Driving Piezoelectric Actuator using BOS1901

Piezoelectric actuators convert force applied on them to voltage, and voltage applied on their terminals to displacement. This behavior can be used to sense pressure applied by a user to a device, but also to provide haptic feedback. The BOS1901 actuator driver supports both capabilities in a single product.

This document explains how to use a BOS1901 to drive a haptic feedback on a piezoelectric actuator. Typical playback sequences are given.

1 System Overview

Since BOS1901 features both sensing and feedback driving, only one circuit is required to fully control the piezoelectric actuator. A typical solution will thus require three components: the piezoelectric actuator, the driver, and the microcontroller (MCU) or application processor.

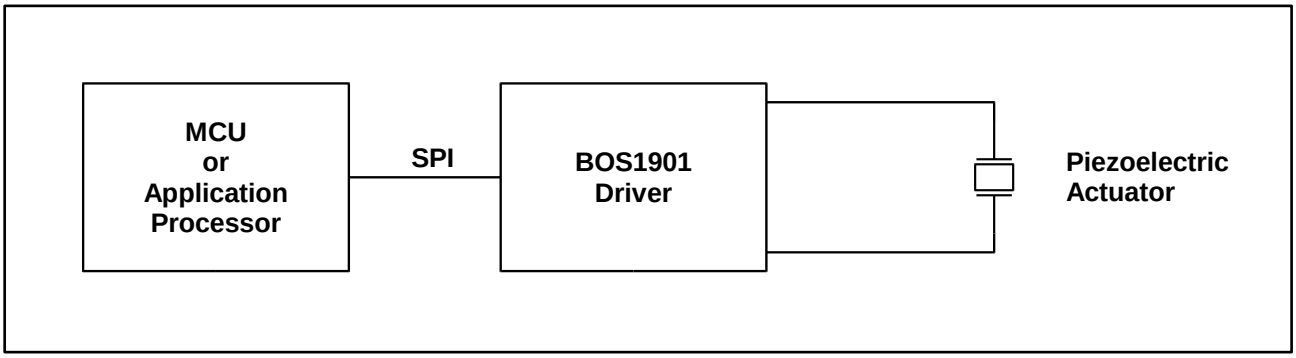


Figure 1 System components using piezoelectric actuator

1.1 Piezoelectric Actuator

The actuator converts voltage to displacement, and displacement to voltage. Through its integration in a device, it can sense how much pressure the user is exerting on the device surface, and it can produce vibrations of various kind to be felt by the user. It is the component that will be in direct interaction with the user.

When compressed, it converts mechanical stress to charges and voltage. This voltage can be measured and processed to determine how the user is interacting with the device, whether is it pushing a button, a screen or any other surface, or releasing it.

When a voltage is applied on its terminals, it tends to deform. Through use of proper voltage waveforms, some effects and vibration can be played to give the user the impression of clicking a button or feeling a texture for example.

1.2 BOS1901 Driver IC

Electrical circuits must be used to measure the voltage on the actuator terminals and to generate the high voltage required to deform it and create a force.

BOS1901 was designed to drive piezoelectric actuators in haptics application, where the signal frequency content is typically below 300 Hz. It was designed to optimize power and be small enough to be integrated into battery-operated mobile devices. It features both sensing and feedback capabilities thereby reducing significantly the bill-of-material.

When sensing, it can read the voltage generated across the actuator and send it to the MCU via the SPI interface. When playing a feedback to be felt by the user, it drives the high voltage required for the actuator to vibrate. Any waveform shape, amplitude and frequency may be played by the BOS1901. Samples are sent via the SPI interface to the internal FIFO to be used at the predefined sampling rate.

The SPI interface operates in full duplex, allowing to write operations and read information at the same time.

1.3 MCU or Application Processor

Although BOS1901 provides voltage measurement and driving capabilities it does not embed detection algorithms. These algorithms reside in the MCU or application processor connected to the BOS1901 by the SPI interface.

This allows any application to use the appropriate complexity for the sensing algorithm, and to change the feedback waveforms given the context and interaction with the user.

