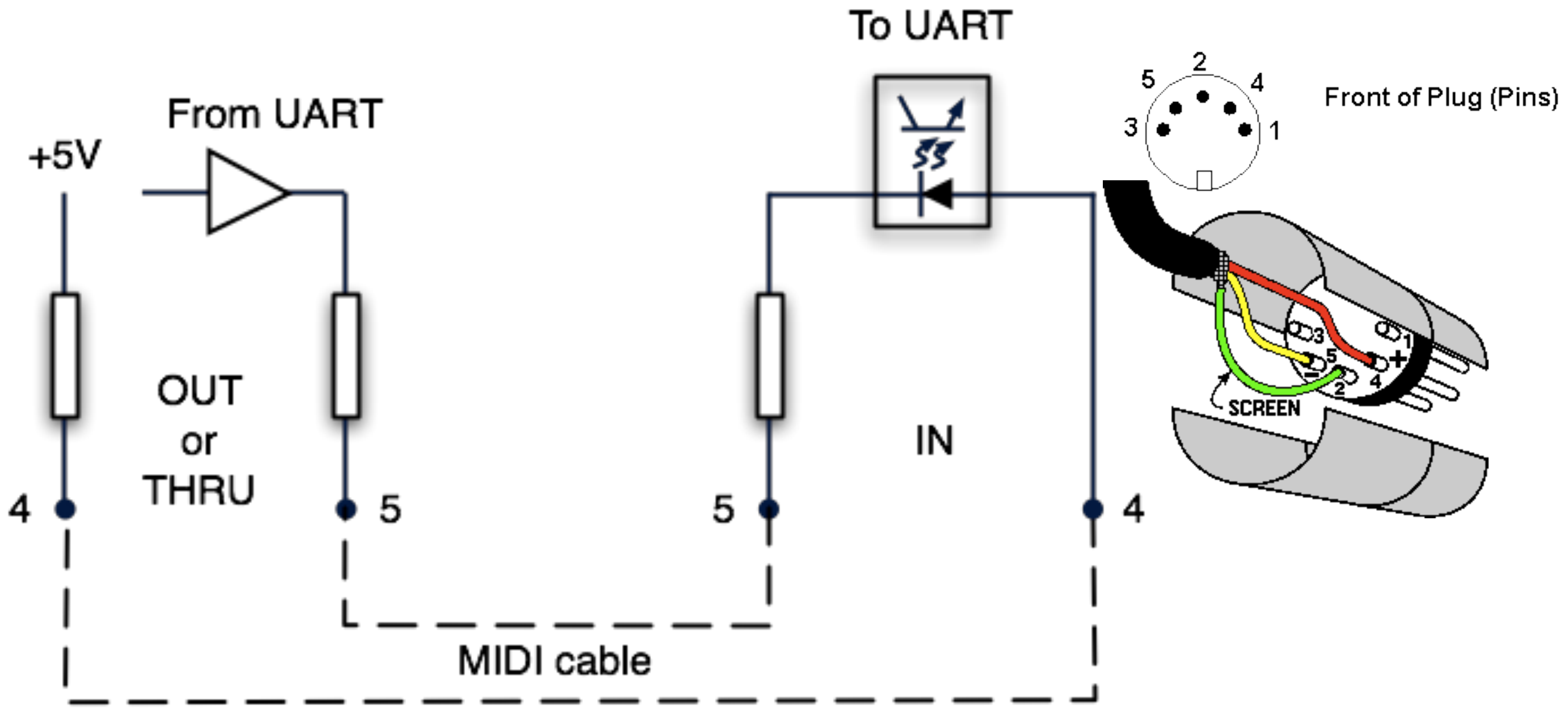
MIDI hardware (4)

- 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.



