AN0596

CS530x and CS4282P Hybrid Gain Control

Introduction

This document provides:

- Explanatory notes on what Hybrid Gain Control (HGC) does, and how it works
- Guidance on designing input path hardware for use with HGC
- Guidance on how host software should control HGC
- Guidance on how to configure and tune HGC
- Configuration and control example using a HGC Mic Pre-Amp circuit

This document should be read in conjunction with the CS530x or CS4282P datasheet, specifically the functional description in Section 4.5.4 ‘Hybrid Gain Control (HGC)’, and the register description in Section 6.5 (or Section 6.6) ‘HGC’.

Table of Contents

1 Overview

Hybrid Gain Control (HGC) is a novel feature of the CS530x (CS5302P, CS5304P, CS5308P, CS5304S and CS5308S) audio ADCs and the CS4282P audio CODEC to simplify and improve system-level control of microphone preamplifier gain in the ADC input path. It is designed for applications where the analog mic preamp gain is stepped and the system provides a near-continuous digital user interface for the mic preamp gain.

In this kind of system, the user interface gain control emulates a traditional continuous gain control like a potentiometer. The user interface controls both the external stepped analog gain and the internal digital gain via software. The digital gain interpolates between the analog gain steps.

Such mic preamp gain controls are common on many pro-audio products including digital mixing consoles and DAW audio interfaces.

HGC offers a number of benefits for implementing this kind of feature:

- Zero-cross-aligned synchronization of analog and digital gain changes, to avoid zipper noise and glitches when changing gain
- Analog and digital make-up gain state stored in a single location, to simplify host control
- Gain control operations controlled via the same host interface as ADC configuration, to reduce the number of I^{2}C or SPI interfaces in the system
- Optionally, mitigation of charge injection noise from gain selection multiplexers in the mic preamp, to reduce zipper noise and allow use of higher-performance multiplexers with higher charge injection
- Control of up to 64 additional general-purpose logic outputs via the same host interface as ADC configuration

HGC may be used with any stepped-gain mic preamp design, where the gain can be controlled via daisy-chained SPI using no more than 32 bits per channel.

2 How HGC Works

2.1 Conventional Gain Implementation

In typical products in which the mic preamp implements switched analog gain and the user interface emulates a continuous gain control, the system software takes the user-interface gain selection and calculates two parameters:

- Analog gain switch selection (e.g. closest value available to the commanded system gain from the user interface control). Typical step size may be around 3 dB.
- Digital interpolated gain selection (difference between the commanded system gain, and the analog gain switch selection). Typical step size may be 0.5 dB or less.

As the user interface gain setting is increased monotonically, the mic preamp gain increases periodically in steps, while the digital gain interpolates between those steps. When the analog gain increases by a step, the digital gain must simultaneously decrease by the same amount. The analog and digital gains sum to produce the system gain, which should be the same as the commanded gain. This is illustrated in Figure 1.

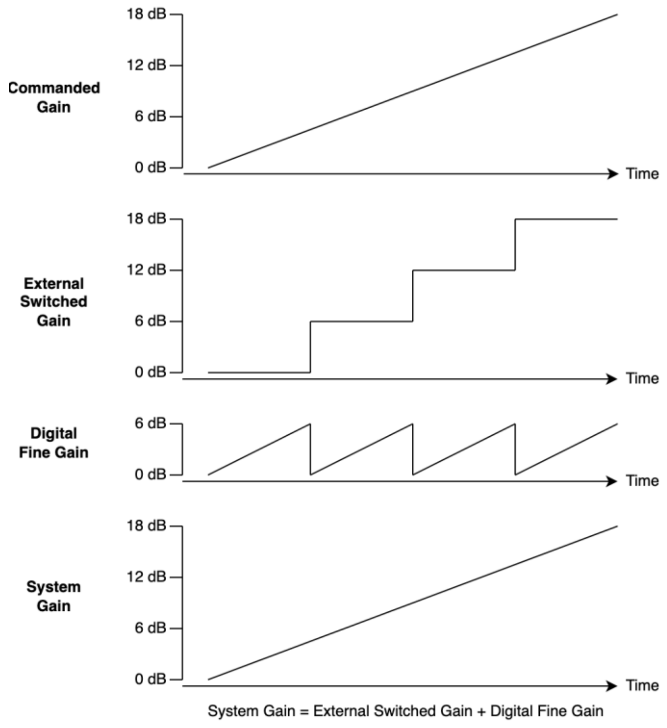

Figure 1: Analog and Digital Gain Step Behavior

The analog gain changes are aligned with analog zero-cross events, so the gain change occurs when the signal amplitude is zero, to minimize audible transients. If the digital gain counter-step is not precisely aligned with the analog gain change, it can produce an audible transient. This alignment can be difficult to achieve because the DSP cores of products are often located some distance from the mic preamp, both physically and in terms of system architecture.

A typical system implementing this scheme, without using HGC, is illustrated in Figure 2.

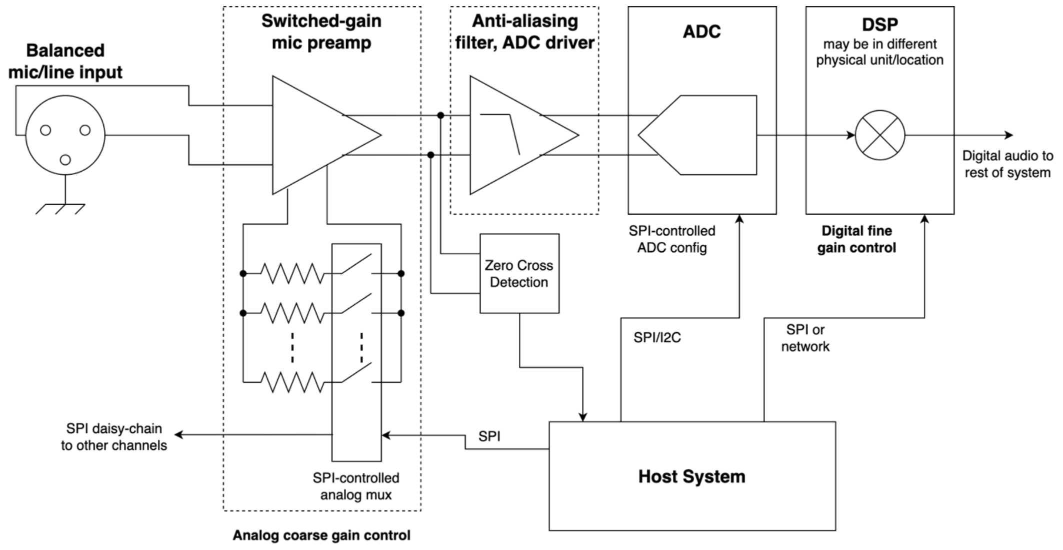
Balanced mic/line input
Switched-gain mic preamp
Anti-aliasing filter, ADC driver
ADC
DSP
may be in different physical unit/location
Digital audio to rest of system
SPI-controlled ADC config
Digital fine gain control
Zero Cross Detection
SPI/I2C
SPI or network
SPI daisy-chain to other channels
SPI
Host System
SPI-controlled analog mux
Analog coarse gain control

Figure 2: Typical Switched-Gain Mic Preamp Input Path, without HGC

2.2 HGC System Architecture

HGC co-locates several elements of the system shown in Figure 2 into the CS530x ADCs and CS4282P CODEC:

- A configurable daisy-chain SPI controller, capable of controlling the gain of up to 8 mic preamps of arbitrary analog design, plus two extra general-purpose output devices of up to 32 bits each.
- Analog zero-cross detectors on each ADC channel, to precisely align the analog gain step changes commanded via SPI with analog zero-cross events, to minimize gain change transients.
- Digital gain control, to implement the ‘interpolation gain’ required to emulate the glitch-free continuous gain control response.
- Transient masking controlled by an exponential moving average audio signal level detector, to reduce audibility of charge injection transients.
- Finite state machine controller, coordinating operation of all the above blocks to execute precisely synchronized, glitch-free gain updates across the stepped analog mic preamp gain and the digital make-up gain.

The resulting simplified system architecture with HGC is illustrated in Figure 3.

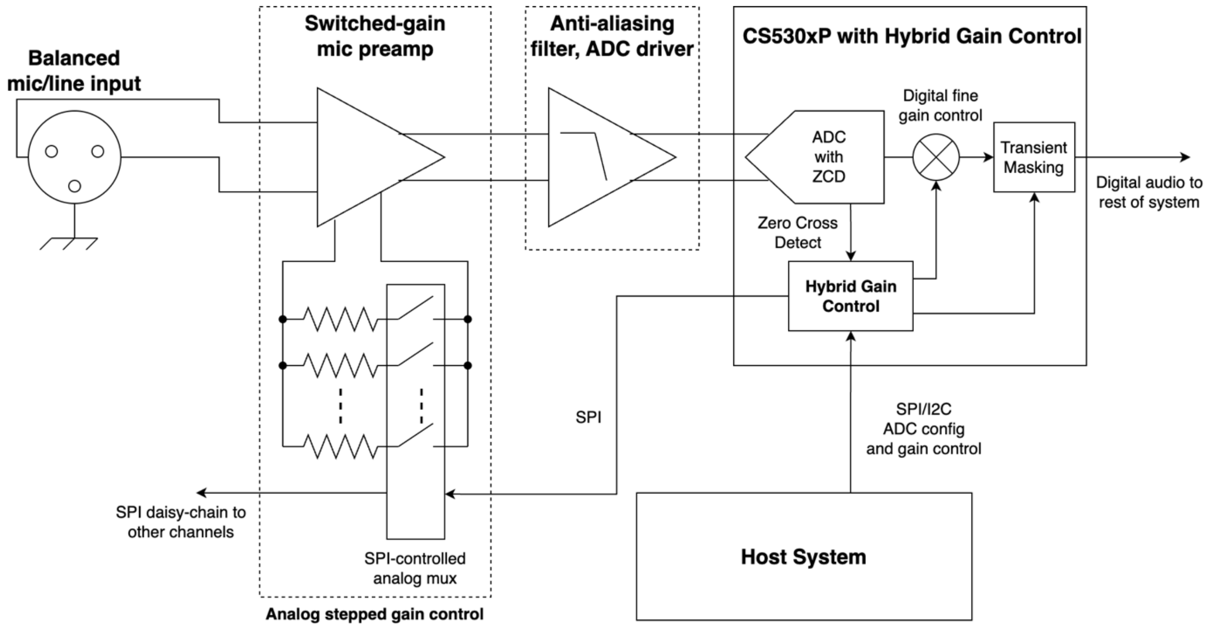
Balanced mic/line input
Switched-gain mic preamp
Anti-aliasing filter, ADC driver
CS530xP with Hybrid Gain Control
ADC with ZCD
Digital fine gain control
Transient Masking
Digital audio to rest of system
Zero Cross Detect
Hybrid Gain Control
SPI
SPI/I2C ADC config and gain control
SPI daisy-chain to other channels
SPI-controlled analog mux
Analog stepped gain control
Host System

Figure 3: Switched-Gain Mic Preamp Input Path with HGC

The zero-cross-aligned analog gain change is executed by HGC, in response to a request from the host for the gain change. Since HGC also has control over the digital audio signal path, it may precisely align the digital gain change with the analog gain change.

