*Application Note*

# *Calculating Conversion Latency and System Cycle Time for Delta-Sigma ADCs*

*Bryan Lizon*

**ABSTRACT**

Many precision data acquisition applications are constrained by a system cycle time, or the total time required to perform one complete loop through all measurement channels. However, it is not always clear how this metric correlates to the information provided in a delta-sigma analog-to-digital converter (ADC) data sheet, which may include multiple features and modes that affect the rate at which the device outputs data. To provide a deeper understanding of how to calculate the system cycle time using the ADC data sheet information, this application note offers a detailed discussion of the most important factors that affect ADC conversion latency. Additionally, key takeaways are summarized and several examples using actual ADCs are provided.

## Table of Contents



## Trademarks

# 1 Introduction

One important consideration when choosing a delta-sigma analog-to-digital converter (ADC) for a specific application is the system cycle time, or the total time required to perform one complete loop through all measurement channels. This cycle time could require reading a single conversion result from a single channel, a single conversion result from multiple channels, or multiple conversion results from multiple channels. Additionally, individual conversion results can require multiple conversion periods to generate settled data. How does a design engineer use the information in the ADC data sheet to select a device that meets the cycle time requirements?

For example, if the cycle time requires retrieving three conversion results per channel on six channels in ten milliseconds, is it sufficient to choose an ADC that samples at 1800 SPS?

(3 conversions / channel) · (6 channels) · (1 / 10 ms) = 1800 SPS

Answering these questions requires a comprehensive understanding of how multiplexed delta-sigma ADCs sample and process data. To that end, this application note explores multiplexed delta-sigma ADC operation in detail, breaking up this broad topic into several subsections:

- Data sheet timing and nomenclature
- What causes conversion latency in a delta-sigma ADC?
- Digital filter operation and behavior
- First conversion versus second and subsequent conversion latency
- Conversion mode
- Programmable delay
- ADC overhead time
- Clock frequency
- Chopping
- Analog settling

Two additional sections at the end of this document summarize important takeaways and provide several examples that illustrate how to apply this information to a practical system.

Before introducing the first topic, note that external factors can impact delta-sigma ADC operation and timing that in turn can have an effect on cycle time. This includes but is not limited to unstable power supplies, low-accuracy clocks, and amplifier overload. This document does not discuss the impact external factors can have on calculating cycle time and assumes an ideal system, unless otherwise noted.

## 2 Data Sheet Timing and Nomenclature

The first step toward understanding delta-sigma ADC timing and operation is to define a common vocabulary that describes this behavior. Table 2-1 identifies five key parameters that serve as the foundation for the rest of this document.

**Table 2-1. Important Delta-Sigma ADC Timing and Operation Parameters .**

| PARAMETER | DEFINITION |
|---|---|
| Conversion period | The time during which the delta-sigma modulator samples the analog input, filters the data, and decimates the output |
| Conversion latency | The time the ADC takes to produce settled output data:<br>• Assumes a settled analog input<br>• Can span a single conversion period or multiple conversion periods<br>• Can include processing time such as ADC overhead or programmable delay |
| Conversion result | Data retrieved by the user after the ADC indicates new results are ready:<br>• Can be a combination of multiple conversions<br>• Can be settled or unsettled |
| Channel scan time | The time to generate the desired number of conversion results for a given channel – can be equal to conversion latency if the system only requires one conversion result per channel |
| System cycle time | The total time required to perform one complete loop through all measurement channels – can be equal to channel scan time if the system only measures one channel |

A delta-sigma ADC data sheet uses the information in Table 2-1 to describe ADC timing behavior in several different ways. One such way that an ADC data sheet communicates timing behavior is using a conversion latency table. This information is important for those devices that have multiple filter types and output data rates (ODRs). For example, Table 2-2 shows the conversion latency in milliseconds for the 24-bit, 40-kSPS, 10-channel ADS1261.

**Table 2-2. ADS1261 Conversion Latency (ms)[1]**

| ODR (SPS) | FIR | SINC1 | SINC2 | SINC3 | SINC4 | SINC5 |
|---|---|---|---|---|---|---|
| 2.5 | 402.2 | 400.4 | 800.4 | 1200 | 1600 | — |
| 5 | 202.2 | 200.4 | 400.4 | 600.4 | 800.4 | — |
| 10 | 102.2 | 100.4 | 200.4 | 300.4 | 400.4 | — |
| 16.6 | — | 60.43 | 120.4 | 180.4 | 240.4 | — |
| 20 | 52.23 | 50.43 | 100.4 | 150.4 | 200.4 | — |
| 50 | — | 20.43 | 40.43 | 60.43 | 80.43 | — |
| 60 | — | 17.09 | 33.76 | 50.43 | 67.09 | — |
| 100 | — | 10.43 | 20.43 | 30.43 | 40.43 | — |
| 400 | — | 2.925 | 5.425 | 7.925 | 10.43 | — |
| 1200 | — | 1.258 | 2.091 | 2.925 | 3.758 | — |
| 2400 | — | 0.841 | 1.258 | 1.675 | 2.091 | — |
| 4800 | — | 0.633 | 0.841 | 1.05 | 1.258 | — |
| 7200 | — | 0.564 | 0.702 | 0.841 | 0.98 | — |
| 14400 | — | — | — | — | — | 0.423 |
| 19200 | — | — | — | — | — | 0.336 |
| 25600 | — | — | — | — | — | 0.271 |
| 40000 | — | — | — | — | — | 0.179 |

(1) Chop mode off, conversion-start delay = 50µs (DELAY[3:0] = 0001)

Table 2-2 provides a conversion latency value for each combination of ODR and filter type using the ADS1261. In this particular case, the ADS1261 conversion latency time is specified using the nominal clock frequency as well as chop mode turned off and a programmable delay of 50µs, as per the table note for Table 2-2. Other ADCs might specify the conversion latency time using different parameters, different conditions, or even

a different format, but the same general information is always provided. This application note delves into the details of these tables to identify the factors contributing to conversion latency and how this relates to overall cycle time.

Additionally, ADC data sheets often use timing diagrams to help visualize the general timing behavior. Figure 2-1 shows an example timing diagram for a typical multiplexed delta-sigma ADC. This timing diagram shows how different conversion results can be comprised of a different number of conversion periods ($t_{CP}$) and conversion latencies ($t_{CL}$), as well as include fixed timing parameters (delay and overhead). The channel scan time ($t_{CH}$) and system cycle time ($t_{CYCLE}$) are also shown. In this example, $t_{CH} = t_{CYCLE}$ because only one channel is measured.

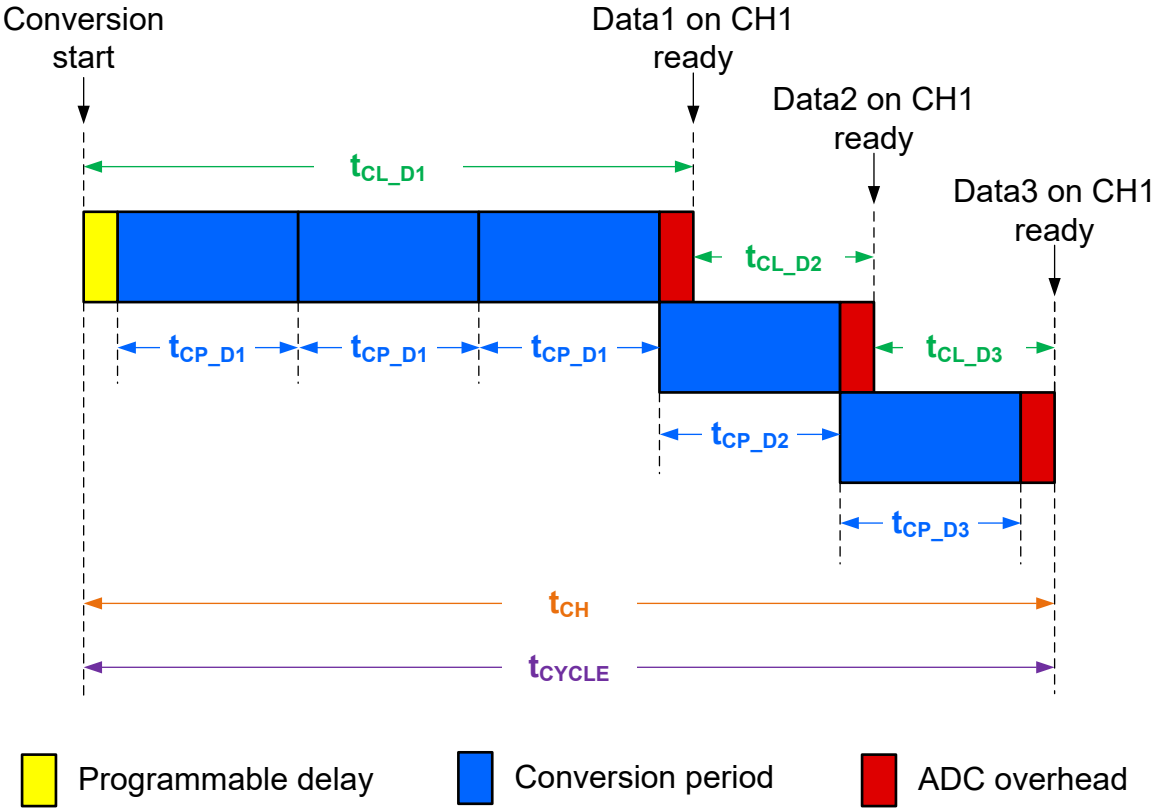



**Figure 2-1. Generic Delta-Sigma ADC Timing Diagram**

Many of the timing parameters defined in Table 2-1 are also shown in Figure 2-1. Furthermore, this application note uses diagrams similar to Figure 2-1 to help visualize how each ADC timing component affects the conversion period, conversion latency, channel scan time, or system cycle time.

Finally, additional ADC features or even some external factors can impact the device timing behavior. These features are often described in separate data sheet sections, making it challenging to determine how each contributes to the overall latency for a specific device. This application note organizes this information into one document to provide a more complete view of ADC operation and how this affects timing.

The rest of this document explores these three components of a delta-sigma ADC data sheet in detail to provide a comprehensive understanding of how multiplexed delta-sigma ADCs sample and process data, and how this contributes to conversion latency and overall system cycle time.

## 3 What Causes Conversion Latency in a Delta-Sigma ADC?

A basic introduction to how a delta-sigma ADC works is helpful to understand why this data converter architecture inherently results in conversion latency. When a conversion start is triggered, the delta-sigma modulator continuously oversamples the analog input using a high-frequency clock, $f_{MOD}$. The modulator outputs a digital bitstream at $f_{MOD}$ that has a ones density proportional to the input signal: the modulator outputs all zeros when the input is at negative full-scale (–FS), all ones when the input is at positive full-scale (+FS), and a proportional number of ones and zeros between these two extremes. Figure 3-1 shows how the analog input signal in black is applied to the delta-sigma modulator, which generates the digital bitstream in green using the high-frequency modulator clock in red.

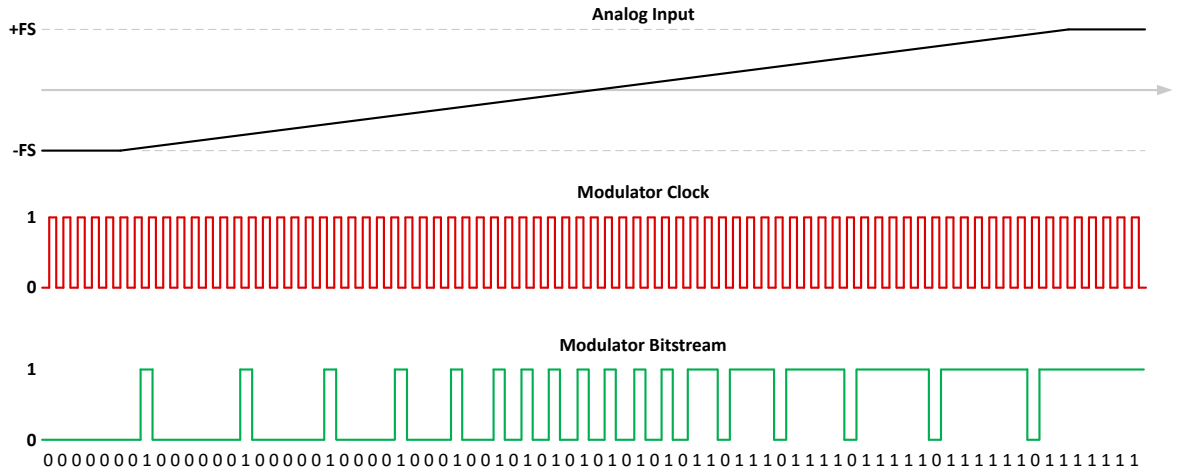



**Figure 3-1. Delta-Sigma Modulator Bitstream Output for an Arbitrary Analog Input**

As each bit of the green modulator bitstream in Figure 3-1 is generated, it propagates through the digital filter to be averaged and decimated. After a well-defined number of clock cycles, a high-resolution output is produced. Figure 3-2 generalizes delta-sigma ADC digital filter behavior using a simplified model.

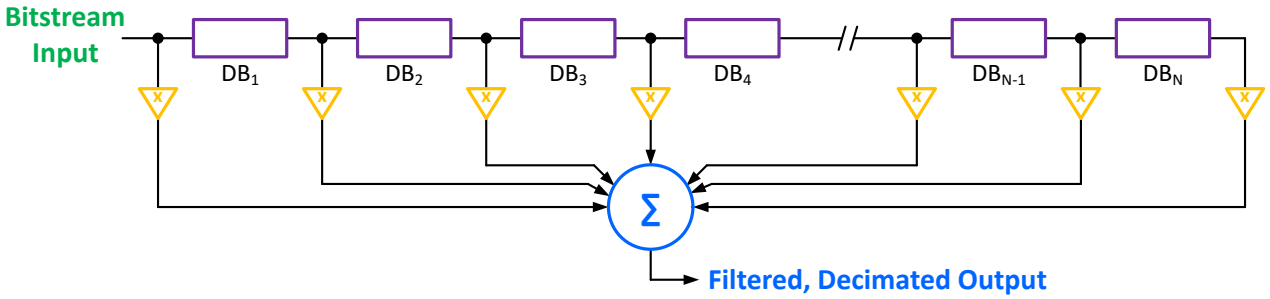



**Figure 3-2. Simplified Digital Filter Model**

The model in Figure 3-2 has N number of stages that are each comprised of one delay block ($DB_X$) in purple and one multiplier in orange, while the summing junction in blue aggregates the information in all stages to produce the filtered, decimated output. As each bit of the green bitstream enters the filter, it progresses through the delay blocks one modulator clock period ($t_{MOD} = 1 / f_{MOD}$) at a time. The digital filter only produces the filtered, decimated output in blue when the first bit in the sequence reaches the last delay block. Assuming that the ADC is sampling continuously, this sequence restarts during the subsequent $t_{MOD}$ period and generates the next output after N number of $t_{MOD}$ periods have elapsed. Therefore, a digital filter with N number of delay blocks and a decimation ratio of N has a conversion period, $t_{CP}$, given by Equation 1:

$$t_{CP} = N \cdot t_{MOD} \quad (1)$$

The variable N in Equation 1 and Figure 3-2 is often referred to as the oversampling ratio (OSR). The OSR determines how many samples are averaged together during one conversion period. As Equation 1 implies, the

larger the value of N (OSR), the longer it takes to generate the output. However, a larger value of N (OSR) generally leads to lower noise because of additional averaging.

As an example of this behavior, Figure 3-3 shows how the red modulator clock and green bitstream from Figure 3-1 are applied to the digital filter model in Figure 3-2. In this example, the digital filter has four delay blocks (N = 4) and the first four bits of the bitstream have an arbitrary value of 1011b.