MIDI interconnection (1)

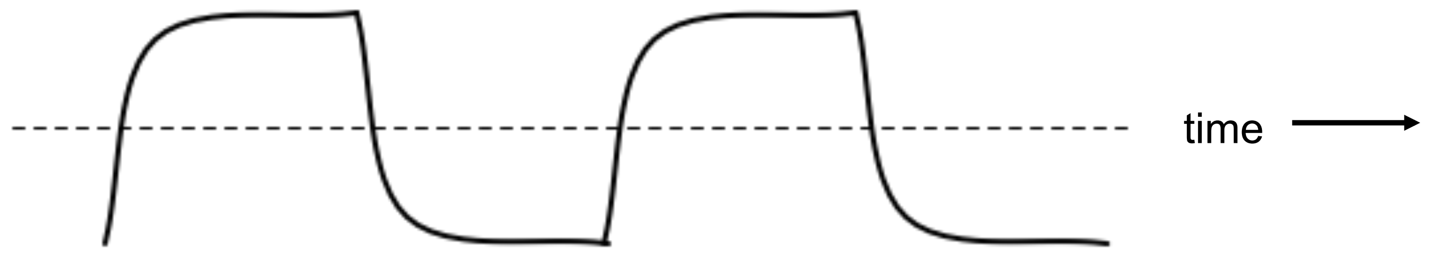
- The hardware uses cables terminated in 180-degree 5-pin DIN connectors, of which only three pins are used (5, 4 and 2).
- Pin 2 is connected to earth in OUT and THRU only



MIDI interconnection (2)

- MIDI data transmission is slow enough to avoid transmission losses and fast enough to make any transmission delay musically negligible.

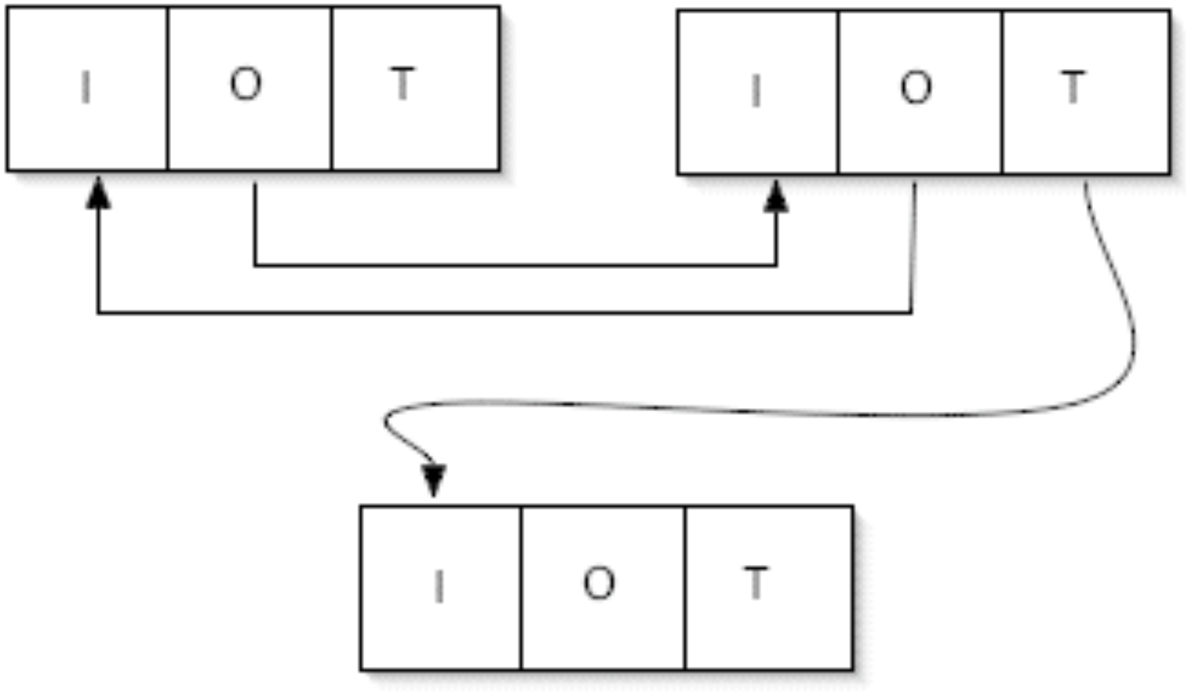
- In the MIDI specification, the opto-isolator is required to have a maximum rise-time of 2μs



- The rise-time refers to the speed of response of the device, if too slow it results on a roll-off of sharp edges (also for fall time) that can introduce errors in the data transmission.