2.3 HGC Mitigates Gain Change Transients

Timing misalignment between the zero-cross-aligned analog gain step change, and the corresponding digital gain counter-step, may cause audible transients as illustrated in Figure 4.

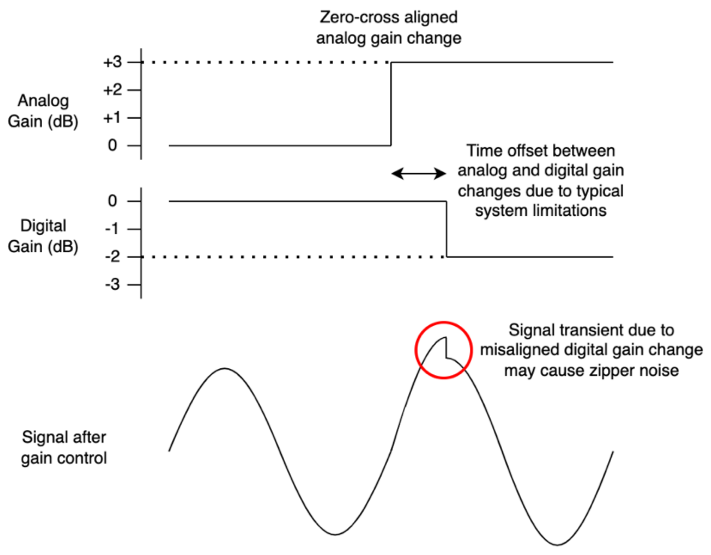
Zero-cross aligned analog gain change
Analog Gain (dB)
+3
+2
+1
0
Time offset between analog and digital gain changes due to typical system limitations
Digital Gain (dB)
0
-1
-2
-3
Signal transient due to misaligned digital gain change may cause zipper noise
Signal after gain control

Figure 4: Timing misalignment of analog and digital gain changes causes audible transient

Figure 5 illustrates how alignment of the analog and digital gain changes by HGC avoids these gain change transients, with the digital gain ramping to the new value over several milliseconds.

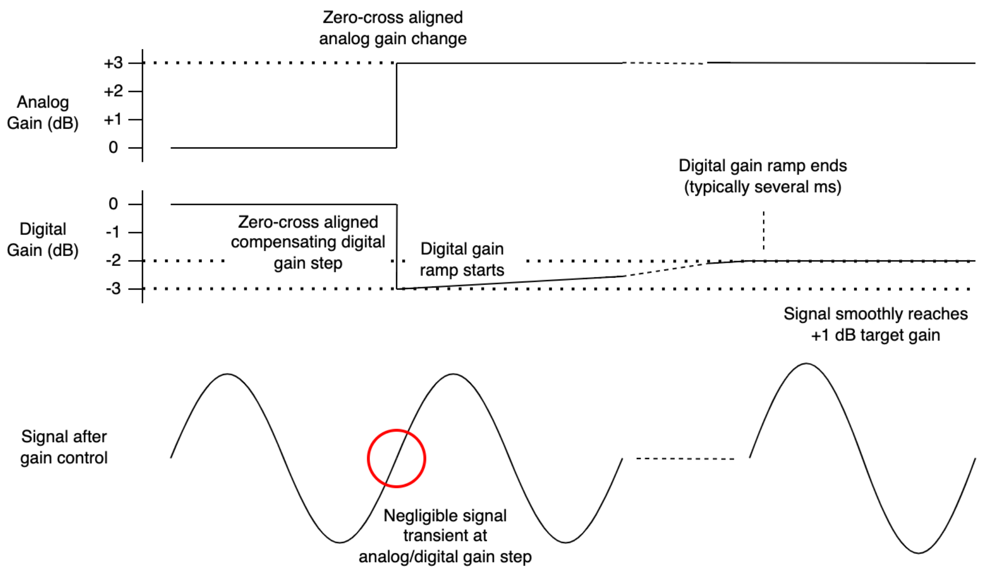
Zero-cross aligned analog gain change
Analog Gain (dB)
+3
+2
+1
0
Digital gain ramp ends (typically several ms)
Digital Gain (dB)
0
-1
-2
-3
Zero-cross aligned compensating digital gain step
Digital gain ramp starts
Signal smoothly reaches +1 dB target gain
Signal after gain control
Negligible signal transient at analog/digital gain step

Figure 5: HGC ensures analog and digital gain changes are aligned, and gain is ramped to avoid gain change transients

2.4 HGC Mitigates Multiplexer Charge-Injection Transients

Gain switching in mic preamps is often implemented using analog multiplexers to switch different value resistors into the circuit. These multiplexers have charge injection, which may cause a transient to be added to the audio when gain is switched.

The audibility of these charge injection transients depends on the mic preamp circuit design and multiplexer selection. If they are sufficiently audible to degrade the user experience for the product when the audio signal level is low, the Transient Masking feature of HGC may be used to reduce the audibility of these transients. This can particularly reduce audibility of zipper noise caused by charge injection when sweeping the user interface gain control with no signal present.

Charge injection transients are typically relatively low in level (e.g. -50 dBFS), have consistent amplitude over mic preamp gain, and have an analog duration that is short relative to the sample period, so in the digital domain they resemble the impulse response of the decimation filter. For a slow roll-off minimum-phase decimation filter, most of their energy is contained in just 2-4 samples.

As illustrated in Figure 6, HGC Transient Masking repeats the audio sample value immediately preceding the charge injection transient for a small number of samples, to mask the spike of the charge injection transient. If applied when the audio level is similar to or lower than the charge injection transient, it reduces the audibility of the transient. Transient Masking is configurable to be applied only when the average audio level preceding the gain change is below a configurable threshold.

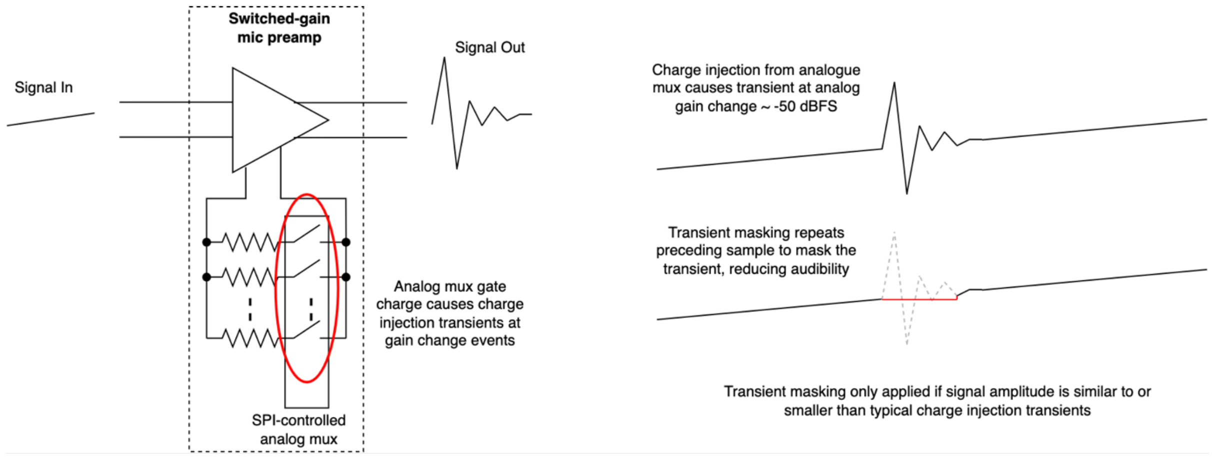
Switched-gain mic preamp
Signal In
Signal Out
Analog mux gate charge causes charge injection transients at gain change events
SPI-controlled analog mux
Charge injection from analogue mux causes transient at analog gain change ~ -50 dBFS
Transient masking repeats preceding sample to mask the transient, reducing audibility
Transient masking only applied if signal amplitude is similar to or smaller than typical charge injection transients

Figure 6: HGC performs transient masking to mitigate charge injection transients from analog multiplexers

Analog multiplexers tend to have an inverse relationship between charge injection and linearity: the lower the distortion level (THD) of a multiplexer, the higher the charge injection and the louder the charge injection transients. Transient Masking may allow use of a lower-distortion, higher charge-injection multiplexer, to help achieve better distortion performance of the mic preamp.

2.5 HGC Operational Sequence for Gain Change

The sequence of events for a gain change is illustrated in Figure 7.