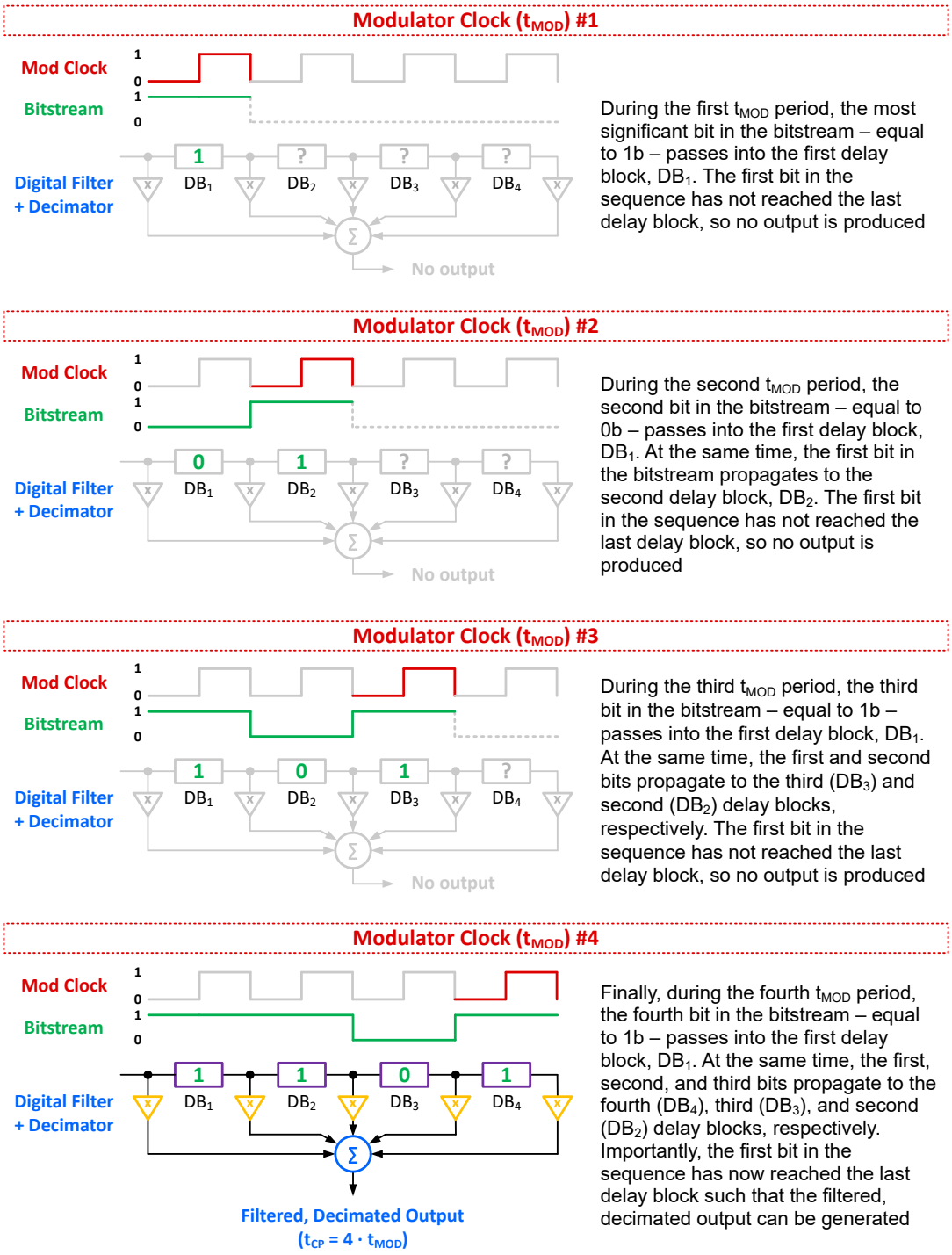



**Figure 3-3. Modulator Bitstream Propagating Through a Digital Filter When N = 4**

As shown during the fourth modulator period in Figure 3-3, $t_{CP} = 4 \cdot t_{MOD}$, which is the time it takes for the bitstream to reach the last delay block in this example. Therefore, the number of digital filter delay blocks is the dominant factor causing delta-sigma ADC conversion latency. The next section expands on this topic by

using the digital filter model in Figure 3-2 to analyze the different types of low-latency filters commonly used in delta-sigma ADCs and how they respond to an analog input signal.

Further discussion of delta-sigma ADC modulator operation and digital filter design are beyond the scope of this article. See the *Digital Filter Types in Delta-Sigma ADCs* application note and TI's Precision Labs ADC content for more information.

## 4 Digital Filter Operation and Behavior

The most common types of digital filters used in delta-sigma ADCs are finite impulse response (FIR) filters such as sinc and wideband. This document focuses on the operation of sinc filters because they typically take five or fewer conversion periods to settle, resulting in low latency. Comparatively, wideband filters can initially take dozens of conversion periods to settle, making them impractical for many multiplexed applications. However, the same general timing and operation principles can be applied to an ADC with a wideband filter.

In terms of the delay from bitstream input to digital output, the simplified digital filter model introduced in the previous section is effectively a first-order sinc (sinc$^1$) filter. Higher-order sinc filters can be approximated as multiple sinc$^1$ filters in series. For example, if a sinc$^1$ filter has N number of delay blocks, a third-order sinc (sinc$^3$) filter has 3 · N delay blocks. Figure 4-1 shows how the digital filter model can be modified for a sinc$^3$ filter, where each of the three stages (S*x*) is comprised of N number of delay blocks.

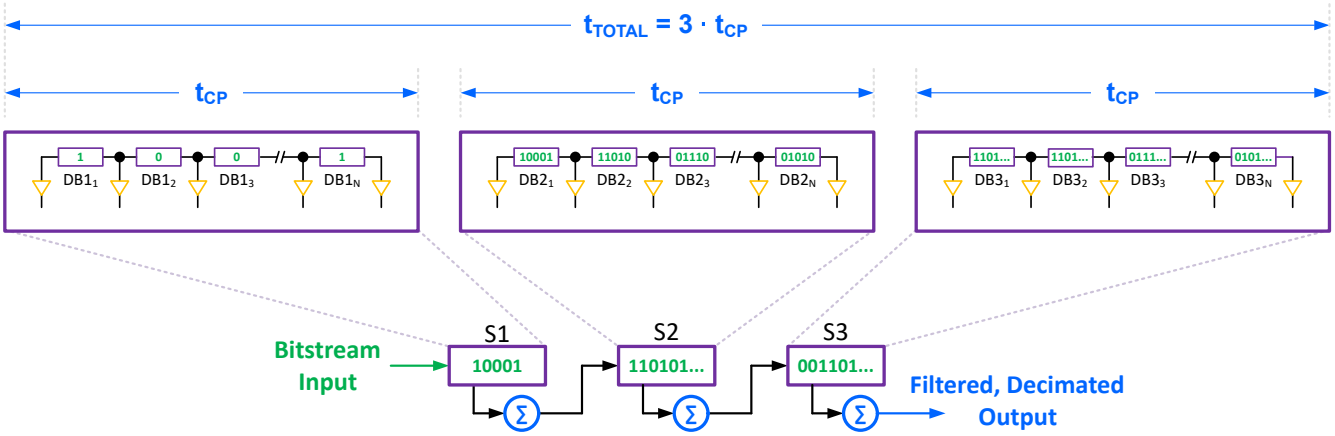



**Figure 4-1. Simplified Sinc$^3$ Digital Filter Model**

The most important result from the simplified sinc$^3$ model in Figure 4-1 is that it takes one conversion period (1 · $t_{CP}$) for the bitstream to reach the end of each stage (S1, S2, or S3), where $t_{CP}$ = N · $t_{MOD}$. The total delay for the bitstream to reach the end of S3 and calculate the filtered, decimated output is $t_{TOTAL}$ = 3 · $t_{CP}$. After this initial output is produced however, higher-order sinc filters can output filtered, decimated data after each conversion period (1 · $t_{CP}$) under certain conditions (described throughout this document). This behavior is possible because the modulator sampling and digital filtering process effectively averages out transient information in the analog input. Therefore, it is typically a good assumption that the digital filter data during any X consecutive conversion periods will be similar enough to generate settled data under most conditions, where X is the sinc filter order. Operating under this assumption enables the noise reduction of a higher-order filter while avoiding the additional multi-conversion period delay required by the first output.

However, this assumption can lead to unsettled conversion results if the analog input does in fact change significantly during the conversion process. For example, Figure 4-2 illustrates how a sinc$^1$, a sinc$^3$, and a fifth-order (sinc$^5$) filter respond when a step input is applied to the ADC after conversion period N ($t_{CP(N)}$) completes.

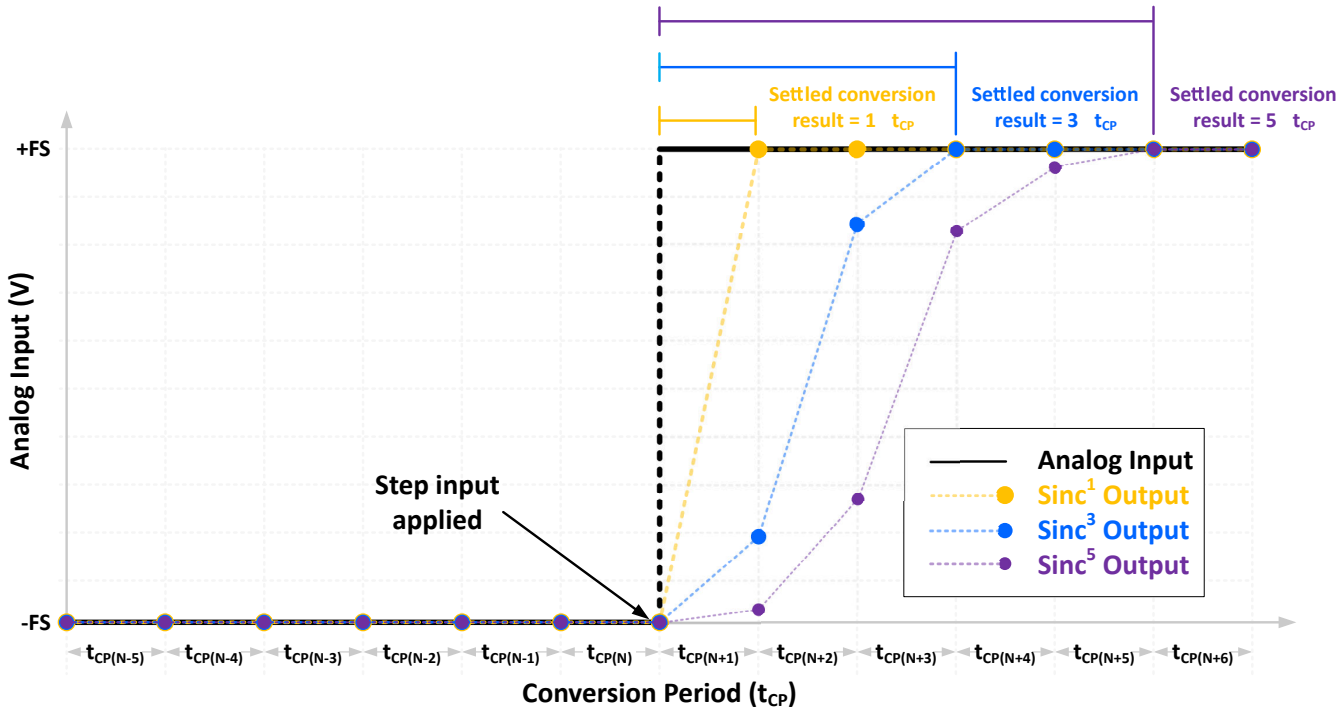



**Figure 4-2. Sinc Filter Response to a Step Input**

**Note**

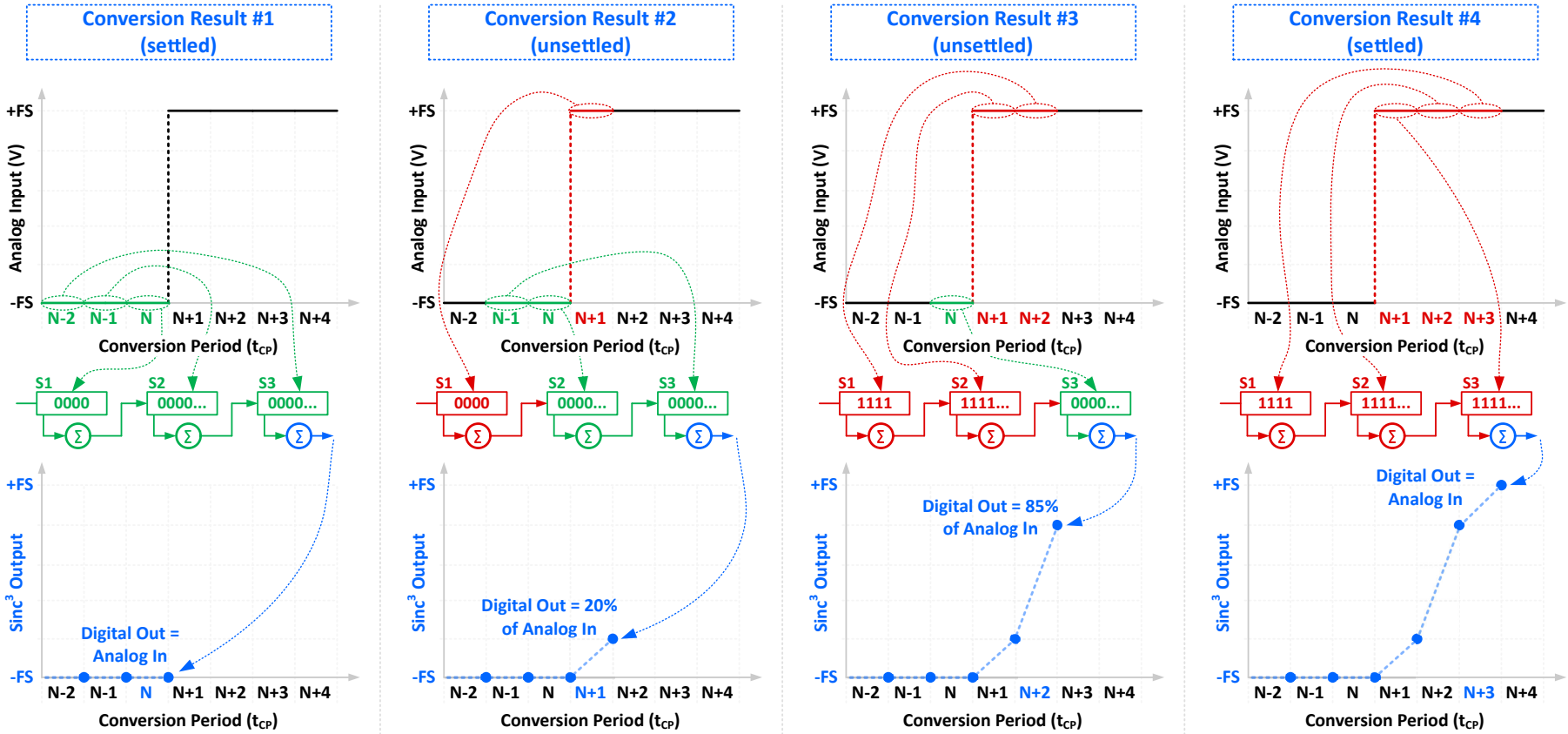
All of the conversion periods are equal in the example system shown in Figure 4-2, and the term "N±*x*" is only used to indicate the conversion period index. For the rest of this document, only the index is used to denote a specific conversion period, such as "N+1" to mean conversion period $t_{CP(N+1)}$.

In Figure 4-2, a –FS input is applied to the selected ADC channel during N-5 through N. During this time, each sinc filter outputs a settled conversion result after each conversion period. However, a +FS step input is applied to the same ADC channel between N and N+1. While this change occurs virtually instantaneously in the analog domain (assuming no analog settling time is required), settled output data is delayed leading to increased conversion latency, $t_{CL}$:

- The orange sinc$^1$ filter has $t_{CL} \cong 1 \cdot t_{CP}$
- The blue sinc$^3$ filter has $t_{CL} \cong 3 \cdot t_{CP}$
- The purple sinc$^5$ filter has $t_{CL} \cong 5 \cdot t_{CP}$

Note that $t_{CL}$ in the previous list is approximate because a settled conversion result can require additional processing time or user-defined delays, as per Table 2-1.

To better understand why settled data is delayed, the simplified sinc$^3$ digital filter model from Figure 4-1 and the blue sinc$^3$ filter response from Figure 4-2 are combined in Figure 4-3 to demonstrate how the analog input propagates through each sinc$^3$ filter stage during conversion periods N-2 through N+3.




Conversion result #1 is the filtered, decimated bitstream from all three filter stages (S1, S2, and S3) during conversion periods N-2, N-1, and N. Since the analog input is at -FS during these three conversion periods – shown in green – the combined information from each filter stage is all 0s. This produces settled data after conversion period N completes. The user can retrieve this settled data (conversion result #1) when the ADC indicates new data is ready. This conversion result is equal to the blue digital output at the end of conversion period N, which correlates very well to the black analog input plot at this time.

For conversion result #2, a +FS step input is applied immediately after conversion period N completes. The ADC modulator samples this input and outputs all 1s to S1, as shown in red. However, S2 and S3 in green still contain information from when the analog input was at -FS. When conversion result #2 is ready to be retrieved, this output contains a combination of data from when the analog input was at -FS (green) and +FS (red). This results in unsettled data at the end of conversion period N+1 that corresponds to approximately 20% of the actual analog input.

Similarly, conversion result #3 is a combination of filtered data corresponding to sampling the input signal when it was at -FS (green) and +FS (red). This still yields an unsettled conversion result at the end of conversion period N+2, though the digital output is now much closer to the true input signal, at approximately 85%.

Finally, conversion result #4 at the end of conversion period N+3 is a settled output because all three filter stages contain information from when the modulator was sampling the +FS input signal, highlighted in red.

