Sensing Piezoelectric Actuator Force using BOS0614

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. BOS0614 actuator driver supports both these capabilities in a single product, enabling haptic button emulation on multiple actuators at the same time.

This document explains how to use the BOS0614 sensing capabilities in various situations. Haptic feedback configuration is covered in another document. However, sensing to haptic sequence and configuration are still treated herein.

1 Fundamentals

Since the BOS0614 features both sensing and haptic feedback driving, only one circuit is required to fully control the piezoelectric actuator, enabling smaller, cheaper integration. A typical solution will thus require three components: the piezoelectric actuator, the driver, and the microcontroller (MCU) or application processor (HOST). Additionally, the BOS0614 offers GPIOs that can be used by the HOST as information input, interrupts, or can be programmed to behave as open-drain outputs, enabling direct button replacement in the device.

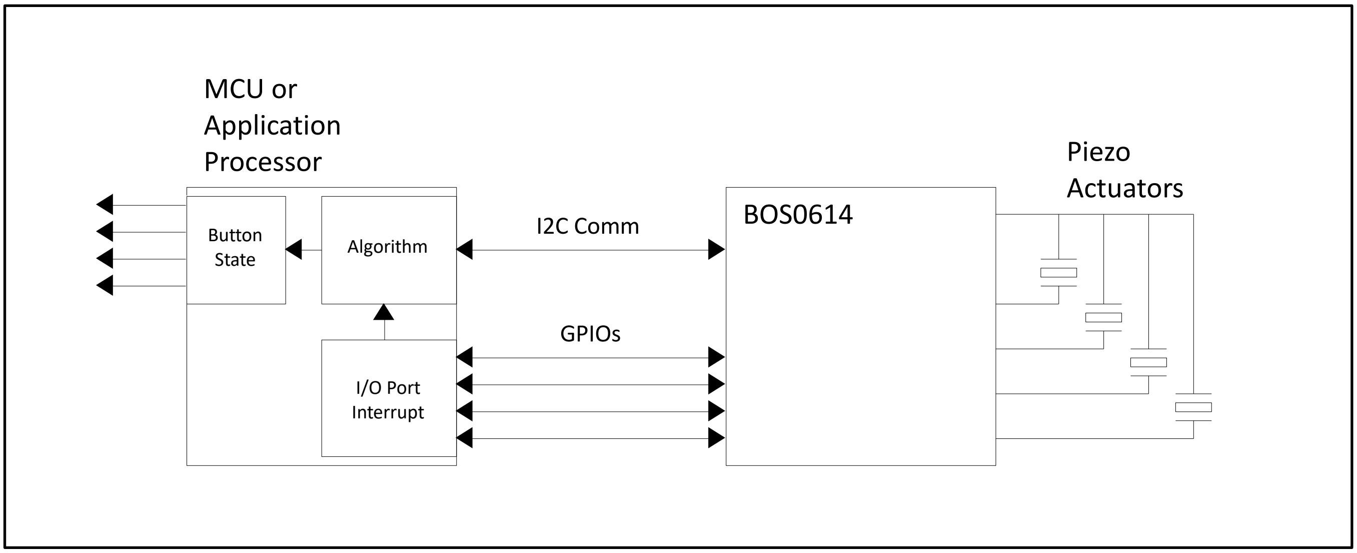


Figure 1 System components using piezoelectric actuator

1.1 Piezoelectric Actuator

The actuator converts voltage to displacement, and displacement to voltage. When placed in a device, the actuator can be used to sense how much pressure the user is exerting on the surface of the device. It can also be used to produce various kinds of vibrations to be felt by the user.

When a force is applied to the actuator, the resulting mechanical stress is converted into charges and voltage. This voltage can be measured and processed to sense how the user is interacting with the device, whether it is pushing a button, a screen, or any other surface, or releasing it.

Conversely, when a high voltage is applied on the actuator terminals, the actuator generates a displacement. When applying voltage waveforms across the actuator terminals, haptic effects and vibrations can be played to give the user the impression of clicking a button or feeling a texture for example.

However, electrical circuits are needed to sense the voltage on the actuator terminals and to generate the voltage required to deform it and create a haptic effect. The BOS0614 does both.

1.2 BOS0614 Driver IC

The BOS0614 is designed to drive piezoelectric actuators in haptics application, where the signal frequency content is typically below 300 Hz. It is designed to optimize power and be small enough to be integrated into battery-operated mobile devices. It features both sensing and haptic feedback capabilities thereby significantly reducing the bill-of-materials. Up to four (4) channels can be sensed at the same time, and while other channels are playing feedback.

For sensing, it measures the voltage generated across the actuator and determines if the required conditions for successful detection have occurred. The state of detection can either be read over the I2C interface or sent to the application processor over the corresponding GPIO output.

When playing haptic feedback, the BOS0614 drives the high voltage required for the actuator to vibrate. Any waveform shape, amplitude and frequency may be played by the BOS0614. Various playback modes are also available (see other application notes or the product datasheet for details).

1.2.1 Embedded Sensing

The BOS0614 has embedded sensing capability. It can be configured to detect when an actuator voltage is above or below a given threshold for some amount of time. It can also detect if the voltage rate change (dV/dt) is above or below a given slope. Combinations of these detection mechanisms can be configured for both events when the actuator is press and released. This allows to emulate a haptic button and the state (pressed or released) of this button is tracked automatically by the BOS0614 for each channel individually. Each channel can be configured independently, each with its own sensing parameters, detection mechanisms, and haptic feedback waveforms.

Direct access to sensed voltage is possible at any time over the I2C interface. This allows for advanced sensing, like voltage profiling or pattern recognition, but such processing needs to run in the application processor or a dedicated MCU. In this case, the BOS0614 is used to read the voltage on each actuator over time. The voltage can be read even while the BOS0614 is configured for automatic detection. This enables external processing to run in parallel with embedded sensing.

Sensing runs independently on all channels and will still be active while a feedback waveform is played on another channel. If one channel has successful detection while a waveform is playing on another channel, the event is queued, and the effect will be played on the channel as soon as the other channel effect finished playing.

1.2.2 Waveform Output Modes

There are four (4) waveform output modes: RAM Synthesis, RAM Playback, FIFO and Direct modes.

RAM Synthesis uses the internal RAM memory to store complex waveforms built from sine entities. The waveform parameters (amplitude, frequency, cycles, and shape) are programmed, and the waveform is constructed by the BOS0614 waveform synthesizer when it is played. When using the embedded sensing feature and it is configured for automatic detection, the RAM Synthesis output mode can also be used to automatically play a haptic feedback upon successful sensing detection.

RAM Playback also uses the internal RAM memory but stores the waveform as sampled waveforms. Using this mode, it is possible to store arbitrary waveforms of any shape (square, triangle, heartbeat, etc). To use this mode in conjunction with the embedded sensing, it is useful to set GPIO outputs as sensing interrupts to the application processor so it can launch the waveform at the correct time. Note both RAM Synthesis and RAM Playback content can coexist in the RAM memory.

FIFO mode uses the 1024-deep samples internal FIFO to play a waveform. While the waveform is played, FIFO space is emptied, and new samples may be sent. This mode is useful when playing content that exceeds the memory size or that changes content often. Note the internal FIFO is making use of the same RAM memory to store the samples. Thus, before using RAM Synthesis or RAM Playback again after using the FIFO mode, the RAM memory must be reprogrammed since previously programmed waveforms have likely been overwritten.

Direct mode is similar to the FIFO mode, but samples must be sent one by one to the BOS0614. Using this mode, it is convenient to set the GPIO output to either NEXT DATA or FIFO EMPTY to indicate to the application processor when to send the next sample. This mode does not use the RAM memory block at all; using the Direct mode will not overwrite the previously programmed memory content. This mode is useful in closed-loop systems where the next sample is calculated at every iteration.

Please consult other application notes and the BOS0614 product datasheet on using these modes.

1.2.3 Configurable GPIOs

BOS0614 has four (4) GPIO output pins, one for each sensing/output channel. They are configurable individually to output various information directly to the application processor which can then use this information as needed, even to create interrupts.

The GPIOs can be configured as push-pull or open-drain. The GPIOs can be used as a direct physical replacement of mechanical tactile switches, using the piezo actuators are buttons instead (see figure below). To do so, configure the GPIOs as open-drain and set their output function to report the button state. The open drain will close and open as the haptic button is pressed and released.

The GPIOs can also be configured as input to allow an external hardware pulse to trigger a waveform directly or change internal button state.

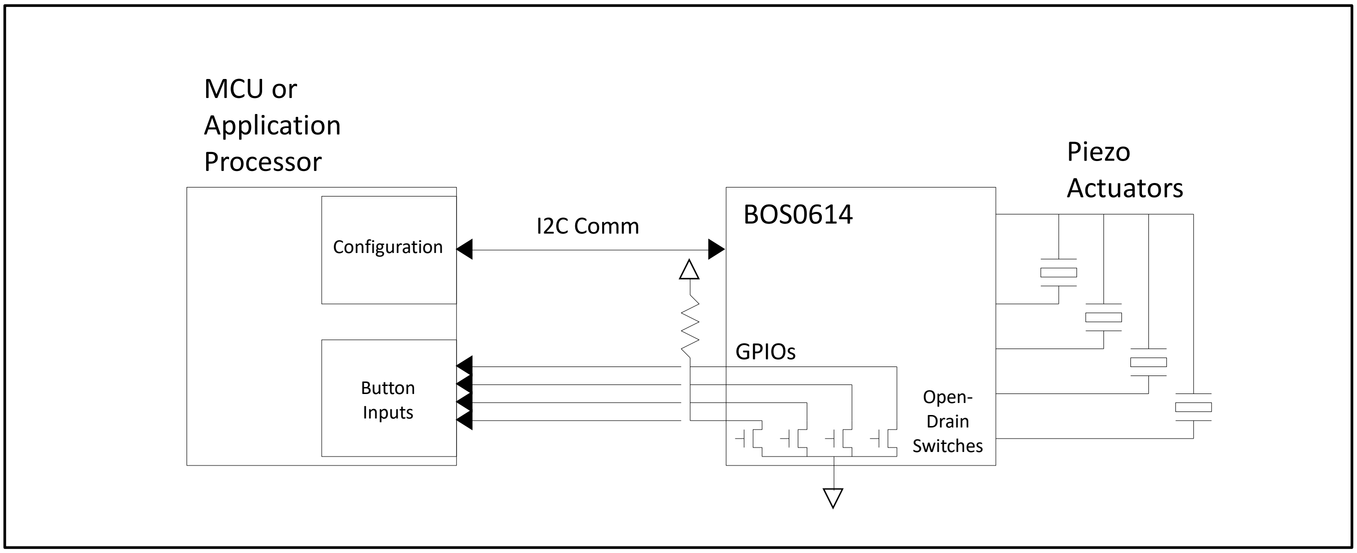


Figure 2 GPIOs configured as open drain can directly replace mechanical switches

1.2.4 ZPS

The Zero Power Sensing (ZPS) feature allows to set the BOS0614 to SLEEP and still maintain the haptic button active. This is useful to implement very low-power haptic button applications, such as power buttons that still needs to act as such even when the device is powered-off or in ship mode.

When the ZPS is active, if the user presses on the piezo actuator, the BOS0614 will wake up and assess if the sensing detection conditions are met. If they are met, the BOS0614 will play the pre-programmed waveform, wait for the release to occur, and return to SLEEP once the haptic button has been released. If the conditions are not met, the BOS0614 will automatically return to SLEEP. While the BOS0614 is in SLEEP, all registers and the RAM memory are maintained.

1.2.5 Powerup State

At powerup, the BOS0614 will have its four (4) channels set up with sensing enabled to detect a voltage positive threshold on a press and a negative slope on a release. GPIOx outputs are set up as open drain and set to give the state of the button, ready to be implemented in a mechanical tactile switch replacement application. ZPS is enabled so the BOS0614 will detect and report the state of any haptic button pressed while in SLEEP mode. Therefore, at powerup the BOS0614 is already set up to detect the haptic button being pressed by default. No waveforms are programmed to play; however, they should be configured at first powerup of the application device, before returning to SLEEP.

1.3 MCU or Application Processor

BOS0614's embedded sensing and automatic feedback capabilities reduce the communication time over the I2C interface and removes the need for a dedicated MCU in most applications. The application processor can program the BOS0614 and let it run in a standalone configuration. The BOS0614 can also be set up to trigger interrupts to the application processor as needed.

If more complex processing is required, the processor can poll the sensed voltage on each channel over the I2C interface and run application-specific algorithms, like voltage profiling or pattern recognition.

In this case, it would fully control the operation mode of the BOS0614, handling its register configuration and initialization, configuring the embedded sensing, processing the voltage readings according to the sense algorithm, and programming the waveforms to be played upon user input detection.

1.4 I2C Communication

The BOS0614 digital interface is an I2C/I3C interface compliant with MIPI® Alliance Specification for I3C^{SM}, version 1.0. Please consult the product datasheet for details on the I3C communication specifics. The current document will make use of the I2C functionality, but everything is also applicable to I3C communication.

The aim of this section is to provide an overview of the most relevant instructions used to configure the BOS0614. This information is also detailed in the product datasheet. It is explained here in simplified form for convenience.

