White Paper
SBAY002–December 2016

Enabling Faster, Smarter, and More Robust System Solutions for SAR ADCs With TI's multiSPI™ Digital Interface

Ameet Bagwe *Precision Analog, SAR ADC*

Learn about TI's multiSPI™ digital interface, a proprietary technology that reduces design complexity and cost, as well as debug cycle times, for SAR ADC-based data-acquisition systems.

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

Industrial systems, like test and measurement, are constantly targeting higher precision and throughput analog-to-digital converters (ADCs). High-precision data conversion provides more bits of data per sample, while higher throughput provides more samples per second. When combined, these two features result in a demand for extremely high transfer rates between the ADC and the host. This allows for the data from one conversion to be transmitted from the ADC to the host before the next conversion data becomes available inside the ADC. In some cases, this combination can result in system constraints that are either technically infeasible or overly complex to address. Extreme system design constraints can result in undesirable consequences such as raising the overall cost of the solution, or causing engineers to spend an inordinate amount of time during initial bring-up and debug of data transfer errors. Occasionally, these challenges can be insurmountable for the legacy serial peripheral interface (SPI) protocol, which is the data transfer protocol offered by most ADCs.

So that system engineers can overcome these challenges, Texas Instruments has developed the multiSPI™ digital interface. multiSPI technology addresses these issues in an innovative way, completely eliminating a number of factors that drive up complexity and cost from the host and board-design perspective. While Texas Instruments introduced the multiSPI digital interface with the ADS9110 18-bit 2 MSPS successive approximation analog-to-digital converter (SAR ADC), multiSPI technology will be present (in various feature sets) on all TI SAR ADCs moving forward. This paper delves into the details of the multiSPI technology, and how a system engineer can take advantage of its various features to optimize the final solution.

2 Typical Host and ADC Communication Setup

Before examining the details of system design issues caused by high-precision and throughput rates, first we need to establish an understanding of the basic setup. Figure 1 illustrates a typical ADC and host communication setup. Note that the pins comprise a superset of what may be used. The four pins enclosed in the dotted box represent the traditional serial peripheral interface (SPI) signals.

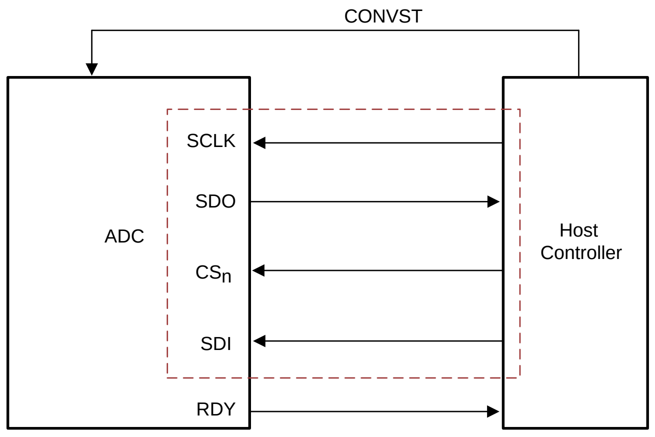

Figure 1. Signals Being Used for Sampling Control and Data Transfer Between a Host and the ADC

Pins used for sampling and data transfer are:

- CONVST (conversion start): The signal that controls the sampling rate of the ADC . Each positive edge on the CONVST indicates the start of a new conversion. The ADC begins to convert the analog input being sampled (sampling continues until the positive edge on the CONVST).
- SCLK (serial clock): The clock used to transfer bits into or out of the ADC.
- SDO (serial data out): The data output line from the ADC. Conversion data or register data is transferred from the ADC to the host on the SDO line.
- CSn (chip select number): The active low chip select line that indicates the data transfer boundaries. Data transfer is considered active when this signal is low. The SDO line is inactive or tri-stated when this signal is high.
- SDI (serial data in): The data input line into the ADC. The host uses this input to configure the ADC (by writing registers in the device).
- RDY (ready): The ready signal goes low when a conversion begins, and goes high once the conversion is complete. The host can sample this signal to decide when the data is ready to be transmitted out of the ADC

Note that the previously-listed pins may be combined or omitted, depending on what the application needs.