The MCU determines the operation mode of the BOS1901, handles its register configuration and initialization, processes the voltage readings to realize the appropriate sensing algorithm, and generates the waveforms to be played to it. All application-specific code is running on the MCU. The BOS1901 is acting mostly as the combination of a driver and sensor circuit.

2 Driving Methodologies

This section describes driving methodologies to play feedback waveforms on the BOS1901 output. Only the driving part is explained.

2.1 Using Internal FIFO

BOS1901 manages data flow using an internal FIFO buffer (First-In, First-Out) that may contain up to 64 entries. Each entry is the 12-bit 2's complement representation of a voltage to play on the driver output.

FIFO entries are created every time data is received on the REFERENCE register (0x0) on the SPI bus.

FIFO entries are transferred one by one to the driver analog output stage at the sampling rate programmed in the PLAY parameter of the CONFIG register (0x5). When playing samples, the driver output stage will interpolate the voltage between the last sample and the new to smooth the voltage waveform and minimize quantization noise.

The FIFO clears itself by playing out all its entries. To clear the FIFO faster, it is possible to change the PLAY parameter to a faster sampling rate. Resetting the IC is not recommended since it also erases the register configuration.

The advantage of using the FIFO is to optimize the communication events between the MCU and the BOS1901. The MCU fills up the FIFO and then performs some other tasks before sending new samples. The FIFO status can be queried. It is possible to determine if the FIFO is full, empty and how much room is available. This information is then used to send the right amount of samples.

The figure below explains how to the FIFO work. In this example, seven samples have already been sent to the FIFO. Next, the data 0x0NNN is written on the SPI interface. This data points to register address 0x0 and contains REFERENCE data 0xNNN. This data is written at the end, to location 8. Then the FIFO is read so the driver can retrieve a sample to output. The first sample that was input was DATA1, so it is extracted and sent to the output driving stage. All other samples are moving up, so in the end there is now only seven samples in the FIFO.