1.4.1 Register and Memory Structure

All data communication with the BOS0614 is sent to and received from the main register map. Writing data to the Main Register block is done by providing the register address followed by the data content to be written into that register. Reading data from the BOS0614 requires first to specify which register address the data must be read from (set in the READ.BC field). Then, all read requests will return the content of this register.

BOS0614 also contains the WaveForm Synthesizer (WFS) command interpreter which manages the RAM Synthesis mode and RAM Playback mode configurations and also manages access to the RAM Memory (as shown in the figure below). The WFS command interpreter is accessed though the REFERENCE register (address 0x0) when the CONFIG.RAM mode is set to RAM Synthesis (0x3) or RAM Playback (0x2).

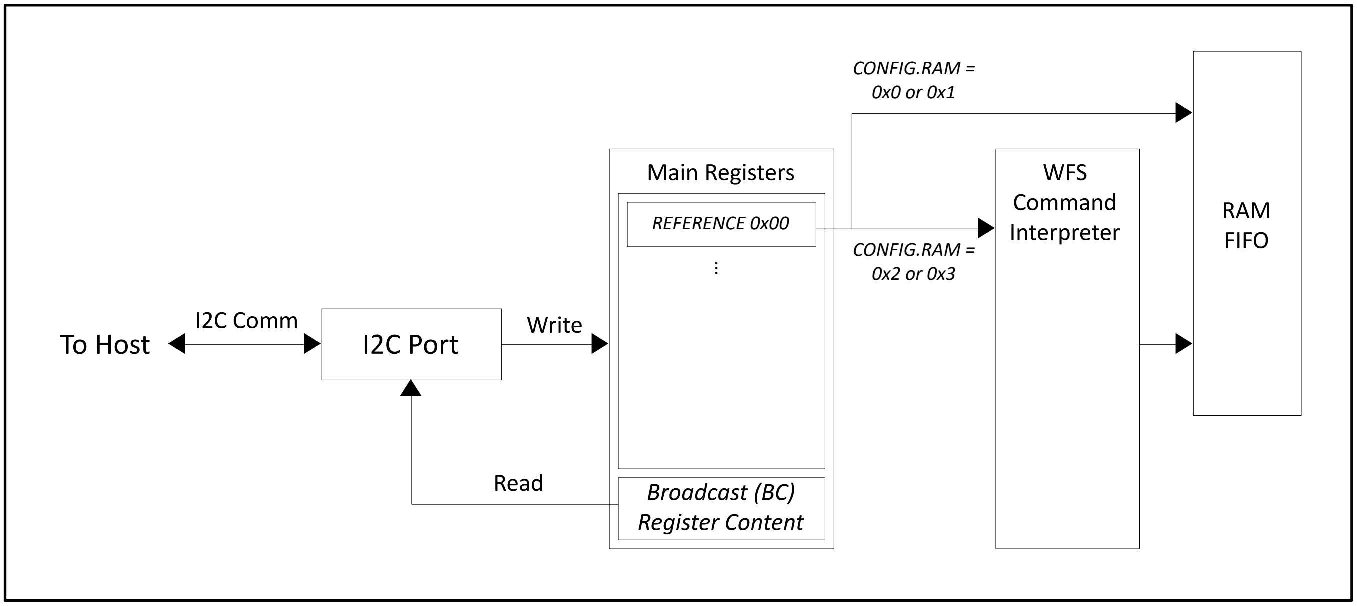


Figure 3 BOS0614 data structure

1.4.2 Writing to the Main Register Map

To write to the main register map, the following I2C transaction should be followed:

1. Send the 7-bit BOS0614 device address (0x2C) and reset the R/W bit (0).
2. Send the 8-bit register address to write to.
3. Send the 16-bit register content in the next two bytes (MSB first).

When writing to the FIFO, it is possible to repeat step 3 as part of the same transaction to send as many samples as needed.

One must be cautious when writing to a register so as to not change the parameter values that should stay the same within that same register. For example, the CONFIG register contains many bits to control the operation of the BOS0614. Therefore the other bits in the register should be re-written at their previous values to avoid unexpected behavior. It is recommended to either read the content of a register prior to changing its content, or to keep track of register values to change only the desired bits.

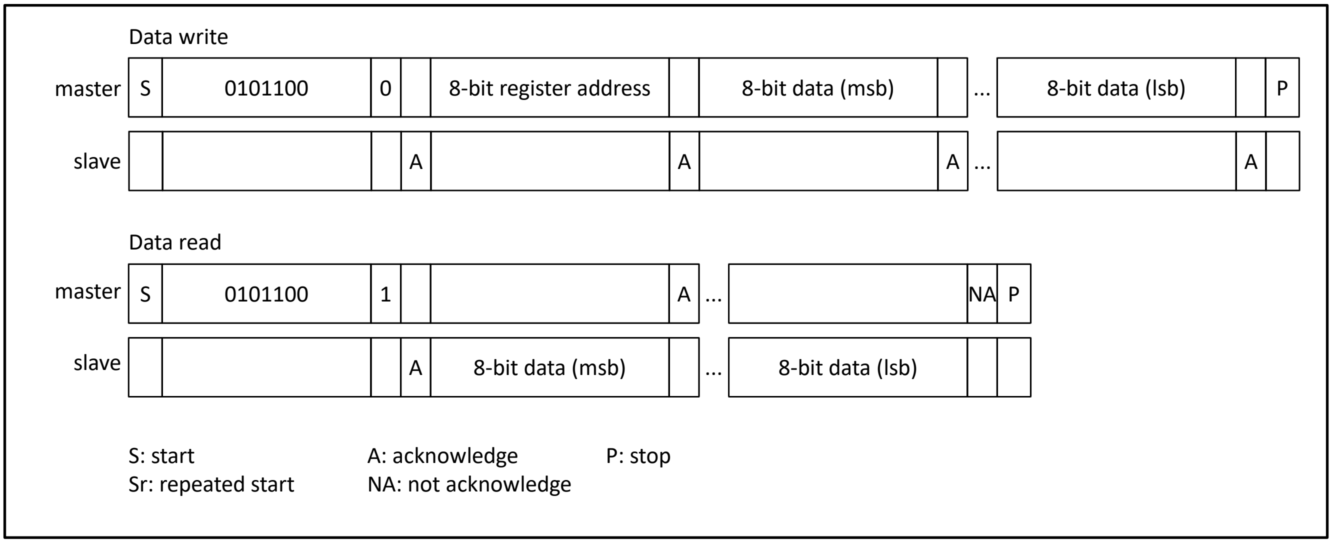


Figure 4: All possible data-transfer sequences with I²C static addressing

1.4.3 Reading from the Main Register Map

Reading a register is done in two steps:

a. Write the register address to read into the READ.BC parameter (BC parameter in the READ register) so the content of the desired register is pushed to the I2C interface upon any read transaction (see the write sequence in the previous section).
b. Read the register by following this transaction:
   1. Send the 7-bit BOS0614 device address (0x2C) and set the R/W bit (1).
   2. Read two bytes of content to get the 16-bit value (MSB is received first).

It is possible to repeat step (b) if multiple values need to be read successively.

The first step (a) need not be repeated for every successive read of the same register. Once READ.BC is set, it will keep its value until changed. Therefore, the content of the register set in READ.BC will be read on every read transaction until READ.BC is changed.

1.4.4 Writing to the WFS Command Interpreter and RAM Memory

The waveform synthesizer (WFS) command interpreter contains the RAM Synthesis and RAM Playback configuration parameters. The WFS command interpreter is also the conduit to manage the RAM Memory content.

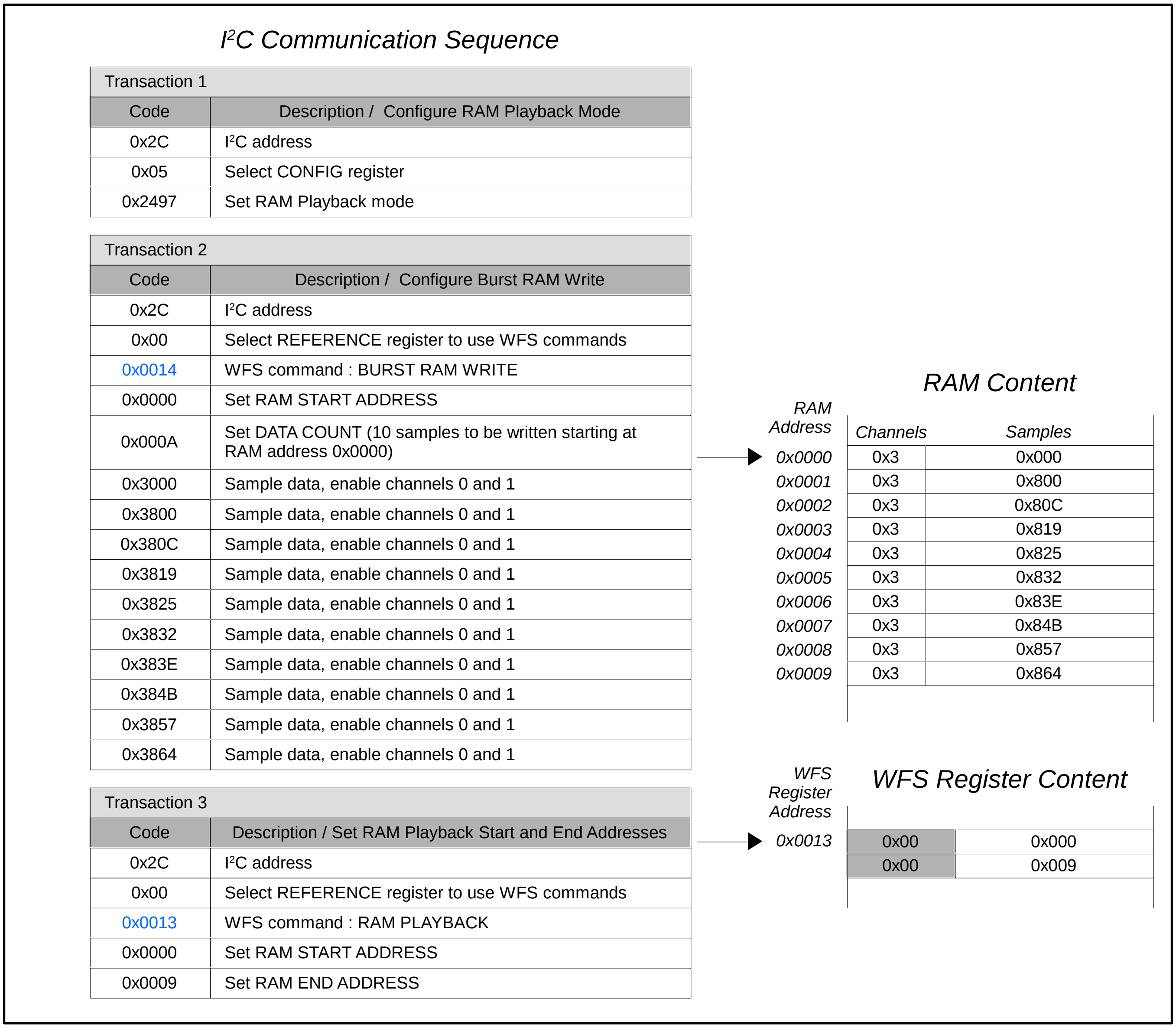
To write to the WFS command interpreter, one must write into the REFERENCE register of the main register map, after setting the RAM mode to either RAM Synthesis or RAM Playback. Following the REFERENCE register (0x00), the next 16-bit word (two bytes) is the WFS command to execute, and the following words are the payload for this command. Note the payload length is different for various commands.

Here is a typical sequence (an example is given in the figure below):

1. I2C transaction: Set CONFIG.RAM to either RAM Synthesis or RAM Playback with an I2C write operation.
   a. I2C address (0x2C) and R/W bit (0).
   b. CONFIG register address (0x05).
   c. CONFIG register content (2 bytes) with RAM set to either 0x3 or 0x2.
2. I2C transaction: access to WFS command interpreter.
   a. I2C address (0x2C) and R/W bit (0).
   b. REFERENCE register address (0x00) to access the WFS command interpreter.
   c. WFS command as16-bit word (2 bytes).
   d. Command Payload as N times 16-bit words depending on the command.