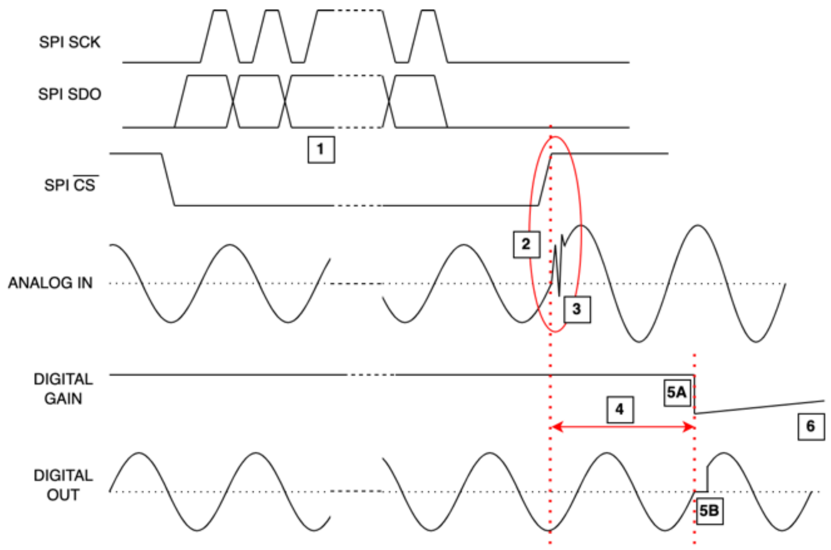

Figure 7: HGC analog gain change sequence

At Point 1 in Figure 7, the HGC SPI interface is sending the gain codes to the daisy-chain of mic preamps to configure one of them for the newly commanded gain value. This gain code depends on the design of the mic preamp. It is written into a control register in the Emerick ADC before the gain change is initiated.

After the HGC SPI interface has shifted the gain bit patterns into the mic preamps, the SPI CSb signal remains asserted low until a zero-cross is detected on the channel that is changing gain (Point 2 in Figure 7). When CSb is consequently de-asserted, it causes the new gain code to take effect in the mic pre-amp, causing (Point 3 in Figure 7):

- An instantaneous gain change in the analog signal path.
- Concurrently with the gain change, a charge injection transient may be added to the analog signal due to the switching action of the gain control multiplexer in the mic preamp.

After the gain change, HGC waits a configurable duration measured in post-decimator audio samples (typically 4-8 samples, Point 4 in Figure 7) to compensate for the signal path latency from the mic preamp, through the analog anti-aliasing filter, ADC and decimation filter.

At this point, HGC initiates a digital gain counter-step (Point 5A in Figure 7), to counteract the analog gain change. It then commences a gain ramp to the final fine-gain value (Point 6 in Figure 7). Ramping the gain over tens of milliseconds instead of an instantaneous gain change, avoids introducing audible transients.

At the same time, if enabled, HGC also may perform a sample-repeat (Point 5B in Figure 7) to mask the charge injection transient.

3 Hardware Design with HGC

3.1 Mic Preamp Design

HGC may be used with any mic preamp design that complies with the following requirements:

- Preamp must have stepped gain control. The gain steps do not need to be equal-sized, and there may be any number of gain steps of any values. The maximum difference between adjacent gain steps is +/-15.875 dB (typical step size is 3–6 dB). See below for further details in case of larger step size.
- Gain must be controllable by daisy-chain SPI (or a logical equivalent), such that the gain can be changed by writing one SPI message into its daisy-chain SPI shift register, of any length between 1 and 32 bits per mic preamp. Coding of SPI message content to gain value is completely configurable in HGC.
- Gain change must take effect immediately on de-asserting SPI CSb at the end of the daisy-chain SPI operation.

A simple and compact implementation is to use daisy-chain-compatible SPI-controlled analog switches to connect different-value resistors into the mic preamp circuit. Two examples of switches that meet these requirements are the Analog Devices ADG1414 and ADGS5414.

Alternatively, a logic shift register such as the 8-bit 74HCS595 may be used with daisy-chain SPI to provide static logic outputs to control analog multiplexers or discrete transistors.

Note: If the difference between analog gain steps is greater than +/-15.875 dB, the HGC still performs the zero-cross aligned gain change, but there is an increased risk of audible artifacts. In this case, HGC generates a digital counter-step of –16 dB or +15.875 dB, before ramping the digital gain to the new target value. This produces a short-term gain anomaly, equal to the difference between the analog gain step size and the digital gain counter-step. Provided the gain anomaly does not cause momentary clipping, the gain anomaly might not be obtrusive or audible, and this mode of operation with analog gain steps larger than 15.875 dB may be acceptable for some applications.

3.2 SPI Daisy-Chain Design

The principle of operation of SPI daisy-chain is shown in Figure 8. Data is transmitted in reverse order (data for the last device in the chain is transmitted first).

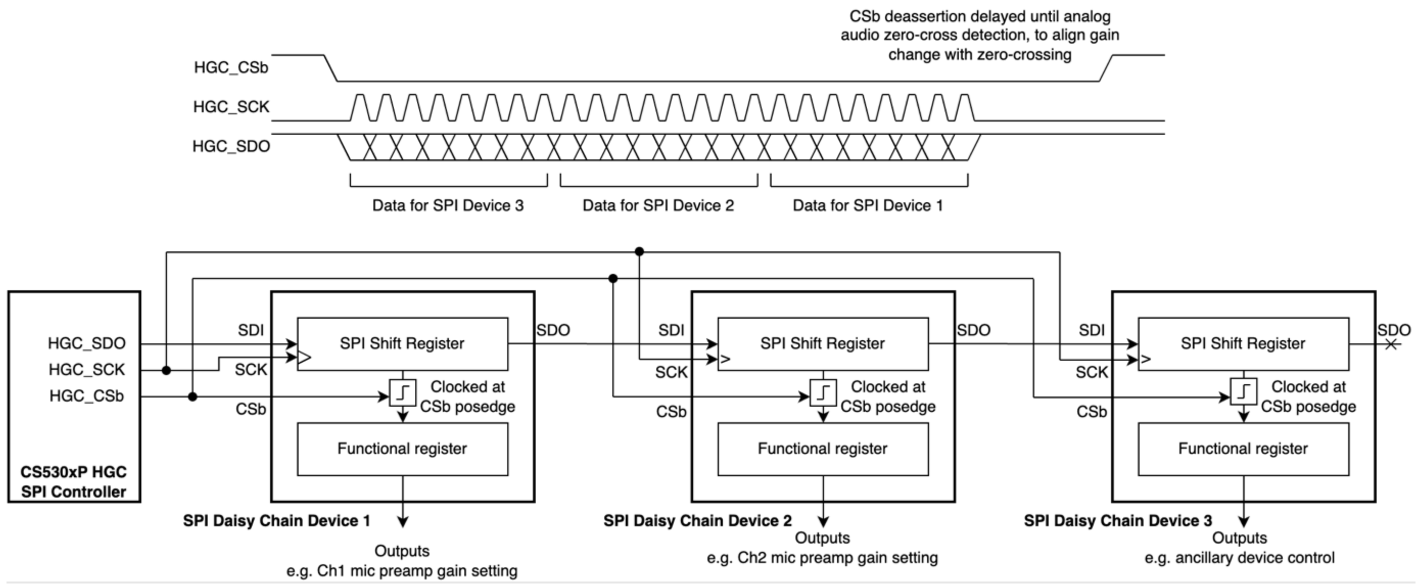

Figure 8: SPI daisy-chain architecture and message format

HGC supports up to 8 channels of mic preamp gain control, plus 2 auxiliary devices. The length of the bit pattern associated with each device is independently configurable between 1 and 32 bits (CHx_BIT_PATT_LENGTH register fields). If its length is set to 0 (reset default value), it is omitted when the SPI daisy chain message is transmitted.

The SPI daisy chain must be connected on the PCB in the same order that the bit pattern length field registers appear in the register map: mic preamp for Channels 1-8, followed by Auxiliary 1, then Auxiliary 2.

Note that the bit-pattern length fields select MSB-aligned data content in the respective bit-pattern fields. The selected bit patterns are transmitted MSB first.

In any example use case, the Channel 1 mic preamp requires 8-bit data to set its gain:

- CH1_BIT_PATT_LENGTH should be set to 0x8
- To set the gain code for the preamp to 0xB3:
  - Write 0xB300 to CH1_BIT_PATT1
  - Note that CH1_BIT_PATT0 is not used

Another example: the Channel 6 mic preamp requires 24-bit data to set its gain:

- CH6_BIT_PATT_LENGTH should be set to 0x18
- To set the gain code for the preamp to 0x8002E1:
  - Write 0x8002 to CH6_BIT_PATT1
  - Write 0xE100 to CH6_BIT_PATT0

The HGC SPI clock frequency can be set to a multiple-of-2 divisor of the 24.576/22.5798 MHz internal master clock frequency, to a maximum of 12.288 MHz (see SCK_DIV field). In most cases, HGC SPI clock phase and polarity (CPOL and CPHA fields) should remain at their default settings of ‘negative’. The minimum inter-message duration (CS_IDLE_DUR), start of transaction delay (CS_FALL_DELAY) and end of transaction delay (CS_RISE_DELAY) fields may be tuned – see the CS530x or CS4282P datasheet for details.

Some SPI daisy-chain devices require initial configuration, and this can be accomplished by writing the configuration data to the devices using the INIT_UPDATE control from a host software initialization routine. INIT_UPDATE transmits the contents of the bit pattern registers out to the SPI daisy chain immediately, without waiting for an analog zero-cross.

For example, the Analog Devices ADGS5414 requires a 16-bit code 0x2900 to be written to it to enable its daisy-chain output, following which it is addressed as an 8-bit daisy-chained device. This can be handled with an initialization routine in host software, setting the bit pattern lengths initially to 16 bits and using the INIT_UPDATE operation to transmit the 0x2900 code to the last ADGS5414 in the SPI daisy-chain. The 0x2900 code propagates through the chain, sequentially enabling the daisy-chain outputs on the devices. After this, the bit pattern lengths are changed to 8 bits, and all devices in the chain can be accessed.

Writing 1 to the INIT_UPDATE bit without changing any bit pattern registers results in no change to the system state, because it writes to all the devices the same values as it wrote last time.

If there are a large number of devices in the SPI daisy chain, attention must be paid to timing and signal integrity of the SPI SCK and CSb signals which may each be driving a large number logic inputs: they may need to be either buffered, or the SPI clock speed reduced (SCK_DIV).

3.3 Auxiliary Outputs

HGC features two extra 32-bit daisy chain device registers ‘AUX1’ and ‘AUX2’, which allows control of arbitrary additional SPI daisy chain devices through the same interface as control of the mic preamp gains. These are intended for control of auxiliary hardware functions associated with the input paths, such as controlling +48 V phantom power, analog high-pass filter, controlling LEDs by the input connectors, or switching between balanced and unbalanced inputs.