**Figure 4-3. Analog Input Propagating Through a Sinc$^3$ Filter to Generate Conversion Results**

The results from Figure 4-2 and Figure 4-3 yield an approximation for the conversion latency, $t_{CL}$, as per Equation 2:

$$t_{CL} \approx x / ODR \quad (2)$$

where

- x is the sinc filter order, which is the x in the notation $sinc^x$

This relationship is implied in the ADS1261 conversion latency table introduced in Section 2. For example, the conversion latency for the 20 SPS ODR is 50.43 ms using a $sinc^1$ filter, which is approximately 1 / ODR. Comparatively, the latency is 150.4 ms using the $sinc^3$ filter, which is approximately 3 / ODR. However, additional factors such as the ADC overhead time or programmable delay can skew this relationship. These factors are discussed in more detail throughout this document.

The digital filter behavior in response to a step input can also be seen in a real ADC. Figure 4-4 recreates an image from the ADS1261 data sheet that reveals how the data ready ($\overline{DRDY}$) pin and $sinc^3$ output (in blue) respond when a step input (in black) is applied to a single input channel.

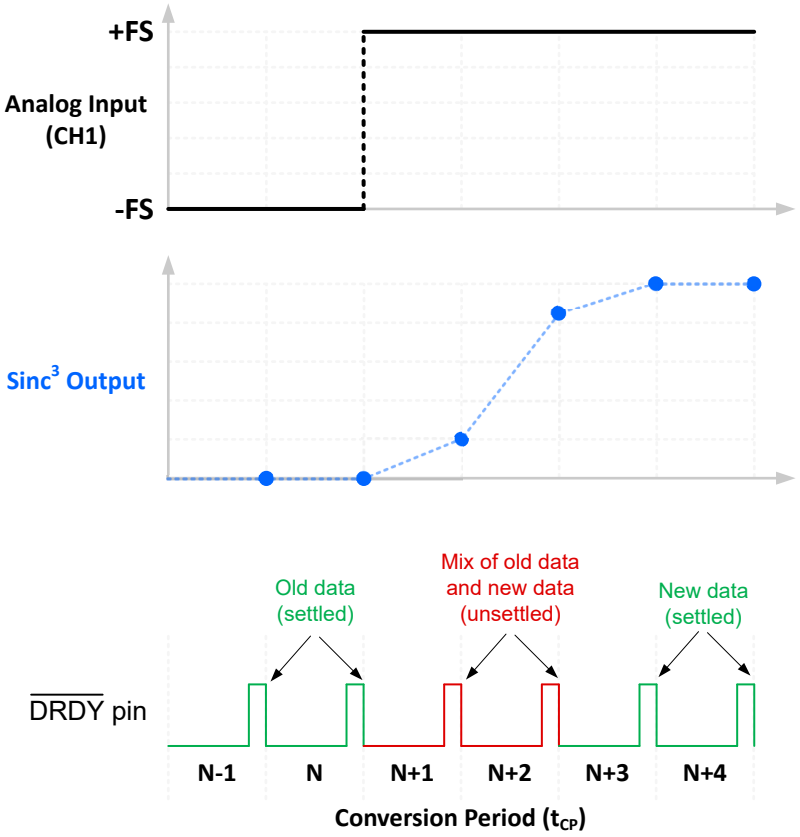



**Figure 4-4. ADS1261 $\overline{DRDY}$ Pin Behavior When a Step Input Occurs**

The important takeaway from Figure 4-4 is that the two conversion results produced immediately after the step input (shown in red) are a mix of old data and new data. However, $\overline{DRDY}$ still transitions high-to-low to indicate new conversion results are ready even though these are comprised of *unsettled* data. In other words, the ADC does not *detect* when a step input is applied to the selected channel. Instead, the delta-sigma modulator continues to sample the input and the digital filter processes this information regardless of significant changes in the analog signal. As Figure 4-4 shows, the ADS1261 requires a number of additional $\overline{DRDY}$ transitions to produce settled conversion results (in green), which depends on the selected filter type. Ultimately, the user must manually identify a step input and then ignore subsequent $\overline{DRDY}$ transitions until a settled conversion result is available.

It is also important to consider if a step change occurs *during* the conversion process, which can lead to additional conversion latency. Figure 4-5 shows a step input occurring immediately *before* conversion period N+1 on a single channel (CH1). Figure 4-6 shows the same response with the input step occurring *during* conversion period N+1.

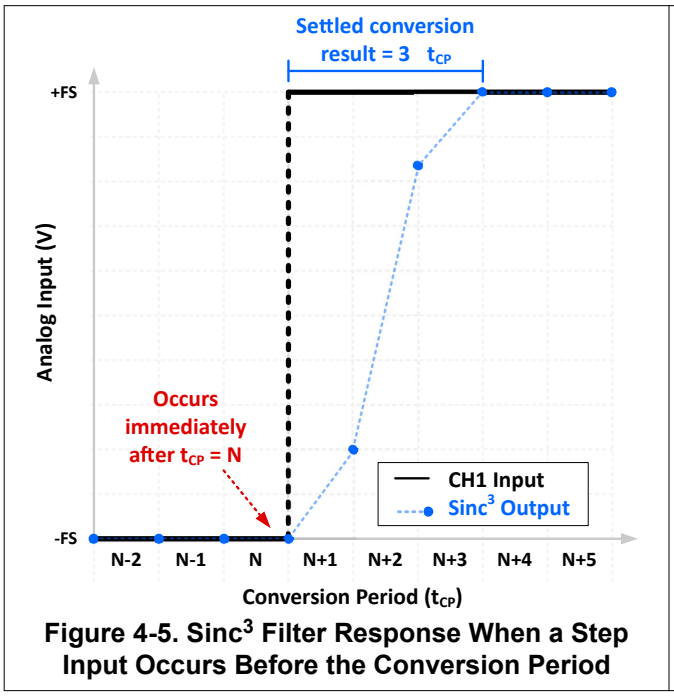



**Figure 4-5. Sinc$^3$ Filter Response When a Step Input Occurs Before the Conversion Period**

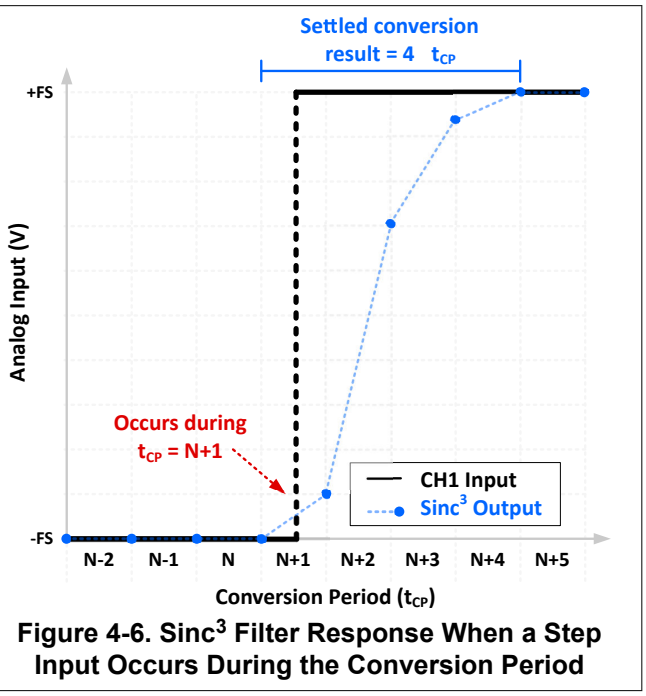



**Figure 4-6. Sinc$^3$ Filter Response When a Step Input Occurs During the Conversion Period**

As Figure 4-6 shows, a sinc$^3$ filter actually requires four conversion periods to produce a settled conversion result when the step input occurs during the conversion process. This additional delay occurs because the first digital filter stage contains sampled data from when the analog input was both –FS and +FS. This information is effectively useless for accurately recreating the input signal and needs to completely clear out of the digital filter before settled conversion results are available. This requires three complete conversion periods for a sinc$^3$ filter, with settled data available at the end of the fourth conversion period.

Equation 3 applies this consideration to Equation 2 and provides an approximation for $t_{CL}$ when the analog input changes significantly during the conversion process:

$$t_{CL} \approx (x + 1) / ODR \quad (3)$$

where

- x is the sinc filter order, which is the *x* in the notation *sinc$^x$*

To avoid reading unsettled conversion results as well as increasing conversion latency, ensure that the input signal has settled to its final value before starting the conversion process.

## 4.1 Unsettled Data Due to an ADC Operation

An ADC operation such as a multiplexer change or a conversion start is similar to an applied step input and its effect on the digital filter. For example, changing from a channel with a –FS input to a channel with a +FS input mimics the step voltage that is applied immediately after conversion period N completes in Figure 4-2. When this occurs, does the user need to identify this action and manually discard these conversion results until settled data is available, similar to a step input?

Fortunately, one key difference between a multiplexer change and a step input is that many ADCs have provisions to automatically identify an ADC operation that can lead to unsettled data. The ADC then waits until the data is settled to indicate a new conversion result is ready. As an example of this behavior, the ADS124S08, a 24-bit, 4-kSPS, 12-channel delta-sigma ADC, automatically restarts the digital filter when certain

register settings are changed – including the INPMUX register – or when a new conversion is triggered. Figure 4-7 shows how the ADS124S08 sinc$^3$ filter and $\overline{\text{DRDY}}$ pin respond after the user initiates a conversion.

### Single-shot conversion mode: Sinc$^3$ Filter

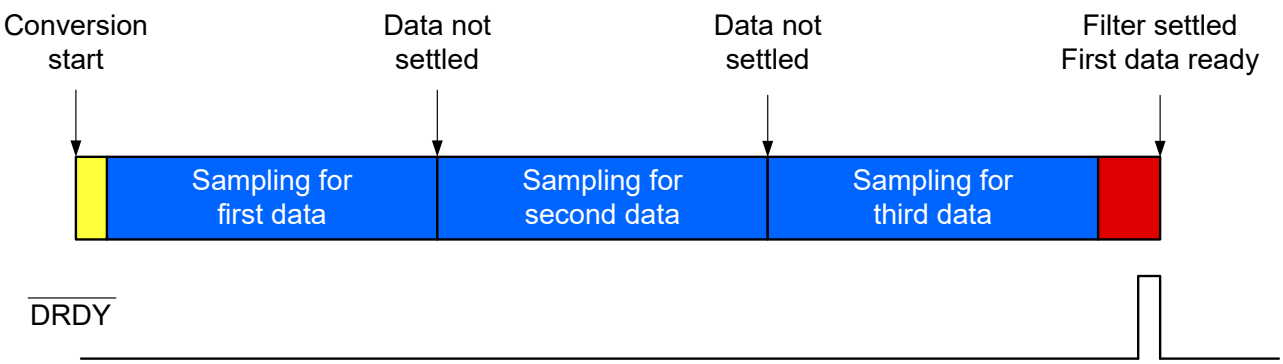



**Figure 4-7. ADS124S08 Sinc$^3$ Filter and $\overline{\text{DRDY}}$ Pin Behavior in Single-Shot Conversion Mode**

In Figure 4-7, $\overline{\text{DRDY}}$ only transitions high-to-low to indicate new data are available after three conversion periods from conversion start (plus processing time). Importantly, the ADS124S08 automatically *hides* unsettled data after a conversion is triggered so that the user does not need to manually discard this information. However, this is not the case for all ADCs. For example, the $\overline{\text{DRDY}}$ pin on the 24-bit, 125-kSPS, 16-channel ADS1258 indicates *all* new conversion results in fixed-channel mode, even those values that are unsettled. Refer to the specific ADC data sheet for more information on how the device handles unsettled data.

Moreover, the ADS124S08 is only able to *hide* unsettled data because it receives the conversion start or register write request. These actions alert the ADC that the input signal is changing and that all information in the digital filter needs to be cleared. As stated in Section 4, the ADC cannot identify unsettled data if the analog input changes significantly while sampling the same channel, such as when a step input occurs. Another action that the ADC does not automatically recognize is when the inputs to an *external* multiplexer are changed. Similar to a step input, the user must manually restart the conversion process after changing the inputs on an external multiplexer.

# 5 ADC Features and Modes that Affect Conversion Latency

While the previous sections describe the causes of conversion latency in delta-sigma ADCs as well as the digital filter operation and behavior in detail, the following subsections discuss several key ADC features and modes that can affect conversion latency. A strong understanding of these ADC behaviors helps reveal how enabling each feature or selecting a specific mode impacts the overall system cycle time.

## 5.1 First Conversion Versus Second and Subsequent Conversion Latency

One important factor that can affect ADC conversion latency is whether settled data is the *first conversion* or a *second* or *subsequent conversion*. The ADS1261 conversion latency values shown in Table 2-2 are valid for the first conversion. As the *ADS126x (ADS1261) Precision, 5-channel and 10-channel, 40-kSPS, 24-bit, delta-sigma ADCs with PGA and monitors* data sheet states in the *Conversion Latency* section, second and subsequent conversion latency is equal to 1 / ODR for all filter types assuming continuous-conversion mode is used and chopping is disabled.

Figure 5-1 illustrates this concept by highlighting settled first conversions (in red) versus settled second and subsequent conversions (in green) for a sinc$^3$ filter that experiences a multiplexer change after conversion period N completes.

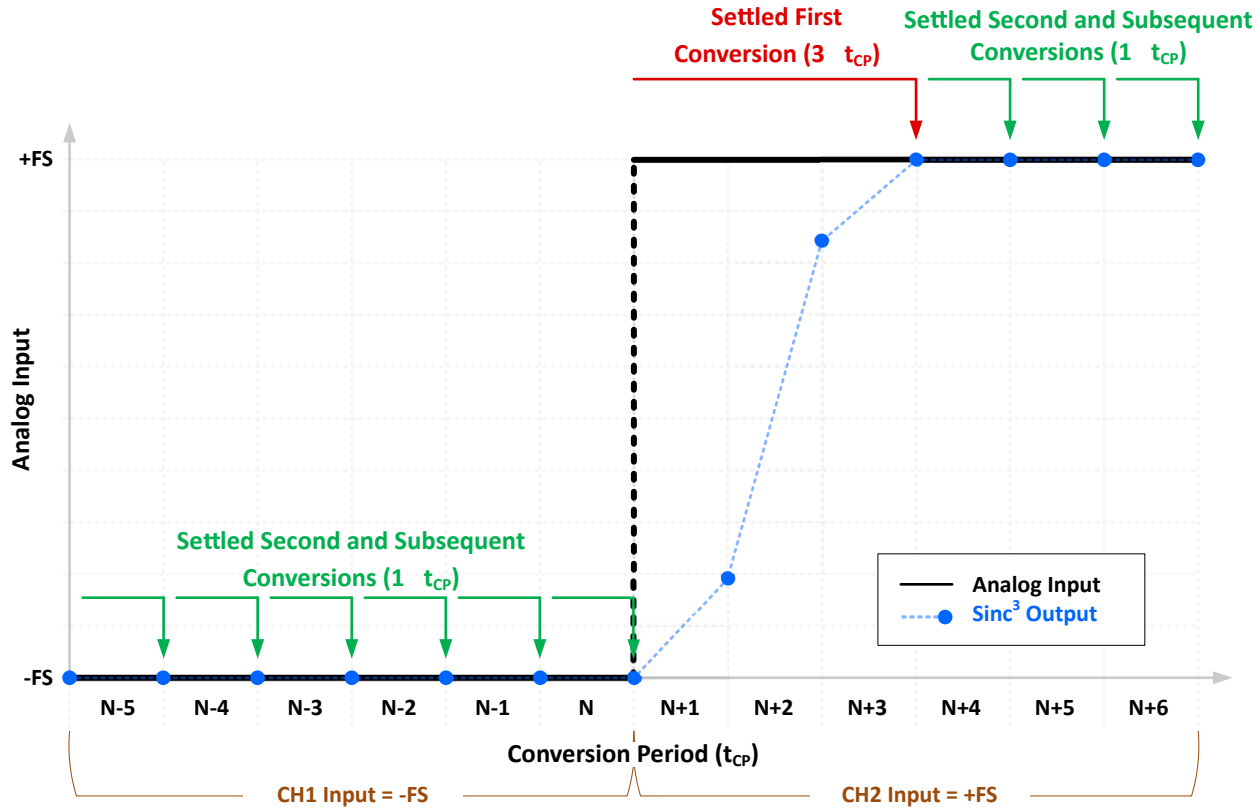



**Figure 5-1. First Conversion vs Second and Subsequent Conversions Using a Sinc$^3$ Filter**

In Figure 5-1, the first settled conversion result after the multiplexer change requires three conversion periods to propagate through the sinc$^3$ filter, and occurs at the end of conversion period N+3. This is shown in red. Importantly, second and subsequent conversion results – N-5 through N for CH1 and N+4 through N+6 for CH2 – all settle in one conversion period, or 1 / ODR. This is shown in green. This result occurs because the input signal is not changing significantly during these conversion periods such that the information in each filter stage is approximately equal. Combining the data in all three filter stages therefore yields settled conversion results at the end of each conversion period. If another multiplexer change occurred after conversion period N+6 for example, the process would need to restart and first conversion latency would apply.

