The Twenty-fourth Annual Interactive Audio Conference
PROJECT BAR-B-Q 2019

Group Report:
A generalized haptic format standard
leveraging existing MIDI infrastructure

Participants: *A.K.A. "Committee on Haptic Augmented MIDI Peripherals (CHAMP)"*
Karen Collins, University of Waterloo
Joshua McMahon, Realtek Semiconductor
Dafydd Roche, Dialog Semiconductor
Brad Lambert, Cirrus Logic
Iroro Orife, Netflix
Kate Stone, Novalia
SPECIAL CONSULTANT: Rick Cohen

Brief statement of the problem(s) on which the group worked:

Haptic devices are increasingly used around the world, and are present in game controllers, mobiles, wearables, force feedback devices, etc. Haptic devices are currently not being used in areas of media where they *could* be used--for instance, to add physical sensation to music and movie content: Outside the West, the #1 way to experience Netflix content is streaming to mobile device, and a way to play back haptic content in the future will be useful.

Today, content creation for VR, games, etc. has a different build for different haptic devices (e.g. phone, Switch, Xbox, etc) and is tightly coupled to hardware. At the moment, there are many companies designing proprietary solutions to the use of haptic devices, but these are not designed for cross-platform implementation. As such, someone designing a game for Switch has to re-design the haptic aspect for phone implementation, etc. A recent attempt to standardize development for VR and AR, called OpenXR was released this year. However, the specification is specific to vibratory devices, is designed for use with game engines only, and relies on code, making it more difficult for novices to programming, (like many sound designers, for instance) to develop haptic content.

We've seen this problem of cross compatibility between devices before, with audio hardware: the response of the audio industry that still holds today was MIDI: Musical Instrument Digital Interface.

Some of the challenges we want to address directly are:

- Content is consumed on many different platforms
  - Non-US markets have mobile as number one platform
  - Hardware is different (Mobile Phones, VR systems, gaming platforms etc)
  - Software/OS is different
  - Haptics is more than just "buzz" vibrations, and is growing rapidly in sophistication
- Creating haptic from audio is hard (and bad)
  - Simply LPF'ing the audio track to create a vibration track for a film for instance potentially creates one long ~90 minutes buzz. It's not something that feels natural.
  - Using audio misses content that may be visual, not auditory, that we want to add haptics to: e.g. glass tapping in heist movie (sound is silence, but want to feel the tap)
- Haptic hardware is varied (transducers)
  - Vibration: ERM, LRA, Piezo etc
  - Thermal: Peltier etc

        - Skin Stretch: Digital Servo's, Solenoids etc.
        - Force Feedback: load cell, strain gauges, tactile force sensors
    - Haptic hardware is varied (edge IC/exciter)
        - Streamed waveform playback
        - Sampled waveform playback from trigger
        - Tuned to transducer
    - Haptic hardware is varied (user interface)
        - Crappy OR old hardware must sensibly de-feature
    - Creating haptic content when it comes to games is typically left to sound designers, who may have little to no prior experience of haptics: therefore, using the language of audio will be beneficial to reduce the learning curves involved, at least when it comes to VR and game engine content.
    - Authoring content must be easier.

MIDI has been used before in the creation of haptic content. Project BBQer Peter "pdx" Drescher, for instance, used MIDI for the creation of haptic vibration on the T-Mobile Sidekick (c. 2002). Ringtones on the phone used MIDI for music, haptic vibration and LED lights. The use of MIDI then is already a tried-and-tested way to distribute haptic information. What we need next is to standardize the use of MIDI for haptics.

A brief statement regarding the group's solutions to those problems, or a statement describing the progress that was made if no solution was determined:

Our solution is to adopt and adapt the MIDI protocol for haptic devices.
MIDI is already used in so many different platforms beyond that available via for instance OSC. Nearly all media hardware from computers to toys already has the ability to run MIDI files: it's also universally understood by musicians and sound designers who are often tasked with adding the haptic content to media at the moment (e.g. in games).

When considering most media playback/interactions systems (e.g. iOS, Linux, Android, Windows etc) all of them have the capability to create, encode, decode and render midi in some manner or another.

Many of the content creation tools (e.g. Unity, Unreal Engine, Adobe Premier, Final Cut Pro, Ableton, Cubase etc) all have basic MIDI creation capabilities. The image below shows how Reaper's MIDI editor was able to create the envelope for a vibro-haptic event, happening on Left Hand Thumb. It is possible, in other words, to use existing software to develop haptic MIDI content.