Many WFS commands can be sent with their payloads as part of the same I2C transaction without the need to re-enter the WFS command interpreter each time. The transaction would look like this:

1. I2C address (0x2C) and R/W bit (0).
2. REFERENCE register address (0x00) to access the WFS command interpreter.
3. WFS command 1.
4. Payload for command 1.
5. WFS command 2.
6. Payload for command 2.
7. WFS command 3.
8. Payload for command 3.
9. …



I^{2}C Communication Sequence

Transaction 1	
Code	Description / Configure RAM Playback Mode
0x2C	I^{2}C address
0x05	Select CONFIG register
0x2497	Set RAM Playback mode

Transaction 2	
Code	Description / Configure Burst RAM Write
0x2C	I^{2}C address
0x00	Select REFERENCE register to use WFS commands
0x0014	WFS command : BURST RAM WRITE
0x0000	Set RAM START ADDRESS
0x000A	Set DATA COUNT (10 samples to be written starting at RAM address 0x0000)
0x3000	Sample data, enable channels 0 and 1
0x3800	Sample data, enable channels 0 and 1
0x380C	Sample data, enable channels 0 and 1
0x3819	Sample data, enable channels 0 and 1
0x3825	Sample data, enable channels 0 and 1
0x3832	Sample data, enable channels 0 and 1
0x383E	Sample data, enable channels 0 and 1
0x384B	Sample data, enable channels 0 and 1
0x3857	Sample data, enable channels 0 and 1
0x3864	Sample data, enable channels 0 and 1

Transaction 3	
Code	Description / Set RAM Playback Start and End Addresses
0x2C	I^{2}C address
0x00	Select REFERENCE register to use WFS commands
0x0013	WFS command : RAM PLAYBACK
0x0000	Set RAM START ADDRESS
0x0009	Set RAM END ADDRESS

RAM Content

RAM Address	*Channels*	*Samples*
0x0000	0x3	0x000
0x0001	0x3	0x800
0x0002	0x3	0x80C
0x0003	0x3	0x819
0x0004	0x3	0x825
0x0005	0x3	0x832
0x0006	0x3	0x83E
0x0007	0x3	0x84B
0x0008	0x3	0x857
0x0009	0x3	0x864

WFS Register Content

WFS Register Address		
0x0013	0x00	0x000
	0x00	0x009

Figure 5: I2C sequencing RAM playback example

2 BOS0614 Sensing

This section describes how to perform sensing operations using the BOS0614. It covers the BOS0614 principle of operation, various sensing modes, and explains the various features useful for sensing. Specific register configurations and WFS commands to use in each situation are also given.

2.1 Principle of Operation

When the piezo actuator is deformed a voltage is generated on its terminals. The BOS0614 measures this with its internal ADC and makes the result available in the main register so it can be read by the application processor. Registers SENSEDATA0 (0x18) to SENSEDATA3 (0x1B) contain the voltage read on each channel. SENSEDATAx are updated continuously with the latest voltage values, allowing polling of these registers to get the voltage change over time.

BOS0614 embeds sensing detection mechanisms which can be configured to detect when the voltage has reached a certain level or slope. When detection is successful, the virtual button state associated with that channel will change between pressed and released states. The state change can also be triggered to the system through the GPIO0 to GPIO3 output pins. The sensing detection can be configured independently for each channel, or all channels may share the same configuration.

BOS0614 supports automatic haptic feedback to be played upon a successful sensing detection. The waveform must be programmed using the RAM Synthesis mode and a different waveform may be played when the piezo is pressed and released. This enables the emulation of a button running autonomously without any intervention from the application processor. Following a successful detection on a channel, the feedback waveform can be played on that channel while sensing on the other channels is still functioning. If a successful detection occurs on channel 3 for example while channel 1 is still playing its feedback waveform, the haptic feedback for channel 3 is queued and played when the haptic feedback has completed on channel 1.

The GPIOx outputs can be configured to provide various types of information to the application processor, namely the button state. By configuring these as open-drain outputs (GPIO.GPIOx = 0x0 and OD = 0x0), the GPIOx pins can directly replace mechanical tactile switches in the system (see Figure 2). In such a case, the piezo actuators become physical buttons to the system.

When capturing on an oscilloscope the piezo voltage while BOS0614 sensing is active, one will note a sawtooth-shaped signal instead of a voltage proportional the force applied on the piezo (see figure below). The BOS0614 uses a patented algorithm to cumulate the small voltage changes, short the piezo periodically to bring that voltage down to zero, and determine the effective piezo signal. This allows greater detection sensitivity and accuracy of the voltage measured. It also reduces noise artifacts when the sensing is successful as the voltage is already close to zero when the haptic feedback starts to play (large sharp voltage transitions would otherwise induce audible noise).

When analyzing the sensed voltage on the oscilloscope, one should pay attention to the direction and slew of the sawtooth to determine if the voltage is rising or falling to determine is the actuator is being pressed or released, and its rate of change to determine how fast the change is occurring. SENSEDATAx registers report the accumulated voltage value. Also, please note the oscilloscope probes may influence measurements, especially with low-capacitance actuators.

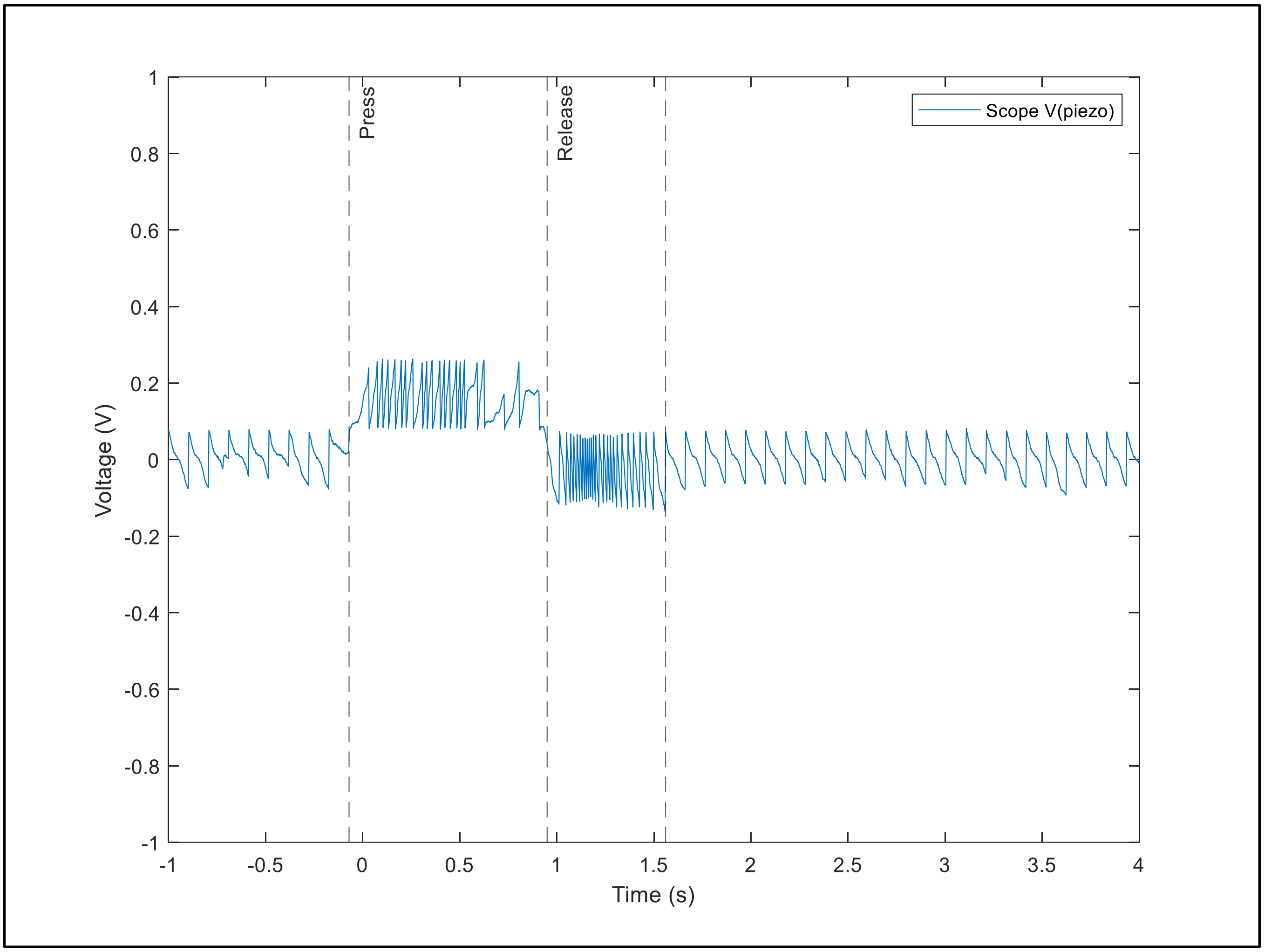


Figure 6: Sawtooth behavior of piezo voltage being pressed and released while sensing with BOS0614

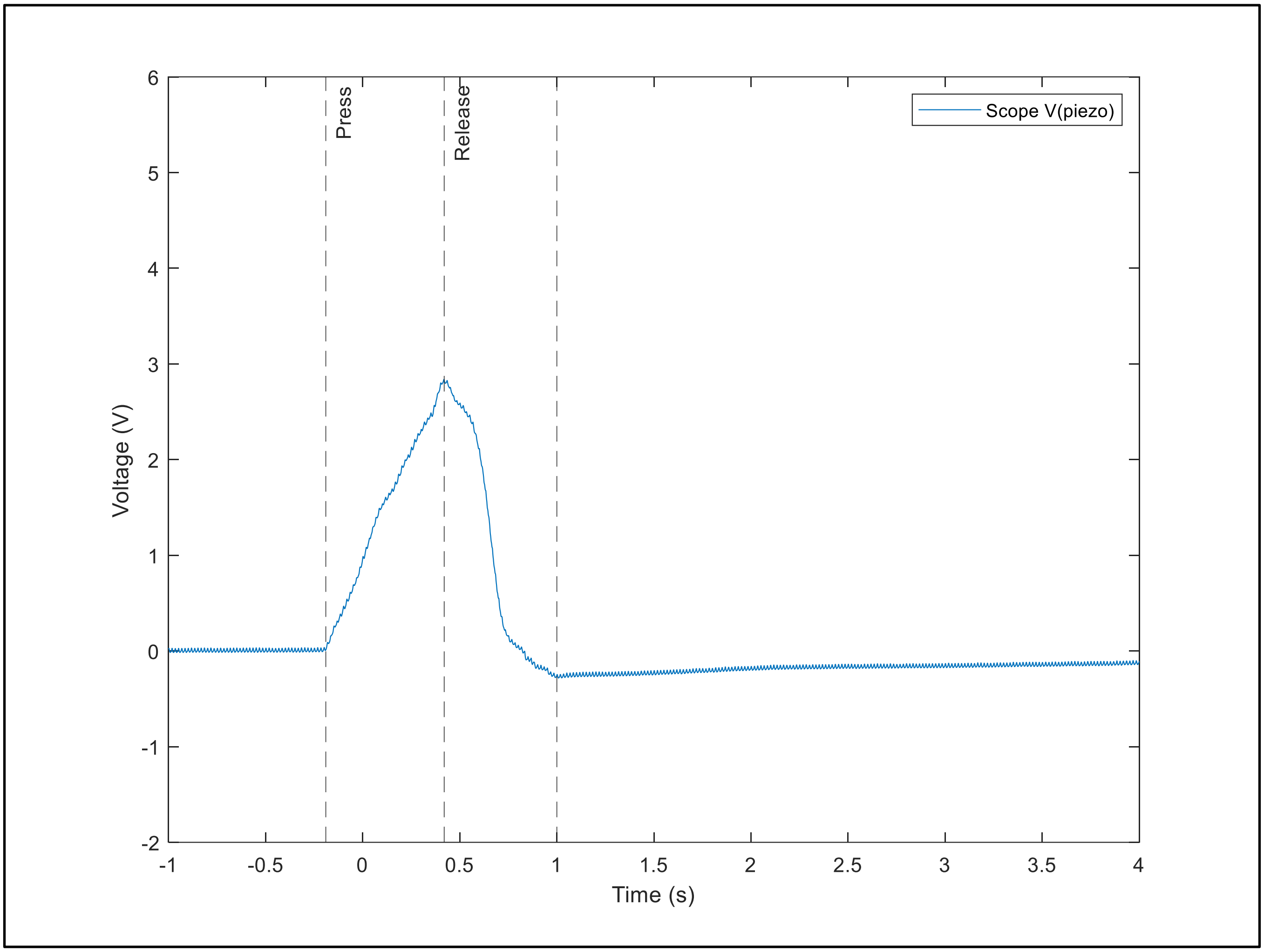


Figure 7: Voltage behavior of an open-circuit piezo voltage while being pressed and released

2.2 Sensing Use Cases

The BOS0614 applications will either use the cases given in this section or be derived from them. Each Use case is described in length in the following sections. A summary table is provided at the end of this section.

2.2.1 Active Sensing

In this use case, a sensing channel is sampled periodically by the application processor at a rate between 100 Hz and 10 kHz. The voltage profile is processed to determine if the actuator has been pressed or released. It can also be processed for advanced gesture recognition (tap, slide, etc.) or complex haptic button behaviors (e.g. multi-level buttons).

This method is simple but requires more system power from the application processor due to increased computation. Once detection occurs, the haptic feedback must be triggered via an I2C command, consuming communication time. Also, this case cannot be used in conjunction with the ZPS feature of the BOS0614 (see below).