This behavior is documented in the *ADS124S0x Low-Power, Low-Noise, Highly Integrated, 6- and 12-Channel, 4-kSPS, 24-Bit, Delta-Sigma ADC with PGA and Voltage Reference* data sheet as well. Figure 5-2 shows how the $\overline{\text{DRDY}}$ pin responds for both the *low-latency* and sinc$^3$ filter in continuous-conversion mode. Note that the term *low-latency* in this case is the name of a specific ADS124S08 digital filter and should not be confused with sinc filters in general, which are often considered low-latency compared to wideband filters (see Section 4).

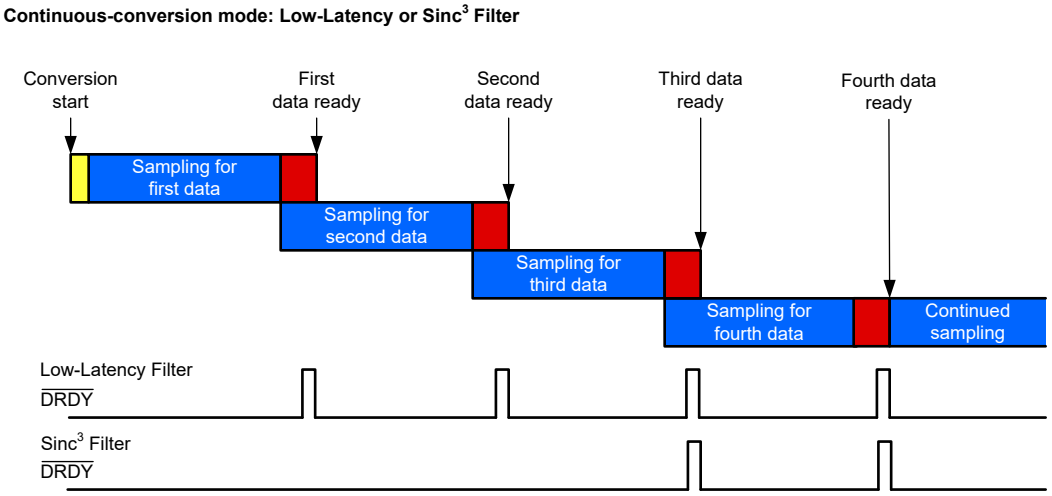



**Figure 5-2. ADS124S08 *Low-Latency*, Sinc$^3$ Filter, and $\overline{\text{DRDY}}$ Pin Behavior in Continuous-Conversion Mode**

The ADS124S08 *low-latency* filter shown in Figure 5-2 is effectively a sinc$^1$ filter that provides settled data in approximately one conversion period (assuming the analog input signal is settled). Comparatively, the sinc$^3$ filter in Figure 5-2 requires three conversion periods after conversion start to provide settled data. A subsequent conversion from the sinc$^3$ filter is available in one conversion period however. As noted in Section 2, this information is often quantified in a conversion latency table. Table 5-1 reports first conversion as well as second and subsequent conversion latency for the ADS124S08 sinc$^3$ filter.

**Table 5-1. ADS124S08 Conversion Latency Table Using the Sinc$^3$ Filter**

| NOMINAL DATA RATE(1) (SPS) | FIRST DATA FOR CONTINUOUS-CONVERSION MODE OR SINGLE-SHOT CONVERSION MODE(2) | | SECOND AND SUBSEQUENT CONVERSIONS FOR CONTINUOUS-CONVERSION MODE | |
|---|---|---|---|---|
| | ms(3) | NUMBER OF $t_{MOD}$ PERIODS(3) | ms(4) | NUMBER OF $t_{MOD}$ PERIODS(4) |
| 2.5 | 1200.25 | 307265 | 400 | 102400 |
| 5 | 600.254 | 153665 | 200 | 51200 |
| 10 | 300.254 | 76865 | 100 | 25600 |
| 16.6 | 180.254 | 46145 | 60 | 15360 |
| 20 | 150.254 | 38465 | 50 | 12800 |
| 50 | 60.254 | 15425 | 20 | 5120 |
| 60 | 50.223 | 12857 | 16.7 | 4264 |
| 100 | 30.254 | 7745 | 10 | 2560 |
| 200 | 15.254 | 3905 | 5 | 1280 |
| 400 | 7.754 | 1985 | 2.5 | 640 |
| 800 | 4.004 | 1025 | 1.25 | 320 |
| 1000 | 3.156 | 808 | 1 | 256 |
| 2000 | 1.656 | 424 | 0.5 | 128 |
| 4000 | 0.906 | 232 | 0.25 | 64 |

(1) Valid for the internal oscillator or an external 4.096-MHz clock. Scales proportional with clock frequency
(2) Conversions start at the rising edge of the START/SYNC pin or on the seventh SCLK falling edge for a START command
(3) Time does not include the programmable delay set by the DELAY[2:0] bits in the gain setting register. The default setting is an additional 14 · $t_{MOD}$, where $t_{MOD} = t_{CLK} \cdot 16$
(4) Subsequent readings in continuous-conversion mode do not have the programmable delay time

As noted throughout this section, first conversion latency applies after an ADC operation, as per Section 4.1. This can include manually triggering a conversion, changing certain ADC settings such as the input channel, or the initial conversion after ADC power up. Refer to the ADC data sheet for more information about any specific actions that can trigger the digital filter to reset such that first conversion latency applies.

First conversion latency does not apply when a step input occurs as per Section 4 because the ADC cannot automatically identify this condition. Instead, the user must detect this event and then manually wait the required time for settled data. Conversely, the user could manually restart the conversion process after confirming the step input has settled. The ADC then automatically waits the first-conversion latency to produce settled data, assuming the device includes this functionality.

## 5.2 Conversion Mode

Closely related to first-conversion versus second-and-subsequent conversion latency is how conversions are triggered by the user. Many ADCs offer multiple modes to trigger a conversion. Examples of these modes include *continuous-conversion*, *single-shot*, or *pulse-convert*. Table 5-2 shows the available conversion modes in the ADS124S08.

**Table 5-2. Conversion Mode Selection in the ADS124S08 Data Rate (DATARATE) Register**

| BIT | FIELD | TYPE | RESET | DESCRIPTION |
|---|---|---|---|---|
| 5 | MODE | R/W | 0h | **Conversion mode selection**<br>Configures the ADC for either continuous-conversion or single-shot mode.<br>0 : Continuous-conversion mode (default)<br>1 : Single-shot conversion mode |

*Continuous-conversion mode* automatically starts a new conversion immediately after the previous conversion period completes. This process continues indefinitely until stopped by the user. Assuming a settled analog input, the ADC outputs settled data after each conversion period for second and subsequent conversions. This process is shown in Figure 5-2.

Conversely, *single-shot mode* produces one conversion result and then waits for user input before restarting the conversion process. When the user requests a new conversion in single-shot mode, the ADC resets the digital filter each time. Therefore, single-shot conversions are always affected by the first-conversion latency that was previously described in Section 5.1. This is true even if the same channel is repeatedly sampled without any ADC operations or step inputs.

The difference between conversion modes may also be included in the conversion latency table. For example, Table 5-1 indicates that first conversion latency applies to both continuous- and single-shot conversion mode for the ADS124S08, while second and subsequent conversion latency only applies to continuous-conversion mode. Refer to the specific ADC data sheet to identify available conversion modes as well as how each mode affects the overall conversion latency.

## 5.3 Programmable Delay

Some ADCs include a *programmable delay* time to accommodate external settling requirements. Table 5-3 shows the ADS124S08 available programmable delay time options in terms of one modulator period, $t_{MOD}$.

**Table 5-3. Programmable Delay Selection in the ADS124S08 Gain Setting (PGA) Register**

| BIT | FIELD | TYPE | RESET | DESCRIPTION |
|---|---|---|---|---|
| 7:5 | DELAY[2:0] | R/W | 0h | **Programmable conversion delay selection**<br>Sets the programmable conversion delay time for the first conversion after a WREG when a configuration change resets the digital filter and triggers a new conversion.<br>000 : 14 · $t_{MOD}$ (default)<br>001 : 25 · $t_{MOD}$<br>010 : 64 · $t_{MOD}$<br>011 : 256 · $t_{MOD}$<br>100 : 1024 · $t_{MOD}$<br>101 : 2048 · $t_{MOD}$<br>110 : 4096 · $t_{MOD}$<br>111 : 1 · $t_{MOD}$ |

The delay shown in Table 5-3 can be used for several reasons, including waiting for an external analog RC filter to settle to a final value, accommodating the PGA start-up time, or ensuring that an integrated voltage reference or current source (IDAC) is stable before starting the conversion process.

Programmable delay time was included in Figure 4-7 and Figure 5-2, though not explicitly mentioned. Figure 5-3 shows a complete diagram for the ADS124S08 *low-latency* and sinc$^3$ filter behavior in single-shot conversion mode. The programmable delay time corresponds to the yellow boxes, as per the legend in the top right.

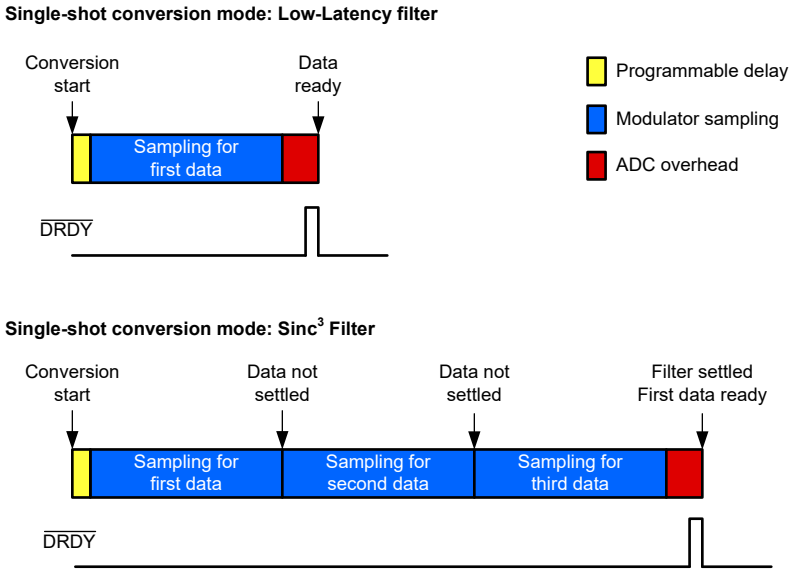



**Figure 5-3. Programmable Delay Using the ADS124S08 *Low-Latency* and Sinc$^3$ Filters in Single-Shot Conversion Mode**

As Figure 5-3 shows, the programmable delay occurs immediately after a conversion start is triggered. Therefore, this time period only needs to be considered when calculating first conversion latency. However, the programmable delay time is not always included in the overall conversion latency table values. Table note #3 for the ADS124S08 conversion latency table specifically states that the conversion latency values do not include the default programmable delay time of 14 · $t_{MOD}$ seconds. Comparatively, the table note for the ADS1261 conversion latency table states that the conversion latency values do include the default 50-µs programmable delay time. Refer to the specific ADC data sheet to determine if that device has a programmable delay feature as well as how that time affects the overall conversion latency.

## 5.4 ADC Overhead Time

Another factor contributing to ADC conversion latency is *ADC overhead time*. This time accounts for any internal ADC functions required to process the converted data before the ADC indicates that a new conversion result is ready. ADC overhead time is defined by the ADC design and therefore cannot be changed by the user.

Unlike programmable delay time, the ADC overhead time is required each time settled data becomes available. However, the conversion process starts as the ADC overhead time begins such that the ADC overhead time only adds to conversion latency during the first conversion. Figure 5-4 highlights the ADC overhead time in red and shows when it occurs during each conversion period for the ADS124S08 *low-latency* filter in continuous-conversion mode.

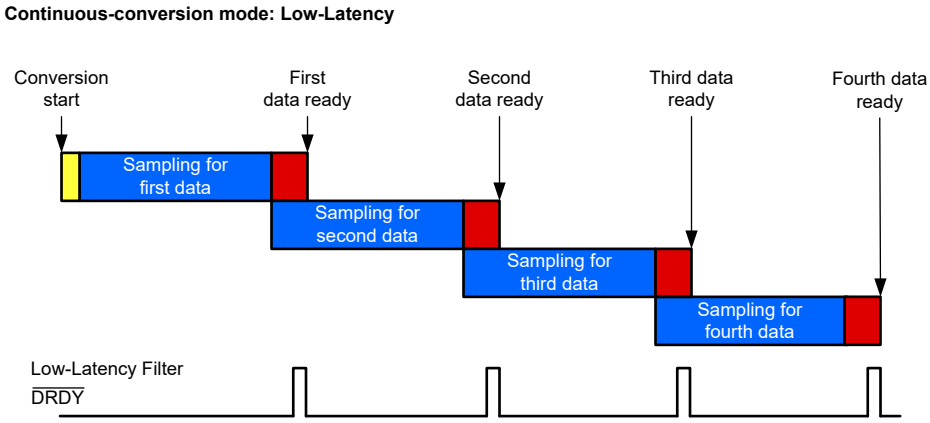



**Figure 5-4. ADC Overhead Time Using the ADS124S08 *Low-Latency* Filter in Continuous-Conversion Mode**

Importantly, Figure 5-4 confirms that the ADC overhead time only contributes to the conversion latency during the first conversion. Second and subsequent conversion results must accommodate the ADC overhead time, but the conversion process begins simultaneously with the ADC overhead such that the overall conversion latency is equal to 1 / ODR.

Higher-order filters follow a similar pattern. Figure 5-5 highlights the ADC overhead time in red and shows when it occurs during the conversion process for the ADS124S08 sinc$^3$ filter in continuous-conversion mode.

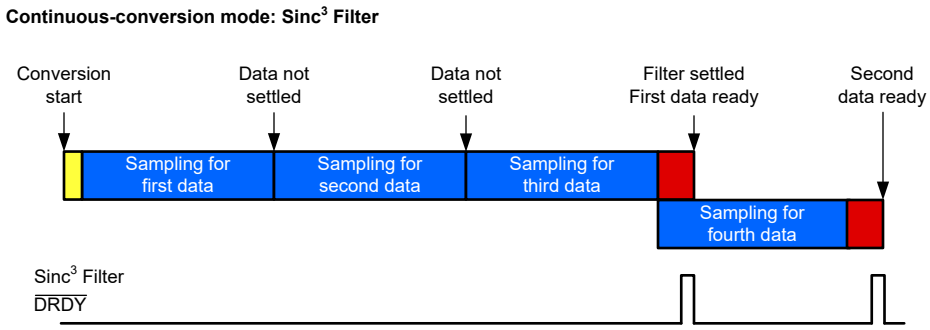



**Figure 5-5. ADC Overhead Time Using the ADS124S08 Sinc$^3$ Filter in Continuous-Conversion Mode**

The unsettled sinc$^3$ data in Figure 5-5 does not require the ADC overhead time because the data is not ready to be processed after the first and second conversion periods. Instead, the ADC overhead begins after the third conversion period ends, and then immediately after each second and subsequent conversion period. However, the conversion process begins concurrently with the ADC overhead time for second and subsequent data such that it does not affect the overall conversion latency. The end result is that the overhead time only affects first conversion latency when using the ADS124S08 sinc$^3$ filter, just as it did for the *low-latency* filter in Figure 5-4.

One important characteristic of ADC overhead time is that it typically requires a fixed amount of ADC clock periods and therefore can be independent of ODR. This means ADC overhead time tends to use a larger

percentage of the overall conversion latency as ODR increases. To verify this claim, refer to the ADS124S08 sinc$^3$ conversion latency values in Table 5-1. As table note #3 states, there is no programmable delay included in the conversion latency. Therefore, the ADS124S08 first conversion latency values using the sinc$^3$ filter are comprised of three conversion periods plus the ADC overhead time, as per Figure 5-5. Since a conversion period is just the time specified for a second or subsequent conversion, it is possible to calculate the ADS124S08 sinc$^3$ filter overhead time, $t_{ADC\_OVERHEAD}$, using Equation 4:

$$t_{ADC\_OVERHEAD} = t_{MOD(FC)} - (3 \cdot t_{MOD(SSC)}) \quad (4)$$

where

- $t_{MOD(FC)}$ = the number of $t_{MOD}$ periods equal to the first conversion latency
- $t_{MOD(SSC)}$ = the number of $t_{MOD}$ periods equal to the second or subsequent conversion latency

It is helpful to consider $t_{ADC\_OVERHEAD}$ in terms of the number of $t_{MOD}$ periods because conversion latency values in milliseconds can include rounding errors that make the results less clear. However, it is possible to use conversion latency values in milliseconds if an ADC data sheet does not quantify conversion latency by the number of $t_{MOD}$ periods. In this case, replace the variables in Equation 4 with their corresponding conversion latency values in milliseconds.

Table 5-4 uses Equation 4 to calculate $t_{ADC\_OVERHEAD}$ for all ODRs using the ADS124S08 sinc$^3$ filter. Table 5-4 also calculates the percent of the total conversion latency used by $t_{ADC\_OVERHEAD}$.