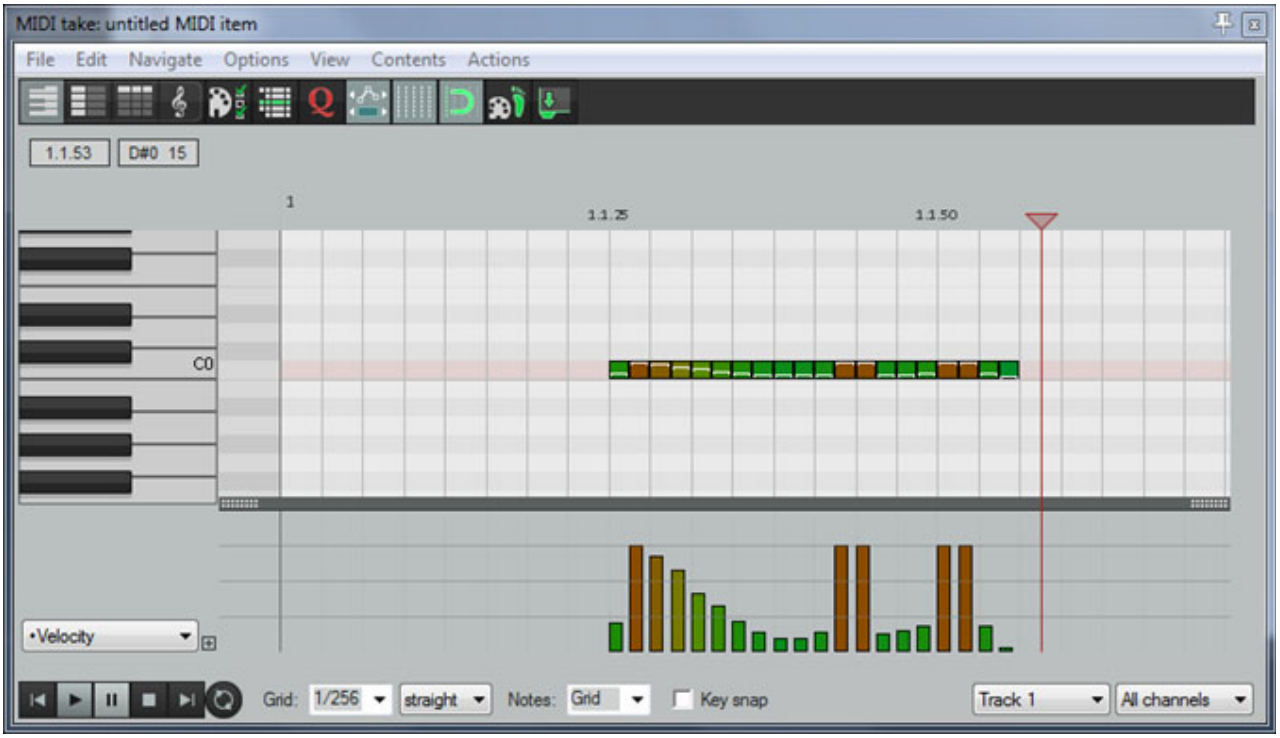


Image 1 - Haptic Amplitude Waveform generated in Reaper's MIDI window

Audio is similar to haptics in many ways, and there are mappings that we can employ like pitch = frequency that make sense and reduces the learning curve for content creators. We want to develop a standard mapping system for haptics that allows basic tools that users already have for the creation and editing of MIDI to edit and create advanced HD haptics in the media that we consume.

This report is not a complete specification, it is the early work and examples of how haptics could be mapped to MIDI, using Vibrohaptics transducers and their drivers as an example. We hope to garner interest in order to expand the initial prototypes and specification.

Use Cases

We imagined a handful of use cases to demonstrate the creation and rendering flow:

Use Case #1 Commuter on a train

Image from (https://www.digitalspy.com/tech/a796740/virgin-trains-launches-netflix-style-streaming-app/)

A commuter on a train is watching a movie on their smartphone. From the media source perspective (e.g. Youtube, Netflix etc), the MIDI file is provided to the client (the mobile phone) along with the subtitle file as part of the MPEG media package. The movie is then streamed in real time. The haptic track has been created in advance by the movie company, by the media provider (e.g. Netflix) or by a third-party (e.g. fans). Standard MIDI files are time stamped, which allows the content to be compatible with the usual fast-forward/rewind controls.

Content is created and authored by professionals in video or audio editing tools that already support MIDI. Custom MIDI “Instruments” could be created to support specific haptics effects. Such simple tools and open access to a standard would even allow the likes of Youtube content creators to add a timestamped “haptics file” to be downloaded before the video begins streaming and allow reliable time-synchronized haptics to be played along with streamed content. The haptics file is in this way analogous to a subtitle file, which are often created and swapped by fans. Such similarity in container and behavior to a subtitle file makes the rollout of Haptics in existing streaming services very easy. The playback app (e.g. Netflix app on an iPhone) would be responsible for the translation/rendering between the timestamped midi-file and the iOS/Android Haptics API.