In some cases, the CONVST and CSn pins may be connected together, either on the board or internally within the ADC. This is useful in pin-constrained applications, or when the host is configured to control both sampling and data transfer with one pin.

The RDY signal is not mandatory for the data transfer. Functionally, it indicates that the ADC has completed its conversion. The host can choose to monitor this signal before it initiates data transfer, or it can wait for the conversion time as specified in the data sheet.

The SDI pin may not be available in devices that do not support any form of configuration or read and write access to internal registers.

The SDO pin gets tri-stated when the CSn pin is high, permitting the SDOs from multiple ADCs to be connected together, thus saving pins on the host. The host can decide which ADC is actively driving the SDO by activating its corresponding CSn pin.

The multiSPI technology (Figure 2) combines the standard SPI signals (red dashed line box in Figure 1) and the RDY pin to offer a number of advanced features. For the remainder of this document, the multiSPI digital interface will refer to all of the signals in Figure 2 except the CONVST. This digital interface has been designed to be completely independent of the sampling state or conversion state of the ADC. The host can choose to read data from the ADC at any point in time, and the data received will reflect the value of the last successfully completed conversion.

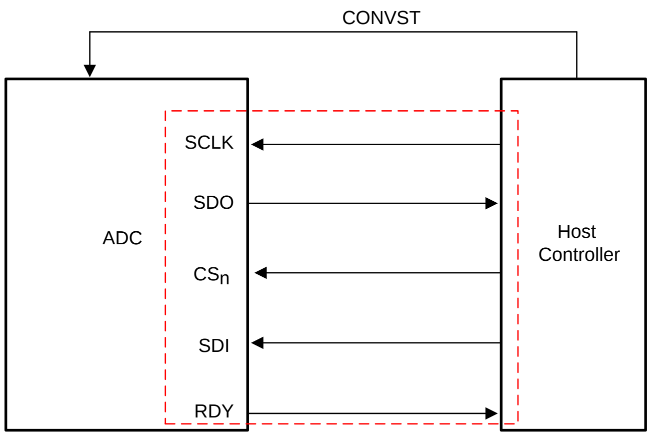

Figure 2. Pins Used by the multiSPI Digital Interface

The host depicted in Figure 1 and Figure 2 can be a field-programmable gate array (FPGA), a digital signal processor (DSP), or a microcontroller. This choice is based on a number of factors like the data transfer capability, data processing bandwidth, on-board memory and, last but not the least, the cost.

2.1 Conversion Frames and Data Transfer Windows

Figure 3 illustrates a couple of conversion frames. Each frame has a well-defined set of characteristics.

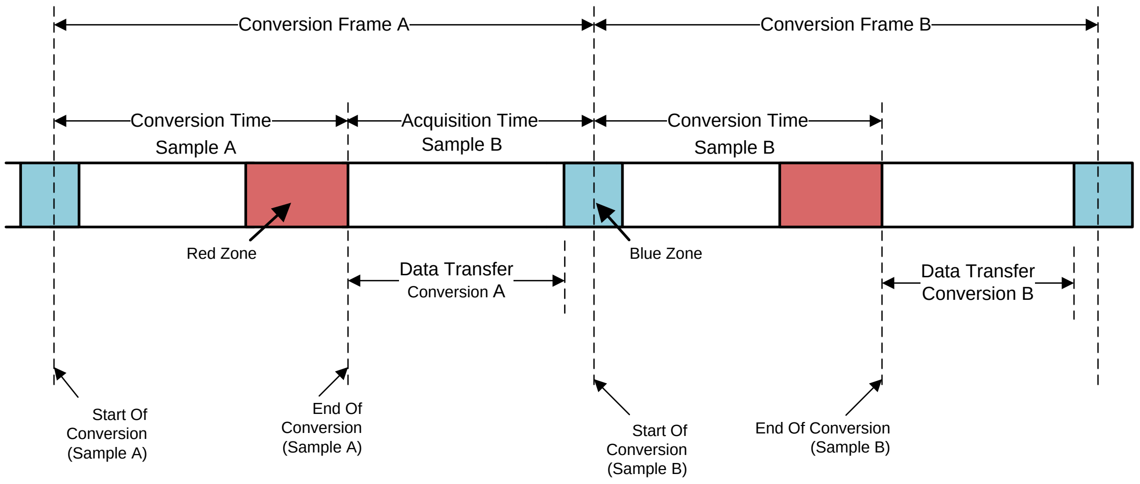

Figure 3. Conversion and Data Transfer Windows Present Within Each Frame

For example, the conversion frame for sample A has the following defining characteristics:

- The point where the frame begins is where the conversion for sample A is said to begin inside the SAR ADC. The time duration between the starting point of consecutive frames essentially determines the sampling rate of the ADC.
- The 'conversion time' for sample A is the time taken by the ADC to prepare the final data corresponding to the analog input acquired for the sample. After the conversion is complete, the ADC resumes sampling and acquisition of the analog input for the next sample (sample B).
- The 'acquisition time' for sample B is the time allowed for the analog input to be acquired. It also coincides with the time when the data value of sample A is transferred from the ADC to the host.

2.2 Critical (Silent) Zones Within Conversion Frames

The blue and red areas in Figure 3 are critical 'silent' zones, when it is recommended that there should be minimal (or zero) digital switching activity at the SDO pin of the AD (or any other digital output pin).

The blue area is the silent zone around the beginning of a conversion (or end of the sample acquisition) where any digital activity can potentially couple into the acquired analog sample as noise.

The red area is the silent zone near the end of conversion (where noise from digital activity can erroneously impact the critical least significant bits (LSBs) being computed).

Digital activity during these zones can result in a degradation of critical ADC parameters such as the differential nonlinearity (DNL) and the signal-to-noise ratio (SNR). The reason for mentioning these 'silent' zones is that an application may attempt to lower the data transfer clock rates while keeping the sampling rate the same, by extending the data transfer (for instance, Frame A) into the blue zone of Frame B, and possibly even into the red zone of Frame B. This is possible in practice, and discussed in Section 4.

3 High Data Transfer Rates: Causes and Effects

If we apply typical data sheet timing requirements to Figure 3, a few things immediately become obvious. As the sampling rates and ADC precision increase, the data transfer window (time available to transfer data bits) shrink. Let us consider an application that needs 18-bit precision at 2 MSPS. This results in each frame being 500 ns long. Now assume that out of the available 500 ns, the typical conversion time is approximately 330 ns. As the data transfer cannot begin until the conversion has been completed, the host has 170 ns to transfer the data before the next conversion cycle begins.

The host functionality should account for one more concern. The data transfer must finish before the silent blue zone begins. Otherwise, there may be degradation to the next conversion. If we assume that the silent blue zone is 15 ns, 155 ns (out of the initial 170 ns) remain for the host to transmit 18 bits from the ADC. The required data transfer rate is approximately 115 Mbps, which, if the extra half-cycle loss in a standard SPI protocol is added, results in an SCLK rate in excess of 120 MHz. Due to various system constraints, this speed is either at the high-end or beyond the capability of most typical ADC or host systems – especially if the host is a microcontroller. Often the only solution is to use a high-performance and possibly more expensive field-programmable gate array (FPGA) as the host, or a solution that increases overall system complexity.

The multiSPI digital interface introduces a number of elegant features that address the data transfer rate, while simultaneously eliminating any burden or complexity at the host end. These features offer the system designer the ability to:

- Run the transfer at extremely high clock speeds, even if individual component delays are large
- Decrease the clock speeds by widening the data transfer window, enabling lower complexity hosts or simpler board designs.
- Decrease clock speeds by increasing the number of data pins or the active clock edges
- Combine any of the above features to achieve the optimal solution for the target application
- Debug the host and ADC interaction easily, reducing design cycle time and iterations

These features are discussed in detail in the rest of this paper, with special notes about how each can be applied to different hosts (FPGAs or microcontrollers).

3.1 High Clock Rates versus System Constraints

Before using the multiSPI digital interface to address the difficulty of supporting high clock rates, look at exactly how the various delays encountered in the system can lead to a seemingly impossible set of constraints. Figure 4 illustrates the typical timing parameters (delays) between the host and the ADC. These are of interest to the system engineer, as they determine the overall clock frequency.

