Pasquale Mainolfi

Appunti per il corso di Musica Elettronica

IL PROTOCOLLO MIDI

Il protocollo MIDI nasce negli anni '80, frutto delle ricerche di D. Smith e C. Wood, gli stessi che nel 1981 pubblicheranno l'articolo titolato The complete SCI MIDI.
MIDI è acronimo di Music Instrument Digital Interface: interfaccia per il trasferimento digitale di dati tra strumenti musicali elettronici. L'insieme, nello specifico, di una interfaccia fisica e, di uno standard di comunicazione, regole e messaggi interpretabili univocamente che permettono agli strumenti di scambiarsi informazioni.
Tale protocollo diventerà vero e proprio standard di comunicazione nel 1982 (MIDI 1.0) e, nel 1983, il primo synth con interfaccia MIDI, il PROFET 600 della SCI (Sequencial Circuits).

L'interfaccia

È una componente hardware che permette la trasmissione seriale di bit alla velocità di 31250 bit/sec in modo asincrono: ogni sequenza di bit (messaggio MIDI) è delimitata da 2 bit (0, 1), il primo posto all'inizio della sequenza e, il secondo alla fine (start, end).
Solitamente un'interfaccia MIDI porta con se una o più porte, tra cui:

1. MIDI In: riceve i dati in ingresso;
2. MIDI Out: trasmette i dati in uscita;
3. MIDI Thru: trasmette in uscita l'esatta copia dei dati ricevuti in ingresso.

I messaggi midi
I messaggi MIDI sono trasmessi in gruppi di 10 bit: 8 bit per il messaggio e, 2 bit destinati allo start e stop.

$$0 \mid x\ x\ x\ x \parallel x\ x\ x\ x \mid 1$$

Due tipologie di byte MIDI:

- *status byte*:inviati per primi, trasmettono il tipo di informaizone, servono a decifrare i byte di dati. Il bit più significativo (MSB, per intenderci quello più a sinistra) dello status byte è uguale a 1. Per tale ragione uno status byte può assumere un valore compreso tra 128 e 255.

$$1xxx \parallel xxxx \rightarrow \text{STATUS BYTE}$$

- *data byte*: servono a comunicare i parametri necessari per un corretto funzionamento dello status byte. Il bit più significativo è pari a 0. Per

tarle ragione un data byte può assumere un valore in base 10 compreso tra 0 e 127.

$$0xxx \;||\; xxxx \;\rightarrow \text{DATA BYTE}$$

Un messagio MIDI può essere, dunque, constituito da una o più sequenze di 1 byte (8 bit) ciascuna. Il primo *status* che serve ad identificare il tipo di messaggio; il successivo o i successivi (non più di due, in ogni caso) sono dei veri e propri data byte che contengono le informazioni significative.
Come si possono distinguere gli status byte dai data byte?
È stato stabilito che i messaggi con MSB pari a 1 contenga una status byte, mentre quelli con MSB uguale a 0 data byte.

Un esempio significativo di messaggio MIDI potrebbe essere:

$$[0|10011111|1][0|00111100|1][0|00100000|1]$$

prima sequenza (senza considerare bit di start e stop): $1001 = \text{note on} \rightarrow \text{status}, 1111 = \text{MIDI ch.}$
seconda sequenza (senza considerare bit di start e stop): $00111100 = \text{note value} \rightarrow \text{data}$
terza sequenza (senza considerare bit di start e stop): $00100000 = \text{note velocity} \rightarrow \text{data}$

Messaggi e struttura

Una linea MIDI 1.0 dispone di 16 diversi canali logici, ognuno dei quali ad identificare una specifica destinazione a cui recapitare una certa informazione.

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

È possibile organizzare i messaggi MIDI in due grandi famiglie: *channel* e *system messages*.