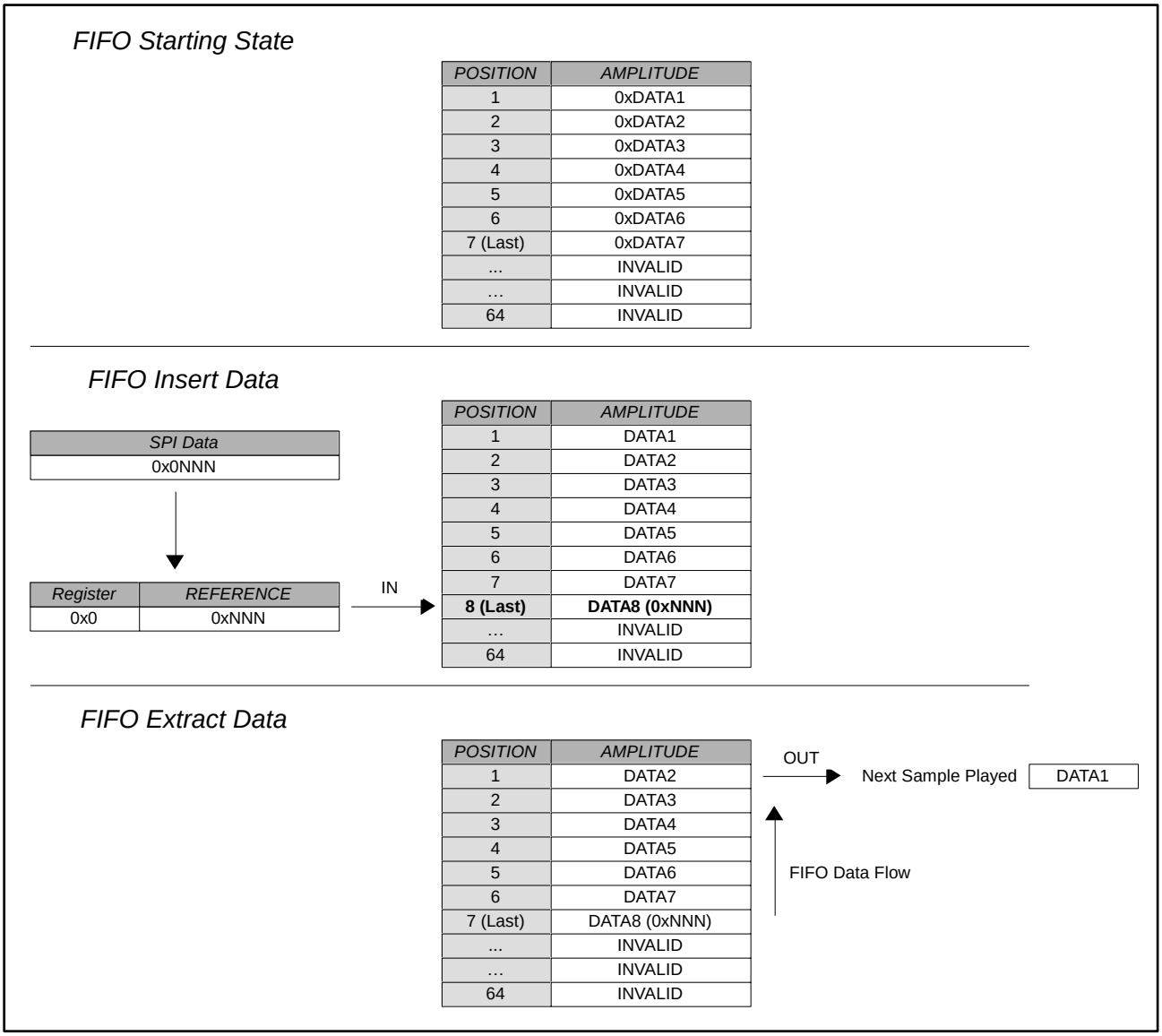


Figure 2 FIFO Principle

2.2 Direct Sampling

There are cases where it is preferred to send samples at a fixed rate and that the samples be played as they are sent. In such cases, the MCU would control the sampling time. Examples include streaming data from the USB port. Data rate is then determined by either the USB master device or the MCU. Another example is when two BOS1901 need to be synchronized. Each driver internal oscillator does not have the same frequency as the other, even after factory-trimming there is still a small difference. This difference creates a phase shift between the output signals of the drivers which increases with time and may become significant.

Direct sampling is achieved by keeping the BOS1901 FIFO empty. While there is still data in the FIFO, when the period defined by the PLAY parameter a new sample is sent to the output driver to be played. If the FIFO is empty when the output stage requests a new sample, then the output voltage is maintained

until new data is sent to the FIFO. Then, when a new sample is sent to the BOS1901, it is automatically sent to the output stage to be played.

To ensure the FIFO stays empty every time a new sample is periodically sent, the driver must be able to play them at a rate slightly faster than the input rate on the SPI interface. This can be done two ways.

The PLAY parameter could be programmed to a higher sampling rate than the MCU will use for the SPI data transfer. Although this method works from a data transfer standpoint, it has the undesirable side effect of distorting the output signal, creating digital quantization noise. Let's assume samples are sent over SPI at an 8 kSPS rate and the driver PLAY parameter is set to 16 kSPS. Each sample will be interpolated from the previous one to the next one in 62.5 µs (16 kSPS), then remain at the final value for another 62.5 µs. This induces discontinuities in the waveform shape and create audible noise.

A better method is to trim the BOS1901 internal oscillator slightly faster than the factory-trimmed value in order to have it play samples slightly faster than the SPI data transfer cadenced by the MCU. Using this method, the BOS1901 will complete playing a sample some small amount of time before the next sample is sent and the output stage will be ready to play it as soon as it is sent to the FIFO.

The basic trimming principle is the following. Operate the MCU to send dummy data to the BOS1901 internal FIFO at a frequency slightly higher than the one intended (about 2.5 %). For example, to drive the BOS1901 at 8 kSPS, set the MCU data rate at 8.2 kSPS. Query the BOS1901 to check if the FIFO accumulates samples. If it does, trim it up one increment. And then repeat this process until the FIFO does not accumulate any sample. This process of trimming the oscillator takes into account both the BOS1901 internal oscillator and the MCU internal oscillator accuracies automatically.

See the application note *BOS1901 Oscillator Trimming* for more details.