MIDI interconnection (3)

- When several devices are connected in series the data passes through as many opto-isolators, resulting on an accumulation of rise-time distortion.
- Additionally, long cables cause unwanted distortion (a max. length of 15m is often recommended)



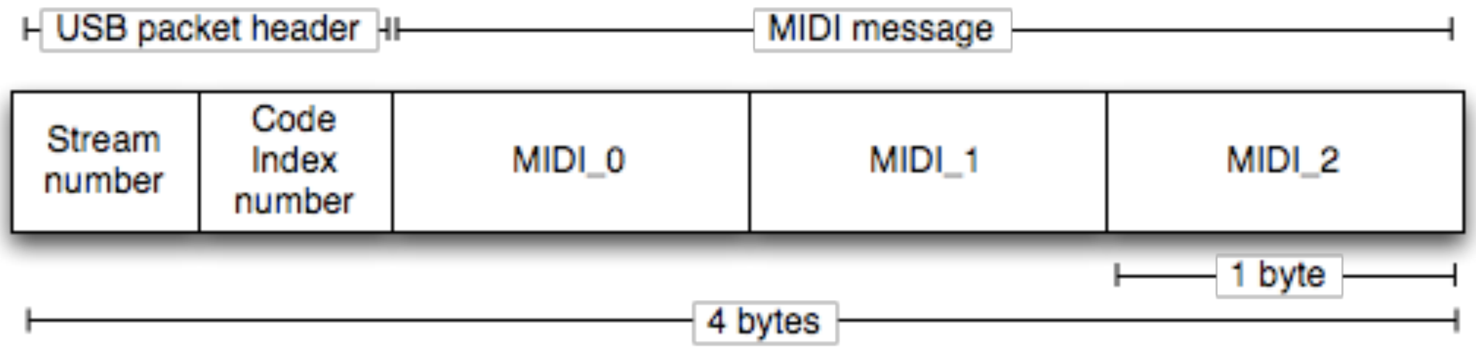
MIDI/computer interface (1)

- Computers are often the central controllers of a MIDI system
- Therefore there is need for a MIDI interface
- In the past, this was often achieved by means of an expansion slot card, or using the so-called MIDI-joystick port

- More recently, external MIDI/USB devices have been used.
- These are widely used for multi-port interfaces, which are able to handle a number of MIDI streams (each controlling up to 16 channels) and distribute separately.
- They allow the synchronized handling of several devices (video recorder, automated mixer, effects, samplers, etc).

MIDI/computer interface (2)

- Recent devices are using universal interfaces (USB, firewire) to transfer MIDI data directly
- There is a USB class definition for MIDI devices that wraps MIDI messages within USB packages.



- It virtualizes the concept of MIDI IN/OUT connectors, allowing the conversion to take place in software.
- A device that receives and transmits USB MIDI events is called a USB MIDI function. It separately process MIDI events from other data transfers (bulk dumps).

Useful References

- Francis Rumsey and Tim McCormick (2002). “Sound and Recording: An Introduction”, Focal Press.
  - Chapter 13: MIDI
- Francis Rumsey (1994). “MIDI Systems and Control”, Focal Press.
  - Chapter 2: Introduction to MIDI Control
- MIDI Manufacturers Association (2002). The complete MIDI 1.0 detailed specification (www.midi.org)