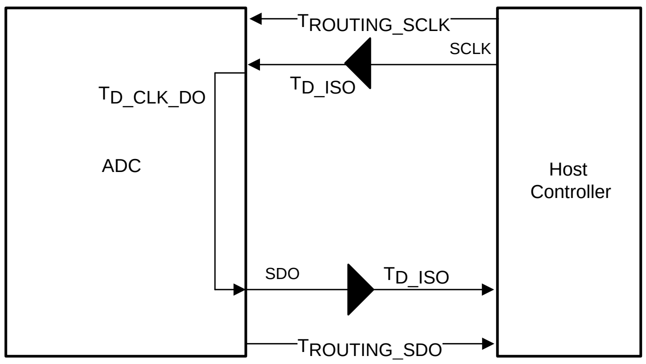

Figure 4. Data Transfer Delays Encountered in a Host or ADC System

Figure 4 concentrates on the data transfer operation from the ADC to the host, which is the speed-limiting operation in the setup. The delays encountered by the signals are:

- $T_{ROUTING\_SCLK}$: SCLK trace routing delay on the printed-circuit-board (PCB)
- $T_{ROUTING\_SDO}$: SDO trace routing delay on the PCB
- $T_{D\_CLK\_DO}$: SCLK to SDO output delay of the ADC
- T_{D_ISO}: The delay caused by any isolators or level translators or digital buffers between the host and ADC

For the sake of simplicity, the input setup time of the host (for receiving the SDO data) is not considered.

Once these delays are determined, the maximum SCLK rate can be calculated from Equation 1 and Equation 2:

$$T_{MIN\_SCLK} = T_{ROUTING\_SCLK} + T_{ROUTING\_SDO} + T_{D\_CLK\_DO} + 2 \times T_{D\_ISO} \quad (1)$$

$$F_{MAX\_SCLK} = \left(\frac{1}{T_{MIN\_SCLK}}\right) \tag{2}$$

Equation 1 and Equation 2 assume that an *entire* SCLK cycle is used for the data transfer. In a true SPI protocol, the data launch and capture happen at opposite edges of the SCLK, which will actually decrease the maximum frequency by a factor of two.

These equations stipulate that the maximum SCLK frequency *cannot* exceed the constraints imposed by the sum of the individual delays. This restriction becomes significantly worse if isolators are present on the data and clock lines. These isolators add to the overall time budget, causing the resultant SCLK speed to be unacceptably low. Usually, board designers can eliminate the routing and isolator delays from the equation by running an extra trace that mirrors the SCLK path from the host to the ADC, and mirrors the SDO path from the ADC back to the host.

However, this approach increases the board design effort and adds the cost of two extra isolators. Finally, even if all of these delay cancellation measures are taken into account, the output delay of the ADC is still a major factor. For example, if the $T_{D\_CLK\_DO}$ parameter is around 15 ns, the SCLK will be limited to 66.67 MHz, even if the rest of the delays in Figure 4 were set to zero. If the other delays are accounted for, the maximum possible SCLK can drop further to 50 MHz or below, severely limiting the application throughput.

3.2 Enabling High SCLK Rates

The multiSPI technology solves the SCLK frequency limitation problem by moving the timing control of the data transfer from the host to the ADC. This mode of operation is called the source-synchronous mode. For ease of explaining how this mode fits into the existing infrastructure, Figure 5 illustrates all the multiSPI digital interface signals with the particular signals of interest (blue lines). The SDI is illustrated with a dashed line because it is not required for the source-synchronous mode. The RDY pin is illustrated as RDY_STRB in Figure 5 to convey that it now serves a dual purpose (explanation follows).