3 Actuator Voltage Sensing

This section describes how to perform the basic operations required to control the BOS1901. It is more precise on the instructions to send over SPI. Please consult the BOS1901 datasheet available on Boréas Technologies website for more details.

3.1 Reading a Register

To drive the BOS1901 reading register content can be useful. This is done in three steps:

1. Setting which register to broadcast on the SPI interface.
2. Sending another word on the SPI interface.
3. Reading the register content received on the SPI interface during step 2.

To set which register will be broadcast, the BC parameter in the CONFIG register must be set to the value of this register. For example, to read the FIFO_SPACE parameter, BC must be set to 0xC, the address of the IC_STATUS register.

Once the BOS1901 has been configured to broadcast the desired register, this register is read by writing any word on the SPI interface and reading SDO at the same time. Be careful to write a word that will not change the configuration or state of the IC unless intended. For example, write the same word again, or write to a read-only register.

It is possible to set the next register to be read while reading the currently set broadcast register. For example, the following sequence is possible:

1. Set BC = 0x1 address. This sets register 0x1 to be read.
2. Write CONFIG with BC = 0x2 address and read SDO at the same time. This reads register 0x1 while setting subsequent broadcasts to 0x2.
3. Write CONFIG with BC = 0x3 address and read SDO at the same time. This reads register 0x2 while setting subsequent broadcasts to 0x3. And so on.

3.2 Register Setup

For driving to work properly a few registers must be configured. Except the following, most registers may be left either to their default state or to the values calculated using the equations given in the datasheet.

Driving is done by sending voltage samples to the BOS1901 FIFO. This FIFO will be read at the sampling rate programmed in the PLAY parameter of the CONFIG register (0x5).

Depending on the situation, setup for driving may also require trimming the internal oscillator. Please read *BOS1901 Oscillator Trimming* application note for details.

Finally, driving requires the SENSE bit in the SUP_RISE register (0x7) to be set to 0 beforehand.

3.3 Bridge Polarity

BOS1901 contains an output bridge to achieve 190 Vpp by alternating the voltage on the piezoelectric actuator from +95 V to -95 V. This is important to understand when testing in order to connect instruments properly, but also when designing the system or device in which the BOS1901 is integrated. See *Probing BOS1901 with an Oscilloscope* application note for details.

When outputting +95 V, BOS1901 is in positive polarity. Pin OUT+ is set to VDD+95 V while pin OUT- is set to VDD. Thus the voltage difference seen by the actuator is +95 V.

When outputting -95 V, BOS1901 is in negative polarity. Pin OUT+ is set to VDD while pin OUT- is set to VDD+95 V. Thus the voltage difference seen by the actuator is -95 V.

In all situations, voltage on OUT+ and OUT- is driven between VDD and VDD+95 V. This means both piezo terminals are above the ground and considered active. Bridge polarity is automatically set by the FIFO value sent in the REFERENCE register.

The figures below show how OUT+ and OUT- are behaving when the programmed voltage is positive (left) and negative (right). In both cases, both OUT+ and OUT- output a positive voltage. It is the voltage difference between both terminals that changes polarity.