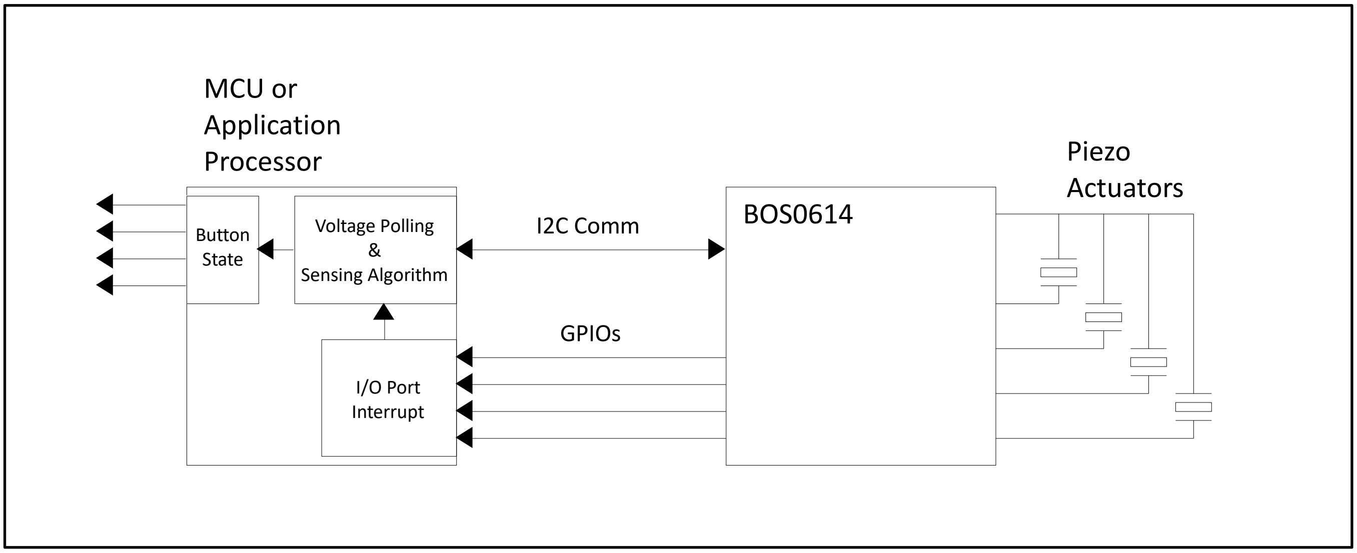


Figure 8: Active sensing diagram

2.2.2 Autonomous Sensing

This case uses the internal capabilities of the BOS0614 to autonomously manage a haptic button. In this case, the BOS0614 will automatically detect if the button is pressed and released, and will automatically trigger the programmed haptic waveform. A different haptic waveform can be programmed for each event. The waveform must be programmed using the RAM Synthesis mode.

This use case minimizes the communication with the application processor, which only needs to program the button behavior and then let go. The application can then either use the GPIO output as information monitoring or poll the BOS0614 as needed.

Note that some channels may be configured for autonomous sensing while others are used in active sensing.

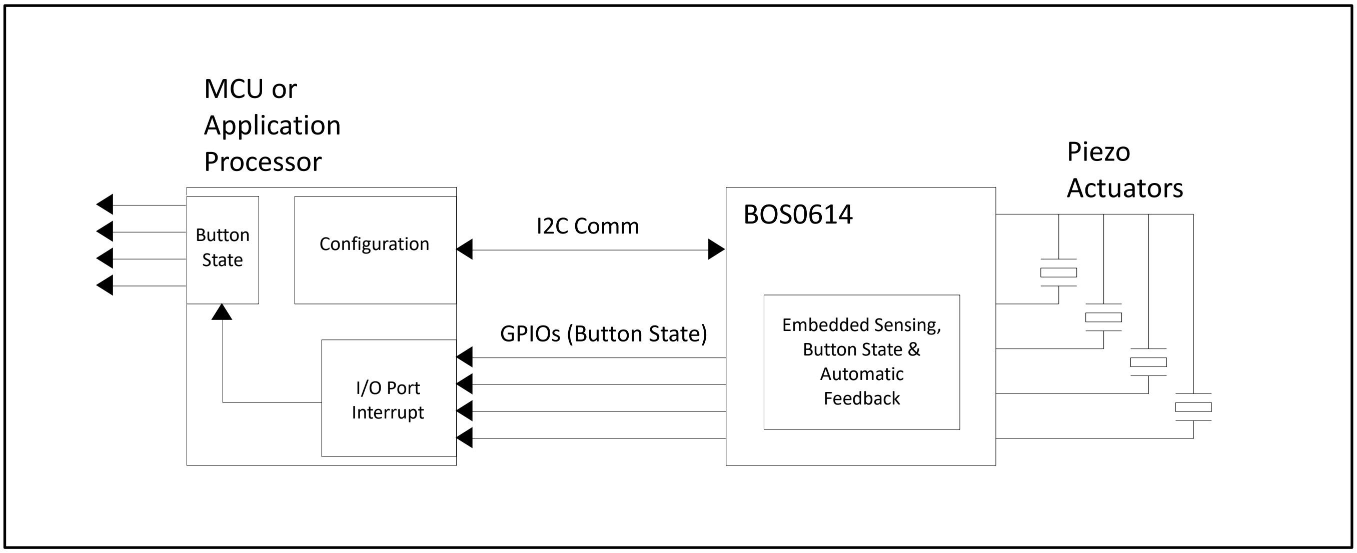


Figure 9: Autonomous sensing diagram

2.2.3 Hybrid Sensing

This use case combines the previous two. Autonomous sensing is first configured to automatically detect some sensing event. Upon detection the BOS0614 issues an interrupt on the GPIO output to tell the application processor to take over and perform active sensing.

This prevents computation when nothing is happening on the actuator, while still allowing the application processor to be used for sensing.

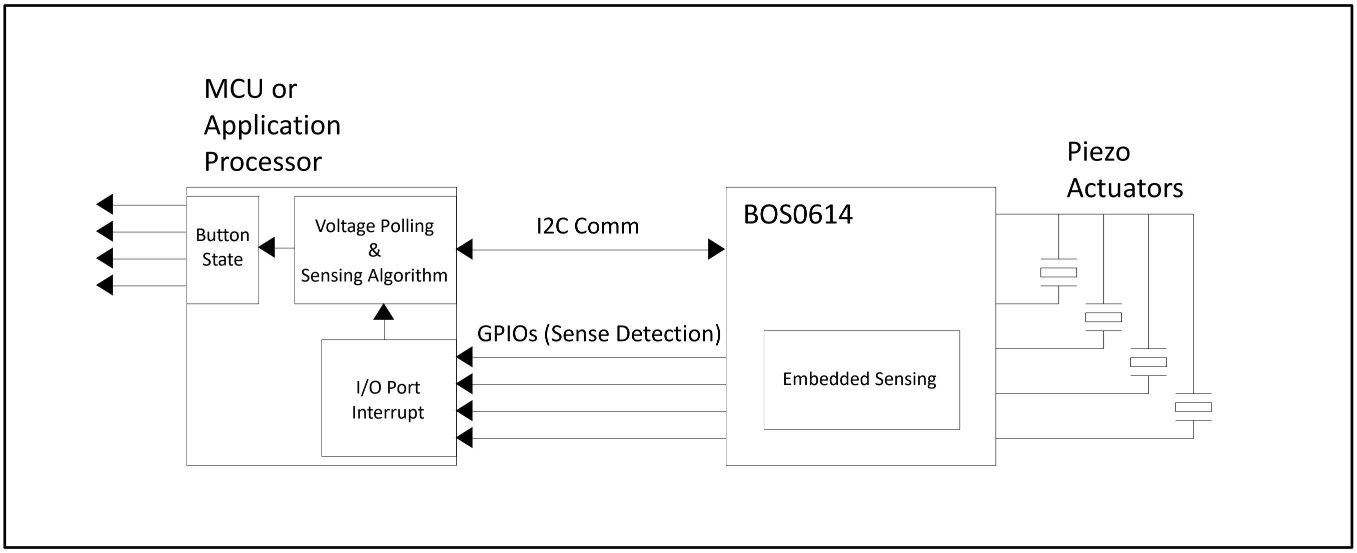


Figure 10: Hybrid sensing diagram

2.2.4 ZPS Sensing

This use case makes use of the BOS0614 ultra low power mode. When using ZPS sensing, all channels are required to be in the same mode since it involves setting the BOS0614 to SLEEP.

When in ZPS sensing, a force applied on the actuator will wake up the BOS0614. If it is configured for autonomous sensing, the sensing conditions are then evaluated. If the press detection is not successful within some amount of time, the BOS0614 will return to SLEEP. This is depicted in the figure below. If the press detection is successful, the state changes to PRESSED and the programmed feedback is played. Then the IC waits for a release to occur and plays the programmed waveform. After some time following the release feedback, the BOS0614 will automatically return to SLEEP.

It is possible to use the ZPS feature without programming the BOS0614 in autonomous sensing. One must make sure the ZPS wake-up is captured by the application processor and that appropriate steps are taken to perform sensing and return to SLEEP when needed.

ZPS sensing is more sensitive to weak or slow voltage signals than autonomous sensing. Therefore, it could also be used in hybrid sensing to launch the application processor active sensing when there is activity on the piezo actuator. This would allow minimizing power when nothing occurs on the piezo actuator while allowing for complex processing to occur when needed.

Note however that is not possible to monitor the BOS0614 while using ZPS sensing since any I2C communication will the BOS0614 will get it out of SLEEP and stop the ZPS sensing.

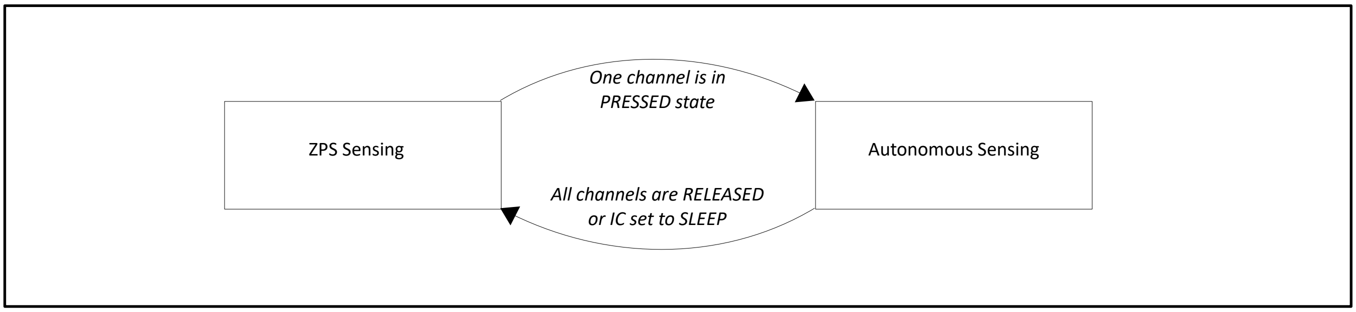


Figure 11: ZPS sensing transition to and from autonomous sensing

2.2.5 Embedded Sensing with Interrupt-Based Feedback

Embedded sensing detection (or autonomous sensing) can only trigger feedback waveforms programmed in RAM Synthesis mode.

To play a waveform using other playback modes (RAM Playback, FIFO mode or Direct mode), it must be triggered by the application processor at the appropriate time. For accurately timed feedback, the BOS0614 can output an interrupt signal on the GPIO pin when a sensing detection occurs. This enables the application processor to program, arm, and trigger that feedback.

When the press feedback has played the BOS0614 switches back to its sense function to detect a release event. The same principle applies for playing the release event feedback.

2.2.6 Summary

Here is a summary of the presented use cases.

Table 1 Sensing use case comparison

USE CASE	PRINCIPLE	ADVANTAGES	CONSIDERATIONS
Active Sensing	MCU polls the voltage on the channels over I2C and processes the data according to the detection algorithm.	Enables more complex algorithms and button behaviors. Enables advanced gesture recognition.	Requires more computing power. Haptic trigger via I2C commands. Incompatible with ZPS sensing.
Autonomous Sensing	BOS0614 sensing using embedded features, haptic is automatically issued upon detection.	Very low computation on MCU. Automatic trigger of haptic. Normal button signal on GPIO. Compatible with ZPS feature.	Incompatible with advanced gesture recognition. Haptic limited to RAM Synthesis.
Hybrid Sensing	BOS0614 sensing using embedded features; when some sensing conditions are met, the MCU is taking over in active sensing.	No computation on MCU while nothing happens on the actuator.	Haptic trigger via I2C commands. Incompatible with ZPS sensing.
ZPS Sensing	BOS0614 in SLEEP mode, wakes up when piezo is pressed, and automatically returns to sleep when the button is released.	Lowest power mode. Automatic trigger of haptic. Power button emulation.	Impossible to monitor IC while in SLEEP.
Embedded Sensing with Interrupt-Based Feedback	BOS0614 sensing using embedded features, then MCU triggers the haptic feedback.	Low sensing computation on MCU. Enables any haptic shape using modes other than RAM synthesis.	Haptic trigger via I2C commands.

2.3 Sensing Detection Mechanisms

As force is increased or decrease on the piezo actuator the voltage between its terminals will increase or decrease respectively. The amount and speed of the force applied can be measured to determine the intent of the user is any given case. This section explains the various detection mechanisms used by the BOS0614 to determine when the haptic button is pressed or released. More complex algorithms can be used by computing the sensed voltage in the application processor. Such algorithms allow to expand on the BOS0614 capabilities such as enabling advanced gesture recognition.

2.3.1 Threshold Method

Applying pressure increases the voltage on the piezo actuator terminals. When the voltage reaches the set threshold (Vth, see figure below), it indicates the user has reached a certain amount of force or pressure on the actuator. Once this threshold is reached, playing haptic feedback is recommended to acknowledge to the user the input was registered, in button applications for example. The feedback may change depending on the context.

The detection could occur when crossing the threshold upward or downward. A different haptic action could be used in each case. A debounce time may also be used to prevent detection of false event due to noise from the system or from accidental bumps to the device frame.