Use Case #2 Gaming and VR

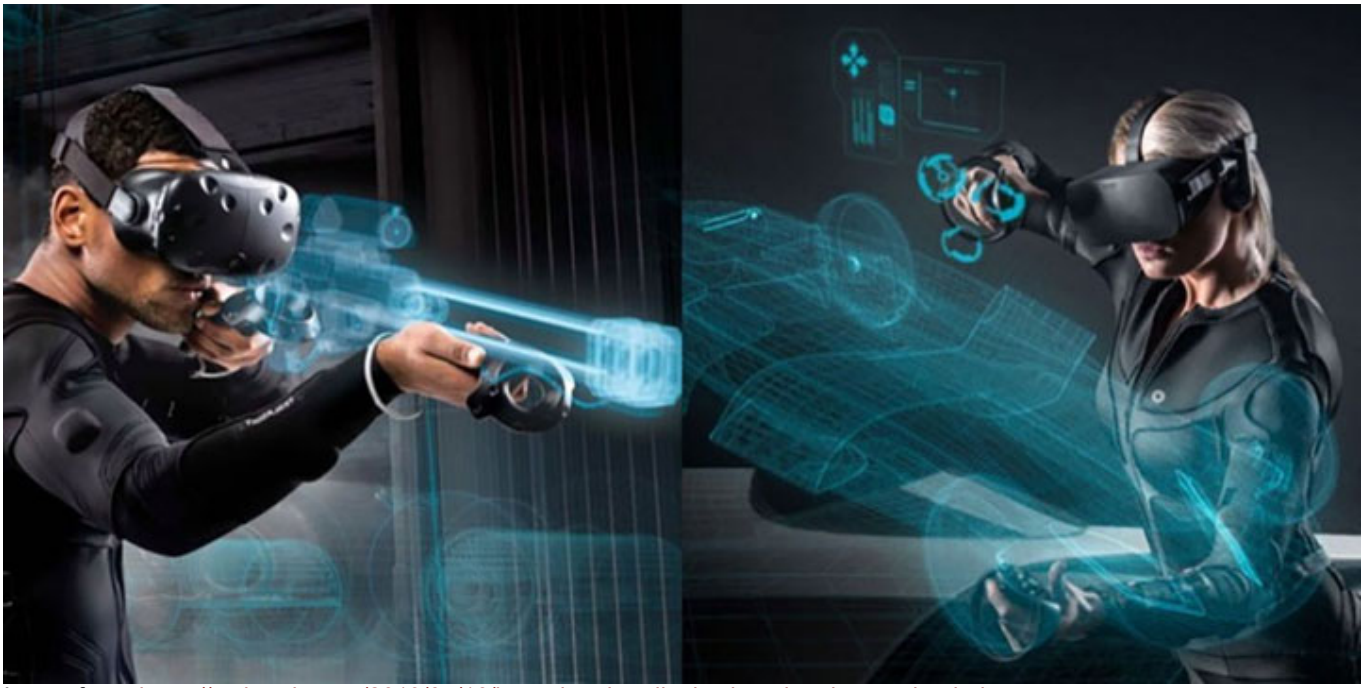


Image from: https://techaeris.com/2019/07/16/how-virtual-reality-is-changing-the-gaming-industry

Gaming and VR have some special needs. Content creation is typically done in custom tools like Unity and Unreal Engine, with specific sound design and effects being created in tools such as Pro Tools, and then integrated into the engine using middleware tools like WWise. Sound designers aren't necessarily haptics experts, but ARE being asked to create haptics effects to match audio effects. (e.g. gunshots, car crashes, Interactive switches, etc).

Game/VR developers are expected to port the media to as many different systems as possible - XBOX, Playstation, Nintendo, Mobile, VR as possible. This is done regularly, porting the visual and audio assets from platform to platform, however, the haptics is typically done for specific hardware. For example:

- Nintendo Switch uses two VibroTactile LRA transducers.
- Mobile Phones typically use one.
- VR systems use two, but are looking to increase that to increase the immersion into the VR environment.

OpenXR offers cross-platform deployment for common VR and gaming engines, but not mobile or future devices.

AAA Games will be expected to be very dynamic in their haptics effects, with envelopes being generated and modified in real time as effects are mixed. For Vibrotactile experiences, this would most likely be either be Mode 0 (Streamed Envelope) or Mode 2 (hifi) (see below for explanation of these modes).

However, shorter, simpler "apps store" style games will likely use pre-canned haptics sequences that matches a standard sound effect - think of Super Mario's Jump sound effect from the 80s. It was the same sound effect, regardless of surface he was jumping on. For quick game development, a standard jump "feel" could be repeated again and again. This would equate to Mode1 in our VibroTactile mode (see below for explanation of these modes)..

Custom made unique haptics effects could also then be bought and licensed for use in games in the way that sound effects are currently sold as part of sound effect libraries.

Use Case #3 Medical Simulation/Training

Image from: https://www.theverge.com/2018/8/14/17670304/virtual-reality-surgery-training-haptic-feedback-fundamentalvr

This content is very similar to the VR/Gaming in their real time creation and rendering, however, the hardware is tightly specialized and coupled to the media. (e.g. surgery simulation etc). Under these conditions, multiple haptics transducers are used, and specific knowledge of the hardware is used. Systems would likely stream the haptics to the transducer using a PCM Audio stream.

Use Case #4 Wearables

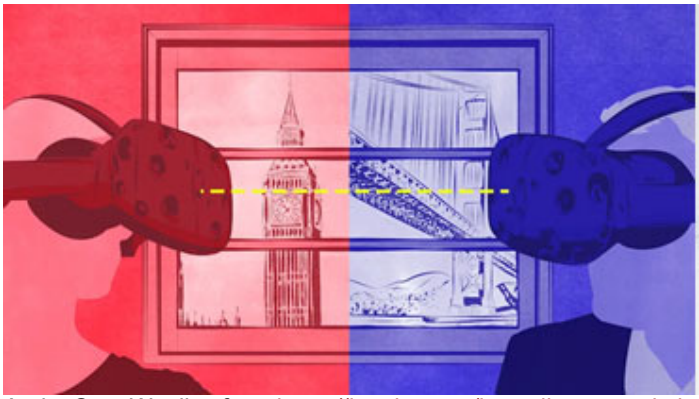
(image from: https://www.wareable.com/news/haptic-wearable-tactile-alerts-alarms-navigation-2618)

The main focus for wearables is lower power consumption, which tends to lend itself as few commands as possible (to minimize the MCU ON time). Such applications fit “sample” based effects well (e.g. different haptic events for different alerts - e.g. one type of haptic to “turn left” another to “turn right” or “text from my daughter” vs. email from work.)

