

Actuator **18** may be, for example, an electromagnetic actuator, an Eccentric Rotating Mass (“ERM”) in which an eccentric mass is moved by a motor, a Linear Resonant Actuator (“LRA”) in which a mass attached to a spring is driven back and forth, or a “smart material” such as piezoelectric, electroactive polymers or shape memory alloys. Memory **20** can be any type of storage device, such as random access memory (“RAM”) or read-only memory (“ROM”). Memory **20** stores instructions executed by controller **12**. Memory **20** may also be located internal to controller **12**, or any combination of internal and external memory.

FIG. 2 is a block diagram of the system architecture of a haptic feedback system **100** in accordance with one embodiment. System **100** includes two or more applications **30** and **32** that call, via software commands, for a specific stored haptic effect to be played. Application program interface (“API”) **34** interprets the software commands and outputs to two parallel paths, one for each type of actuator. In other embodiments, one or more paths may be implemented. In FIG. 2, the left-side path is for a piezo actuator **50** while the right-side path is for an LRA actuator **52**.

Coupled to each path is a digitized stream envelope (“DSE”) construct **36** which includes stored haptic effects. In one embodiment, four different haptic effects are stored, but any number may be stored. Each path includes a driver **38** and **40** and a service provider interface **39** and **41** that include information on the specific type of actuator for the path. This allows the API to be hardware/actuator independent.

Each path further includes, for each actuator, a drive signal **42** and **46** and an electrical drive circuit **44** and **48**. Finally, actuators **50** and **52** generate the vibration or other desired haptic effect.

In one embodiment, the piezo actuator driver expects a differential sine wave control signal. One embodiment generates a sine wave in software at a 5 kHz rate and outputs that as a pulse width modulation (“PWM”) signal. A simpler control signal may be used to achieve the same feel on the device. However, in one embodiment, the piezo circuit requires a PWM signal with fixed frequency in the 20-50 kHz range, whose duty cycle is expected to be updated every 200 μ s.

In one embodiment, there are at least 3 possible physical configurations for the LRAs:

1. Case mounting—In this configuration, the LRAs are mounted rigidly to the interior of the device casing. The LRA is driven at its resonant frequency (from 175 Hz to 185 Hz), and the entire device is actuated.
2. Screen mounting—matched frequency. The screen is floated on a suspension, mechanically isolated from the casing, and the LRAs are rigidly mounted to the screen. The LRAs is typically driven at the LRA resonant frequency, and the suspension is tuned to reinforce that frequency.
3. Screen mounting—bi-modal. The screen is floated on a suspension, mechanically isolated from the casing, and the LRAs are rigidly mounted to the screen. The LRA can be driven at the LRA resonant frequency in which case the suspension is tuned to transmit most of the vibration to the casing. The LRA can also be driven at the system’s natural frequency, with the suspension tuned to a higher frequency than the LRA’s own resonant frequency (typically approx. 500 Hz). In this configuration, the LRA can be used both as an event alerting system (Silent/Manner mode vibration alerter) or as a touch-screen feedback system (for button press tactile confirmation).

In one embodiment, many variants of the known VibeTonz® LRA drive circuit from Immersion Corp. can be used. Generally, these drive circuits require one PWM at 20-50 kHz fixed frequency, variable duty cycle, 8-bit duty cycle resolution, 50%=no output, with a fine granularity in the 20-30 kHz range (as the carrier frequency is a multiple of the LRA drive frequency in many designs). The circuit also requires a General Purpose Output (“GPO”) for AMP_ENABLE control.

Alternately, an even simpler amp circuit could be used, because a 5 or 8 kHz output rate from the driver would allow output of sine or square waveforms at the LRA resonant frequency. This eliminates the need to fine-tune the PWM frequency, as the resonant frequency would itself be encoded.

FIG. 3 is a block diagram of the software architecture in accordance with one embodiment. One embodiment of the architecture includes the following:

1. No custom effect playback for applications. Only predefined effects (“stored haptic effects”), stored in Digitized Streaming Envelope (“DSE”) constructs stored in the driver software can be played back.
2. No real-time generation. Effect playback is based on a pre-recorded control signal, not generated in real-time. The control signal is computed using an effect design tool.
3. Multiple application support. Multiple applications could register with the API simultaneously. Supporting multiple applications requires some form of API client marshaling, where the API must determine whose request is most important. Two approaches are possible:
 - a. Last caller wins. When multiple applications try to use the vibration resource simultaneously, the last caller interrupts whatever was playing before and its effect plays.
 - b. Priority scheme. The API could support a concept of high/medium/low priority on effect playback launch. When launching an effect, the caller specifies the priority to be used. Playback succeeds when priority is equal to or higher than the current effect’s playback priority.
4. Designed for portability. Like VibeTonz®, ANSI C only, no dynamic memory allocation.

In one embodiment, the Driver Access Interface (“DAI”) consists of lower-level functions that provide the bulk of the API functionality. In one embodiment, the DAI is designed to be implemented as a functional interface, which is easily wrapped in a serial protocol. The driver is a timed loop that executes the following commands:

1. Look at API/Driver shared memory. If playback is scheduled, retrieve current DSE pointer from memory, along with any other required driver/effect state information.
2. If state is “playing effect”, decode DSE, extract PWM value to be applied to PWM. Then write to PWM. Sleep until next sample.
3. If state is “finished iteration check for repeat”, and if effect is to be repeated, decrement repeat value, set up gap timing and sleep until gap time expires.

In one embodiment, DSE construct **36** of FIG. 2 is a set of magnitudes, or strengths, over time, for a number of effects. The samples are stored in a lossless compact format. Files containing DSE information use the .dse file extension.

FIG. 4 is a block diagram of the file format used in one embodiment for the DSE construct. The DSE 1.0 Header Block contains file format version information, number of effects, location of the Effect Storage Block, location of the Effect Set Name Block, and file size. The Effect Storage