**Table 5-4. Calculating $t_{ADC\_OVERHEAD}$ for the ADS124S08 Sinc$^3$ Filter**

| ODR (SPS) | $t_{MOD(FC)}$ ($t_{MOD}$ PERIODS) | $t_{ADC\_OVERHEAD}$ ($t_{MOD}$ PERIODS) | % OF TOTAL |
|---|---|---|---|
| 2.5 | 307265 | 65 | 0.02% |
| 5 | 153665 | 65 | 0.04% |
| 10 | 76865 | 65 | 0.08% |
| 16.6 | 46145 | 65 | 0.14% |
| 20 | 38465 | 65 | 0.17% |
| 50 | 15425 | 65 | 0.42% |
| 60 | 12857 | 65 | 0.51% |
| 100 | 7745 | 65 | 0.84% |
| 200 | 3905 | 65 | 1.66% |
| 400 | 1985 | 65 | 3.27% |
| 800 | 1025 | 65 | 6.34% |
| 1000 | 808 | 40 | 4.95% |
| 2000 | 424 | 40 | 9.43% |
| 4000 | 232 | 40 | 17.24% |

An important takeaway from Table 5-4 is that $t_{ADC\_OVERHEAD}$ is not constant across all ODRs using the ADS124S08 sinc$^3$ filter. Instead, $t_{ADC\_OVERHEAD} = 65 \cdot t_{MOD}$ periods for ODR < 1000 SPS and $t_{ADC\_OVERHEAD} = 40 \cdot t_{MOD}$ periods for ODR ≥ 1000 SPS. This behavior results from the specific digital filter architecture, and can be different for other filters within the same ADC. Moreover, Table 5-4 confirms the claim that $t_{ADC\_OVERHEAD}$ tends to become a larger percentage of the overall conversion latency as ODR increases. In fact, $t_{ADC\_OVERHEAD}$ is almost 20% of the total conversion time when ODR = 4000 SPS, compared to just 0.02% at ODR = 2.5 SPS.

As stated previously, ADC overhead time is defined by the ADC design. This means that the ratio of the ADC overhead time to the total conversion latency is fixed for a given ADC digital filter architecture and ODR. However, a $t_{MOD}$ period – and therefore the conversion latency – can be changed by modifying the ADC clock frequency.

## 5.5 Clock Frequency

The ADC *clock frequency* plays an important role in determining the conversion latency values reported in the ADC data sheet. Typically, these latency values are reported with the default clock frequency, $f_{CLK}$. For example, Table note #1 in the ADS124S08 conversion latency table states that the sinc$^3$ filter conversion latency values are reported with $f_{CLK}$ = 4.096 MHz. However, choosing a different clock frequency, $f_{CLK\_NEW}$ , proportionally changes the resulting latency time when reported in milliseconds.

Table 5-5 shows the number of $t_{MOD}$ periods, the default first-conversion latency, and the default ODR values for the ADS124S08 sinc$^3$ filter. Table 5-5 also calculates the first-conversion latency and ODR values when $f_{CLK\_NEW}$ = 4.5 MHz, which is the maximum clock frequency allowed by the ADS124S08.

**Table 5-5. Calculating How Changing the Clock Frequency Affects First-Conversion Latency and ODR**

| # OF $t_{MOD}$ PERIODS | $f_{CLK}$ = 4.096 MHz | | $f_{CLK\_NEW}$ = 4.5 MHz | |
|---|---|---|---|---|
| | FIRST-CONVERSION LATENCY (ms) | ODR (SPS) | FIRST-CONVERSION LATENCY (ms) | ODR (SPS) |
| 307265 | 1200.254 | 2.5 | 1092.498 | 2.7 |
| 153665 | 600.254 | 5 | 546.365 | 5.5 |
| 76865 | 300.254 | 10 | 273.298 | 11 |
| 46145 | 180.254 | 16.6 | 164.071 | 18 |
| 38465 | 150.254 | 20 | 136.765 | 22 |
| 15425 | 60.254 | 50 | 54.845 | 55 |
| 12857 | 50.223 | 60 | 45.714 | 66 |
| 7745 | 30.254 | 100 | 27.538 | 110 |
| 3905 | 15.254 | 200 | 13.885 | 220 |
| 1985 | 7.754 | 400 | 7.058 | 439 |
| 1025 | 4.004 | 800 | 3.645 | 879 |
| 808 | 3.156 | 1000 | 2.873 | 1099 |
| 424 | 1.656 | 2000 | 1.507 | 2197 |
| 232 | 0.906 | 4000 | 0.825 | 4395 |

Importantly, the number of $t_{MOD}$ periods in Table 5-5 is unaffected by changes in the clock frequency. However, the resulting conversion latency decreases while the ODR values increase when $f_{CLK\_NEW}$ = 4.5 MHz, enabling faster conversion results.

Changing the clock frequency also impacts the programmable delay. As Table 5-3 shows, the ADS124S08 programmable delay is specified in terms of $t_{MOD}$ periods. For example, the default delay is 14 · $t_{MOD}$, which is 3.42 µs when $f_{CLK}$ = 4.096 MHz. This delay reduces to 3.11 µs when $f_{CLK\_NEW}$ = 4.5 MHz. Ensure that the system still has the required delay when changing the value of the clock frequency.

One final thing to consider about the clock frequency is the tolerance. The clock frequency tolerance changes $f_{CLK}$ and therefore directly impacts the conversion latency as described throughout this section. For example, the ADS124S08 internal oscillator has a maximum accuracy tolerance of ±1.5%, which translates to a conversion latency variation of ±1.5%. Ultimately, consider how the clock frequency tolerance – whether the clock is internal or external to the ADC – might impact systems with very strict timing constraints.

## 5.6 Chopping

Many delta-sigma ADCs offer *chopping* functionality to help reduce errors and improve accuracy. Chopping is a sampling technique that averages two conversions together: one with normal polarity, and another with reverse polarity, which generates a final conversion result that is virtually free from offset or mismatch errors. Some examples of different chopping techniques include:

- Input (global) chop – remove offsets internal to the ADC, similar to constant offset calibration
- Bridge chop (AC excitation) – typically used with Wheatstone bridge circuits to remove offsets from the signal conditioning circuitry external to the ADC
- IDAC chop (rotation) – used with two-IDAC, 3-wire RTD systems to eliminate the effects of IDAC mismatch

Chopping impacts conversion latency because multiple conversions are required to determine a single, chopped conversion result. Moreover, the digital filter resets after each conversion even though the same channel is being sampled because the input polarity is swapped. This behavior is described and quantified in the ADC data sheet. For example, Figure 5-6 shows how the ADS124S08 processes data for both the *low-latency* and sinc$^3$ filters when global chop is enabled.

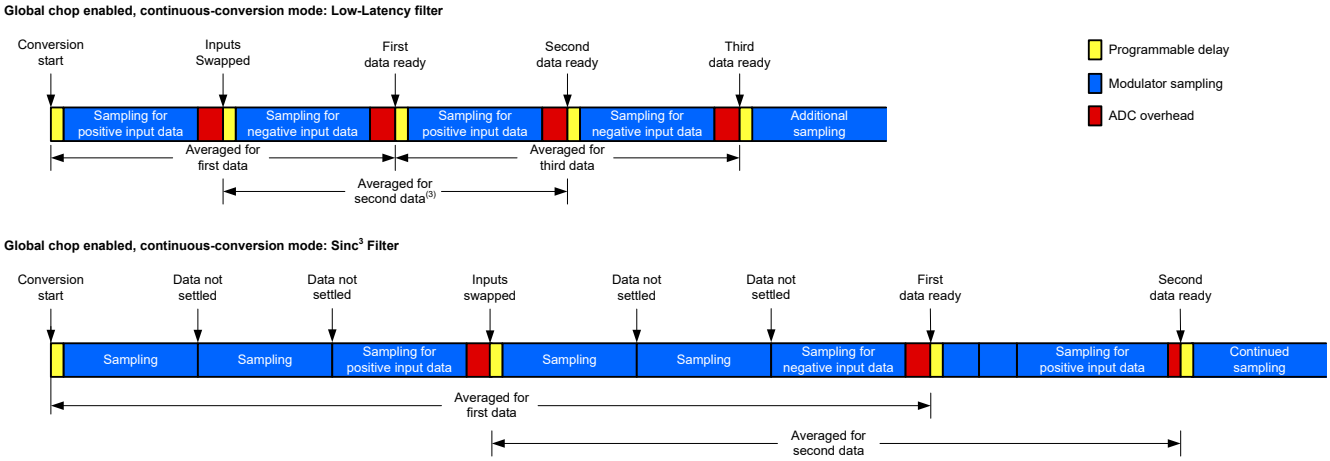



**Figure 5-6. Global Chop Conversion Mode Sequences Using the ADS124S08 *Low-Latency* and Sinc$^3$ Filters**

After initiating a conversion, the *low-latency* and sinc$^3$ filters in Figure 5-6 both take two complete first conversion latency periods – each including programmable delay and ADC overhead time – before the first conversion result is ready. As noted previously, this is because the input signal polarity is swapped after each conversion, requiring the digital filter to reset each time. As an example, it takes 30.254 ms to get a settled, first conversion result from the ADS124S08 sinc$^3$ filter at an ODR = 100 SPS as per Table 5-1. When global chop mode is enabled, this time doubles to 2 · 30.254 ms = 60.508 ms, resulting in an effective data rate of 16.5 SPS for first conversion data.

Second and subsequent conversions follow a similar process as outlined in Section 5.2. If the ADC is in single-shot mode and the user triggers a second conversion on the same channel, the entire process restarts. This requires two new conversions to be averaged together, and therefore two additional first conversion latency periods to generate one, settled conversion result. If the ADC is in continuous-conversion mode, a second or subsequent conversion averages the previous conversion with the current conversion to generate the next settled conversion result. This behavior only requires one additional first conversion latency period. Second and subsequent conversion behavior in continuous-conversion mode is also shown in Figure 5-6. Continuing the previous example where the first conversion result took 60.508 ms at ODR = 100 SPS with global chop enabled, generating a second or subsequent conversion result in continuous-conversion mode takes 30.254 ms.

Not all ADCs offer chopping functionality, nor is all chopping behavior the same. Refer to the specific ADC data sheet to determine how to calculate conversion latency when chopping functionality is enabled.

# 6 Analog Settling

While the previous sections discussed how certain features and modes integrated into the ADC affect conversion latency and cycle time, one common external factor that affects timing is analog settling. External signal conditioning circuitry such as amplifiers or filters has a finite bandwidth. Additionally, some ADCs have internal analog filters that have a well-defined settling time. As a result, an analog signal takes some amount of time to propagate through these components before it is sampled by the ADC. This analog signal could be an input from a sensor, or it could be a biasing signal such as a current source or an excitation voltage. In any case, it is necessary to account for analog settling time in the overall conversion latency. Otherwise, the ADC will sample an unsettled signal that will show up as an error in the ADC conversion result. This noise can be erroneously attributed to crosstalk or other errors even though it is actually the result of sampling an unsettled signal.

As an example, a simple low-pass RC filter used for anti-aliasing has some time constant, τ, that might prohibit the input signal from settling before the ADC starts converting. Figure 6-1 shows the differential filter circuit commonly used at the input to a delta-sigma ADC on the left and its corresponding settling time in the plot on the right.

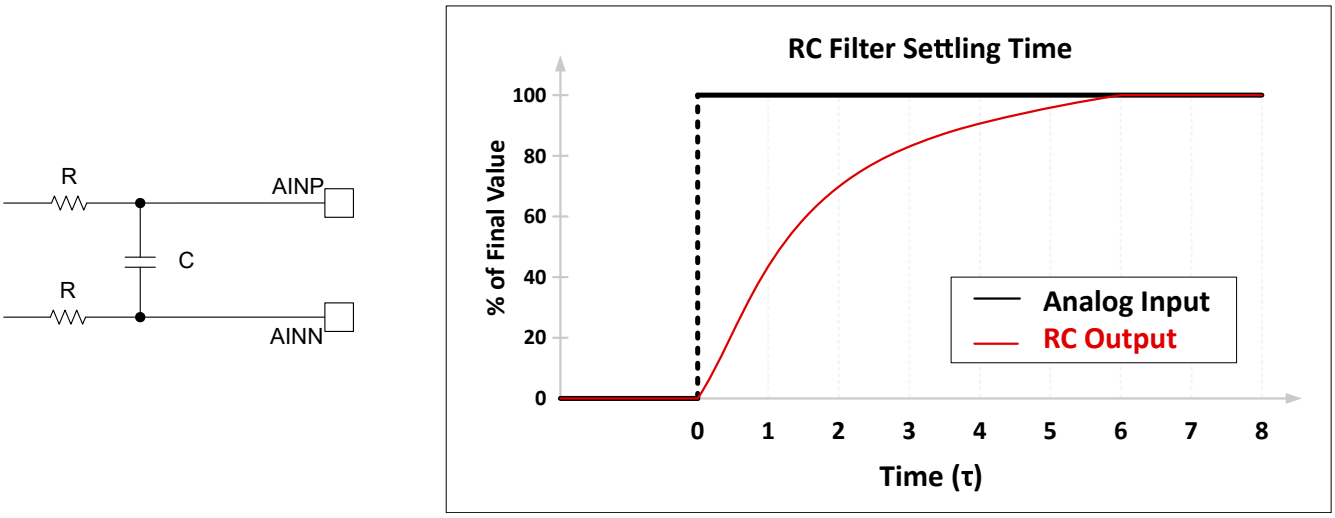



**Figure 6-1. Step Response and Settling Time for an RC Filter**

As shown, Figure 6-1 assumes there is initially 0 V across the capacitor in the example system. Then, at τ = 0, the 5-V analog step input is applied to the capacitor, which is the black plot shown in Figure 6-1. The capacitor cannot respond to this voltage immediately, and instead takes some time to ramp to the applied value, as per the response shown in red. While this plot appears to show that the output settles after approximately 5 · τ, many high-resolution delta-sigma ADCs can distinguish much finer analog signals than the RC output amplitude at 5 · τ. In fact, it takes more than 17 · τ for the RC output signal to reach ½ of a least significant bit (LSB) for a 24-bit ADC. Waiting 17 · τ might be unnecessary for some applications, though even 20-bit resolution takes almost 15 · τ to settle to ½ of an LSB. The relationship between ADC resolution and the number of time constants to settle to ½ LSB ($\tau_{LSB}$) can be calculated using Equation 5.

$$\tau_{LSB} = \ln(2^{N+1}) \quad (5)$$

where

- N is the ADC resolution

Table 6-1 uses Equation 5 to calculate the number of time constants required for the analog filter to settle to ½ LSB for several common ADC resolution values.

**Table 6-1. RC Filter Settling Time for Common ADC Resolutions**

| RESOLUTION (BITS) | $\tau_{LSB}$ |
|---|---|
| 16 | 11.78 |
| 18 | 13.17 |
| 20 | 14.56 |
| 22 | 15.94 |
| 24 | 17.33 |

One important caveat to the information in Table 6-1 is that the actual RC output settling time depends on the magnitude of the ADC LSB as well as the magnitude of the change in the input voltage. If the ADC reference voltage is small or the gain is large, it is often impractical to settle to ½ LSB because the LSB size is well below the inherent noise of the ADC. Instead, target the magnitude of the system noise at the desired data rate and gain settings. Additionally, if the applied voltage changes from 4.99 V to 5 V for example, it will not be necessary to wait the time specified in Table 6-1 to reach the corresponding ADC resolution. Therefore, consider analog settling time when the input signal changes very quickly, when the value of τ is large, or when the magnitude of the input signal changes significantly after each conversion.

As mentioned previously, some ADCs include a programmable delay that occurs immediately before the conversion process to account for external factors such as a multiplexer change or analog settling. For example, assume that a design calls for 20-bit resolution and includes an RC anti-aliasing filter where τ = 15 µs. Table 6-1 reveals that 14.56 · τ seconds are required to settle to 20-bit resolution, which is a total analog settling time of 14.56 · 15 µs = 218.4 µs. The ADS124S08 programmable delay options in Table 5-3 – where $t_{MOD}$ = 3.9 µs when $f_{CLK}$ = 4.096 MHz – determine that the system requires at least 218.4 / 3.9 = 55.9 · $t_{MOD}$ periods to accommodate the analog settling time. Therefore, set DELAY[2:0] = 010b to wait 64 $t_{MOD}$ periods and allow enough time for the RC output to fully settle before the ADC starts the conversion process.

Ultimately, it is important to consider how external signal conditioning circuitry can impact analog settling time because this directly adds to the overall ADC conversion latency.