The auxiliary device bit pattern length fields AUXn_BIT_PATT_LENGTH must be set to have the same bit length as the auxiliary devices in the daisy chain. For example, if using a single 74HCS595 8-bit shift register (as described in Section 4.1), the bit pattern length would be set to 8 bits. The Qn outputs of the 74HCS595 would take the value of the data written to the AUXn_BIT_PATT0/1 fields, every time a SPI daisy chain transaction occurs – that is, on either a channel gain change event, or on the INIT_UPDATE bit being set.

Although the auxiliary device registers are structured as two devices each up to 32 bits, they can be used as up to 64 bits of SPI daisy chain shift register control split arbitrarily across multiple devices: for instance, eight 74HCS595 shift registers connected sequentially.

If not using any auxiliary devices, AUXn_BIT_PATT_LENGTH should both be zero.

3.4 ADC Configuration

HGC requires the ADC to be configured in Software Mode, not Hardware Mode, because HGC requires the host to dynamically interact with internal registers.

Transient Masking is only recommended for use with the slow roll-off minimum-phase decimation filter type. This is because charge injection transients appear in the digital domain to have a similar shape to the decimation filter impulse response. The slow roll-off minimum phase impulse response has no pre-ringing and minimal post-ringing, so it causes the effect of charge injection transients in the digital domain to be shorter and easier to mask than with other filter types.

The ramp rate of the HGC digital gain after a synchronized gain change is controlled by the IN_RAMP_RATE_DEC and IN_RAMP_RATE_INC fields of the IN_RAMP_SUM register. They can be set to different rates if the application requires it, but for most applications it is recommended that these two fields should have the same value.

HGC is compatible with all other configuration settings of the ADC. HGC SPI operates only at 3.3 V.

4 Host Software Control of HGC

4.1 Commanding a Gain Change

To command a synchronized analog/digital gain change, the host must write four pieces of information to the HGC registers in CS530x and CS4282P:

- CHx_BIT_PATT_0/1: The bit pattern to set the mic preamp to the selected analog gain setting (the coding depends entirely on the design of the mic preamp and its daisy-chain SPI control interface, see Section 3.1).
- CHx_ANA_GAIN: The absolute analog gain value corresponding to the new bit pattern for the selected mic preamp, expressed to the nearest 0.125 dB.
- CHx_DIG_GAIN: The value of the digital interpolation gain required to achieve the desired system gain, expressed to the nearest 0.125 dB.
- CHx_UPDATE: The signal to indicate that the three items above have all been written, such that HGC may now execute the synchronized gain change.

The algorithm to choose which analog gain step to select for a particular system gain request is defined in host software.

4.1.1 Gain Change Command Timing

The host may command a gain change at any time, and it will be executed within a time bound of approximately the number of active channels multiplied by the zero-cross timeout configuration setting (20 ms max). In practice, it is likely to execute much faster than that due to audio zero-cross detection.

HGC operates using an internal state machine that executes requested gain update operations for each channel in turn, based on availability of the daisy-chain SPI port. This availability is dominated by the time waiting for zero-cross for prior requested gain updates and is bounded by the zero-cross timeout.

The HGC internal state machine checks the CHx_UPDATE bit for each channel in turn, and if it has been set since the last gain operation on that channel, HGC executes the gain update for that channel according to the register fields listed above. Each gain update takes a non-deterministic but bounded time to complete due to the zero-cross alignment.

If a gain update operation is commanded for a channel (as above) before a preceding gain update operation on that same channel has executed, the most recently written gain update operation will be executed, and the previously pending gain update operation will be dropped.

Writing 1 to CHx_UPDATE causes HGC to immediately “snapshot” the CHx_BIT_PATT, CHx_ANA_GAIN and CHx_DIG_GAIN register values, for use in the gain update operation when the state machine is ready. This ensures that gain update operations are safe from timing collisions between the host writing registers to command an operation, and the HGC state machine reading registers to execute gain updates. Changing any of the CHx_BIT_PATT, CHx_ANA_GAIN and CHx_DIG_GAIN registers without setting CHx_UPDATE has no effect on any previously commanded and pending gain updates on that channel.

4.1.2 Usage of CHx_ANA_GAIN Absolute Analog Gain Information

The CHx_ANA_GAIN fields are used to determine the difference in mic preamp analog gain between the previous setting and the new setting. This difference is used to determine the size and direction of the digital gain counter-step, to initially counter the analog gain step before the digital gain is ramped to the new value specified in CHx_DIG_GAIN. The absolute values don’t affect HGC operation – only the difference between the previous and new values is used.

4.1.3 Commanding Gain Changes with Auto-Incrementing Register Writes

The register layout facilitates auto-incrementing host access operations, to improve efficiency of the host interface. A gain update for a single channel where the bit pattern is up to 16 bits in length can be performed with a single auto-incrementing write starting at CHx_BIT_PATT_1, and a length of 3 registers (and if the bit pattern is more than 16 bits long, the register write starts at CHx_BIT_PATT_0 with a length of 4). A single auto-incrementing host access may also write gain update commands for all the channels (or any set of consecutive channels) with high efficiency as they are arranged sequentially.

4.2 Configuring HGC

From reset defaults, the minimum configuration necessary to use HGC is:

- Set the bit pattern length fields for channels with mic preamps (CHx_BIT_PATT_LENGTH), according to the length of the SPI daisy chain shift register for each channel (see Section 3.1).
- Set the bit pattern length fields for any auxiliary devices (see Section 3.3).
- Set DIG_GAIN_DELAY according to the tuned value (see Section 5.2).
- If transient masking is not required, clear the TM_EN bit.

If transient masking is enabled, the transient masking tuned parameters TM_DELAY, TM_HOLD_TIME, TM_LD_THRESH must be set. See Section 5.3.

If required by the HGC SPI daisy chain hardware timing constraints, SPI tuning parameters SCK_DIV, CPHA, CPOL, CS_IDLE_DUR, CS_RISE_DELAY and CS_FALL_DELAY may need to be changed from defaults.

4.3 Example Configuration and Control

In this section, an example hardware system is described, then the host register writes to configure, initialize and use HGC with that hardware system are described.

4.3.1 Hardware Characteristics

For this example, there is a two-channel ADC, with a mic preamp that gives +4 to +18 dB gain in approximately 3 dB steps. The mic preamp gain is controlled by an 8-bit shift register in the SPI daisy chain. The gain coding for the mic preamp, with gain values rounded to the nearest 0.125 dB (to match the gain resolution of HGC), is shown in Table 1.

Table 1 Example Gain Coding

Gain (dB)	Bit pattern
+4.0	0xC0
+5.375	0xA0
+8.875	0x88
+11.5	0xF0
+15.0	0xA2
+18.0	0xC6

Additionally, there is an 8-bit auxiliary shift register in the SPI chain, the least-significant two output bits of which enable +48 V phantom power for the two inputs.

4.3.2 HGC Configuration Settings

The registers in Table 2 show the initial settings of HGC for this hardware configuration, as deltas from reset defaults. It assumes that tuning has found DIG_GAIN_DELAY should have value of 1 sample, transient masking is not required, and all other defaults are satisfactory.

Table 2 HGC Configuration

Register	Register Value	Field(s)	Bits	Field Value	Meaning
CH1_CONFIG	0x0008	CH1_BIT_PATT_LENGTH	7:0	0x08	Length 8 bits
CH2_CONFIG	0x0008	CH2_BIT_PATT_LENGTH	7:0	0x08	Length 8 bits
AUX1_CONFIG	0x0008	AUX1_BIT_PATT_LENGTH	7:0	0x08	Length 8 bits
PATH_DELAY	0x0001	DIG_GAIN_DELAY	7:0	0x01	Delay 4 samples
TM	0x0000	TM_EN	0	0x0	TM disabled
PAD_HGC_SPI	0x0001	HGC_SPI_EN	0	0x1	Enable HGC SPI

The registers in Table 2 are represented in the following pseudo-code:

```
# Configure HGC settings
writeReg("CH1_CONFIG", 0x0008)      # ch1 bit patt length 8
writeReg("CH2_CONFIG", 0x0008)      # ch2 bit patt length 8
writeReg("AUX1_CONFIG", 0x0008)     # aux bit patt length 8
writeReg("PATH_DELAY", 0x0001)      # dig gain delay = 1, TM not used
writeReg("TM", 0x0000)              # disable transient masking
writeReg("PAD_HGC_SPI", 0x0001)     # enable HGC SPI outputs
```

4.3.3 HGC Asynchronous Initialization

To initialize mic preamp gains to +4 dB and ensure that phantom power is off (logic outputs at zero), the appropriate gain values are set and then immediately written out to the SPI daisy chain using the INIT_UPDATE function. This initializes the state of all external devices in a single operation, with no zero-cross alignment.

The register writes for the HGC initialization (deltas from reset defaults) are shown in Table 3.

Table 3 Asynchronous Initialization

Register	Register Value	Field(s)	Bits	Field Value	Meaning
CH1_BIT_PATT_1	0xC000	CH1_BIT_PATT_1	15:0	0xC000	8-bit pattern 0xC0
CH1_GAIN_0	0x0020	CH1_ANA_GAIN	10:0	0x020	Analog gain +4 dB
CH1_GAIN_1	0x0000	CH1_DIG_GAIN	7:0	0x00	Digital gain 0 dB
CH2_BIT_PATT_1	0xC000	CH2_BIT_PATT_1	15:0	0xC000	Bit pattern 0xC0
CH2_GAIN_0	0x0020	CH2_ANA_GAIN	10:0	0x020	Analog gain +4 dB
CH2_GAIN_1	0x0000	CH2_DIG_GAIN	7:0	0x00	Digital gain 0 dB
AUX1_BIT_PATT_1	0x0000	AUX1_BIT_PATT_1	15:0	0x0000	Bit pattern 0x00
CONTROL	0x0001	INIT_UPDATE	0	0x1	Trigger initialization update

The registers in Table 3 are represented in the following pseudo-code:

```
# Initialize mic preamp gain and +48V via HGC
writeReg("CH1_BIT_PATT_1", 0xC000)  # ch1 bit patt 0xC0 for 4 dB
writeReg("CH1_GAIN_0", 0x0020)       # ch1 abs analog gain 4 dB
writeReg("CH1_GAIN_1", 0x0000)       # ch1 init dig gain 0 dB
writeReg("CH2_BIT_PATT_1", 0xC000)  # ch2 bit patt 0xC0 for 4 dB
writeReg("CH2_GAIN_0", 0x0020)       # ch2 abs analog gain 0 dB
writeReg("CH2_GAIN_1", 0x0000)       # ch2 init dig gain 0 dB
writeReg("AUX1_BIT_PATT_1", 0x0000) # aux1 bit patt for +48V off
writeReg("CONTROL", 0x0001)         # immediate SPI init update
```

4.3.4 Digital-Only Gain Change

To subsequently set Channel 1 gain to +5.125 dB (comprising analog gain of 4 dB, digital gain of +1.125 dB), as a zero-cross-aligned and ramped gain change, the host would perform the register write shown in Table 4.