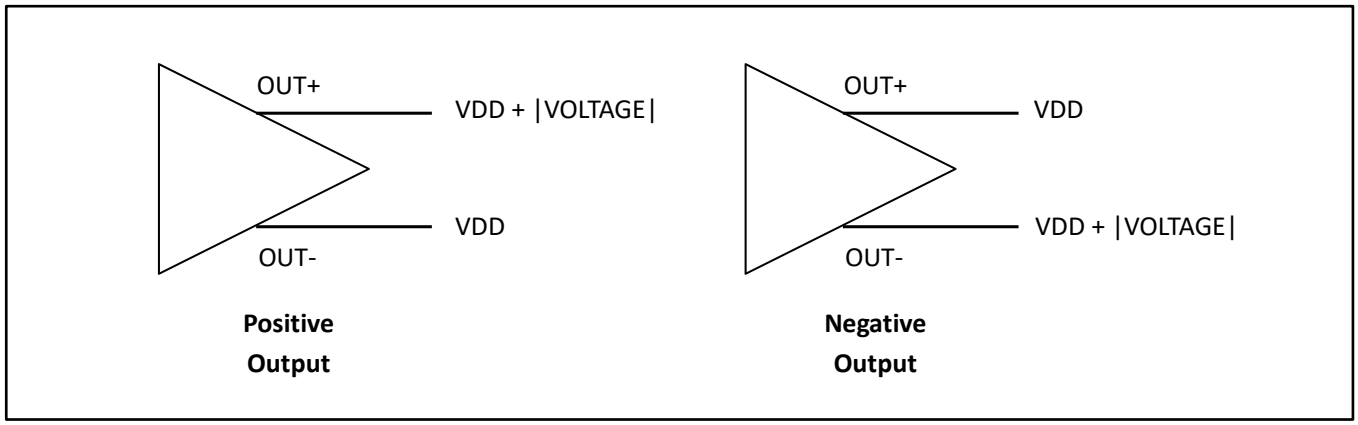


Figure 3 OUT+ and OUT- voltage level when programming a positive (left) and a negative (right) voltage

3.4 Sending Voltage to Output

When the output is enabled (OE = 1) and sensing is disabled (SENSE = 0), FIFO data is automatically sent to the output at the programmed sampling rate (PLAY). Therefore, sending data to play is only a matter to sending samples to the BOS1901 FIFO.

This is done by writing the FIFO parameter in the REFERENCE register (0x0). This is valid for both FIFO-based driving and direct sampling.

The equation that converts voltage to FIFO 12-bit 2's complement representation is given in the product datasheet and below for convenience.

$$Amplitude = \frac{V * (2^{11} - 1)}{V_{ref} * FB_{ratio}}$$

Where *V* is the voltage to play, *Amplitude* is the 12-bit code to send over SPI, V_{ref} is 3.6 V and FB_{ratio} is 31.

Usually waveforms should start and end to 0 V. If the voltage of the signal drops below 2 V, it should be made to either reach code 0x0000 or cross it to reach a negative voltage before rising back up, otherwise the internal logic might get in a condition where the output does not respond anymore and a reset is required to get out of that state.

3.5 Initiating the Playback

Driving requires the SENSE bit in the SUP_RISE register (0x7) to be set to 0 beforehand. Then, the output must be enabled by setting the OE bit to 1 in the CONFIG register. When the OE bit is raised to 1, the output driver automatically starts reading the waveform contained in the FIFO.

It is not necessary to fill the FIFO entirely before enabling the output. For example, a single full sine cycle of 200 Hz played at 8 kSPS will only need 41 samples to play completely. It is possible to only send the 41 samples and then enable the output and let it play out.

It is possible to keep the OE bit set to 1 while the FIFO is empty, and then start to fill the FIFO. This is convenient when switching between sensing and driving. OE is always kept at 1, only the SENSE bit is changed between 0 when driving and 1 when sensing.

If the FIFO is empty when OE is set to 1 (and SENSE = 0), the output voltage will stay to the last programmed value. As a new sample is programmed in the FIFO, it is immediately played.

3.6 Piezo Creep

When a waveform finished playing, the actuator will tend to continue moving and this will induce a voltage change on the actuator. This is referred to as piezo creep.

For driving haptic feedback purposes, it might not cause issues. However, since piezoelectric actuators can be used for both sensing and feedback, waveforms playback phases are generally followed by sensing phases. And this voltage change can be high enough in some cases to exceed the sensing threshold, thereby inducing an undesired threshold detection and creating undesired effects.

To prevent problems related to piezo creep, it is often necessary to actively drive the actuator to its rest value for some time until the actuator has settled enough for the following voltage rise to be within tolerable limits.

After playing a positive waveform, piezo creep is stabilized by maintaining a small negative voltage (0x0FFF) for some amount of time.

After playing a negative waveform, piezo creep is stabilized by maintaining a small positive voltage (0x0001) for some amount of time.

Read *Sensing Piezoelectric Actuator Force using BOS1901* application note for more details.