# 7 Important Takeaways

This application note provides a detailed explanation of different ADC timing components, how multiplexed delta-sigma ADCs sample and process data, and how to use the information in the ADC data sheet to select a device that meets system cycle-time requirements. The following list summarizes the key points presented in this document:

- Delta-sigma modulator data takes some time to propagate through the digital filter, resulting in conversion latency
- Sinc filter conversion latency is approximately equal to a number of conversion periods equivalent to the sinc filter order. For example, a $sinc^3$ filter typically has three conversion period latency
- ADCs do not have a way to identify significant changes in the input signal, such as when a step input occurs. Ensure that the input signal is settled before starting the conversion process
- Under certain conditions, some ADCs can *hide* unsettled data from the user
- First conversion data is subject to conversion latency proportional to the sinc filter order, while second and subsequent conversion data can be approximated as 1 / ODR in most cases
- Single-shot mode requires the digital filter to reset after each conversion result, providing all data at a rate equivalent to first conversion latency. Continuous-conversion mode typically provides second and subsequent data at a rate of 1 / ODR
- Some ADCs include a programmable delay to account for external analog settling before the conversion process begins. This delay occurs once when conversions are triggered and scales with the clock frequency
- Each ADC includes an overhead time to process data before the conversion result is available. This time affects the first conversion latency only, cannot be changed by the user, and scales with the clock frequency
- The ODR, the programmable delay, and the ADC overhead time scale with the clock frequency. This enables the user to increase or decrease the conversion latency compared to the nominal value without changing any other system settings
- Chopping swaps the polarity of the input signal and averages two conversions to produce a settled conversion result. As such, the digital filter must reset after each conversion, requiring two first conversion latency periods to generate a single, settled output
- External signal conditioning circuitry has some finite bandwidth that can cause analog inputs to be unsettled when the ADC conversion process begins. The programmable delay can be used to account for any analog settling delay, though this increases the total conversion latency

# 8 Cycle Time Calculation Examples

This section provides several different examples that apply the information presented in this document to arbitrary system parameters. These examples help demonstrate how the cycle time is calculated as well as how cycle time is affected by different design requirements. To keep the analysis simple, these examples assume no analog settling or step inputs. However, these factors should always be considered in a real design.

## 8.1 Example #1: Using the ADS124S08

Table 8-1 reports the system parameters that determine the cycle time in Example #1:

**Table 8-1. System Parameters for Example #1**

| PARAMETER | VALUE |
|---|---|
| ADC | ADS124S08 |
| ODR | 1000 SPS |
| Filter type | $sinc^3$ |
| Clock frequency | 4.096 MHz (default) |
| Conversion mode | Continuous |
| Programmable delay | $14 \cdot t_{MOD}$ (default) |
| Chopping | Disabled |
| Conversions per channel | 3 |
| # of channels | 2 |

First, this example uses the default clock frequency, $f_{CLK}$, of 4.096 MHz, allowing the nominal conversion latency values reported in the data sheet to be used. Next, it is necessary to consider both first conversion versus second and subsequent conversion latency for this example because there are multiple conversions required per channel and continuous-conversion mode is used.

Reviewing Table 5-1, the ADS124S08 conversion latency using the $sinc^3$ filter and ODR = 1000 SPS is 3.156 ms for first conversion data ($t_{FC}$) and 1 ms for second and subsequent conversions ($t_{SSC}$). These times include the ADC overhead but do not include the programmable delay, $t_{DELAY}$, where applicable. Equation 6 calculates the value of $t_{DELAY}$ in microseconds using the default value of $t_{DELAY} = 14 \cdot t_{MOD}$ and $f_{CLK}$ = 4.096 MHz:

$$t_{DELAY} = 14 \cdot t_{MOD} = 14 \cdot (16 / f_{CLK}) = 54.69\ \mu s \quad (6)$$

The ADS124S08 programmable delay does not affect $t_{SSC}$, though it does affect $t_{FC}$. The first conversion latency including programmable delay, $t_{FC\_TOTAL}$, is calculated by Equation 7:

$$t_{FC\_TOTAL} = t_{FC} + t_{DELAY} = 3.156\ ms + 0.055\ ms = 3.211\ ms \quad (7)$$

Finally, there is no need to consider additional latency due to chopping. Since each channel includes one first conversion and two second or subsequent conversions, the scan time for one channel, $t_{CH}$, is given by Equation 8:

$$t_{CH} = 1 \cdot t_{FC\_TOTAL} + 2 \cdot t_{SSC} = 1 \cdot 3.211\ ms + 2 \cdot 1\ ms = 5.211\ ms \quad (8)$$

Equation 9 calculates the cycle time, $t_{CYCLE}$, using the result from Equation 8:

$$t_{CYCLE} = \#\ of\ channels \cdot t_{CH} = 2 \cdot 5.211\ ms = 10.422\ ms \quad (9)$$

Ultimately, the cycle time for this example is 10.422 ms for 6 conversion results. Figure 8-1 depicts a timing diagram for the example system given the design parameters.

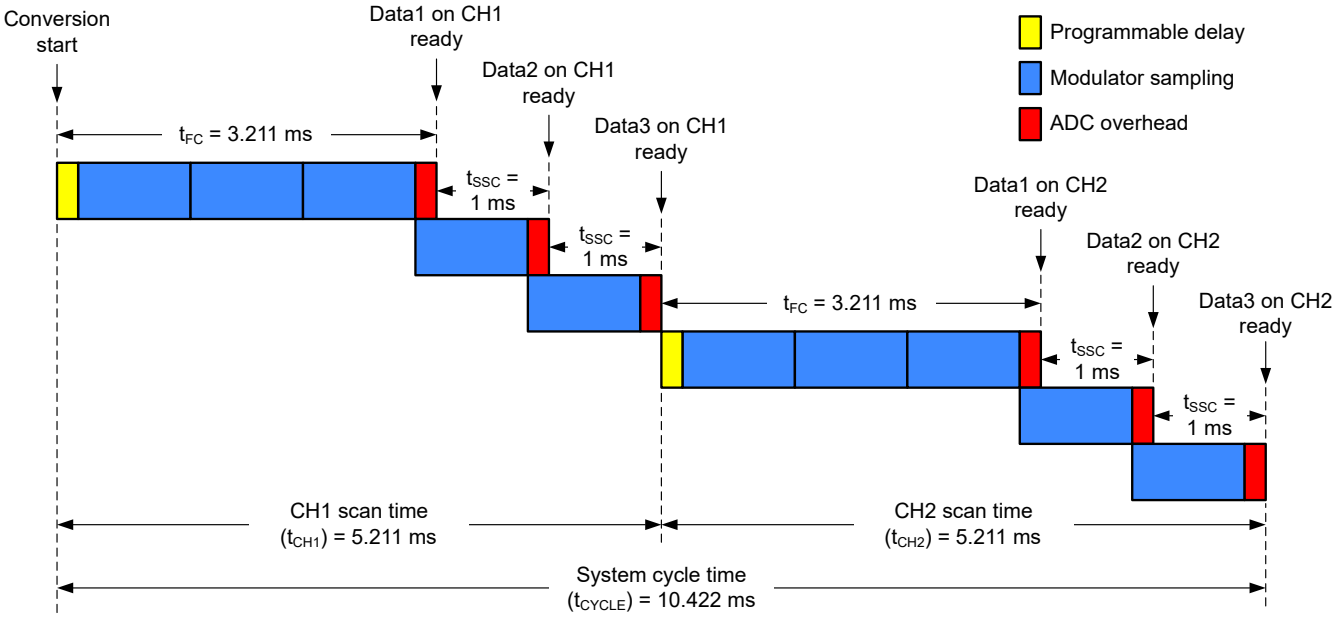



**Figure 8-1. Timing Diagram for Example #1**

## 8.2 Example #2: Changing the Conversion Mode

Table 8-2 reports the system parameters that determine the cycle time in Example #2:

**Table 8-2. System Parameters for Example #2**

| PARAMETER | VALUE |
|---|---|
| ADC | ADS124S08 |
| ODR | 1000 SPS |
| Filter type | $sinc^3$ |
| Clock frequency | 4.096 MHz (default) |
| Conversion mode | Single-shot |
| Programmable delay | 14 · $t_{MOD}$ (default) |
| Chopping | Disabled |
| Conversions per channel | 3 |
| # of channels | 2 |

The only difference between Example #1 and Example #2 is that Example #2 now uses single-shot conversion mode. This choice means that all three conversion results on each channel are subject to first conversion latency.

Reviewing Table 5-1, the ADS124S08 first conversion latency using the $sinc^3$ filter and ODR = 1000 SPS is 3.156 ms for first conversion data ($t_{FC}$). This time assumes that the default clock frequency, $f_{CLK}$, of 4.096 MHz is used, which is the case for this example. Also, this time includes the ADC overhead but does not include the programmable delay, $t_{DELAY}$. Equation 10 calculates the value of $t_{DELAY}$ in microseconds using the default value of $t_{DELAY}$ = 14 · $t_{MOD}$ and $f_{CLK}$ = 4.096 MHz:

$$t_{DELAY} = 14 \cdot t_{MOD} = 14 \cdot (16 / f_{CLK}) = 54.69\ \mu s \quad (10)$$

The ADS124S08 programmable delay applies to each conversion using single-shot mode, resulting in a first conversion latency including programmable delay, $t_{FC\_TOTAL}$, as per Equation 11:

$$t_{FC\_TOTAL} = t_{FC} + t_{DELAY} = 3.156\ ms + 0.055\ ms = 3.211\ ms \quad (11)$$

Finally, there is no need to consider additional latency due to chopping. Equation 13 calculates the cycle time, $t_{CYCLE}$, using the scan time for one channel, $t_{CH}$, that results from Equation 12. This assumes that the user starts the next conversion on each channel immediately after the previous conversion result is ready:

$$t_{CH} = 3 \cdot t_{FC\_TOTAL} = 3 \cdot 3.211\ ms = 9.633\ ms \quad (12)$$

$$t_{CYCLE} = \#\ of\ channels \cdot t_{CH} = 2 \cdot 9.633\ ms = 19.266\ ms \quad (13)$$

Ultimately, the cycle time for this example is 19.266 ms for 6 conversion results. Figure 8-2 depicts a timing diagram for the example system given the design parameters.

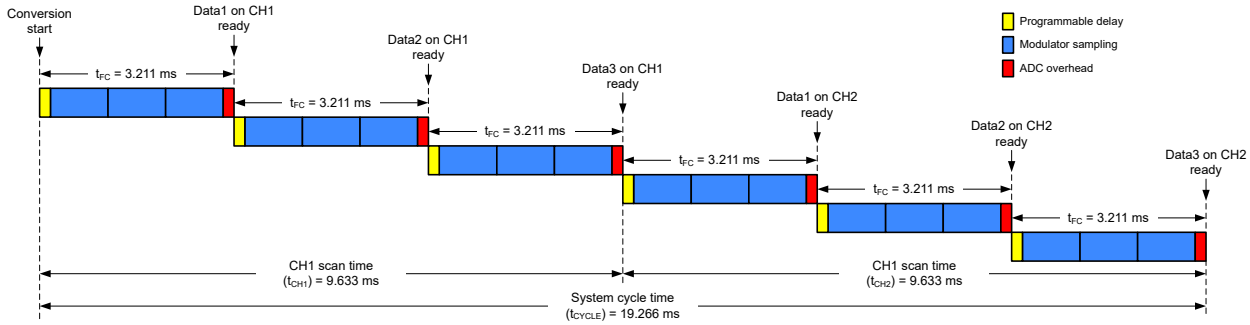



**Figure 8-2. Timing Diagram for Example #2**

## 8.3 Example #3: Changing the Filter Type

Table 8-3 reports the system parameters that determine the cycle time in Example #3:

**Table 8-3. System Parameters for Example #3**

| PARAMETER | VALUE |
|---|---|
| ADC | ADS124S08 |
| ODR | 1000 SPS |
| Filter type | *Low-latency* |
| Clock frequency | 4.096 MHz (default) |
| Conversion mode | Single-shot |
| Programmable delay | 14 · $t_{MOD}$ (default) |
| Chopping | Disabled |
| Conversions per channel | 3 |
| # of channels | 2 |

Example #3 keeps all of the same system parameters as Example #2 with the exception of switching to the ADS124S08 *low-latency* filter. This choice reduces the conversion latency for each first conversion.

While not shown in this document, the *ADS124S08* data sheet identifies the first-conversion latency, $t_{FC}$, using the *low-latency* filter at ODR = 1000 SPS as 1.156 ms. This time assumes that the default clock frequency, $f_{CLK}$, of 4.096 MHz is used, which is the case for this example. Also, this time includes the ADC overhead but does not include the programmable delay, $t_{DELAY}$. Equation 14 calculates the value of $t_{DELAY}$ in microseconds using the default value of $t_{DELAY}$ = 14 · $t_{MOD}$ and $f_{CLK}$ = 4.096 MHz:

$$t_{DELAY} = 14 \cdot t_{MOD} = 14 \cdot (16 / f_{CLK}) = 54.69\ \mu s \quad (14)$$

The ADS124S08 programmable delay applies to each conversion using single-shot mode, resulting in a first conversion latency including programmable delay, $t_{FC\_TOTAL}$, as per Equation 15:

$$t_{FC\_TOTAL} = t_{FC} + t_{DELAY} = 1.156\ ms + 0.055\ ms = 1.211\ ms \quad (15)$$

Finally, there is no need to consider additional latency due to chopping. Since all three conversions use single-shot mode and therefore are subject to $t_{FC\_NEW}$, the scan time for one channel, $t_{CH}$, is given by Equation 16. This assumes that the user starts the next conversion on each channel immediately after the previous conversion result is ready:

$$t_{CH} = 3 \cdot t_{FC\_TOTAL} = 3 \cdot 1.211\ ms = 3.633\ ms \quad (16)$$

Equation 17 calculates the cycle time, $t_{CYCLE}$, using the result from Equation 16:

$$t_{CYCLE} = \#\ of\ channels \cdot t_{CH} = 2 \cdot 3.633\ ms = 7.266\ ms \quad (17)$$

Ultimately, the cycle time for this example is 7.266 ms for 6 conversion results. Figure 8-3 depicts a timing diagram for the example system given the design parameters.

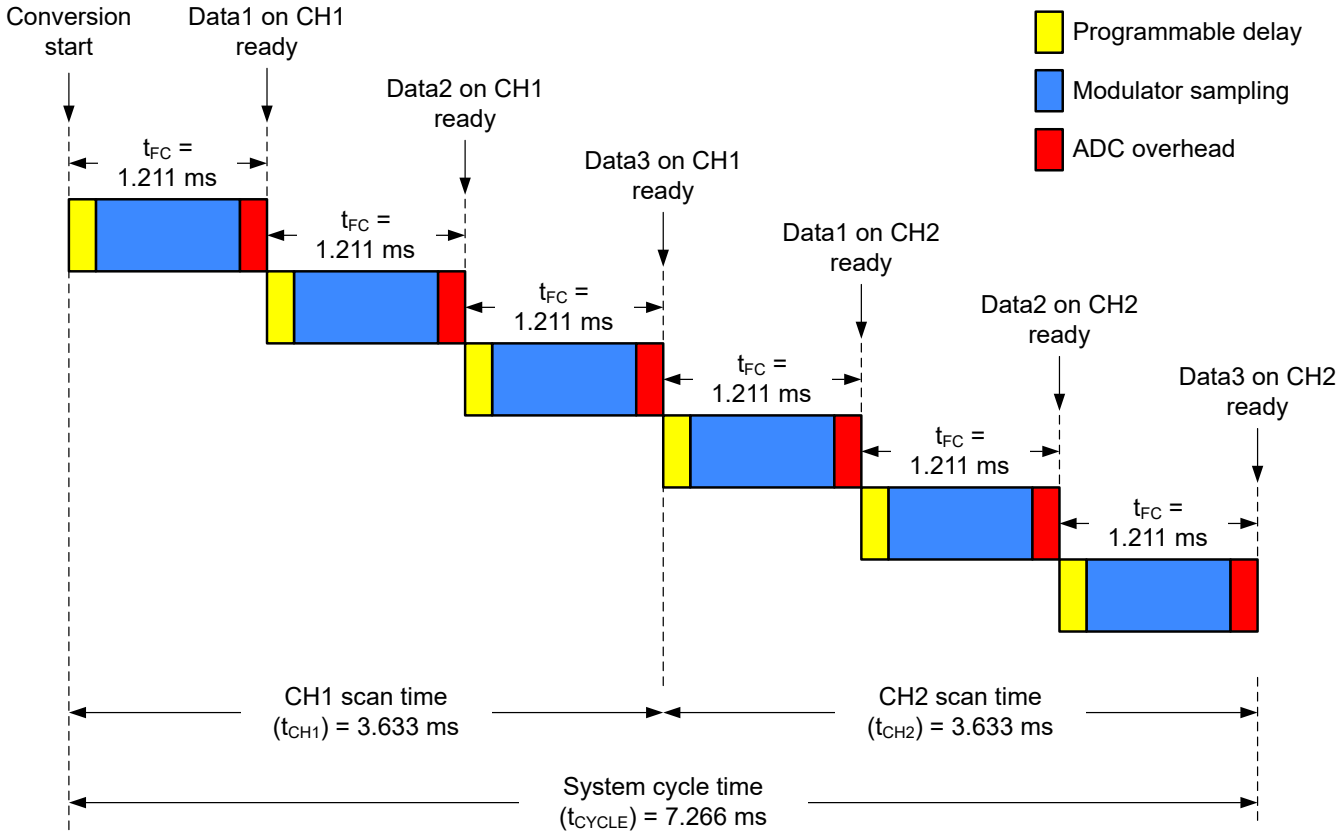



**Figure 8-3. Timing Diagram for Example #3**

## 8.4 Example #4: Changing the Clock Frequency

Table 8-4 reports · the system parameters that determine the cycle time in Example #4:

**Table 8-4. System Parameters for Example #4**

| PARAMETER | VALUE |
|---|---|
| ADC | ADS124S08 |
| ODR | 1000 SPS |
| Filter type | *Low-latency* |
| Clock frequency | 3 MHz |
| Conversion mode | Single-shot |
| Programmable delay | 14 · $t_{MOD}$ (default) |
| Chopping | Disabled |
| Conversions per channel | 3 |
| # of channels | 2 |

Example #4 keeps all of the same system parameters as Example #3, though the default clock frequency, $f_{CLK}$, of 4.096 MHz has been changed to a new clock frequency, $f_{CLK\_NEW}$, of 3 MHz. This choice directly affects the conversion latency and the programmable delay, and indirectly affects the ODR.