Table 4 Digital-Only Gain Change

Register	Register Value	Field(s)	Bits	Field Value	Meaning
CH1_GAIN_1	0x8009	CH1_DIG_GAIN	7:0	0x09	Digital gain +1.125 dB
		CH1_UPDATE	15	0x1	Gain update

The registers in Table 4 are represented in the following pseudo-code:

```
writeReg("CH1_GAIN_1", 0x8009)       # CH1_DIG_GAIN +1.125 dB, set CH1_UPDATE
```

In the example above, the analog gain has not changed, so CH1_ANA_GAIN and CH1_BIT_PATT do not need to be updated. The CH1_DIG_GAIN field is encoded in 0.125 dB steps, so +1.125 dB equates to 0x09. The top bit of CH1_GAIN_1 is the CH1_UPDATE bit, which signals that the Channel 1 gain and bit pattern registers contain the correct values for a gain update to execute.

4.3.5 Analog Gain Change

To subsequently set Channel 1 gain to 6.5 dB (comprising analog gain of +5.375 dB with a corresponding bit pattern of 0xA0, digital gain of +1.125 dB), as a zero-cross-aligned and ramped gain change, the host would perform the register writes shown in Table 5.

Table 5 Analog Gain Change

Register	Register Value	Field(s)	Bits	Field Value	Meaning
CH1_BIT_PATT_1	0xA000	CH1_BIT_PATT_1	15:0	0xA000	8-bit pattern 0xA0
CH1_GAIN_0	0x002B	CH1_ANA_GAIN	10:0	0x02B	Analog gain +5.375 dB
CH1_GAIN_1	0x8009	CH1_DIG_GAIN	7:0	0x09	Digital gain +1.125 dB
		CH1_UPDATE	15	0x1	Gain update

The registers in Table 5 are represented in the following pseudo-code:

```
writeReg("CH1_BIT_PATT_1", 0xA000)  # ch1 bit patt 0xA0 +5.375 dB
writeReg("CH1_GAIN_0", 0x002B)       # CH1_ANA_GAIN +5.375 dB
writeReg("CH1_GAIN_1", 0x8009)       # CH1_DIG_GAIN +1.125 dB, CH1_UPDATE set
```

In this example, the absolute analog gain (field CH1_ANA_GAIN) has increased from its previous value of +4 dB to a new value of +5.375 dB, a step of +1.375 dB. This will cause HGC to perform a digital gain counter-step of -1.375 dB relative to its previous value of +1.5 dB, executed 1 sample (DIG_GAIN_DELAY) after the zero-cross-aligned analog gain change. After this step, the digital gain will ramp to the new value of +1.125 dB.

To subsequently set the Channel 1 gain to +9.125 dB (comprising analog gain of +8.875 dB, bit pattern 0x88, and digital gain of +0.5 dB), the host would perform the register writes shown in Table 6.

Table 6 Analog & Digital Gain Change

Register	Register Value	Field(s)	Bits	Field Value	Meaning
CH1_BIT_PATT_1	0x8800	CH1_BIT_PATT_1	15:0	0x8800	8-bit pattern 0x88
CH1_GAIN_0	0x0047	CH1_ANA_GAIN	10:0	0x047	Analog gain +8.875 dB
CH1_GAIN_1	0x8004	CH1_DIG_GAIN	7:0	0x04	Digital gain +0.5 dB
		CH1_UPDATE	15	0x1	Gain update

The registers in Table 6 are represented in the following pseudo-code:

```
writeReg(“CH1_BIT_PATT_1”, 0x8800)  # ch1 bit patt 0x88 +8.875 dB
writeReg(“CH1_GAIN_0”, 0x0047)       # CH1_ANA_GAIN +8.875 dB
writeReg(“CH1_GAIN_1”, 0x8004)       # CH1_DIG_GAIN +0.5 dB, CH1_UPDATE set
```

This will cause HGC to increase the external analog gain from +5.375 dB to +8.875 dB, a step of +3.5 dB, followed 1 sample (DIG_GAIN_DELAY) later by a digital gain counter step of -3.5 dB relative to its previous value of +1.125 dB. The digital gain will then ramp to the new value of +0.5 dB.

4.3.6 Set Auxiliary Output

To set Bit 0 of the AUX 1 device (to turn on +48 V phantom power in this example), the host would set the AUX1 bit pattern then use the INIT_UPDATE function to immediately update the SPI devices.

Table 7 Auxiliary Output Control

Register	Register Value	Field(s)	Bits	Field Value	Meaning
AUX1_BIT_PATT_1	0x0100	AUX1_BIT_PATT_1	15:0	0x0100	8-bit pattern 0x01
CONTROL	0x0001	INIT_UPDATE	0	0x1	Trigger initialization update

The registers in Table 7 are represented in the following pseudo-code:

```
writeReg(“AUX1_BIT_PATT_1”, 0x0100) # aux1 0x01 for ch1 +48V on
writeReg(“CONTROL”, 0x0001)         # immediate SPI init update
```

The host must ensure that there are no channel gain updates pending when using INIT_UPDATE in this way, because the operation writes-out to the external SPI devices the current content of all the bit pattern registers.

5 Tuning HGC

5.1 SPI Hardware Tuning

In most cases, the default SPI settings, which give a SPI clock frequency of 11.2896 or 12.288 MHz, should work with no problems. To test SPI data integrity, write arbitrary bit patterns to the bit pattern registers corresponding to devices in the SPI daisy chain, use INIT_UPDATE to transmit them, and check on your hardware (e.g. with a scope) for correct function of the SPI daisy chain. In case of signal integrity issues, the SPI parameters SCK_DIV (SPI clock frequency divider), CS_IDLE_DUR, CS_RISE_DELAY, CS_FALL_DELAY, CPHA and CPOL may be adjusted to resolve problems. See the device datasheet for further details of these parameters.

5.2 Digital Gain Delay Tuning

The field DIG_GAIN_DELAY defines the number of samples that HGC waits from a zero-cross-aligned analog gain change occurring (caused by de-assertion of SPI CSb after sending the new gain code), until the digital gain counter-step executes to compensate for the analog gain step. The purpose of this delay is to compensate for the signal path latency from the analog domain to the digital domain, to ensure that the digital gain counter-step is precisely aligned with the analog gain step.

This latency is dominated by the decimation filter, and the decimation filter has different latency characteristics for each combination of sample rate multiplier, linear/minimum phase response, and slow/fast roll-off response. DIG_GAIN_DELAY must be tuned separately for each filter type and sample rate supported by the product.

The following process is recommended to tune DIG_GAIN_DELAY.

1. Ensure that HGC is functioning correctly, such that analog and digital gain changes execute as expected.
2. Disable Transient Masking and ensure zero-cross timeout is set to 20 ms.
3. Set DIG_GAIN_DELAY to a few samples more than the latency of the currently selected filter. The latency can be read from the group-delay specification in the device datasheet. For example, for 96 kHz minimum-phase slow roll-off, the group delay is 4.1/Fs, corresponding to a latency of 4.1 samples. In this case, set DIG_GAIN_DELAY to an initial value of 0x4, corresponding to a delay of 7 samples.
4. Listen to the ADC output signal path, and configure an analog source signal that is easy to identify impulsive transients on, such as a sine tone or solo piano music.
5. Repeatedly trigger system gain changes that cause the analog gain to change (i.e. just switching either side of the ‘decision boundary’ for selecting between two adjacent analog gain steps), with a corresponding digital gain counter-step. Listen carefully for transients caused by the gain change.
6. Decrease DIG_GAIN_DELAY in single decrements while repeating Step 5, until the audibility of the gain change transients is minimized.

DIG_GAIN_DELAY can alternatively be tuned by capturing the ADC output in a DAW and visually examining it in a waveform view, using a visually predictable stimulus such as a sine tone (Figure 9). Misaligned digital gain changes are visible as glitches in the waveform.

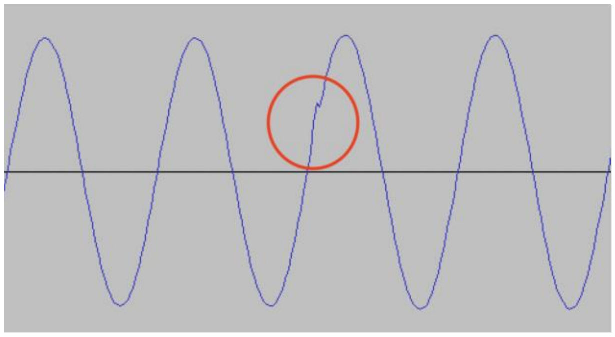

Figure 9: Waveform view of ADC output, indicating glitch caused by misaligned digital gain step

Their offset from the zero-crossing point can be measured, and used to modify the value of DIG_GAIN_DELAY to align the digital gain updates with the zero-crossing. The same glitch illustrated in Figure 9 is illustrated zoomed-in in Figure 10, showing that the digital gain step glitch is measured to be 4 samples after the zero-crossing. In this case, reducing the value of DIG_GAIN_DELAY by 4 correctly aligns the digital gain step with the zero-crossing.