On the Vibrotactile side, this is best taken care of by Mode 1 - Sample playback.

On the creation side, this is best handled either by a DAW (Digital Audio Workstation) or a simple midi-editor.

Use Case #5 Remote Physical Long Distance Link



Art by Sam Woolley from https://kotaku.com/long-distance-relationships-suck-but-vrs-made-it-easie-1775615168
Teslasuit Image from fashnerd.com
https://fashnerd.com/2019/01/does-the-teslasuit-represent-the-future-of-full-body-haptic-technology/

While seemingly humorous at first glance, the possibility of remote physical links is an interesting one. It is our belief that beyond gaming, adult content is a strong driver of technological innovation in haptics. Beyond the purely sexual, physical long-distance touch is an intriguing use of haptics. In a Kotaku post (https://kotaku.com/long-distance-relationships-suck-but-vrs-made-it-easie-1775615168) writer Nathan

Grayson wrote a fascinating article about the power of VR to provide closeness in an interaction with a loved one: “We held hands. We hugged a couple times. It was just really nice, despite the lack of so much as a controller rumble when we touched.”

That article was written in 2016, but even then, the lack of haptics technology was noticeable. Haptics has a true potential to make such interactions much closer, using VibroTactile (touching someone's arm), Thermal (warmth of other users hand), Skin Stretch (a hug) etc.

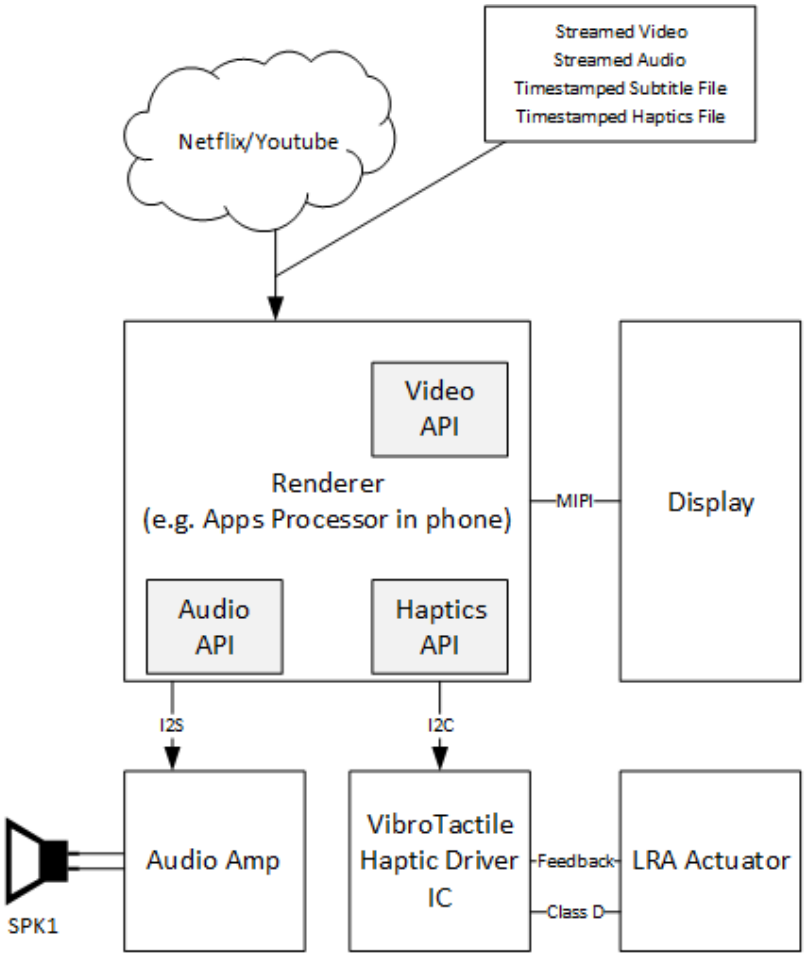
On the vibrotactile side, such content wouldn’t be expected to be repeated and would likely be dynamically authored and streamed (potentially from one client to another). The low bandwidth nature of MIDI would be an excellent method to transmit from one user on one side of the planet to another, and allow the receiver rendering device to downmix to whichever transducers were in the receiver.

On the content creation side, this was probably one of the most complex use cases as individual interactive objects would require their own characteristics to be created. (e.g. playing piano for your partner - force, temperature, texture of the key’s alone would require significant authoring skills - likely custom tools would be required within Unity etc).

This topic also introduced the concept that differing human gender may require different actuators, placed differently, with different capabilities. Not every human being is the same. That could be addressed by allocating a specific number of NOTES to “unallocated body location #” and using MIDI’s SysEx commands to name the actuator positions for the renderer, which would then allocate the haptics signals as required for the user (e.g. user may be wearing a female haptics suit, where and what transducers should be actuated). We anticipate leaving blank a number of possible body locations in our specification for either expansion of existing content or personalized touch locations.