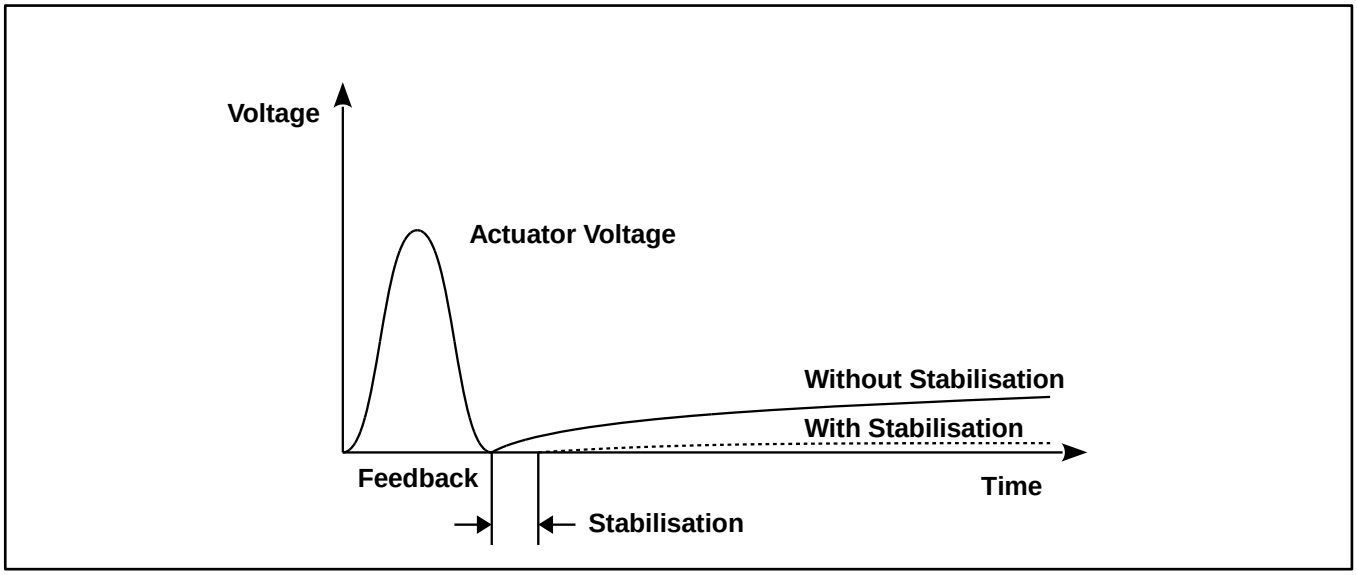


Figure 4 Piezo creep following feedback waveform

3.7 FIFO Management and Monitoring

IC_STATUS register contains information about FIFO state and other information that can be read at runtime such as error flags. FIFO-related parameters are

1. FULL: indicates if the FIFO is filled with data.
2. EMPTY: indicates if the FIFO is empty.
3. FIFO_SPACE: 6-bit unsigned integer indicating how much space is available in the FIFO.

These parameters can be read regularly to determine when MCU should dedicate resources to send new samples.

4 Typical Driving Sequences

This section provides high-level typical sequences of driving an actuator using both FIFO-management and direct sampling methods. Please refer to the companion example code for further details.

4.1 FIFO Management Example

Initialization

1. Reset the IC
   - CONFIG register / RST = 1 – this bit auto-clears.
2. Set SUP_RISE register
   - SENSE = 0 – to disable sensing
   - All other parameters to default values
3. Set CONFIG register
   - BC = IC_STATUS – to monitor FIFO status
   - OE = 1 – in this example, we enabled the output first; output will start at 0 V by default and then follow the waveform samples sent to the FIFO.
   - PLAY = 7 – 8 kSPS sampling rate

Playing the waveform

4. Send 64 samples to the REFERENCE register (0x0xxx)
5. Do something else in the MCU.
6. Check if there is room in the FIFO.
   - If yes, send as many samples as possible.
   - If not, check again later.
   - For example, in our code example, in sensing.c file, function sensingPressFeedback, we wait until the FIFO is completely empty before sensing another batch of 64 samples.
   - Repeat until all waveform samples have been sent.
7. When the waveform is completed, stabilize the piezo (piezo creep).
   - Either send as many samples as needed for the stabilization delay to occur with the programmed sampling rate; or
   - Program the stabilization voltage and wait the stabilization delay.
8. Disable the output, unless it is need for another waveform or for sensing.