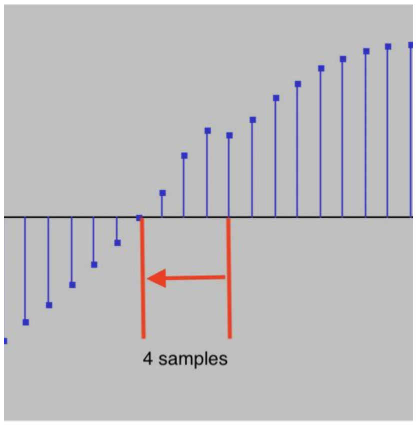

Figure 10: Zoomed-in view of misaligned digital gain step, showing it is measured to be 4 samples late

5.3 Transient-Masking Tuning

5.3.1 Transient-Masking Tuning Parameters

Transient Masking features a number of tunable parameters:

- TM_DELAY: similar to DIG_GAIN_DELAY, it is the number of samples from the zero-cross detection to the onset of sample-repeat transient masking. If the product that CS530x and CS4282P is being integrated-into supports multiple sample rate multipliers, each must be tuned separately.
- TM_HOLD_TIME: the number of samples for which the masking applies (i.e. the number of times the sample prior to the transient is repeated)
- TM_LD_THRESH: the measured average audio signal level below which Transient Masking is enabled.
- TM_LD_TIME: time constant of the exponential moving average level detector used with TM_LD_THRESH

Additionally, the level detector-based suppression of Transient Masking can be enabled/disabled per channel with the CHx_TM_LD_EN bits. Disabling level-based suppression of Transient Masking can be useful for tuning and debugging.

5.3.2 Transient-Masking Tuning Steps

To tune Transient Masking, the following steps are recommended:

1. Tune DIG_GAIN_DELAY as described in Section 5.2. Ensure that Transient Masking is disabled.
2. Using a sine wave signal source, configure signal source and monitoring levels such that charge injection transients are clearly audible over the sine tone.
3. Capture a waveform view of the ADC output in a DAW while triggering analog gain changes via HGC, and observe charge injection transients. Estimate the peak level of the charge injection transient in dBFS.
4. Enable Transient Masking, and disable Transient Masking Level Detector (CHx_TM_LD_EN) for the active channel. Set TM_DELAY to the same number of samples as DIG_GAIN_DELAY (note that different settings are required for the same delay, between these two fields). Set TM_HOLD_TIME to a large value such as 5 samples.
5. Capture more audio while triggering analog gain changes. The waveform view should show that the charge injection transient has been replaced by a horizontal portion of the waveform of length 5 samples.
6. Repeat Step 5, adjusting TM_DELAY until the masked (horizontal) portion of the waveform starts as shortly before the charge injection transient as possible.
7. Listening to the audio, reduce TM_HOLD_TIME until the loudness of the glitches is minimized.
8. Set the level detector threshold TM_LD_THRESH to the estimated peak level of the charge injection transients obtained in Step 3 and enable the level detector for the active channel.
9. Vary the signal source level and monitoring level. Confirm that transient masking only occurs at low signal levels where the charge injection transients are audible. Adjust TM_LD_THRESH to minimize audibility of both charge injection transients and Transient Masking over a wide audio level range, with a range of real-world program material.

It is important to note that while the charge injection transient amplitude remains approximately constant with gain, the audible effect of transient masking scales with gain. This means that as signal level increases significantly beyond the level of the charge injection transients, transient masking will become more audible than charge injection transients. The level detector threshold is used to prevent this – the threshold should be set for the signal amplitude at which charge injection transients and transient masking are equally audible with real program material.

Note that when using a sine tone, charge injection transients and Transient Masking itself will be much more audible than when using real-world program material such as music and speech.

5.3.3 Transient-Masking Tuning Waveforms

Tuning of Transient Masking delay and hold time is illustrated in the following waveform captures. Figure 11 shows a typical charge injection transient with a low-amplitude (around -50 dBFS) sine wave signal, and with the recommended minimum phase slow roll-off filter selected.

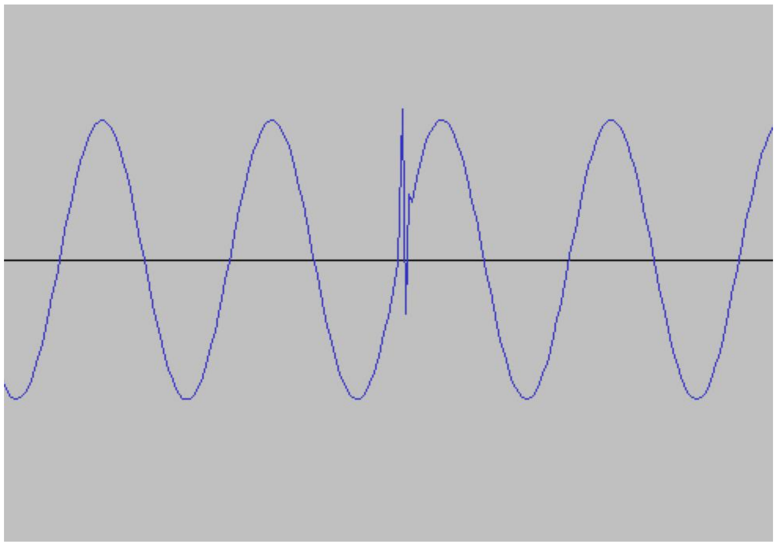

Figure 11: ADC output showing typical charge-injection transient with minimum-phase slow roll-off decimation filter

Figure 12 shows the same charge injection transient as Figure 11, zoomed-in on the time axis to show individual samples.

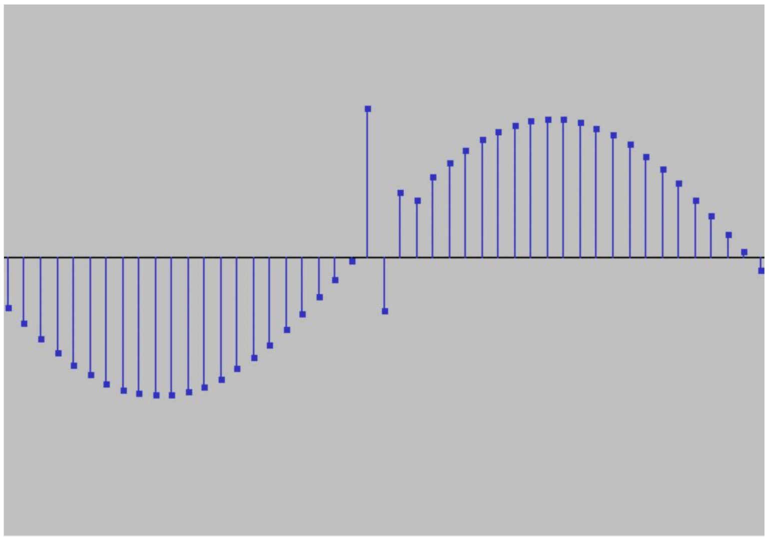

Figure 12: Zoomed-in view of ADC output illustrating same charge injection transient as Figure 11

Figure 13 illustrates misaligned Transient Masking which is occurring later than the charge injection transient. Transient Masking causes the current sample value to be repeated a number of times, creating a small flat portion of the waveform followed by a step to the current signal amplitude.

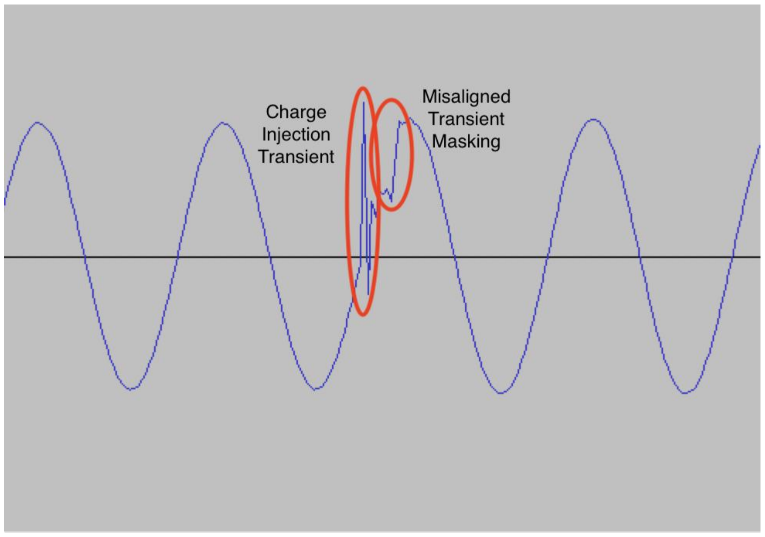

Figure 13: Misaligned Transient Masking

Figure 14 shows the same misaligned transient masking, zoomed-in on the time axis. It can be seen that the transient masking is starting about 5 samples after the zero-cross, which is immediately prior to the charge injection transient. This can be corrected by reducing the value of TM_DELAY by 5. It can also be seen that the transient masking duration is significantly longer than the charge injection transient, which has most of its energy in the first two samples.

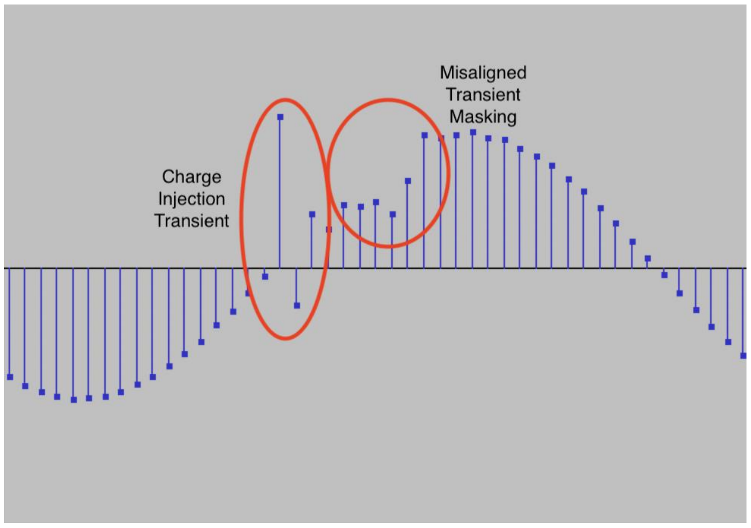

Figure 14: Misaligned Transient Masking (zoomed-in view)

Figure 15 shows the result of reducing TM_DELAY to correctly align the onset of transient masking with the zero-cross, and also reducing TM_HOLD_TIME to just two samples. This creates a signal glitch that is much smaller and much less audible than the charge injection transient being masked (compare with Figure 11).