While not shown in this document, the *ADS124S08* data sheet identifies the first-conversion latency, $t_{FC}$, using the *low-latency* filter at ODR = 1000 SPS as 296 · $t_{MOD}$ periods. Using the conversion latency in terms of $t_{MOD}$ periods instead of milliseconds enables easier calculations because the number of $t_{MOD}$ periods is independent of clock frequency.

The conversion latency of 296 · $t_{MOD}$ periods includes the ADC overhead but does not include the programmable delay, $t_{DELAY}$. Using the default value of $t_{DELAY}$ = 14 · $t_{MOD}$, the total first-conversion latency, $t_{FC\_TOTAL}$, is given by Equation 18:

$$t_{FC\_TOTAL} = t_{FC} + t_{DELAY} = 296 \cdot t_{MOD} + 14 \cdot t_{MOD} = 310 \cdot t_{MOD} \quad (18)$$

Next, Equation 19 calculates one $t_{MOD}$ period in terms $f_{CLK\_NEW}$ using the ADS124S08:

$$t_{MOD} = 16 / f_{CLK\_NEW} \quad (19)$$

At $f_{CLK\_NEW}$ = 3 MHz, $t_{MOD}$ = 5.33 µs. Therefore, $t_{FC\_TOTAL}$ = 310 · 5.33 µs = 1.652 ms

Finally, there is no need to consider additional latency due to chopping. Equation 21 calculates the cycle time, $t_{CYCLE}$, using the scan time for one channel, $t_{CH}$, that results from Equation 20. This assumes that the user starts the next conversion on each channel immediately after the previous conversion result is ready:

$$t_{CH} = 3 \cdot t_{FC\_TOTAL} = 3 \cdot 1.652 \text{ ms} = 4.956 \text{ ms} \quad (20)$$

$$t_{CYCLE} = \#\text{ of channels} \cdot t_{CH} = 2 \cdot 4.956 \text{ ms} = 9.912 \text{ ms} \quad (21)$$

Ultimately, the cycle time for this example is 9.912 ms for 6 conversion results. Figure 8-4 depicts a timing diagram for the example system given the design parameters.

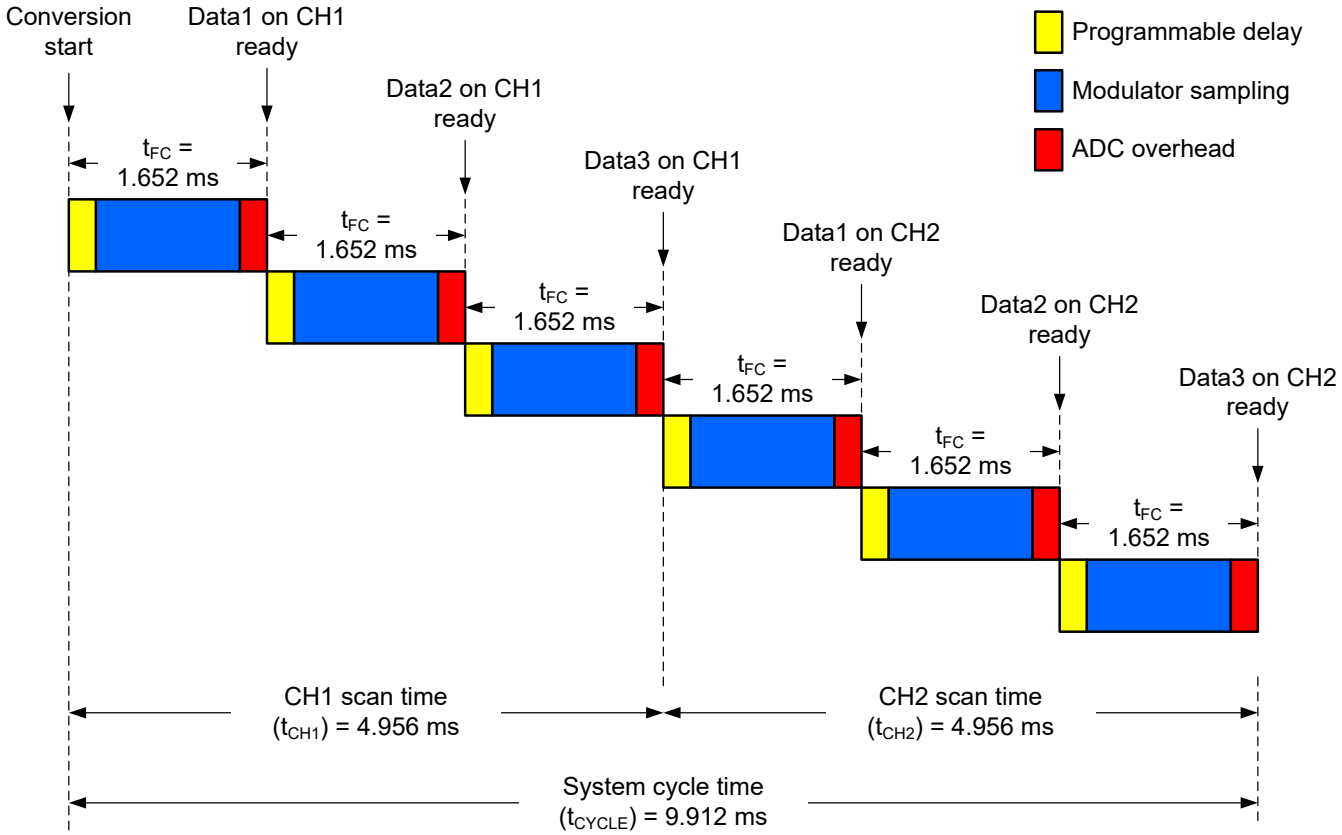



**Figure 8-4. Timing Diagram for Example #4**

## 8.5 Example #5: Enabling Chop and Reducing the Number of Conversions per Channel

Table 8-5 reports the system parameters that determine the cycle time in Example #5:

**Table 8-5. System Parameters for Example #5**

| PARAMETER | VALUE |
|---|---|
| ADC | ADS124S08 |
| ODR | 1000 SPS |
| Filter type | *Low-latency* |
| Clock frequency | 4.096 MHz (default) |
| Conversion mode | Single-shot |
| Programmable delay | $14 \cdot t_{MOD}$ (default) |
| Chopping | Enabled |
| Conversions per channel | 2 |
| # of channels | 2 |

Example #5 is similar to Example #3, though in this case the ADS124S08 chopping feature is turned on. Also, the number of conversions per channel has been reduced from 3 to 2. When chopping is enabled and single-shot mode is used, two conversions must be averaged to generate each conversion result. This increases the overall conversion latency.

While not shown in this document, the *ADS124S08* data sheet identifies the first-conversion latency, $t_{FC}$, using the *low-latency* filter at ODR = 1000 SPS as 1.156 ms. This time includes the ADC overhead but does not include the programmable delay, $t_{DELAY}$.Equation 22 calculates the value of $t_{DELAY}$ in microseconds using the default value of $t_{DELAY} = 14 \cdot t_{MOD}$ and $f_{CLK} = 4.096$ MHz:

$$t_{DELAY} = 14 \cdot t_{MOD} = 14 \cdot (16 / f_{CLK}) = 54.69\ \mu s \quad (22)$$

The ADS124S08 programmable delay applies to each conversion using single-shot mode, resulting in a first conversion latency including programmable delay, $t_{FC\_TOTAL}$, as per Equation 23:

$$t_{FC\_TOTAL} = t_{FC} + t_{DELAY} = 1.156\ ms + 0.055\ ms = 1.211\ ms \quad (23)$$

Finally, there is a need to consider additional latency due to chopping because each output is an average of two conversions. Since producing each conversion using single-shot mode requires $1 \cdot t_{FC\_TOTAL}$, a single conversion result with global chop enabled requires $2 \cdot t_{FC\_TOTAL}$. Equation 25 calculates the cycle time, $t_{CYCLE}$, using the scan time for one channel, $t_{CH}$, that results from Equation 24. This assumes that the user starts the next conversion on each channel immediately after the previous conversion result is ready:

$$t_{CH} = 2 \cdot (2 \cdot t_{FC\_TOTAL}) = 4 \cdot 1.211\ ms = 4.844\ ms \quad (24)$$

$$t_{CYCLE} = \#\ of\ channels \cdot t_{CH} = 2 \cdot 4.844\ ms = 9.688\ ms \quad (25)$$

Ultimately, the cycle time for this example is 9.688 ms for 4 conversion results. Figure 8-5 depicts a timing diagram for the example system given the design parameters.

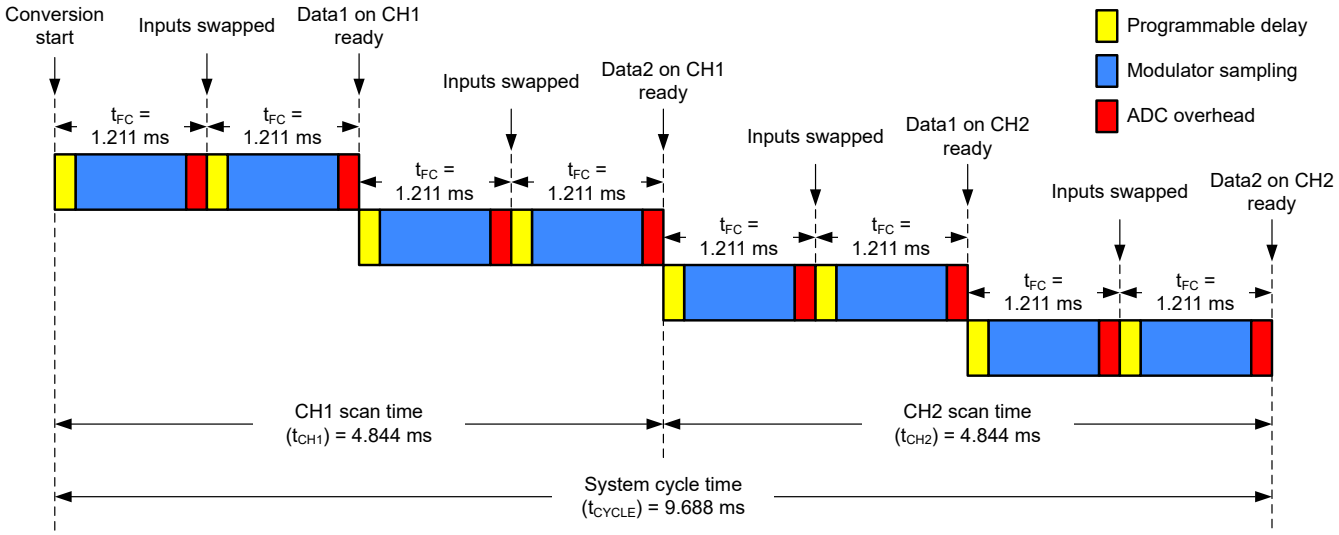



**Figure 8-5. Timing Diagram for Example #5**

## 8.6 Example #6: Scanning Two Channels With Different System Parameters

Table 8-6 reports the system parameters that determine the cycle time in Example #6:

**Table 8-6. System Parameters for Example #6**

| PARAMETER | | VALUE |
|---|---|---|
| ADC | | ADS124S08 |
| Clock frequency | | 4.5 MHz |
| # of channels | | 2 |
| Channel 1 (CH1) | ODR | 800 SPS |
| | Filter type | *Low-latency* |
| | Conversion mode | Continuous |
| | Programmable delay | $14 \cdot t_{MOD}$ (default) |
| | Chopping | Enabled |
| | Conversions per channel | 2 |
| Channel 2 (CH2) | ODR | 200 SPS |
| | Filter type | $Sinc^3$ |
| | Conversion mode | Single-shot |
| | Programmable delay | $256 \cdot t_{MOD}$ |
| | Chopping | Disabled |
| | Conversions per channel | 3 |

Example #6 introduces different parameters for each of the two channels. This requires analyzing CH1 and CH2 separately to determine their respective scan times, $t_{CH1}$ and $t_{CH2}$.

To determine $t_{CH1}$, consider that the ADC is operating at 800 SPS, the *low-latency* filter is used, the ADC is operating in continuous-conversion mode, the default programmable delay has been selected, and chop is enabled. Additionally, this system does not use the default clock frequency, $f_{CLK}$, of 4.096 MHz. Instead, it is necessary to accommodate a new clock frequency, $f_{CLK\_NEW}$, of 4.5 MHz.

First, identify the first conversion latency for CH1, $t_{FC\_CH1}$, using the ADS124S08 *low-latency* filter at 800 SPS. Even though continuous-conversion mode is used for CH1, second and subsequent conversion latency does not apply because global chop is enabled. While not shown in this document, the *ADS124S08* data sheet identifies that $t_{FC\_CH1} = 360 \cdot t_{MOD}$ periods using the *low-latency* filter at ODR = 800 SPS. Using the conversion latency in terms of $t_{MOD}$ periods instead of milliseconds enables easier calculations because the number of $t_{MOD}$ periods is independent of clock frequency.

The time specified by $t_{FC\_CH1}$ includes the ADC overhead but does not include the programmable delay, $t_{DELAY}$. Using the default value of $t_{DELAY} = 14 \cdot t_{MOD}$, the total first conversion latency for CH1, $t_{FC\_CH1\_TOTAL}$, is given by Equation 26:

$$t_{FC\_CH1\_TOTAL} = t_{FC\_CH1} + t_{DELAY} = 360 \cdot t_{MOD} + 14 \cdot t_{MOD} = 374 \cdot t_{MOD} \quad (26)$$

Next, Equation 27 calculates one $t_{MOD}$ period in terms of $f_{CLK\_NEW}$ using the ADS124S08:

$$t_{MOD} = 16 / f_{CLK\_NEW} \quad (27)$$

At $f_{CLK\_NEW} = 4.5$ MHz, $t_{MOD} = 3.56$ µs. Therefore, $t_{FC\_CH1\_TOTAL} = 374 \cdot 3.56$ µs = 1.331 ms.

As described in Section 5.6, each conversion result when global chop is enabled is an average of two conversions. Producing the data for each conversion requires $1 \cdot t_{FC\_CH1\_TOTAL}$ due to input swapping, and therefore the first conversion result with global chop enabled requires $2 \cdot t_{FC\_CH1\_TOTAL}$. However, in continuous conversion mode, the previous conversion can be averaged with the next conversion to generate the second conversion result. This second conversion result is therefore only subject to $1 \cdot t_{FC\_CH1\_TOTAL}$. Figure 8-6 shows how this behavior affects the conversion latency for each conversion result from CH1.

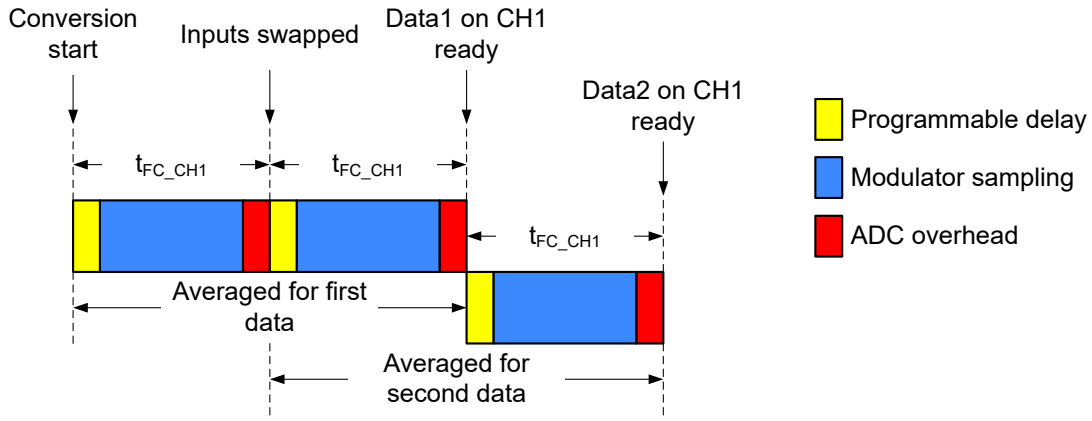



**Figure 8-6. Timing Diagram for CH1 in Example #6**

Figure 8-6 reveals that the scan time for CH1, $t_{CH1}$, is given by Equation 28:

$$t_{CH1} = 3 \cdot t_{FC\_CH1\_TOTAL} = 3 \cdot 1.331 \text{ ms} = 3.993 \text{ ms} \quad (28)$$

Note that CH1 only yields two conversion results even though $t_{CH1}$ is comprised of $3 \cdot t_{FC\_CH1\_TOTAL}$. This is specifically due to global chop being enabled and operating in continuous-conversion mode.