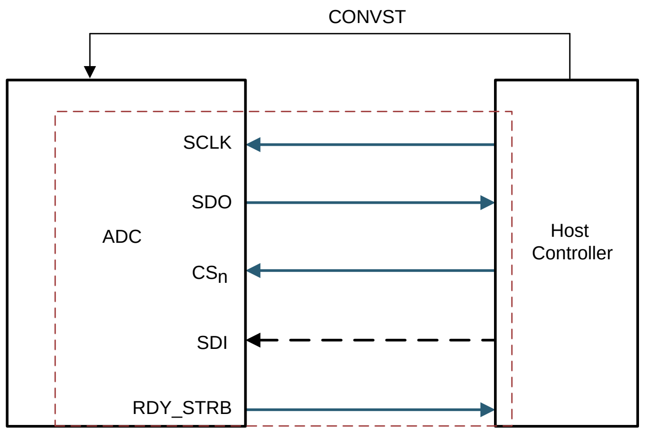

Figure 5. Signals Used for Source Synchronous Data Transfer

In the source-synchronous mode, the RDY pin of the ADC functions as a strobe (STRB) for the data exported on the SDO pin. In simplified terms, the STRB now acts as an accompanying clock for the data. Each new data bit on the SDO pin is launched with a corresponding positive edge of the STRB. The STRB and SDO pins need to be routed to the host over length-matched traces, where the host can simply use the STRB as a clock to capture the SDO into its input data buffers and FIFOs. The steps involved are:

1. The host activates CSn to indicate the start of a data transfer, and starts providing SCLK pulses.
2. The ADC returns STRB pulses and SDO bits to the host where the STRB is used to capture the SDO.
3. The device sends only as many STRB cycles as there are valid data bits. As such, the host does not need to control the number of SCLK pulses generated, as long as it generates the minimum number of pulses needed for receiving all data bits. For example, if the ADC is providing 18 bits of data, if the host provides 24 SCLK pulses, the ADC still only generates 18 STRB pulses. This ensures that only 18 bits are captured inside the buffer and FIFO in the host.

4. The only thing that the control logic in the host needs to do is to wait for the desired number of bits to be captured and then deactivate the CSn to complete the frame transfer.

The waveform in Figure 6 illustrates a source synchronous data transfer. The figure shows an extra SCLK cycle at the end, which is not reflected as a cycle on the STRB because the ADC stops generating STRB pulses once it has transmitted out the LSB. Additionally, the SDO and STRB signals start aligned from the ADC, even though they will have a delay with respect to the SCLK at the input of the ADC. As long as the SDO and STRB are routed as length or delay matched traces, they will arrive aligned at the host. This places the negative edge of the STRB right in the middle of the stable data eye of the SDO. Now the STRB negative edge can be used reliably as the clock for capturing the SDO.

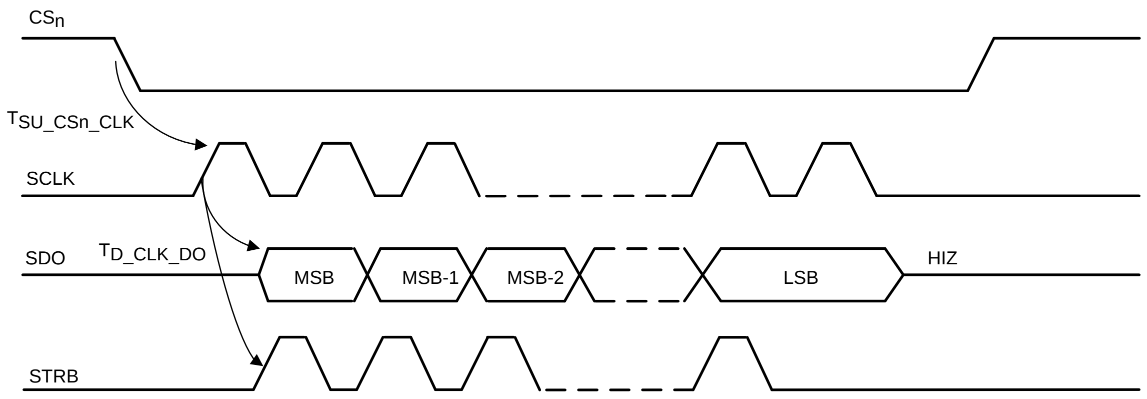

Figure 6. Source Synchronous Data Transfer

The source-synchronous mode of signaling brings in a number of advantages, both from a performance and system design perspective.

The delays listed in Equation 1 have no bearing on the maximum clock (STRB) frequency. This is because the STRB and SDO travel in the same direction. The frequency can be as high as the SDO or STRB switching rates supported by the host, the ADC and the intermediate isolators or buffers on the signal chain. The ADS9110 supports SCLK rates up to 100 MHz, which should meet the requirements of most high-throughput applications.

Board design constraints can be relaxed as there is no need to locate the host and ADC extremely close to each other. The only requirement is that the SDO and STRB traces have to be length-matched, but the overall trace length is no longer a constraint for the board designer.