Channel messages:
possono essere inviati ad uno dei canali disponibili e, si dividono in:

1. *channel voice message*. Vengono generati durante l'esecuzione. Sono:
    1. **note on** e **note off**, per comunicare l'inizio e la fine di un dato evento sonoro. Un messaggio **note on** specifica il canale (0 - 16), l'altezza della note (0 - 127) e l'intensità con una escursione dinamica 0 - 127. Un messaggio di **note off** ordina di smettere di suonare una certa nota su di un certo canale, non interrompe il suono, ma solo il suo invio. È un messaggio che si genera quando viene rilasciato il tasto premuto in precedenza. Anch'esso è costituito da due data byte, uno che specifica la nota rilasciata e, l'altro l'intensità di rilascio.
    2. **after touch polifionico**, comunica i valori di pressione che si esercitano su di un certo tasto dopo che questo è stato premuto con una escursione che va da 0 a 127. Controlla vari parametri, come ad esempio, il vibrato, il tremolo, etc...
    3. **piptch bend**, modifica l'intonazione di una nota (glissando) in esecuzione, simula l'effetto bending. È un messaggio composto da due data byte per i valori di intervento. Può essere inviato via joystic o rotella; il range frequenziale dipende dallo slave, solitamente ± 2 semitoni.
    4. **program change**, serve ad inviare un cambio di programma (da 0 a 127 preset). Richiama il contenuto di una locazione di memoria.
    5. **control change**, serve ad inviare un cambio di controllo. Contiene 128 diverse variazioni di messaggio che fanno riferimento alle informazioni di espressione.

Message	nibble 1	nibble 2 (canale)
Note on	1001	xxxx
Note off	1000	xxxx
Poliphonic key pressure	1010	xxxx
Pitch bend	1110	xxxx
Control change	1011	xxxx
Channel pressure	1101	xxxx
Program change	1100	xxxx

1. *channel mode message*. Riguardano il controllo generale di un dispositivo e non dei singoli canali. Controllano il modo di funzionamento delle voci di uno strumento e sono:
    1. **all sound off**: silenzia tutte le note che erano state attivate con il messaggio note on;
    2. **all note off**: silente tutte le note... ferma tutto!

3. **local control**: attiva o disattiva la tastiera localmente in modo da permettere il controllo dei parametri del synth via MIDI IN da uno strumento esterno;
4. **omni mode off**: il dispositivo risponderà solo ai messaggi voice su un numero limitato di canali;
5. **omni mode on**: risponderà comunque al messaggio;
6. **mono mode on**: comanda di selezionare mono mode;
7. **poly mode on**: comanda di rispondere polifonicamente ai messaggi in entrata.

Message	data byte 1	data byte 2
Omni on	0111	1100
Omni off	0111	1101
Mono	0111	1110
Poly	0111	1111

1. *system messages*. Contengono informazioni di sistema e possono essere ricevuti da qualsiasi dispositivo MIDI. Sono individuati dalla stessa ultima combinazione di bit, ovvero hanno il 1° nibble sempre uguale a 1111, mentree il 2° nibble, non essendo destinato al canale, identifica il tipo di messaggio. Si dividono in:
    1. **system common message**, danno istruzioni generali relative a tutto il sistema. Sono:
        1. **midi timecode quarter frame**: messaggi nel formato ore, minuti, secondi e frame;
        2. **song position pointer**: indica il punto di una sequenza MIDI in cui deve posizionarsi il puntatore durante l'esecuzione;
        3. **song select**: selezione una song tra quelle disponibili nel dispositivo;
        4. **tune request**: intona due dispositivi.
    2. **system real time message**: organizzano la sincronizzazione del sistema. Sono:
        1. **midi clock**: inviati 24 volte per quarto, restituiscono un tempo relativo, tempo musicale e, non assoluto come accade per il midi timecode quarter frame in formato SMPTE;
        2. **start-continue-stop**: controllano i puntatori di tutti i dispositivi MIDI collegati;
        3. **active sensing**: inviato ogni 300ms, serve a tenere attiva la connessione tra master e slave;
        4. **system reset**: riporta tutti i dispositivi collegati ai valori predefiniti.
    3. **system exclusive message**: sono destinati alle informazioni specifiche di ogni produttore.