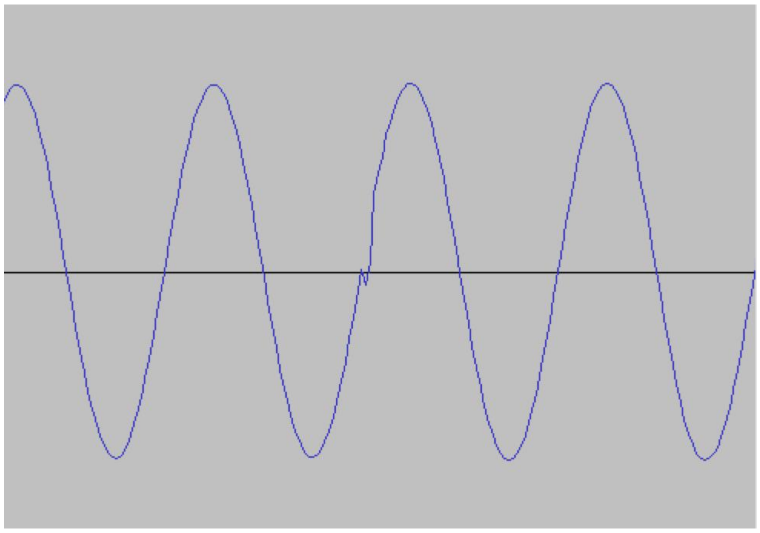

Figure 15: Well-aligned Transient Masking (compare with charge-injection transient in Figure 11)

Figure 16 shows the well-aligned transient masking, zoomed-in on the time axis.

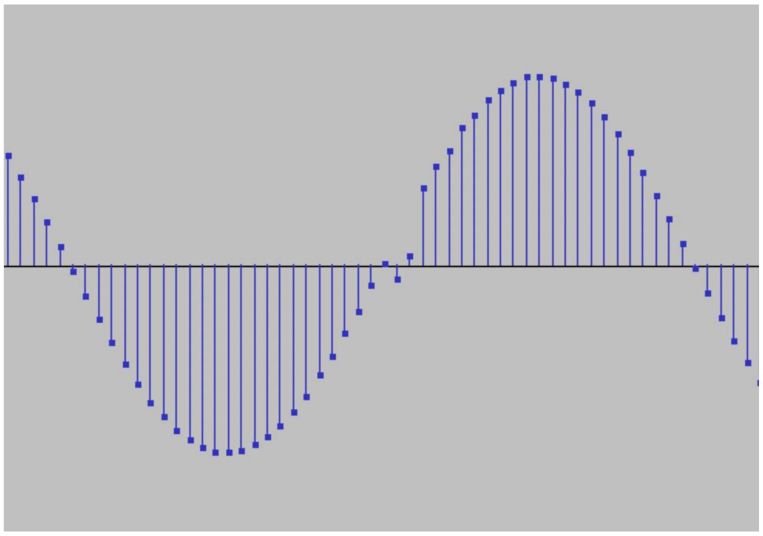

Figure 16: Well-aligned Transient Masking (zoomed-in view)

6 HGC Configuration and Control Example

This section describes an example application using HGC with a microphone pre-amp circuit for the CS530x or CS4282P devices. Note that the circuit described here is intended to demonstrate the HGC and is not a reference design for HGC hardware – see Section 3 for hardware design with HGC. The example described is for a 2-channel ADC device.

The circuit in Figure 17 shows the HGC control interface connected to two 8-bit shift registers (74HCS595) in a daisy-chain configuration. The two shift-register devices provide logic outputs for the IN1 and IN2 channels, respectively.

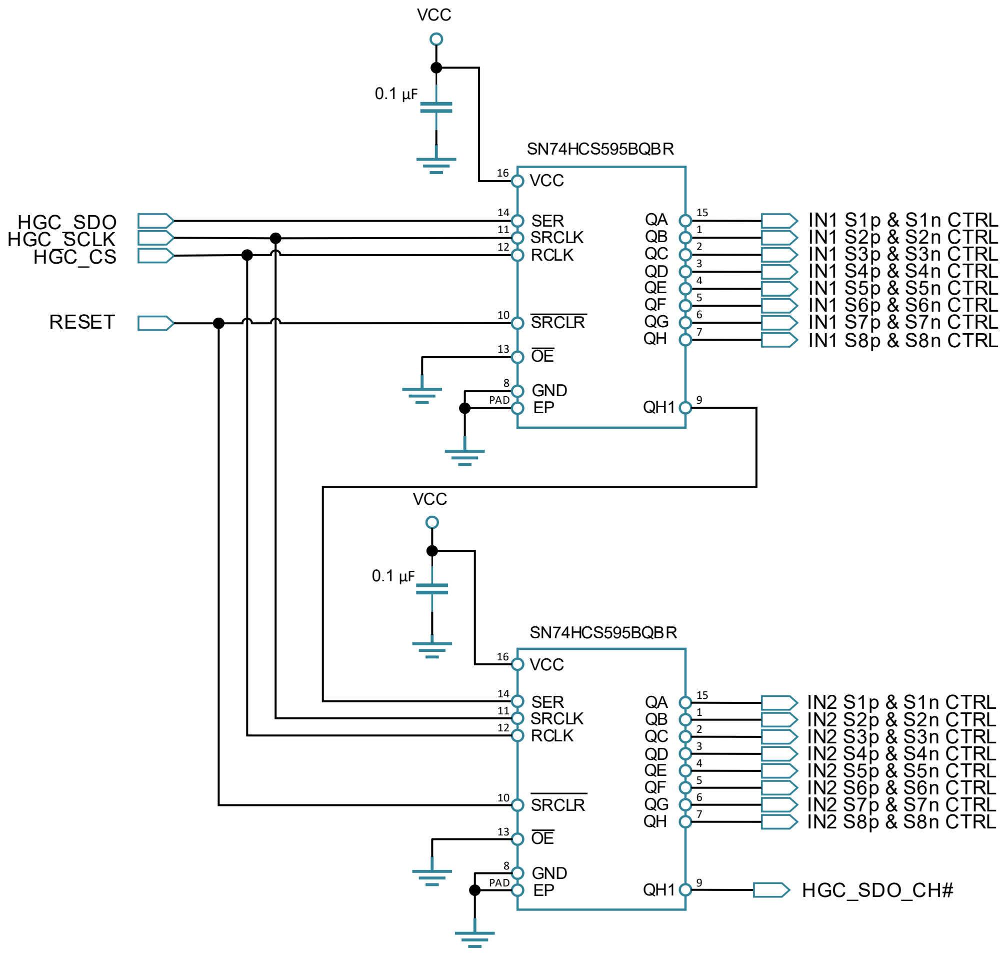

Figure 17: HGC Example Shift-Register Circuit

The circuit in Figure 18 shows a microphone pre-amp, with switched resistors providing configurable gain. The switched resistors are controlled by the shift-register logic outputs (see Figure 17). The circuit shown is implemented separately for the IN1 and IN2 channels.

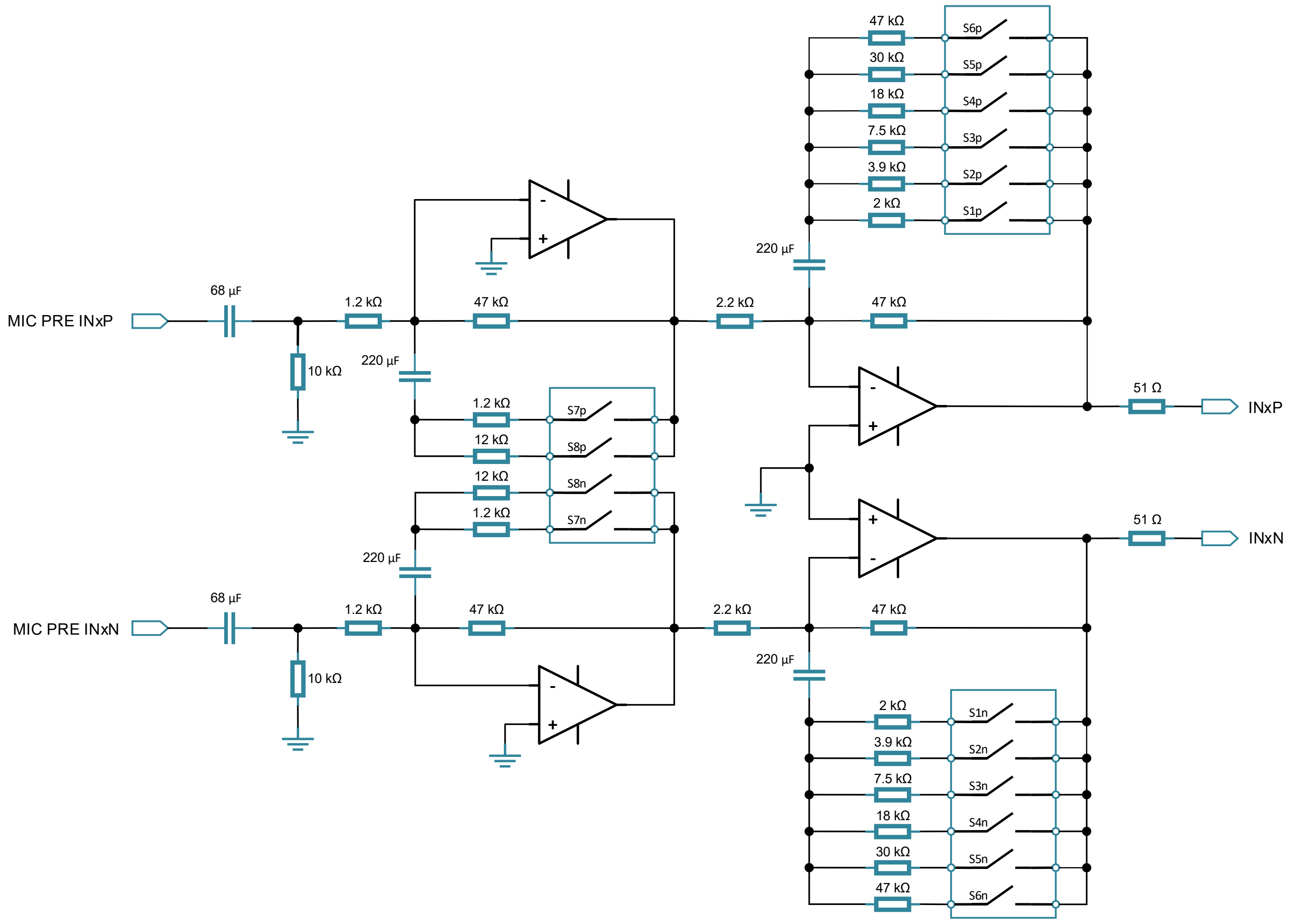

Figure 18: HGC Example Microphone Pre-Amp Circuit

The gain for the microphone pre-amp example circuit (Figure 18) is configured by closing the paired switches 1 to 8 in different combinations. The analog gain and associated control bit pattern are shown in Table 8.