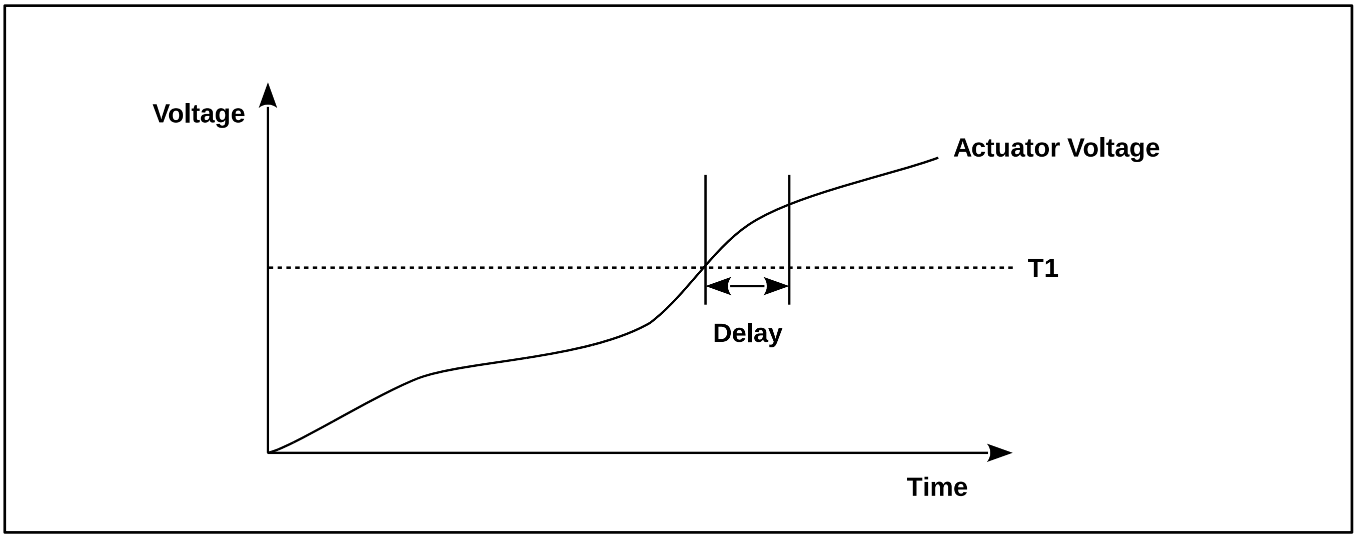


Figure 12 Threshold detection

2.3.2 Slope Method

In some cases, it might be more convenient to detect a rate of voltage change rather than only its level. For example, when trying to detect quickly repetitive press events a slope detection will be able to distinguish the events independently of the average pressure applied. In such an application, the repeated presses might be too quick and not reach a single voltage level every time. It also allows to detect pressure applied quickly and for a short amount of time, even though it is low in amplitude.

Calculating a moving average window across the measured voltage signal enables better resolution on the voltage change and increases the effective number of bits per value.

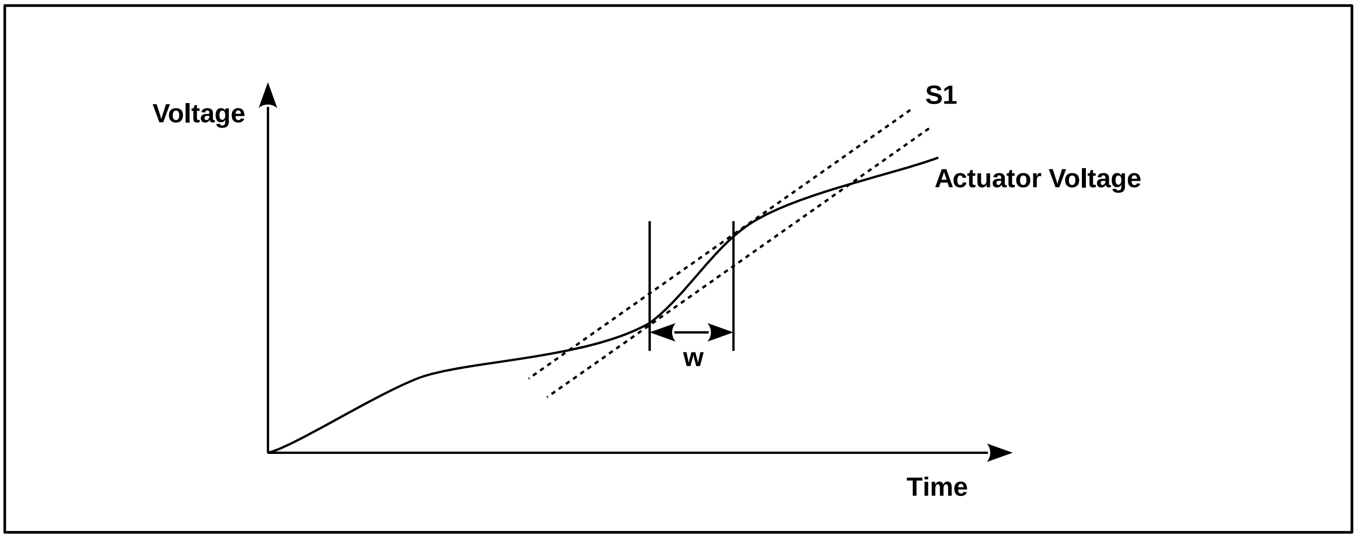


Figure 13 Slope detection

2.3.3 Combination of Both Mechanisms

More complex detection algorithms can be built from the above methods. This allows for more complex behaviors to be detected, and to increase detection robustness.

For example, slope detection might be combined with a minimum threshold to avoid false positives at low forces.

Another example is depicted in the following figure. The press event can be registered at lower amplitude (Vth1) if the detected slope (Slope Threshold) is high enough (Vth1 AND Slope Threshold). Past a given voltage threshold (Vth2) the press is registered regardless of the slope detected to avoid the user applying too much pressure on the device and guarantee detection.

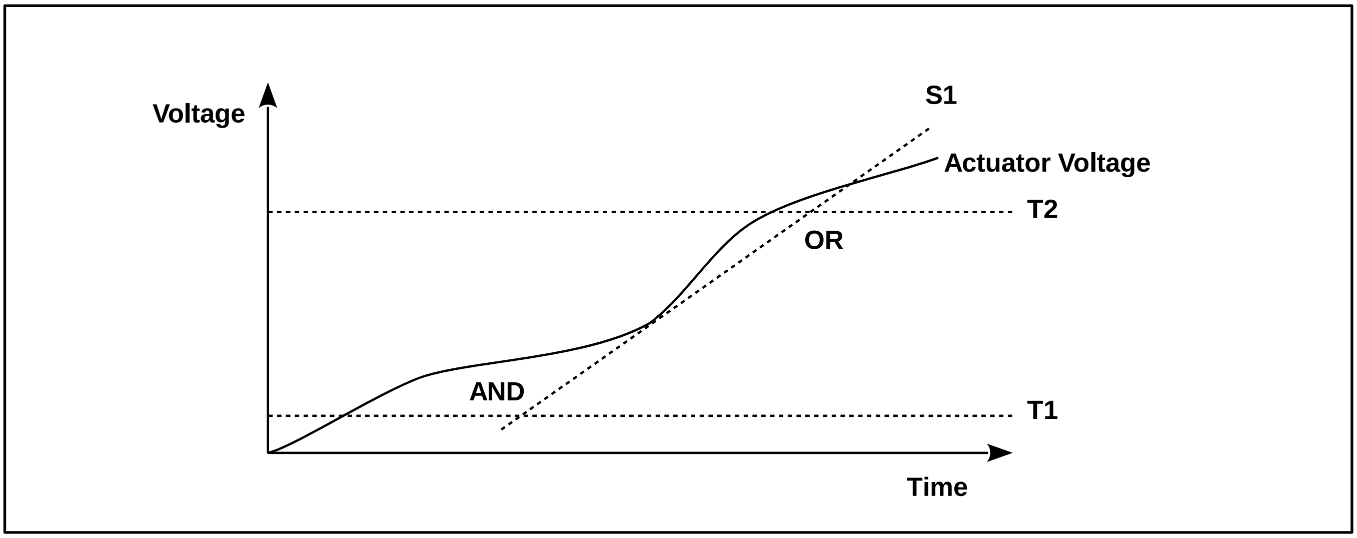


Figure 14 Combining threshold and slope detections

2.3.4 BOS0614 Embedded Sensing Combinations

BOS0614 features threshold and slope detection mechanisms. Each channel can be configured independently with a combination of two (2) threshold (T1, T2) and the two (2) slope mechanisms (S1, S2). The mechanisms are configured by setting the associated bits (Tx, Sx) in the corresponding channel SENSEx register (SENSE0 to SENSE3).

T1 and S1 normally applies for press detection and T2 and S2 for release detection. However, disabling both the threshold and slope bits for the same button state transition (press, release) results in a more complex combination also using the other state threshold mechanism. For example, if T1 = 0 and S1 = 0, then the press detection will use both threshold mechanisms and the first slope mechanism as criteria for detection (see table below). If not careful, the same mechanism could be used for both the press and the release detection, leading to undesired results. If a press condition is reached while the release condition is also evaluated true, then the press detection will not occur. The same is true if release condition is reached while the press condition is also true. Detection mechanisms should be configured to avoid these situations.

Table 2: Press event triggering conditions configuration, where x is the channel number

Bit SENSEx.T1 Value	Bit SENSEx.S1 Value	Condition to trigger a press event
0x0	0x0	(S1x & T1x) | T2x = 1
0x1	0x0	T1x = 1
0x0	0x1	S1x = 1
0x1	0x1	S1x & T1x = 1

Table 3: Release event triggering conditions configuration, where x is the channel number

Bit SENSEx.T2 Value	Bit SENSEx.S2 Value	Condition to trigger a release event
0x0	0x0	(S2x & T2x) | T1x = 1
0x1	0x0	T2x = 1
0x0	0x1	S2x = 1
0x1	0x1	S2x & T2x = 1

2.3.5 Double Threshold Method using Active Sensing

Using active sensing, more elaborate detection mechanisms and algorithms can be used. The following method is a combination of two thresholds. Depending on the amount of pressure the user applies, the first or second threshold might be reached, and a different action may be taken depending on the level reached.

To ensure reliable detection, delays should be used. The delay associated with the lower threshold should be longer than the one with the higher threshold.

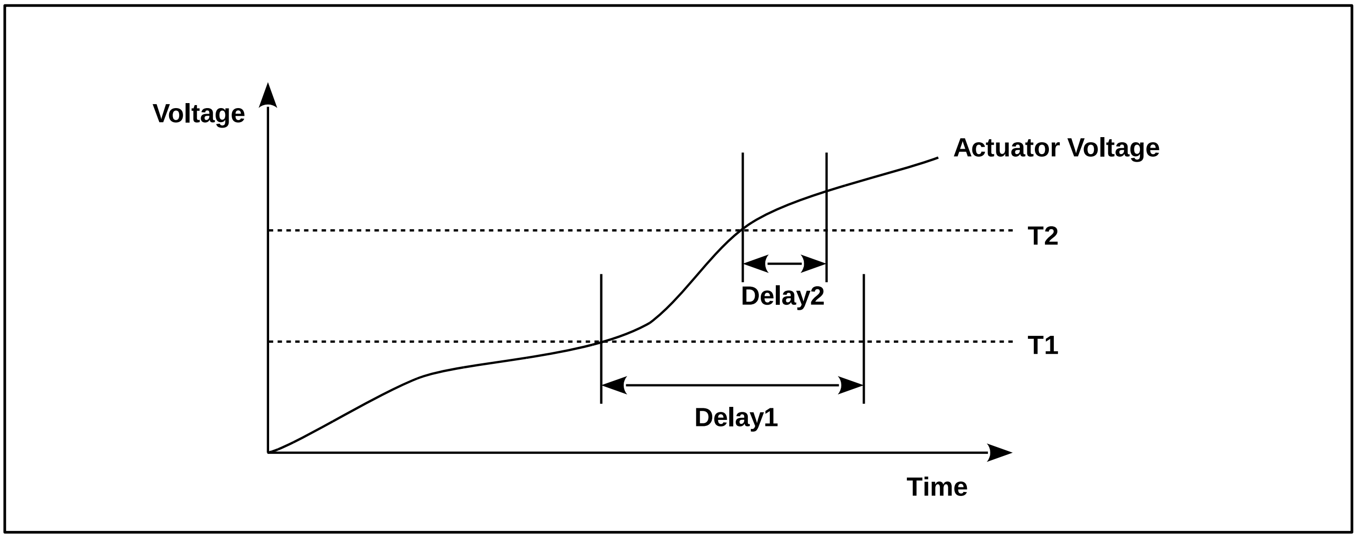


Figure 15 Double threshold detection

2.3.6 Bidirectional Method using Active Sensing

BOS0614 can detect voltage rising or falling in both the positive and negative ranges. This enables bidirectional sensing detection enabling multi-level buttons, or haptics in gradually increase force scenarios. With proper processing and state machines, complex user interface controls can be implemented.

For example, a two-level button can issue a feedback different at each level, and the system action at each level can be different. Moreover, the sequence of levels is arbitrary. The user may reach the first level, then either reach the second level or release the button entirely. This enables buttons with behaviors similar to the focus and capture button on most DLSR cameras.