Message	data byte 1	data byte 2
System exclusive start	1111	0000
MTC quarter frame	1111	0001
Song position pointer	1111	0010
Song select	1111	0011
Tune request	1111	0110
System exclusive stop	1111	0111
MIDI clock	1111	1000
Start	1111	1010
Continue	1111	1011
Stop	1111	1100
Active sensing	1111	1110
System reset	1111	1111

Struttura di un file MIDI

I file MIDI devono avere una firma MThd (Hex: 4D 54 68 64) all'inizio del file. I file MIDI sono organizzati in *chunk* di dati, ognuno dei quali preceduto da un'intestazione di 8 byte: una firma di 4 byte (MThd o MTrk) utilizzata per identificare il tipo di chunk, seguita da sequenza di 4 byte che definisce la lunghezza del chunk come numero di byte che seguono questa intestazione. Tutti i valori dei dati sono memorizzati in formato Big-Endian (il byte più significativo prima). La somma delle dimensioni di tutti i chunk restituisce la dimensione totale del file MIDI.

Nota: quanto segue è riportato nella [REF. 1].
Header:
Il "Header Chunk" consiste in una stringa che indica l'intestazione, un indicatore di lunghezza, il formato del file MIDI, il numero di tracce nel file e un valore di timing che specifica le unità di tempo delta.

header: "MThd" + header_length + format + n + division

- "MThd" (4 byte): la stringa MThd, o in notazione esadecimale: 0x4d546864 si trova all'inizio del file MIDI ed indica che si tratta effettivamente di un file MIDI
- header_length (4 byte): lunghezza del chunk di intestazione (sempre di 6 byte - la dimensione dei tre campi successivi)
- format (2 byte): 0 = formato file a singola traccia, 1 = formato file a più tracce, 2 = formato file a più canzoni
- n (2 byte): numero di tracce che seguono
- division (2 byte): unità di tempo per il timing delta. Se il valore è positivo, indica i bpm. Se il valore è negativo, i tempi delta sono in unità compatibili con SMPTE.

Truck chunk:
Il "Track Chunk" consiste in una stringa identificativa, un indicatore di lunghezza

che specifica la dimensione della traccia e dati di eventi effettivi che compongono la traccia.

track_chunk = “MTrk” + length + track_event [+ track_event . . .]

- “MTrk” (4 byte): la stringa MTrk segna l’inizio 4di una traccia
- length (4 byte): il numero di byte nella traccia che seguono questo numero
- track_event: un evento di traccia sequenziato

Track event: Un “Track Event” consiste in un tempo delta dal precedente evento e in uno dei tre tipi di eventi.

track_event = v_time + midi_event | meta_event | sysex_event

- v_time: un valore a lunghezza variabile che specifica il tempo trascorso (tempo delta) dall’evento precedente a questo evento
- midi_event: qualsiasi messaggio del canale MIDI come note-on o note-off
- meta_event: un evento meta nel formato Standard MIDI File (SMF)
- sysex_event: un evento esclusivo di sistema nel formato Standard MIDI File (SMF)

Meta Event Gli eventi meta sono che consistono in un prefisso fisso, un indicatore di tipo, un campo di lunghezza e dati di evento effettivi.

meta_event = 0xFF + meta_type + v_length + event_data_bytes

- meta_type (1 byte): tipi di eventi meta:

Type	Event
0x00	Sequence number
0x01	Text event
0x02	Copyright notice
0x03	Sequence or track name
0x04	Instrument name
0x05	Lyric text
0x06	Marker text
0x07	Cue point
0x20	MIDI channel prefix assignment
0x2F	End of track
0x51	Tempo setting
0x54	SMPTE offset
0x58	Time signature
0x59	Key signature
0x7F	Sequence specific event

- v_length: lunghezza dei dati dell’evento meta espressa come un valore a lunghezza variabile
- event_data_bytes: i dati effettivi dell’evento.

System exclusive event
Un evento esclusivo di sistema può assumere due forme:

- sysex_event = 0xF0 + data_bytes 0xF7 (lo stream di dati MIDI includerebbe 0xF0)
- oppure sysex_event = 0xF7 + data_bytes 0xF7 (0xF0 viebe omesso)

References

1. Standard MIDI File Structure: https://ccrma.stanford.edu/~craig/14q/midifile/MidiFileFormat.html
2. Standard MIDI-File Format: https://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html
3. Outline of the Standard MIDI File Structure: https://www.ccarh.org/courses/253/handout/smf/
4. H. M. de Oliveira, R. C. de Oliveira, Understanding MIDI: A Painless Tutorial on Midi Format, 2017, https://arxiv.org/pdf/1705.05322.pdf