Table 8 HGC MIC PRE-AMP Circuit Analog Gain and Control Bit Pattern

Analog Gain	Control Bit Pattern								CH#_BIT_PATT_1
(dB)	S8	S7	S6	S5	S4	S3	S2	S1	
0	0	1	0	0	1	0	0	1	0x4900
1.25	0	1	0	1	1	1	1	0	0x5E00
3	0	1	0	0	0	1	1	0	0x4600
4.875	0	1	0	0	1	0	1	0	0x4A00
6.375	0	1	0	0	0	0	1	0	0x4200
8.875	0	1	0	0	1	1	0	0	0x4C00
11.5	0	1	0	0	0	1	0	0	0x4400
14	1	0	0	0	1	0	0	1	0x8900
16.875	1	0	0	0	0	1	1	0	0x8600
18.875	1	0	0	0	1	0	1	0	0x8A00
20.375	1	0	0	0	0	0	1	0	0x8200
22.75	1	0	0	0	1	1	0	0	0x8C00
25.375	1	0	0	0	0	1	0	0	0x8400
27.875	0	0	0	0	1	0	0	1	0x0900
29.25	0	0	0	1	1	1	1	0	0x1E00
30.75	0	0	0	0	0	1	1	0	0x0600
32.625	0	0	0	0	1	0	1	0	0x0A00
34.25	0	0	0	0	0	0	1	0	0x0200
36.625	0	0	0	0	1	1	0	0	0x0C00
39.25	0	0	0	0	0	1	0	0	0x0400
42.25	0	0	0	1	1	0	0	0	0x1800
45.375	0	0	0	0	1	0	0	0	0x0800
48.25	0	0	0	1	0	0	0	0	0x1000
50.5	0	0	1	0	0	0	0	0	0x2000
56.5	0	0	0	0	0	0	0	0	0x0000

Note: The CH#_BIT_PATT_0 field value is 0x00 for the HGC Mic Pre-Amp example circuit.

6.1 HGC Configuration Writes

The following writes are performed on power-up, before enabling the ADC inputs. These register writes enable the HGC function and configure the combined analog and digital gain to 0 dB.

- Enable HGC function
  - **Reg 0x3D20 Data 0x1**
    - HGC_SPI_EN = 1 (Enable HGC function on CONFIG2, CONFIG3, and CONFIG4 pins)
- HGC Interface and Tuning settings
  - **Reg 0x201C Data 0x3**
    - CH2_TM_LD_EN = 1 (Enable transient-masking level-detect on Channel 1)
    - CH1_TM_LD_EN = 1 (Enable transient-masking level-detect on Channel 2)
  - **Reg 0x201E Data 0x90D**
    - TM_LD_THRESH = –56 dBFS (Transient masking is applied if the signal level is below the threshold)
    - TM_LD_TIME = 8192 samples (Configure the transient-masking level-detect time constant)
  - **Reg 0x2014 Data 0x3**
    - TM_DELAY = 0 FS samples (Set delay from analog gain update to start of transient-masking period)
    - DIG_GAIN_DELAY = 3 FS samples (Set delay from analog gain update to digital gain update)
  - **Reg 0x2018 Data 0x201**
    - TM_HOLD_TIME = 2 FS samples (Transient masking hold time)
    - TM_EN = 1 (Transient masking enable)
  - **Reg 0x2010 Data 0x17**
    - STEP_RAMP_ENABLE = 1 (Enables the digital gain to compensate for step changes in the analog gain)
    - ZC_TIMEOUT = 20 ms (Time out for Zero Cross detection)
  - **Reg 0x2020 Data 0x0**
    - SCK_DIV = Divide by 2
  - **Reg 0x2022 Data 0x944**
    - CS_IDLE_DUR = 128 clock cycles
    - CS_RISE_DELAY = 10 clock cycles
    - CS_FALL_DELAY = 11 clock cycles
- Bit pattern length for each channel
  - **Reg 0x2030 Data 0x8**
    - CH1_BIT_PATT_LENGTH = 8 bits
  - **Reg 0x2034 Data 0x8**
    - CH2_BIT_PATT_LENGTH = 8 bits
- Set CH1 path gain to 0 dB
  - **Reg 0x2062 Data 0x4900**
    - CH1_BIT_PATT_1 = 0x4900 (Switches S1, S4 & S7 closed)
  - **Reg 0x2064 Data 0x0000**
    - CH1_ANA_GAIN = 0 dB (The value must match the external analog gain of the SPI bit pattern)
  - **Reg 0x2066 Data 0x0001**
    - CH1_UPDATE = 1 (Apply the gain and SPI bit pattern at the next scheduling opportunity, zero-cross aligned)
    - CH1_DIG_GAIN = 0 dB (Channel 1 digital gain)
- Set CH2 path gain to 0 dB
  - **Reg 0x206A Data 0x4900**
    - CH2_BIT_PATT_1 = 0x4900 (Switches S1, S4 & S7 closed)
  - **Reg 0x206C Data 0x0000**
    - CH2_ANA_GAIN = 0 dB (The value must match the external analog gain of the SPI bit pattern.)
  - **Reg 0x206E Data 0x0001**
    - CH2_UPDATE = 1 (Apply the gain and SPI bit pattern at the next scheduling opportunity, zero-cross aligned)
    - CH2_DIG_GAIN = 0 dB (Channel 2 digital gain)

6.2 HGC +20 dB Gain Control Writes

The following writes configure the combined analog and digital gain to +20 dB. This is implemented using analog gain of 18.785 dB, and digital gain of 1.125 dB.

- Set CH1 path gain to 20 dB
  - **Reg 0x2062 Data 0x8A00**
    - CH1_BIT_PATT_1 = 0x8A00 (Switches S2, S4 & S8 closed)
  - **Reg 0x2064 Data 0x0097**
    - CH1_ANA_GAIN = 18.875 dB (The value must match the external analog gain of the SPI bit pattern)
  - **Reg 0x2066 Data 0x8009**
    - CH1_UPDATE = 1 (Apply the gain and SPI bit pattern at the next scheduling opportunity, zero-cross aligned)
      CH1_DIG_GAIN = 1.125 dB (Channel 1 digital gain)
- Set CH2 path gain to 20dB
  - **Reg 0x206A Data 0x8A00**
    - CH2_BIT_PATT_1 = 0x8A00 (Switches S2, S4 & S8 closed)
  - **Reg 0x206C Data 0x0097**
    - CH2_ANA_GAIN = 18.875 dB (The value must match the external analog gain of the SPI bit pattern)
  - **Reg 0x206E Data 0x8009**
    - CH2_UPDATE = 1 (Apply the gain and SPI bit pattern at the next scheduling opportunity, zero-cross aligned)
      CH2_DIG_GAIN = 1.125 dB (Channel 2 digital gain)

6.3 HGC +12 dB Gain Control Writes

The following writes configure the combined analog and digital gain to +12 dB. This is implemented using analog gain of 11.5 dB, and digital gain of 0.5 dB.

- Set CH1 path gain to 12 dB
  - **Reg 0x2062 Data 0x4400**
    - CH1_BIT_PATT_1 = 0x4400 (Switches S3 & S7 closed)
  - **Reg 0x2064 Data 0x005C**
    - CH1_ANA_GAIN = 11.5 dB (The value must match the external analog gain of the SPI bit pattern)
  - **Reg 0x2066 Data 0x8004**
    - CH1_UPDATE = 1 (Apply the gain and SPI bit pattern at the next scheduling opportunity, zero-cross aligned)
    - CH1_DIG_GAIN = 0.5 dB (Channel 1 digital gain)
- Set CH2 path gain to 12 dB
  - **Reg 0x206A Data 0x4400**
    - CH2_BIT_PATT_1 = 0x4400 (Switches S3 & S7 closed)
  - **Reg 0x206C Data 0x005C**
    - CH2_ANA_GAIN = 11.5 dB (The value must match the external analog gain of the SPI bit pattern)
  - **Reg 0x206E Data 0x8004**
    - CH2_UPDATE = 1 (Apply the gain and SPI bit pattern at the next scheduling opportunity, zero-cross aligned)
    - CH2_DIG_GAIN = 0.5 dB (Channel 2 digital gain)

7 Conclusion

Hybrid Gain Control simplifies the implementation of digitally controlled mic preamp gain control, where the user interface emulates a continuous gain control and the mic preamp analog gain is stepped. It incorporates analog zero-cross detection and digital interpolation gain.

It assures precise alignment of the digital interpolation gain step to the zero-cross-aligned analog gain step change, for smooth, glitch-free gain changes.

Transient Masking reduces the audibility of charge injection transients caused by the switching action of multiplexers used to change gain of the mic preamp, especially when the signal level is low such that charge injection transients become audible. In particular, it reduces zipper noise when the gain user interface control is swept with little or no signal present. It can also allow use of lower-distortion multiplexers that have higher charge injection.

HGC is designed to be easy to control from existing software/firmware architectures that implement control of switched analog gain and digital interpolation gain and may be used with any switched-gain mic preamp design that can be controlled via daisy-chain SPI. HGC also provides control of up to 64 additional general-purpose logic outputs to control auxiliary functions associated with the front-end audio signal path.

8 Revision History

Revision History

Revision	Changes
R1 AUG 2023	• Initial version.
R2 MAR 2026	• HGC register name and field name updates • Included other devices that support HGC (CS4282P, CS5304S, and CS5308S) • Added example configuration and control information

Contacting Cirrus Logic Support
For all product questions and inquiries, contact a Cirrus Logic Sales Representative.
To find the one nearest you, go to www.cirrus.com.