Proposed Solution

In each of the use cases above, there is a Media Renderer (e.g. Apps processor in a mobile phone). Haptics Transducers or Driver ICs are expected to be connected to the the Apps processor.



Example flow for Commuter use case

In existing media creation and playback (e.g. making a game that has to work on Switch, Oculus and iOS) haptics content needs to be created separately for each end equipment. The content creator must know how to directly control the end equipment's Haptic actuator API. This has future implications for newer hardware that has more/different haptics actuators and drivers.

We propose that the creative tools are being used without knowledge of the end equipment. it should only care about the actual body positions and haptic effects known to date. The content renderer is then responsible for "downmixing" the content down to the number of transducers that the device has. This is very similar to Dolby Atmos' Object-Based Audio rendering scheme, where the rendered has individual sound objects and 3D data that the renderer uses to render to a stereo headphone mix, a stereo living room mix, 5.1 surround, 7.1 surround etc.

For example, the MIDI-Haptic file may call for the vibrotactile content on each of the 10 fingers to be actuated, but the Renderer (e.g. smartphone) may only have one haptic driver. It is then the role of the renderer to process and mix together the appropriate fingertip data into the single haptic driver API available in the Operating system.

This creates backwards and forwards compatibility for everything from a single Vibrotactic ERM vibration motor through to a multi-haptic-type-transducer haptics suit!

Parameters

Multiple parameters needs to be mapped into the MIDI space. The first thing to explore is the number of

options per parameter, the number of parameters, and the potential for expansion into the future. Each transducer type will also need to have different parameters passed to it (that is, a Peltier element has different configuration parameters than a Vibrotactile actuator).

Different kinds of Haptics senses/transducers

There are 7 kinds of haptics senses that can be used at the time of publication. The spec needs to address these types of Haptics senses. We leave one open for user-defined option. Transducer type is set by writing to the MIDI Continuous Controller (sometimes also called Control Change) message CC14. CC14 can be set per channel, allowing multiple channels to be used for the same kind of transducer, with other different configurations.

Haptic Sense	Typical Transducer and Accuracy required	Measurements	Value written to CC14
Vibration - Vibrotactile	ERM, LRA, Piezo	measured in Hz (frequency) and amplitude in dB ?	0 (Default)
Thermal Feedback	thermoelectric, conduction, radiation	Temperature - Freezing to Boiling in C Degrees	1
Skin Stretch (Laterotactile)	shape changers: shape memory actuators, pneumatic actuators, microelectro-mech actuators	--Displacement distance (up to 12mm) -- direction/rotation in degrees -speed (mm/s)	2
Pressure/Force Feedback: Resistive versus Active, Biomimetic vs non-Biomimetic.	strain gauges (Foil/metal) strain gauges(silicon) force sensing resistors, ultrasonics, pneumatics	Stiffness: measured in N/mm (0-100)? Degrees of Freedom (DOF: 1-6) Direction Mass (Grams) ? torque? continuous vs. rotational force?	3
Electrotactile	electrodes	measured in voltage	4
Pain	Microfluidic actuators, magnetic sensors?		5
(remains open)	(remains open)		6

Each of the senses/transducers require a different set of parameters. For example, Vibration transducers (ERM, LRA, Piezo) require amplitude, time, frequency of vibration (for off resonant drive).

Locations of the Body

Multiple positions on the body need to be accounted for, and some must be left for custom content. This then potentially gets multiplied by the TYPE of transducers. The team looked at and discussed the various individual parts of the human body and how many transducers might be required. Body location is set using the NOTE on a piano roll. e.g. Left hand Thumb may be C0, Left Hand Index C#0 etc. Using the entire

music note selector in MIDI, we have 128 options, which is more than the existing haptic suits on the market (usually 80-88 for the entire body). Any additional beyond 128 can be set using SysEx commands (to be defined), or by using a second transducer type channel, assuming it isn't being used. Actual mapping to be determined.

Proposed Supported Body Locations	Quantity of Each
Hands	2
Fingers	10
Palm	2
Back of Hand	2
Wrist	2
Head	1
Forehead Left	1
Forehead Right	1
Jaw Left	1
Jaw Right	1
Top	1
Back	1
Lips	1
Tongue	1
Torso	
Front Array	16
Back Array	16
Left Side	8
Right Side	8
Right glute	2
Left glute	2
Legs	
Right quadriceps	2
Left quadriceps	2
right hamstring	1
left hamstring	1
right shin	1
left shin	1
right calf	1
left calf	1
top of left foot	1
top of right foot	1
sole left	1
sole right	1

Modes of operation:

Each transducer shall have multiple different modes of operation that maps the MIDI data presented into different decoding. Mode of operation is set on CC15, allowing a different mode of operation for channel. As multiple channels can use the same transducer type, it allows some transducers to be in one mode, whilst others are in another.

For example, Ch0 could have Vibrotactile that are simple streamed envelope haptics (running at resonant frequency) while Ch1 could be in Triggered Patterns, where the stored sample is a heartbeat effect that

gets played repeatedly to build tension.