3 Configuring the BOS0614 for Sensing

3.1 Initial Configuration

At first powerup or after a software reset (CONFIG.RST = 1), the BOS0614 will have the following default state:

- all four (4) channels have sensing enabled (SENSECONFIG.CHx = 1);
- all four channels share the same sensing configuration (SENSECONFIG.SAME = 1);
- press is detected on a threshold (T1 = 1, S1 = 0);
- release is detected on a slope (T2 = 0, S2 = 1);
- GPIOx outputs are set as open drain (CONFIG.OD = 0);
- GPIOx outputs are set to report button state (GPIO.GPIOx = 0);

If one or more of these parameters need to be changed to prevent unwanted detection, it should be set up soon after the powerup or the reset. For example, setting all SENSECONFIG.CHx bits to 0 will disable sensing on all channels. SENSECONFIG.SAME = 0 will avoid unwanted detection on other channels once the first channel (0) is configured.

3.2 Enabling Sensing and Reading the Piezo Voltage

For sensing to operate properly, it must first be calibrated. This only need to be done once. First enable the sensing on channel 0 (SENSECONFIG.CH0 = 1), then run the sensing calibration (SENSECONFIG.CAL = 1). The calibration time is approximately the time set in CONFIG.SHORT. It is possible to read SENSECONFIG.CAL to know when the calibration has finished since this bit self-clears.

The BOS0614 automatically senses the voltage on the actuator once a channel sensing is enabled (SENSECONFIG.CHx = 1). Reading the voltage only requires reading the content of the corresponding SENSEDATAx register. The data is returned as a signed 16-bit value with 220 µV LSB resolution. Even though the voltage capture on an oscilloscope returns a sawtooth shape, the content of SENSEDATAx is the reconstructed voltage on the actuator.

When doing active sensing or monitoring, the application processor or MCU will poll SENSEDATAx and determine the detection condition based on its value and variation. While using the embedded sensing detection, SENSEDATAx can still be read and used by the application processor for monitoring.

3.3 Configuring the BOS0614 for Autonomous Sensing

Using the embedded sensing feature is required for autonomous sensing. Follow the sequence below to configure all relevant parameters. Depending on the sensing type, not all parameters may be needed. See the product datasheet for specific values to use. The BOS0614 example codes also provide a good reference for parameters to set in each case. For example, press and release detection must be configured, and the RAM synthesis feedback waveforms must also be programmed.

1. Start with sensing disabled on all channels:
    a. Set SENSECONFIG.CHx = 0.
2. Program the waveforms for press and release in RAM synthesis mode. This playback mode is required for autonomous sensing, but it is optional otherwise. See other application notes or the product datasheet for specifics of programming a waveform in RAM synthesis.

3. Configure sensing for press and/or release detection of all channels used:
    a. Configure the channel sensing behavior (SENSEx):
        i. .AUTOP & .AUTOR: enable (1) the programmed waveform to start playing automatically when the press and release detection are successful.
        ii. .WVP & .WVR: indicates which programmed waveform to start playing automatically when the press and release detection are successful.
        iii. .Tx & .Sx: indicates which combination of detection mechanisms to use for detection. Be careful with the combination used (see section 2.3.4).
    b. Configure the press threshold detection parameters (SENSExP):
        i. .REP: sets the hold time the voltage must meet the threshold before the sensing is successful.
        ii. .AB: indicates if the voltage must be above or below the threshold. Normally a press is detected when the voltage is above the threshold and a release is detected when the voltage is below the threshold, unless the piezo actuator has an inverted behavior.
        iii. .THRESHOLD: sets the voltage threshold to reach. This is a signed 12-bit value with 1.66mV LSB resolution.
    c. Configure the release threshold detection parameters (SENSExR):
        i. Same as above.
    d. Configure the slope detection parameters (SENSExS):
        i. .ABS1 & .SLOPE1 are associated with S1.
        ii. .ABS2 & .SLOPE2 are associated with S2.
        iii. .ABSx: indicates if the slope must be above or below the slope threshold.
        iv. .SLOPEx: sets the slope threshold to reach as signed 7-bit with 2.2mV/ms LSB resolution.
    e. Set the stabilization time (see section 4.1):
        i. TC.TCP: sets the press waveform stabilization time as 5-bit unsigned with 3.2ms LSB resolution.
        ii. TC.TCR: sets the release waveform stabilization time as 5-bit unsigned with 3.2ms LSB resolution.
        iii. TC.PC: enables (1) automatic stabilization at the end of the waveform using TCP and TCR times.
    f. SENSECONFIG.SAME: set (1) if all channels should all use the configuration of channel 0 or reset (0) if all channels are configured individually.
4. Enable sensing for each channel used (SENSECONFIG.CHx). If using the same configuration for all channels, they still must be enabled individually.
5. Configure GPIO output pin behavior (GPIO.GPIOx). Configuring to output the button state (0x0) is very convenient in autonomous sensing because the GPIO will directly indicate the state of the haptic button at any moment without the need to read the state over I2C communication. Configuring the GPIOs as open drain (CONFIG.OD = 0x0), the GPIO pin can also physically replace mechanical switches.

3.4 Activating the ZPS Feature

The ZPS (Zero Power Sensing) feature allows a piezo press to wake up the BOS0614 automatically without I2C communication. This enables haptic button behavior at very low consumption (few µA). When a piezo on a channel with ZPS active is pressed, the BOS0614 wakes up to perform sensing detection of the press using the preprogrammed settings. If the press is not successfully detected, the BOS0614 will return to SLEEP. If the press is detected, the MCU or application processor needs to initiate communication with the BOS0614 to keep it awake for further operation, otherwise the chip will return to sleep automatically 100 ms after the button has been released. GPIO outputs can be used to generate interrupts to the MCU when the press is detected.

To enable the ZPS feature on a channel, that channel sensing must be enabled (SENSECONFIG.CHx = 1) before the chip is set to SLEEP.

The following registers will determine how the ZPS feature will behave:

a. SENSECONFIG.ZPS_SENS: sets the ZPS sensitivity (default is high).
b. SENSECONFIG.ZPS: sets if the ZPS threshold should be detected as a press event directly, of if the programmed sensing configuration must be met for the press event to be valid (default is the programmed configuration).

The figure below illustrate a detected press sequence using the ZPS feature. The blue curve shows the voltage as captured from an oscilloscope. The red curve shows the BOS0614 VREG pin. VREG reflects the state of the BOS0614 : it is down when the BOS0614 is in SLEEP, and it is up when the BOS0614 is awake.

The figure starts with BOS0614 in SLEEP with sensing preconfigured. As the user pressed on the actuator, voltage rises and triggers the ZPS internal module, which wakes up the BOS0614 to perform sensing as shown by the VREG curve. As force increases on the actuator (fast increasing sawtooths), the voltage detection threshold is reached, and the press haptic feedback is played. Then, the BOS0614 goes back to sensing and waits for haptic button (piezo actuator) to be released. Once the release is detected, the release haptic feedback is played. Then the BOS0614 returns to sensing for press detection. Since no other press event is detected within 100 ms, the BOS0614 then automatically returns to SLEEP.

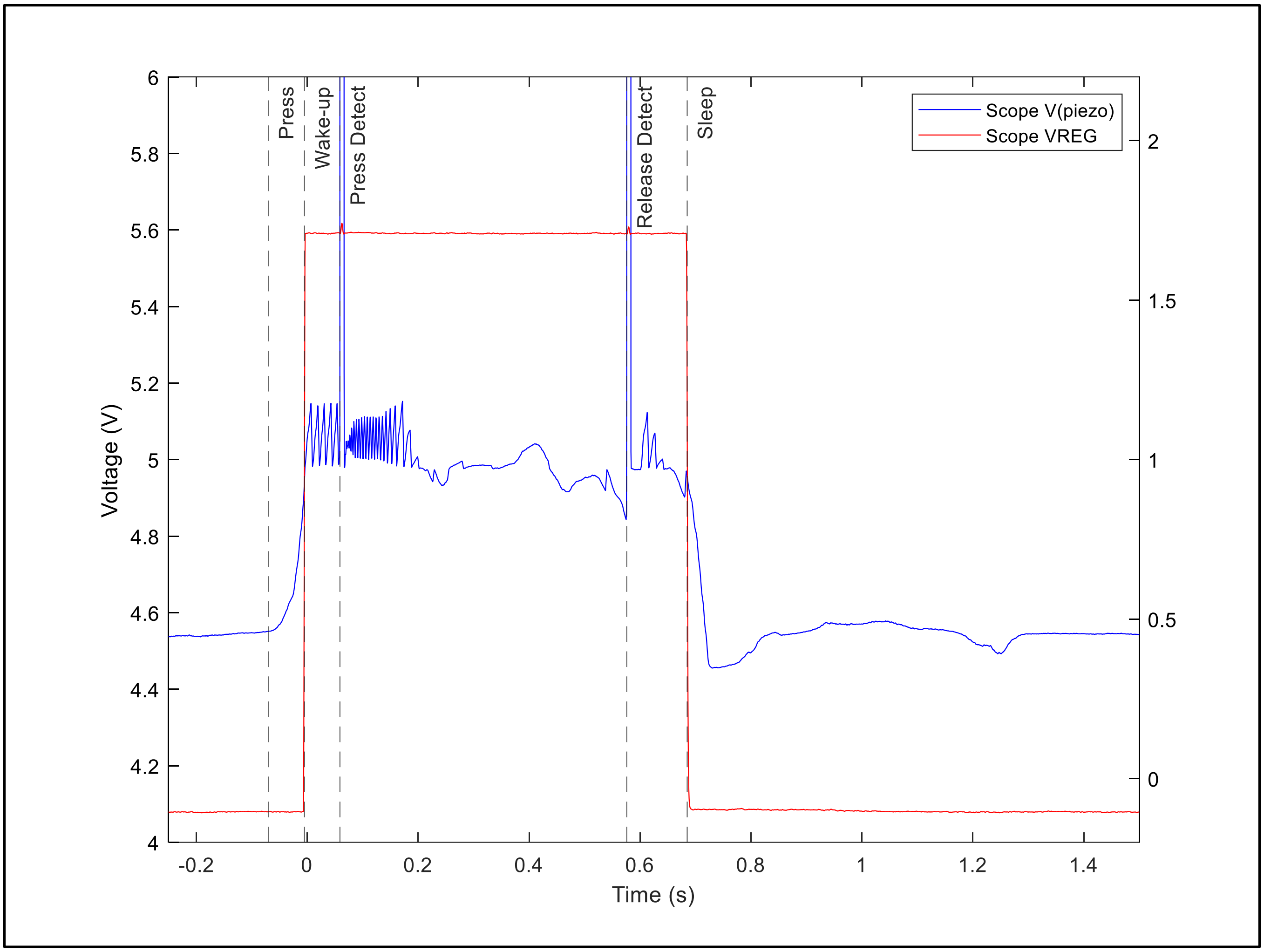


Figure 16 ZPS Sensing typical sequence

3.5 Configuring the Sensing to Trigger MCU Feedback

When using playback modes other than RAM Synthesis for the haptic feedback, the autonomous sensing cannot be used. An alternate method is to use the embedded sensing to detect the press and the release events and to use the GPIO output pin to tell the MCU the haptic must be played via I2C commands.

Here is a possible configuration:

a. Configure the channel sensing behavior (SENSEx):
   i. .AUTOP & .AUTOR = 0: disables auto play.
   ii. .Tx & .Sx: indicates which combination of detection mechanisms to use for detection. Be careful with the combination used (see section 2.3.4).
b. Configure the press threshold detection parameters (SENSExP):
   iii. .REP: sets the hold time the voltage must meet the threshold before the sensing is successful.
   iv. .AB: indicates if the voltage must be above or below the threshold. Normally a press is detected when the voltage is above the threshold and a release is detected when the voltage is below the threshold unless the piezo actuator has an inverted behavior.
   v. .THRESHOLD: sets the voltage threshold to reach. This is a signed 12-bit value with 1.66mV LSB resolution.
c. Configure the release threshold detection parameters (SENSExR):
   vi. Same as above.
d. Configure the slope detection parameters (SENSExS):
   vii. .ABS1 & .SLOPE1 are associated with S1.
   viii. .ABS2 & .SLOPE2 are associated with S2.
   ix. .ABSx: indicates if the slope must be above or below the slope threshold.
   x. .SLOPEx: sets the slope threshold to reach as signed 7-bit with 2.2mV/ms LSB resolution.
e. Set the stabilization time (see section 4.1):
   xi. TC.TCP: sets the press waveform stabilization time as 5-bit unsigned with 3.2ms LSB resolution.
   xii. TC.TCR: sets the release waveform stabilization time as 5-bit unsigned with 3.2ms LSB resolution.
   xiii. TC.PC: enables (1) automatic stabilization at the end of the waveform using TCP and TCR times. Note: only use this feature is using the RAM Playback or RAM Synthesis modes. Using FIFO or Direct play, set TC.PC = 0.
f. SENSECONFIG.SAME: set (1) if all channels should all use the configuration of channel 0 or reset (0) if all channels are configured individually.
g. Set GPIO.GPIOx = 0x1: this indicates to the MCU a sensing event has triggered on this channel. When this event is detected, the MCU will arm and trigger the waveform to play via I2C commands. See the product datasheet or other application notes on how to play waveforms using the other playback modes.