4.2 Direct Sampling

Initialization

1. Reset the IC
   - CONFIG register / RST = 1 – this bit auto-clears.
2. Trim the internal oscillator – See other application note.
3. Start MCU 8 kHz timer.
4. Set SUP_RISE register
   - SENSE = 0 – to disable sensing
   - All other parameters to default values

5. Set CONFIG register
    - OE = 1 – in this example, we enabled the output first; output will start at 0 V by default and then follow the waveform samples as they are sent.
    - PLAY = 7 – 8 kSPS sampling rate

Playing the waveform

6. For all waveform samples
    - Wait for the MCU 8 kHz timer to expire.
    - Send the next waveform sample to the REFERENCE register (0x0xxx)
7. When the waveform is completed, stabilize the piezo (piezo creep).
    - Either send as many samples as needed for the stabilization delay to occur with the programmed sampling rate; or
    - Program the stabilization voltage and wait the stabilization delay.
8. Disable the output, unless it is need for another waveform or for sensing.

4.3 Code Example

Code examples can be found on the Boréas Technologies website in the technical documents section, under BOS1901-KIT Firmware. Look for the sensing and driving examples. The comments explain each step.

They both follow the sensing phases explained in the application note *Sensing Piezoelectric Actuator Force using BOS1901*. Pressing the actuator generate simulated click-button behavior.

The sensing example uses FIFO-based playback for the feedback pulses.

The driving example uses timer-based (direct sampling) playback for the feedback pulses.

Study more closely the following functions:

- sensing.c
    - sensingWaitFifoEmpty: waits until the FIFO is empty to exit.
    - sensingInit: initialization instructions for sensing and driving.
    - sensingPressFeedback and sensingReleaseFeedback: play the feedback waveforms.
    - sensingPressCreepStabilization and sensingReleaseCreepStabilization: stabilize the piezo creep of the actuator.
    - sensingFifoDriveMain: main function, initializes then runs an infinite loop.
- driving.c
    - drivingInit: initialization instructions for sensing and driving.
    - drivingTrimming: main oscillator trimming function, determines if TRIM_OSC need to be changed.
    - drivingSoftwareTrimming: trimming utility function, retrieves TRIM_OSC value and increases it.
    - drivingPressFeedback and drivingReleaseFeedback: play the feedback waveforms.
    - drivingPressCreepStabilization and drivingReleaseCreepStabilization: stabilize the piezo creep of the actuator.
    - drivingFifoDriveMain: main function, initializes then runs an infinite loop.

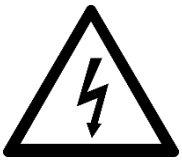
5 Related Parts

	PART NUMBER	DESCRIPTION
1	BOS1901	BOS1901 Piezo Haptic Driver with Digital Front End

6 Document Changes

ISSUE	DATE	Document Number	CHANGES
1	March 2020	BT001DAN03.01	Original document

7 Notice and Warning



Danger High Voltage!

Electric shock possible when connecting board to live wire. Board should be handled with care by a professional. For safety, use of isolated test equipment with overvoltage and/or overcurrent protection is highly recommended.

ESD Caution

This product uses semiconductors that can be damaged by electrostatic discharge (ESD). When handling, care must be taken so that the devices are not damaged. Damage due to inappropriate handling is not covered by the warranty.

The following precautions must be taken:

- Do not open the protective conductive packaging until you have read the following and are at an approved anti-static workstation.
- Use a conductive wrist strap attached to a good earth ground.
- If working on a prototyping board, use a soldering iron or station that is marked as ESD-safe.
- Always disconnect the microcontroller from the prototyping board when it is being worked on.
- Always discharge yourself by touching a grounded bare metal surface or approved anti-static mat before picking up an ESD - sensitive electronic component.
- Use an approved anti-static mat to cover your work surface.

Oscilloscope measurements:

Both OUT+ and OUT- are active outputs. When measuring these signals using an oscilloscope, use a separate probe on each output. Never connect the ground of a probe to one of these outputs. Doing so might damage the BOS1901-KIT and/or your oscilloscope. For more information please consult the *Probing BOS1901 with an Oscilloscope* application note available for download on Boréas website.