Modes of operation for all the different transducer types are open for continued work. For the sake of example, Vibrotactile modes of operation are below. They are based on existing smartphone/gaming/VR Haptic Driver IC Technology (DA7280)

Example: VibroHaptic Modes of operation

We envisage 3 basic modes of operation: These are dependent on the level of detail required (i.e. fidelity) and dynamic versus “baked” content.

Vibration Mode of Operation	Explanation
0x00 0x01 0x02	Streamed Envelope Triggered Patterns Hi-Fi Mode

1. Streamed Envelope: Creator decides what content to push to end user, this is a pre-defined stream of data sent to device e.g.a MIDI file containing all amplitude envelopes of vibration mode. Some additional control of the vibration frequency is possible using CC16 values.
2. Triggered patterns: equivalent to a MIDI DLS patch, downloadable patterns or “samples” of haptic content that can be shared amongst users, in the same way that wavetable synth patches, VSTs could be shared in music creation tools. Users could create MIDI files, or content creators could create MIDI files that can be played alongside media (like sample/wavetable patch). Sample data is stored in the end equipment and triggered from individual note-on commands.
3. HIFI mode: HI-FI mode is the equivalent of streaming audio (e.g. WAV): It tells the end device instead of playing the MIDI version, to use the streamed wave file content.

Mapping of Music MIDI to VibroHaptic MIDI

Initially our feeling was to use channel, instrument, note, and velocity. This will not account for all use cases. As such, we switched to using Control Change Messages to set parameters such as frequency. Initially, we set the transducer type statically in the spec (e.g. Ch0 is always Vibro Tactile, Ch1 is Thermal) - however, that limits the amount of parallel transducers (e.g. haptics suit with 100+ Vibro-Haptics transducers, 16 of them could be played simultaneously with different Mode of operations).

Music MIDI	**VibroHaptic MIDI Mode 0x00 usage**
Channel	NULL - Used to allow parallel transducers
Note (on and off)	Body Position
Velocity (set per NoteON event)	Amplitude
Continuous Controller 16 (set per channel) - CC16	Frequency (0=Up to the Renderer, Any other value Midi-Audio Frequency = eg. A4 = 440Hz)
Continuous Controller 15 (set per channel) - CC15	Haptic Mode of Operation
Continuous Controller 14 (set per channel) - CC14	Transducer type

Examples in Hex code

in hex code: https://people.carleton.edu/~jellinge/m108f13/pages/04/04StandardMIDIFiles.html

Example: Haptic Headshot

Channel	Note	Amplitude	Explanation
0xB0 0xB0 0x90 0x90 0x90 0x90 0x80	0x0E 0x0F 0x69 0x69 0x69 0x69 0x69	0x00 0x00 0x7F 0xB0 0x50 0x10 0x00	Set MIDI Ch0 to Vibe type of actuator Vibration Mode of Operation = 0 (Amplitude Envelope) 9 is note on, 0 is the vibration channel, 0x69 is the haptic transducer location (Audio A4) note the decreasing amplitude to create the sawtooth. 8 is note off

Example: Thermal Hot Then Cold

Channel	Note	Temperature	Explanation
0xB1 0x91 0x81 0x91	0x0E 0x69 0x69 0x69	0x01 0x42 0x00 0x09	Set MIDI Ch1 to Thermal type of actuator 9 is note on, 1 is the vibration channel, 69 is the thermal transducer location heat to 42 degrees C 8 is note off. Device returns to room/users temperature. 9 is note on. cool to 9 degrees C

Items from the brainstorming lists that the group thought were worth reporting:

Product safety

- Integrators are advised not to mess up
  - Existing haptic vendors are aware of many safety issues and are already addressing them
  - End equipment must meet safety standards

Known issues

- Wave table standard may be problematic for IP
- Do we need to define what the “audio” channel actually looks like in Hi-Fi???
- Future: MIDI 2 spec providing advanced options for more channels, etc.

Future Items required before bringing to a Spec group

- This document only covers suggested mapping for VibroTactile Mode 0 systems. Midi Mapping needs to be completed for other VibroTactile Modes AND other Sensory types.
- Body positions list should be mapped and populated.
- We also identified an opportunity to define input haptic midi streams for accessories. E.g., a premium beverage simulator that can render cold effects on it’s output but also capture and stream

accelerometer data. With a standard Bluetooth midi device

Glossary

ERM - Eccentric Rotating Mass
LRA - Linear Resonant Actuator
MOO - Mode of operation
CC - Continuous controller

Web Resources of use

MIDI driving LEDs, being controlled in Logic https://www.youtube.com/watch?v=AYx55nj5vSo

https://www.nap.edu/read/4761/chapter/8#175

https://www.midi.org/articles-old/details-about-midi-2-0-midi-ci-profiles-and-property-exchange

https://www.soundonsound.com/reviews/sonalog-gypsy-midi

List of Midi Continuous Controllers (and how they are typically mapped): http://www.nortonmusic.com/midi_cc.html

The essentials of the midi protocol
https://ccrma.stanford.edu/~craig/articles/linuxmidi/misc/essenmidi.html

Incomplete Specification In-Work

Use of special terms:

This document adopts Section 13.1 of the IEEE Standards Style Manual, which dictates use of the words “shall”, “should”, “may”, and “can” in the development of documentation, as follows:

- The word **shall** is used to indicate mandatory requirements strictly to be followed in order to conform to the Specification and from which no deviation is permitted (shall equals is required to).
- The use of the word **must** is deprecated and shall not be used when stating mandatory requirements; must is used only to describe unavoidable situations.
- The use of the word **will** is deprecated and shall not be used when stating mandatory requirements; will is only used in statements of fact.
- The word **should** is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (should equals is recommended that).
- The word **may** is used to indicate a course of action permissible within the limits of the Specification (may equals is permitted to).
- The word **can** is used for statements of possibility and capability, whether material, physical, or causal (can equals is able to).
- All sections are normative, unless they are explicitly indicated to be informative.