4 Other Considerations

The following topics should also be considered as they can impact sensing performance and user experience. They have been put in this section to keep the previous ones easier to read.

4.1 Piezo Creep

After a waveform finished playing, the actuator tends to continue to move and this can induce a voltage change (or creep) on the actuator. This voltage change can be high enough to affect the force required to trigger the next press, or in some cases to exceed the sensing threshold, thereby inducing a premature threshold detection and creating undesired effects.

For example, let’s consider using the actuator as a haptic button that is playing a pulse when the force press threshold is reached and another pulse when the force release threshold is reached. After the second pulse (release) is played, piezo creep could induce voltage to rise again while the user is still in the process of releasing the button. This voltage rise may trigger an unwanted pulse such that two pulses are felt when releasing the actuator. In certain cases, depending on the algorithm, the haptic button may also get stuck in the pressed state.

To prevent problems related to piezo creep, it is often necessary let the piezo settle within tolerable limits before reenabling the sensing. This settling time can be shortened by forcing the actuator to 0 V or by shorting the actuator terminals for some time (stabilization time). The stabilization time to use is dependent on the sensing algorithm used, the actuator size, its internal capacitance, and the peak voltage of the feedback waveform. A delay of a few milliseconds will suffice in most cases, however, the appropriate value should be determined at testing to prevent the sensing to automatically trigger haptic in all possible conditions.

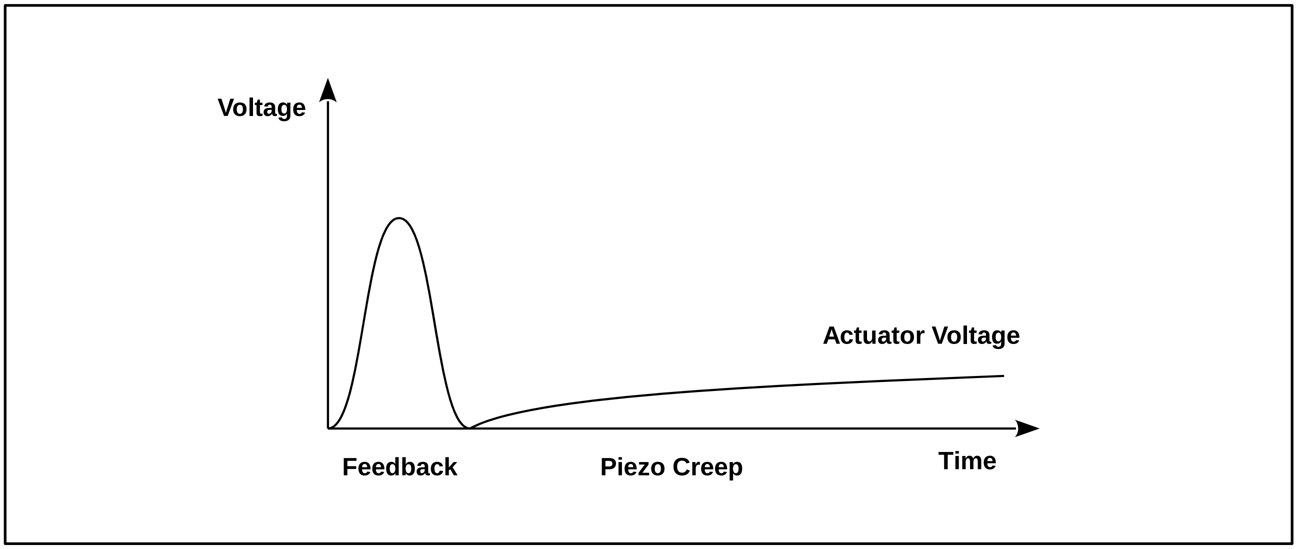


Figure 17 Piezo creep following feedback waveform

The BOS0614 provides parameters to perform this stabilization automatically. TC.TCP and TC.TCR set the time the actuator will be shorted once the waveform has finished playing following a press and release detection respectively. TC.PC enables this feature. Note it is possible to play a waveform on another channel while the first one is being stabilized.

Several strategies can be used to keep it at a practical value. For example, after a press haptic feedback, immediately before the release sensing phase, it is recommended to keep the stabilization delay short, and use the combination of both threshold and slope mechanisms for the release detection. The short stabilization delay will reduce some of the effect of the creep, yet still allow to quickly detect a voltage fall that would occur as the user removes the force on the actuator. Using this approach, the release is still reliably detected as a negative voltage slope, despite piezo creep.

For the release haptic feedback, reducing the stabilization delay will allow the piezo voltage to still rise to noticeable values. Then the press detection mechanism can be changed to a combination of threshold and slope to filter out detection from a piezo creep effect, while still detecting user-induced press input. One way to achieve this is to combine one positive threshold for press detection and one positive slope detection: the detection would occur if both the threshold has been reached and the voltage rate change is high enough. Since both a force press and the piezo creep may reach the threshold, the slope detection will only allow user-induced presses to be detected. A second threshold could be defined at a higher voltage to also trigger the haptic press if the user reached that force but slower.

Note that the BOS0614 sensing range is ±3.6 V. If the stabilization delay is too short, the creep voltage rise may exceed the voltage sensing range, in which case the information will be lost in the saturated sensed voltage value. When adjusting the sensing configuration and the creep stabilization delays, it is useful to probe the voltage on an oscilloscope and to read and plot SENSEDATAx values. A logic analyzer is also very useful to retrieve the SENSEDATAx values.

4.2 SENSEDATAx Drift Monitoring

SENSEDATAx is the voltage accumulated on the actuator as tracked by the BOS0614. When the sensing is activated for a channel, SENSEDATAx value is reset. As the voltage rises or falls, SENSEDATAx will follow and accumulate increments of voltage changes between the moments the actuator is shorted. When the sensing detection conditions are met, the haptic waveform is played and the SENSEDATAx value is automatically reset.

However, some of the internal variables used in the BOS0614 are prone to drift. If the actuator is pressed and released without reaching the detection conditions, some error may accumulate on SENSEDATAx, leaving an offset on SENSEDATAx even if no force is applied on the actuator. This will offset the force required to trigger the next press. As this drift increases it may become very hard to trigger the press.

Consequently, some routine needs to be implemented to make sure this drift doesn`t prevent the button to trigger as intended. One method is given in the BOS0614 example codes that can be downloaded from the Boréas website. In a nutshell, the algorithm checks for an offset on SENSEDATAx while no force is applied on the actuator and resets the reading to 0. If SENSEDATAx is stable enough at an offset significant enough, it indicates a drift that should be corrected.

Drift monitoring is only needed when applied to active, autonomous and hybrid sensing. It is not needed using ZPS sensing.