The RDY pin of the ADC is re-used as the STRB, so the ADC does not impose a pin count impact on the system.

Figure 6 illustrates one final advantage of the source-synchronous mode. As the STRB and SDO are both generated by the ADC, the SCLK pin can now be considered redundant for data transfer operations from the host to the ADC. The ADC can generate the STRB using its own internal oscillator, thereby allowing the host to keep the SCLK switched off for most of the application. The only exception is when it is needed along with the SDI to configure registers inside the ADC.

When the ADC is set up to use its own internal oscillator to generate the pulses on the STRB, it is in internal clock source mode. If configured to use the SCLK to generate the STRB, then the ADC is in SCLK reflect mode. Refer to the device data sheet for more details on how you can select between these two modes.

Furthermore, when the internally-sourced option is enabled, the resultant STRB can be pre-divided by factors of 1, 2, and 4 (can vary across devices) to offer extra frequency options for the system designer. In the ADS9110, the internal source mode can generate STRB frequencies of 100 MHz, 50 MHz, and 25 MHz.

When using the SCLK reflect mode to generate the STRB, the system designer can use Equation 3 to approximate the total time required for the overall data transfer, or the time for the CSn to be held active:

$$T_{DATA\_TRANSFER} = N \times T_{SCLK} + T_{D\_CLK\_DO} + T_{SU\_CSn\_CLK}$$

where

- N = Number of bits to be transmitted (3)

In Equation 3, you may wonder why the delay $T_{D\_CLK\_DO}$ still appears in the overall data transfer time. This is because this delay reflects the time taken for a change on the SCLK to reflect as a change on the SDO and STRB. In normal SPI modes, this delay impacts T_{SCLK} and appears as a delay in all the bits being transmitted. In turn, this increases the amount of time taken in data transfer per frame, and lowers overall throughput or sampling rate of the system. In the source-synchronous mode, this delay appears only once, while T_{STRB} can be much lower. This enables lower time spent for the overall data transfer and consequently higher throughput. The final parameter ($T_{SU\_CSn\_CLK}$) is the setup time requirement between CSn going low and the first positive edge on the SCLK.

Figure 7 illustrates a source synchronous data transfer using the internal clock source mode where the STRB and SDO are generated, using the internal clock of the ADC.

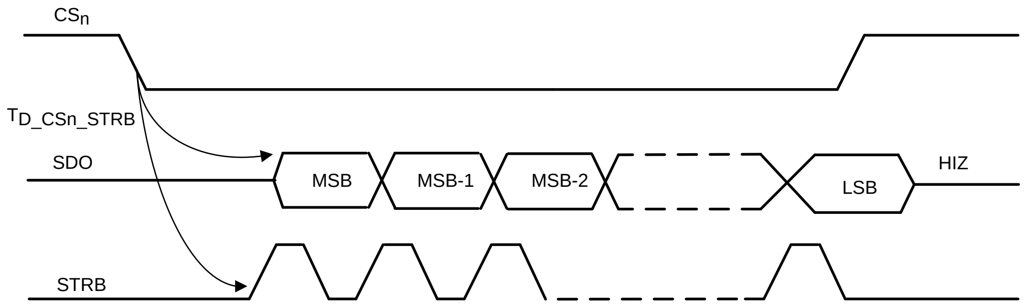

Figure 7. Source Synchronous Data Transfer With an Internal Clock Source Mode for STRB Generation

In this mode, the system designer can use Equation 4 to approximate the total time required for the overall data transfer, or the time for the CSn to be held active:

$$T_{DATA\_TRANSFER} = N \times T_{STRB} + T_{D\_CSn\_STRB}$$

where

- N = Number of bits to be transmitted (4)

In Equation 4, the last parameter on the right is the delay taken by the ADC to start generating the STRB pulses after it first detects a falling edge on the CSn.

3.2.1 Using the multiSPI Source-Synchronous Mode With FPGAs

The architecture of FPGAs allows for a seamless integration with the source-synchronous mode of the multiSPI (Figure 8).