Protocol

This body of work is based on the protocol work done in MIDI 1.0. Standards documents are available from

http://www.midi.org

High Level Overview (Informative)

- Content is generated in the standard MIDI Format
  - Pre-Authored content may be shared in standard midi-file format (including timestamps) to be rendered alongside the Audio and Video content. (similar to a subtitle file)
  -
- Each Channel can have a Haptic Actuator (HAP-TYPE) and Mode Of Operation (MOO)
  - Each HAP-TYPE can have multiple different MOOs, on a per channel basis.
    - e.g. HAP-TYPE Vibration can have MOO:0x00 on Ch0, and MOO:0x01 on Ch1.
  - HA)-TYPE is specified per channel, using CC14
  - MOO is specified per channel using CC15
- NOTE ON/OFF events start and stop the haptic event for all HAP-TYPES and MOO
- NOTE KEY Parameters address the target body location should be driven giving a range of 127 locations.
  - The renderer shall decide which haptic accessories it has, that should be driven.
    - example - 5 individual left hand fingers should simply be routed to a left-hand-palm vibration if that is all that is available.
- Velocity, CC16 and CC values other than CC14/15 are MOO Specific. Their conversion and usage is defined by each MOO specifically.

Body Location List

Being worked on in a separate document until complete

Vibrotactile Transducers (CC14: 0x00)

Vibration Haptic Transducer, that covers ERM, LRA, Piezo and other Vibration. These transducers are either DC or AC wave driven. ERM's are DC (with a PWM'd amplitude) or LRA (use on-resonance tracking or off-resonance tracking).

Transducer type needn't be specified by the media creator. It is the role of the Hardware Vendor to decide for these signals should be best converted for the transducers they have.

Vibration Mode of Operation	Explanation
0x00 0x01 0x02	Streamed Envelope Triggered Patterns Hi-Fi Mode

1. Streamed Envelope: Creator decides what content to push to end user, this is a pre-defined stream of data sent to device e.g.a MIDI file containing all amplitude envelopes of vibration mode. Some additional control of the vibration frequency is possible using CC16 values.
2. Triggered patterns: equivalent to a DLS patch, downloadable patterns or "samples" of haptic content that can be shared amongst users, in the same way that wavetable synth patches, VSTs could be shared in music creation tools. Users could create MIDI files, or content creators could create MIDI files that can be played alongside media (like sample/wavetable patch). Sample data is stored in the end equipment and triggered from individual note-on commands.
3. HIFI mode: HI-FI mode is the equivalent of streaming audio (e.g. WAV): It tells the end device instead of playing the MIDI version, to use the streamed wave file content.

MOO 0x00 Envelope Streaming

This mode of operation uses the NOTE Parameter to describe the location on the body and uses the VELOCITY Parameter to transmit updates to the current amplitude. Timing information is at the discretion of the renderer.

CC16 is used to transmit the AC frequency mode, if it's supported by the transducer.. See below for details.

Most AC driven transducers still have an extremely resonant frequency response. This is best tracked locally using local hardware (such as DA7280 LRA driver IC's).
Alternatively, some developers may choose to run the LRA in an open-loop "off resonance" mode, where the frequency is fixed and updated by the content creator. This is mostly done for interesting audio effects from the LRA,in systems where you may not have an audio loudspeaker (for example, a gaming controller).

Midi Parameter	Description	Formula to Hex
NOTE	Body Location	See table.
VELOCITY	amplitude between 0% and 100%	0x00 = 0% 0x64 -> 0x7F = 100%
CC16	AC Frequency	0x00 - Resonant Frequency Tracking 0x01->0x7F Audio Frequency matched to Piano-Range of midi. (e.g. 0x45 = Middle A = 440Hz)