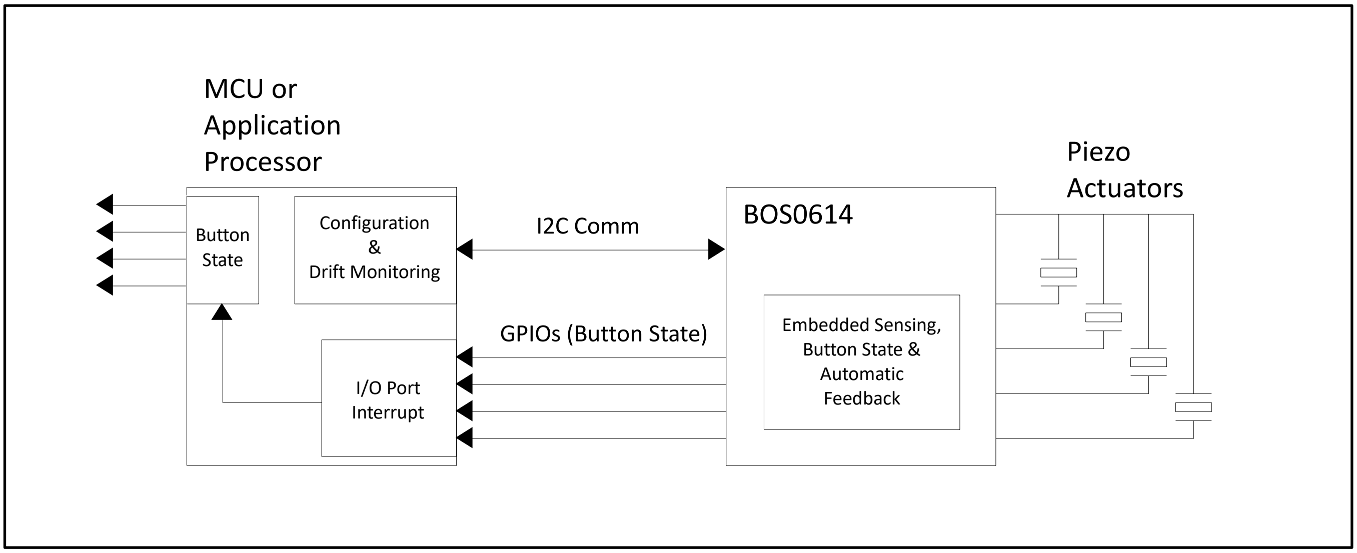


Figure 18: Autonomous sensing with drift monitoring diagram

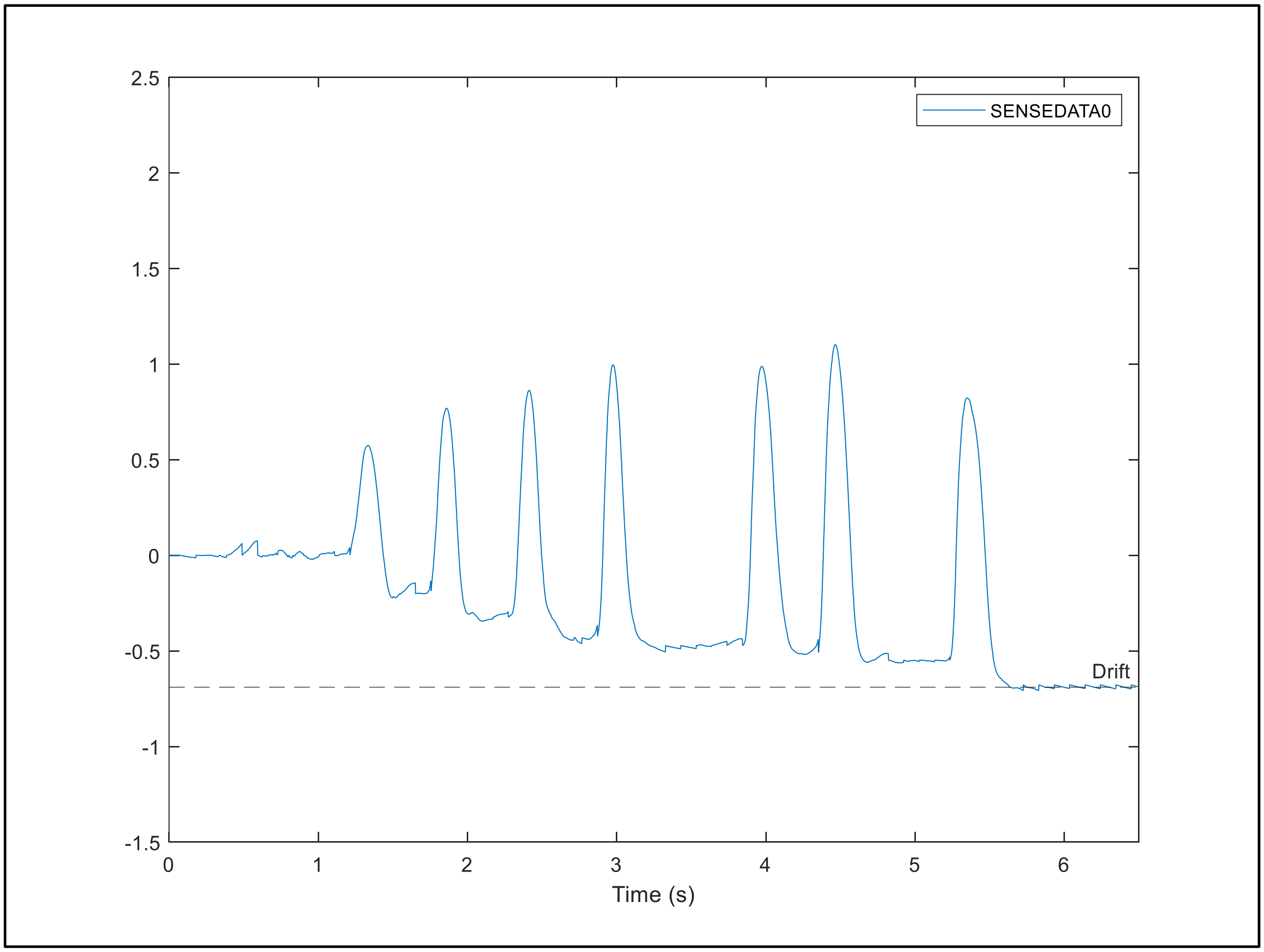


Figure 19: SENSEDATA0 drift under multiple presses and releases

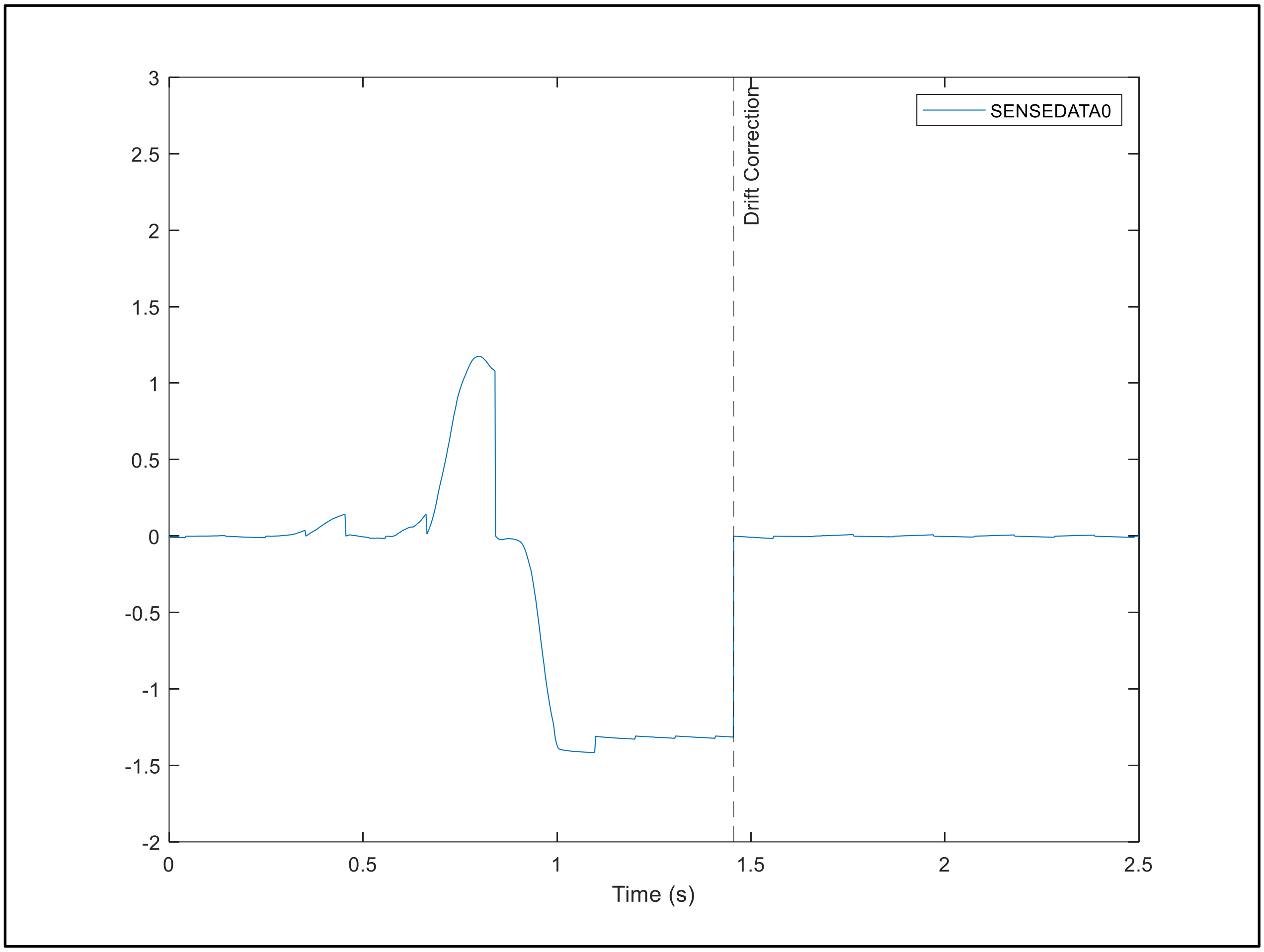


Figure 20: SENSEDATA0 drift correction

4.3 Minimum Load Requirement

If the output capacitive load on the BOS0614 is too low, automatic negative drift may occur on SENSEDATAx on that channel due to leakage. One way to check if the capacitance value is too low is to monitor SENSEDATAs for a gradual drift when the actuator is not subject to any force. Connecting an oscilloscope probe on the output node (OUTx) will increase the leakage and will exacerbate the issue. If connecting the probe causes the issue, this indicates the capacitance value to too close to the limit.

Solutions to this problem are:

1. using a piezo actuator with more capacitance;
2. adding a capacitor in parallel with the actuator;
3. increase the values of SENSECONFIG.SCOMP and SENSECONFIG.SAMP.

Note that adding a capacitor to the actuator (option 2) may attenuate the sensing voltage variations for the same applied force.

Voltage changes below SENSECONFIG.SAMP within the time defined by SENSECONFIG.SCOMP are ignored and not cumulated into SENSEDATAx. Increasing SENSECONFIG.SCOMP will zero the output channels sooner, so it is zeroed before the voltage crosses the SENSECONFIG.SAMP value and the voltage change is accumulated into SENSEDATAx.

Also consider that increasing temperature slightly increases the leakage. Consider this when checking for leakage and while making adjustments.

4.4 Sensing Interface Calibration

The sensing interface must be calibrated before it can be used reliably. This operation need only be run once and is not automatic. To perform calibration,

1. Enable sensing on the first channel (SENSECONFIG.CH0 = 1).
2. Run the sensing calibration (SENSECONFIG.CAL = 1).
3. Wait for the calibration to finish by polling SENSECONFIG.CAL (this bit self-clears). The calibration duration is approximately equivalent to the time set by CONFIG.SHORT.

The example code uses a BOS0614 driver that automatically calibrates the sensing when the driver is initialized. See this code for an example of calibration.

5 Sensing Examples

Code examples can be found on the Boréas Technologies website in the BOS0614-KIT Documentation page. The ZIP file contains a code project with many targets, one for each example. Many examples are provided, including sensing examples listed below.

The examples can be compiled and run from VSCode IDE. Instructions to install VSCode and the toolchain, and how to build and run the examples are given in the README.md file contained in the ZIP file. The code can be ported and modified as needed.

The code includes the BOS0614 driver that can be reused in any application. It can also be modified if more functionality is needed.

This section gives a short description of each sensing example provided.

5.1 Active Sensing (ExampleSwButton)

This example demonstrates how to program the IC for active sensing while running sensing detection algorithms in the MCU. This is useful when running detection algorithms exceeding the IC capabilities. The MCU activates sensing and polls the voltage periodically. Sensing detection criteria is assessed in the MCU algorithm using a combination of threshold and slope methods. Haptic feedback is played using the RAM Synthesis mode. Define instructions at the top of the example file set the sensing mechanisms to use and the parameters to configure, along with feedback waveform shape to use. Drift monitoring is used when sensing for the actuator to be pressed.

The sections below give an overview of each phase of sensing. Letters in the following figure indicate each phase of the sequence. The voltage is conceptually representing the voltage produced when force is exerted on the piezo actuator (B, E, H), and the voltage forced onto the actuator to produce the haptic feedback (C, F).

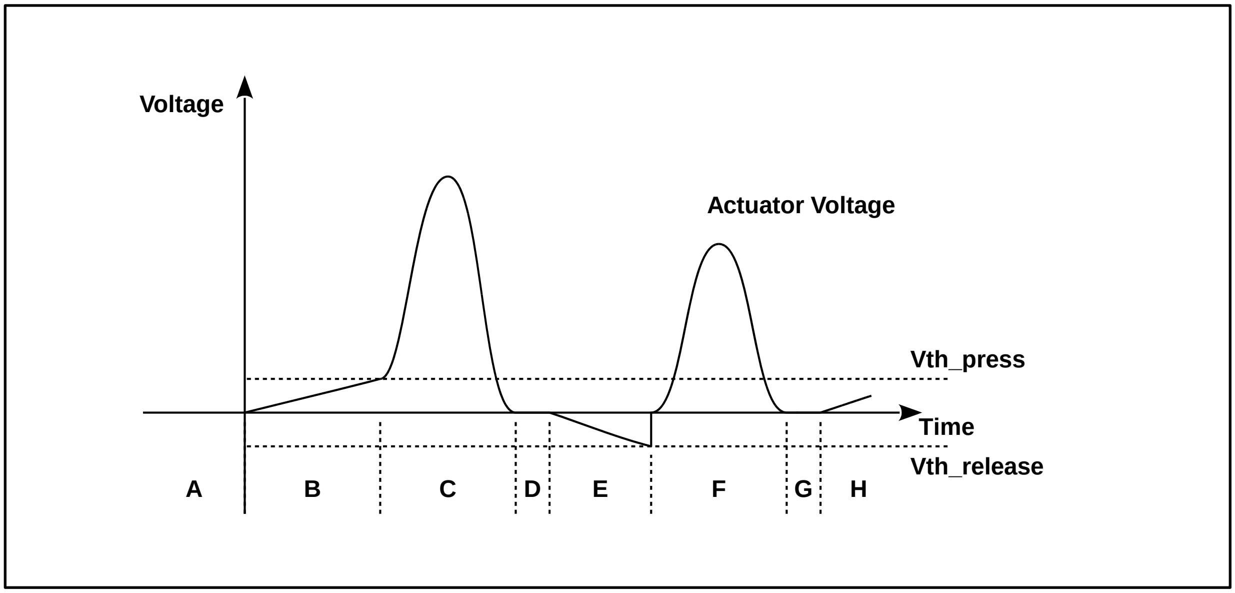


Figure 21 Typical sensing and feedback sequence emulating a click/release button

Phase A: Sensing Initialization

This phase initializes the parameters in the sensing mechanisms structures to be used later.

Entering Phase B: Press Sensing Setup

Enable sensing.

Resets the sensing mechanisms variables.

Resets the drift monitoring variables.

Phase B: Sensing for Press

The voltage is polled from the IC.

Each sensing mechanism is evaluated, then the combination of all active sensing mechanism is evaluated to determine if a press is detected.

SENSEDATAx is monitored for drift. If a drift is found, then SENSEDATAx is reset by disabling and then reenabling the sensing.

Phase C: Press Feedback Waveform

The sensing is disabled, and the haptic feedback is triggered.

Phase D: Creep Stabilization

This phase is implicit to phase C in the code example. The stabilization is programmed to automatically execute when the waveform has finished played via TC.TCP.

Entering Phase E: Release Sensing Setup

Enable sensing.

Resets the sensing mechanisms variables.

Phase E: Sensing for Release

The voltage is polled from the IC.

Each sensing mechanism is evaluated, then the combination of all active sensing mechanism is evaluated to determine if a press is detected.

SENSEDATAx is not monitored for drift. Usually, a button is maintained pressed at a relatively constant force and drift is not significant.

Phase F: Release Playback Waveform

The sensing is disabled, and the haptic feedback is triggered.

Phase G: Creep Stabilization

This phase is implicit to phase F in the code example. The stabilization is programmed to automatically execute when the waveform has finished played via TC.TCR.

5.2 Autonomous Sensing (ExampleHwButton)

All relevant code for this example is in the initialization function. It first programs the press and release haptic feedback waveforms in the waveform synthesizer (WFS). Then the sensing configurations are setup as a combination of threshold and slope for the press detection, and as slope detection only for the release detection. The sensing runs without intervention from the MCU.

The GPIO output is configured to output the button state.

The sensing runs without intervention from the MCU. The example routine function only checks the button state by reading the GPIO pin and reports its state to the user via the associated channel LED.

SENSEDATAx is not monitored for drift. One can repeatedly press softly on the actuator without triggering the haptics to witness the impact of the drift on subsequent force required to trigger the haptic.

5.3 Drift Monitoring (ExampleMonitoring)

This example is the same as the previous but adds the drift monitoring in the example routine. Comparing with the previous example, one can experience how more repeatable the force required to trigger the haptic becomes.

5.4 ZPS Sensing (ExampleZpsButton)

This example shows how to program autonomous sensing while enabling ZPS for automatic wake-up. The feedback in autonomous sensing is configured the same as before, then the BOS0614 is put to sleep. The SLEEP LED on the board lights up automatically when the chip is in sleep.

The example routine only monitors the state the GPIO output to show activity on all channels using LEDs.

When the button is pressed, the BOS0614 will get out of SLEEP automatically. If no press is detected (the force is insufficient for the sensing conditions to be met), the chip will automatically return to SLEEP. After a release is detected, the chip will also return to sleep automatically.

6 Related Products

	PRODUCT NAME	DESCRIPTION
1	BOS0614	Four-Channel Piezo Haptic Driver with Integrated Sensing

7 Document History

ISSUE	DATE	DOCUMENT NUMBER	CHANGES
1	October 2022	BT005EAN02.01	Original document

8 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 sides of the actuator (OUTx and VDD) 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 the actuator terminals. Doing so might damage the BOS0614-KIT and/or your oscilloscope. For more information please consult the *Probing BOS0614 with an Oscilloscope* application note available for download on Boréas website.

Information relating to products and circuits furnished herein by Boréas Technologies Inc. is believed to be reliable. However, Boréas Technologies assumes no liability for errors that may appear in this document, or for liability otherwise arising from the application or use of any such information which may result from such application or use. The products, their specifications and the information appearing in the document are subject to change by Boréas Technologies without notice. Trademarks and registered trademarks are the property of their respective owners.

Boréas's products are provided subject to the Boréas's Terms & Conditions available at:

https://www.boreas.ca/pages/general-terms-and-conditions