To determine $t_{CH2}$, consider that the ADC is sampling at 200 SPS, the $sinc^3$ filter is used, the ADC is operating in single-shot conversion mode, the programmable delay is $256 \cdot t_{MOD}$, and chop is disabled. Additionally, the same clock frequency from CH1 is used for CH2, where $f_{CLK\_NEW} = 4.5$ MHz.

First, Table 5-1 identifies that the first conversion latency for CH2, $t_{FC\_CH2}$, is equal to $3905 \cdot t_{MOD}$ periods using the ADS124S08 $sinc^3$ filter at 200 SPS. Using the conversion latency in terms of $t_{MOD}$ periods instead of milliseconds enables easier calculations because the number of $t_{MOD}$ periods is independent of clock frequency.

The time specified by $t_{FC\_CH2}$ includes the ADC overhead but does not include $t_{DELAY}$. Using the example value of $t_{DELAY} = 256 \cdot t_{MOD}$, the total first conversion latency for CH2, $t_{FC\_CH2\_TOTAL}$, is given by Equation 29:

$$t_{FC\_CH2\_TOTAL} = t_{FC\_CH2} + t_{DELAY} = 3905 \cdot t_{MOD} + 256 \cdot t_{MOD} = 4161 \cdot t_{MOD} \quad (29)$$

Therefore, $t_{FC\_CH2\_TOTAL} = 4161 \cdot 3.56$ µs = 14.813 ms because $t_{MOD}$ was calculated to be 3.56 µs for CH1 when $f_{CLK\_NEW} = 4.5$ MHz. Additionally, the use of single-shot conversion mode means that all three conversion results required by CH2 are subject to first conversion latency. Assuming that the user starts the next conversion on CH2 immediately after the previous conversion result is ready, Equation 30 calculates $t_{CH2}$:

$$t_{CH2} = 3 \cdot t_{FC\_CH2\_TOTAL} = 3 \cdot 14.813 \text{ ms} = 44.439 \text{ ms} \quad (30)$$

Finally, there is no need to consider additional latency for CH2 due to chopping. Equation 31 calculates the cycle time, $t_{CYCLE}$, by summing the scan time for each channel:

$$t_{CYCLE} = t_{CH1} + t_{CH2} = 3.993 \text{ ms} + 44.439 \text{ ms} = 48.432 \text{ ms} \quad (31)$$

Ultimately, the cycle time for this example is 48.432 ms for 5 conversion results. Figure 8-7 depicts a timing diagram for the example system given the design parameters.

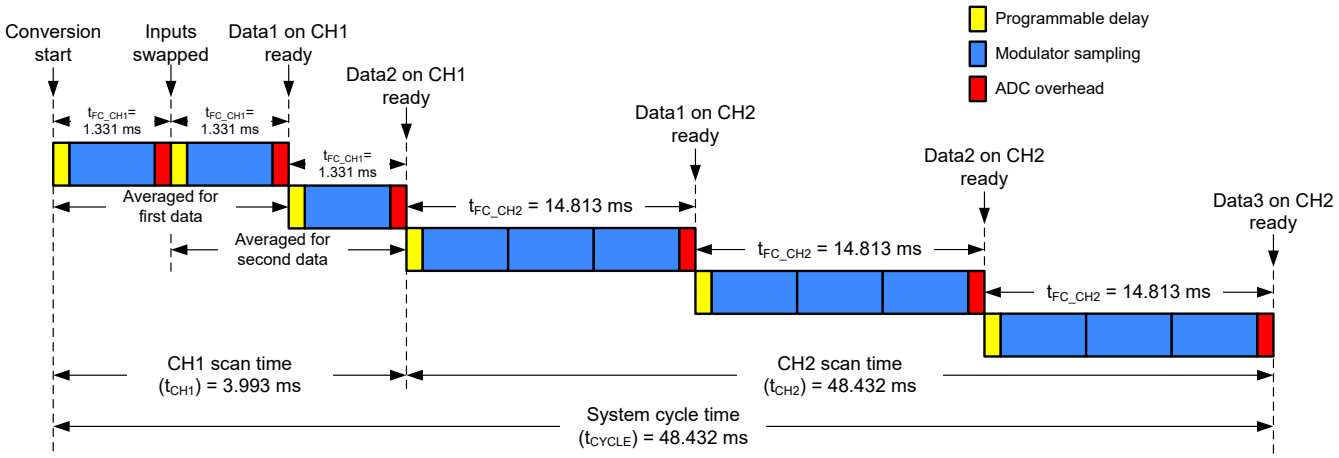



**Figure 8-7. Complete Timing Diagram for Example #6**

## 8.7 Example #7: Using the ADS1261

Table 8-7 reports the system parameters that determine the cycle time in Example #7:

**Table 8-7. System Parameters for Example #7**

| PARAMETER | VALUE |
|---|---|
| ADC | ADS1261 |
| ODR | 4800 SPS |
| Filter type | $sinc^4$ |
| Clock frequency | 7.3728 MHz (default) |
| Conversion mode | Continuous |
| Programmable delay | 50 µs (default) |
| Chopping | Disabled |
| Conversions per channel | 3 |
| # of channels | 2 |

Unlike the previous examples that used the ADS124S08, Example #7 uses the ADS1261. Consequently, the default clock frequency and programmable delay time are different, as are the ODR and filter type options. However, the process for determining the cycle time remains the same.

Refer to Table 2-2 to identify the ADS1261 first conversion latency, $t_{FC}$, using the $sinc^4$ filter and ODR = 4800 SPS. This is given as 1.258 ms and includes the default programmable delay time of 50 µs as well as any ADC overhead. Second and subsequent conversion latency, $t_{SSC}$, is not provided directly in the ADS1261 data sheet. Instead, the *Conversion Latency* section in the ADS1261 data sheet states that $t_{SSC}$ = 1 / ODR when continuous conversion mode is used and chop is disabled. Since both of these conditions are true for this example, $t_{SSC}$ is given by Equation 32:

$$t_{SSC} = 1 / ODR = 1 / 4800 = 0.208 \text{ ms} \quad (32)$$

Finally, there is no need to consider additional latency due to chopping. Equation 34 calculates the cycle time, $t_{CYCLE}$, using the scan time for one channel, $t_{CH}$, that results from Equation 33:

$$t_{CH} = 1 \cdot t_{FC} + 2 \cdot t_{SSC} = 1 \cdot 1.258 \text{ ms} + 2 \cdot 0.208 \text{ ms} = 1.674 \text{ ms} \quad (33)$$

$$t_{CYCLE} = \#\text{ of channels} \cdot t_{CH} = 2 \cdot 1.674 \text{ ms} = 3.348 \text{ ms} \quad (34)$$

Ultimately, the cycle time for this example is 3.348 ms for 6 conversion results. Figure 8-8 depicts a timing diagram for the example system given the design parameters.

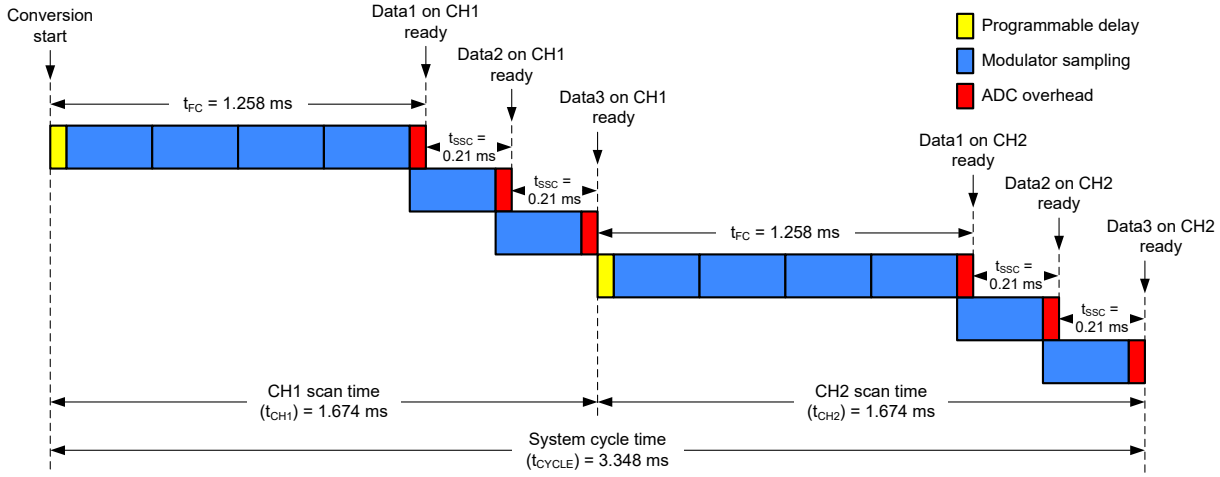



**Figure 8-8. Timing Diagram for Example #7**

## 8.8 Example #8: Changing Multiple Parameters Using the ADS1261

Table 8-8 reports the system parameters that determine the cycle time in Example #8:

**Table 8-8. System Parameters for Example #8**

| PARAMETER | VALUE |
|---|---|
| ADC | ADS1261 |
| ODR | 600 SPS |
| Filter type | $sinc^2$ |
| Clock frequency | 4 MHz |
| Conversion mode | Pulse-convert |
| Programmable delay | <Not default - see example description> |
| Chopping | Enabled |
| Conversions per channel | 2 |
| # of channels | 2 |

Example #8 uses the same ADC from Example #7 (ADS1261), though almost all of the system parameters have been changed. These changes include a 60-SPS ODR, a $sinc^2$ filter, a 4-MHz clock frequency ($f_{CLK\_NEW}$), pulse-convert mode (similar to single-shot mode), chopping enabled, and two conversion results per channel instead of three. Additionally, the programmable delay has been changed from the default value, though the specific value in milliseconds depends on the clock frequency and is calculated during the example.

Similar to Example #5 where chop was enabled using single-shot mode, both conversion results on each channel using the new example parameters require two first-conversion latency periods. Table 2-2 identifies that the first conversion latency, $t_{FC}$, is 33.76 ms for the ADS1261 using the $sinc^2$ filter and ODR = 60 SPS. In this case, $t_{FC}$ is derived using the default clock frequency, $f_{CLK}$, of 7.3728 MHz and includes the default 50-µs programmable delay, $t_{DELAY\_DEFAULT}$. It is therefore necessary to scale $t_{FC}$ based on $f_{CLK\_NEW}$ = 4 MHz and apply the new programmable delay.

Unlike the ADS124S08, the ADS1261 data sheet only provides $t_{FC}$ in milliseconds, not $t_{MOD}$ periods. To translate to a new conversion latency value, $t_{FC\_NEW}$, first remove $t_{DELAY\_DEFAULT}$. Then, scale $t_{FC}$ by the ratio of $f_{CLK}$ to $f_{CLK\_NEW}$ as per Equation 35:

$$t_{FC\_NEW} = (t_{FC} - t_{DELAY\_DEFAULT}) \cdot (f_{CLK} / f_{CLK\_NEW}) \quad (35)$$

Applying the values from this example yields a value for $t_{FC\_NEW}$ given by Equation 36:

$$t_{FC\_NEW} = (33.76\ ms - 0.05\ ms) \cdot (7.3728\ MHz / 4\ MHz) = 62.134\ ms \quad (36)$$

The programmable delay options in the ADS1261 are also specified in milliseconds and referenced to $f_{CLK}$. Perform a similar operation as shown in Equation 36 to scale any new programmable delay value, $t_{DELAY\_NEW}$, for $f_{CLK\_NEW}$. For this example, choose a nominal programmable delay, $t_{DELAY\_NOM}$, of 328 µs from the MODE1 register in the ADS1261 data sheet. Then, scale $t_{DELAY\_NOM}$ by the ratio of $f_{CLK}$ to $f_{CLK\_NEW}$ to get a value for $t_{DELAY\_NEW}$ as per Equation 37:

$$t_{DELAY\_NEW} = t_{DELAY\_NOM} \cdot (f_{CLK\_NOM} / f_{CLK\_NEW}) \quad (37)$$

In this example, $t_{DELAY\_NEW}$ = 328 µs · (7.3728 MHz / 4 MHz) = 0.605 ms. As a result, the total conversion latency, $t_{FC\_TOTAL}$, is given by Equation 38:

$$t_{FC\_TOTAL} = t_{FC\_NEW} + t_{DELAY\_NEW} = 62.134\ ms + 0.605\ ms = 62.739\ ms \quad (38)$$

As stated previously, chopping requires two first-conversion latency periods ($2 \cdot t_{FC\_TOTAL}$) per conversion result. With two conversion results per channel and assuming that the user starts the next conversion immediately after the previous conversion result is ready, Equation 40 calculates the cycle time, $t_{CYCLE}$, using the scan time for one channel, $t_{CH}$, that results from Equation 39:

$$t_{CH} = 2 \cdot (2 \cdot t_{FC\_TOTAL}) = 4 \cdot 62.739 \text{ ms} = 250.956 \text{ ms} \quad (39)$$

$$t_{CYCLE} = \#\text{ of channels} \cdot t_{CH} = 2 \cdot 250.956 \text{ ms} = 501.912 \text{ ms} \quad (40)$$

Ultimately, the cycle time for this example is 501.912 ms for 4 conversion results. Figure 8-9 depicts a timing diagram for the example system given the design parameters.

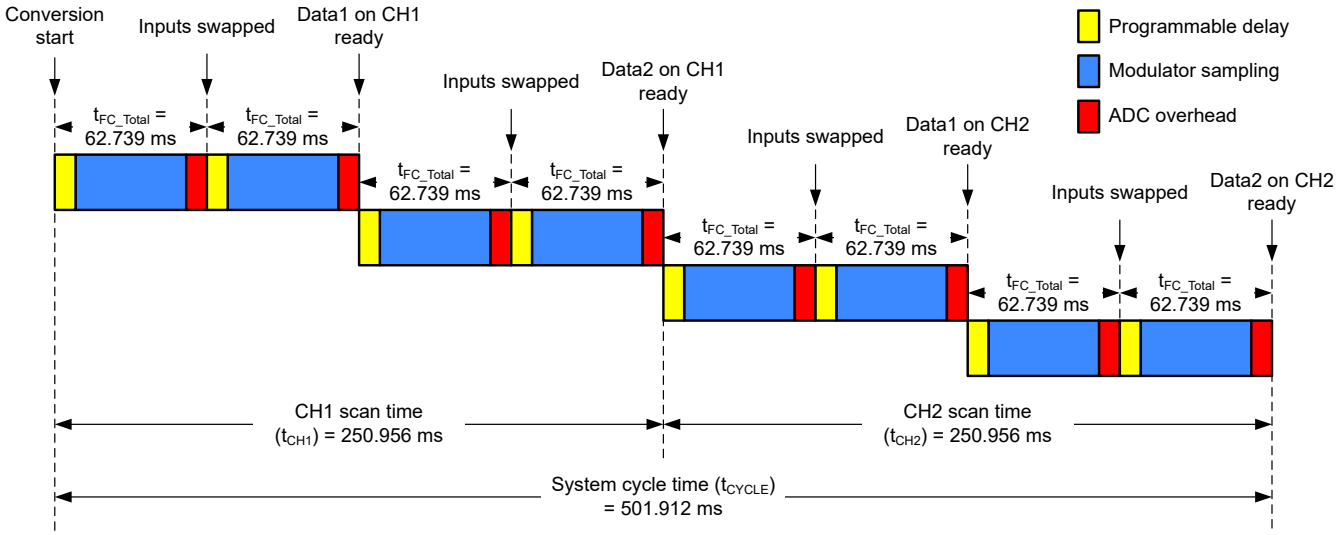



**Figure 8-9. Timing Diagram for Example #8**

## 9 Summary

Understanding all of the factors that affect ADC conversion latency is imperative to help confidently choose an ADC that meets the desired system cycle time. This application note offers a detailed discussion of these important factors as well as provides several examples to enable a deeper understanding of how to calculate the system cycle time using information from the ADC data sheet.

# 10 Revision History

| Changes from Revision * (March 2022) to Revision A (March 2024) | Page |
|---|---|
| • Updated the numbering format for tables, figures, and cross-references throughout the document | 1 |

# IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265