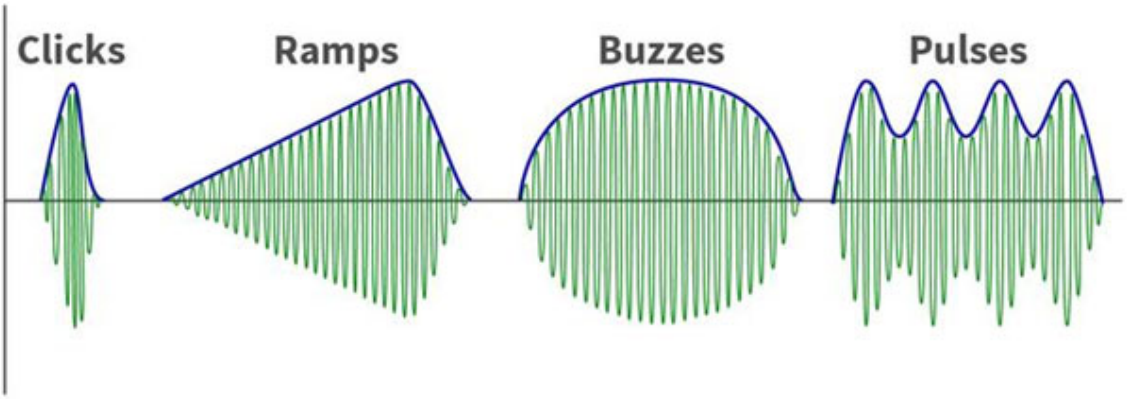
MOO 0x01 Triggered Sequences

This mode of operation uses the NOTE Parameter to describe the location on the body and uses the VELOCITY Parameter to tell the rendering device with pre-downloaded sequence to playback. Pre-Programmed sequences can either be downloaded before playback (e.g. at start of media, playback, game level etc) or during run time.

CC7 is used to scale the sequence amplitude between 0 and 100%.

CC16 is used to transmit the AC frequency mode, if it's supported by the transducer.. See below for details.

Most AC driven transducers still have an extremely resonant frequency response. This is best tracked locally using local hardware (such as DA7280 LRA driver IC's).



https://www.precisionmicrodrives.com/content/which-haptic-effects-should-you-use/

Alternatively, some developers may choose to run the LRA in an open-loop "off resonance" mode, where the frequency is fixed and updated by the content creator. This is mostly done for interesting audio effects

from the LRA,in systems where you may not have an audio loudspeaker (for example, a gaming controller).

NOTE: A method to transmit samples needs to be added to this section.
SDS (https://dpmv3.wordpress.com/2016/04/12/sds-the-midi-sampling-dump-standard/) could be used.

Midi Parameter	Description	Formula to Hex
NOTE	Body Location	See table.
VELOCITY	Pattern Selection	0x00 = Pattern 0 0x7F = Pattern 127
CC7	Volume (Scaling of sample)	0x00 = 0 0x64 -> 0x7F = 100%
CC16	AC Frequency	0x00 - Resonant Frequency Tracking 0x01->0x7F Audio Frequency matched to Piano-Range of midi. (e.g. 0x45 = Middle A = 440Hz)

MOO 0x02 HiFi VibroHaptics

Advanced users of haptics may prefer to use an additional audio track as their absolute, or envelope signal. This mode is set up to simply say “Haptics Data for Body Location X can be found on Audio Track Y, in Absolute or Envelope form”

Absolute simply means data should be treated in a blind-passthrough manner. A DAC to a Class D, to the actuator. Envelope more will simply be resonance below a fast moving envelope.

Note and Velocity refer to the body position and the audio channel to use.

Midi Parameter	Description	Formula to Hex
NOTE	Body Location	See Table
VELOCITY	Audio Channel	0x00 Channel 0 0x7F Channel 127
CC16	AC Frequency Mode	0x00 - Use Audio Ch. as DAC. 0x01 - Audio Ch. is Envelope, use resonance 0x02->7F - Audio Ch. Is Envelope, use MIDI note frequency.

Thermal Transducers (CC14: 0x01)

Thermal transducers may be something as simple as a Peltier element with a closed loop controller attached to it. Media authors needn’t concern themselves with the method of control. The end system could either be open-loop (calibrated) or closed loop.

MOO 0x00 Direct Temp

NOTE ON signals will set the transducer to a fixed temp and hold.

NOTE OFF will disconnect drive signal and allow the transducer to come to equilibrium with the user (skin temperature?)

MIDI Parameter	Description	Formula to Hex

NOTE	Body Location	
VELOCITY	Temperature in Celsius	0x00 = 0 Celsius 0x7E = 127 Celsius
CC16		

Visual LED Transducers (CC14: 0x02)

LEDs may not be haptic as such, but the ability to light up specific parts of the body along with vibration could be very useful (e.g. showing a user playing an Augmented Reality game where they were hit, along with the vibration!)

MOO 0x00 HSV Mode

NOTE ON signals will set the transducer to a fixed temp and hold.

NOTE OFF will disconnect drive signal and allow the transducer to come to equilibrium with the user (skin temperature?)

Midi Parameter	Description	Formula to Hex
NOTE	Body Location. NoteON = LED ON NoteOff = LED OFF	
VELOCITY	Value (in the HSV)	0x00 = 0% 0x64 = 100%
CC7	Saturation (set before the Note command)	0x00 = 0% 0x64 = 100%
CC16	Hue (set before the Note command)	Hue is traditionally a value between 0 and 360. Value = Hex * 3 e.g. Dec120 = 360

Electrical Stimulation Transducers (CC14: 0x03)

TBD

MOO 0x00 HSV Mode

TBD

Midi Parameter	Description	Formula to Hex
NOTE	Body Location.	
VELOCITY		
CC7		
CC16		

Appendix A: Document Templates

TEMPLATE: HAP-TYPES and MOOs

HAP-TYPE NAME

[Insert Description of the Haptics Actuator] - .e.g Vibration Haptic Transducer, that covers ERM, LRA, Piezo and other Vibration.

MOO 0x00 NAME

[Insert Description of the AIM of this MOO]
[What Functionality does it aim to achieve]

Midi Parameter	Description	Formula to Hex	
NOTE			
VELOCITY			
CC16			