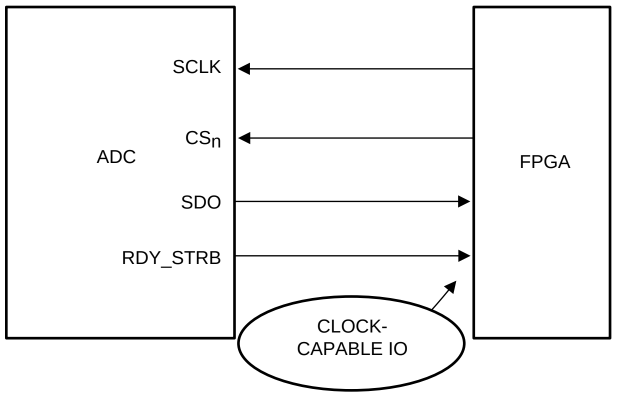

Figure 8. Using Source-Synchronous Mode With FPGA-Based Hosts

The board designer needs to ensure that the RDY_STRB signal from the ADC to the host is routed on a clock-capable IO of the FPGA. The SDO can be routed to any of the FPGA IOs. However, be careful to follow the bank grouping rules of the FPGA because a regional clock buffer in an FPGA can only service a select group of IO pins. Once the traces are routed this way, the FPGA logic designer can implement a simple receive first-in, first-out (FIFO) that uses the STRB as a clock pin to capture the SDO. The CSn generated by the FPGA can be used internally within the FPGA to allow the FIFO to distinguish between frame boundaries, and hence store results of different conversions in different data words.

4 Techniques for Decreasing the SCLK Rates

There are avenues to further reduce the frequency of SCLK rate without decreasing the overall throughput or the sampling rate. To understand this better, revisit Figure 3, which is re-illustrated as Figure 9 with one modification. The data result from conversion of sample A is transferred from the ADC to the host in the next frame.

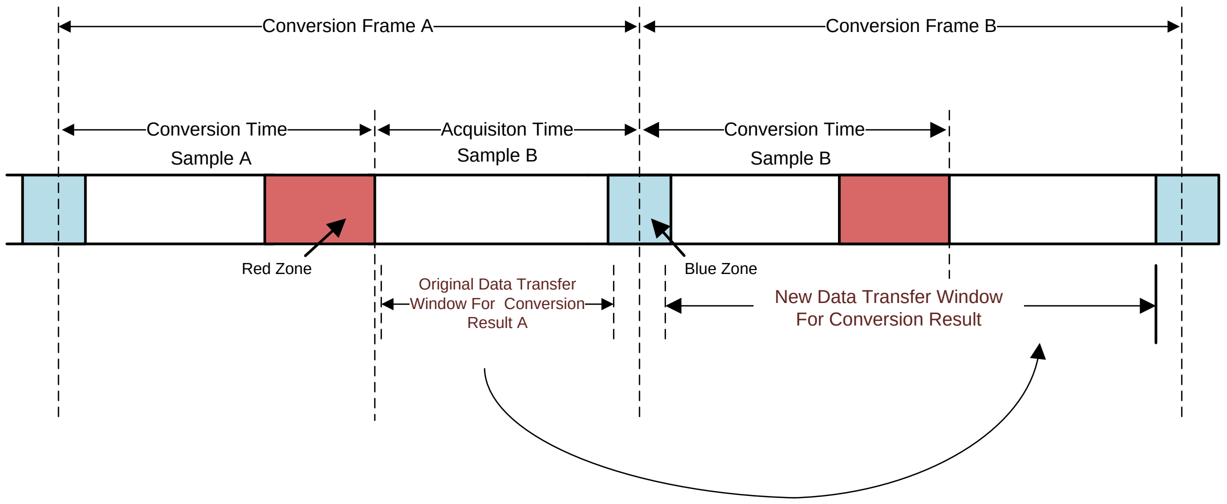

Figure 9. Frames With Data Conversion and Data Transfer Windows

Taking a closer look at Figure 9, the overall sampling rate is the same as Figure 3, except that the data transfer operation for the result of conversion A was moved entirely into frame B. To differentiate this from the usual data transfer window, identify it as the Latency 1 transfer mode. This nomenclature is used because the conversion generated in one frame is available to the host in the *next* frame (latency of 1 frame).

Figure 9 illustrates that Latency 1 allows the host significantly more time to transfer out the data bits. In turn, this enables a lowering of the SCLK rate without compromising the overall sampling rate. A simple calculation using the timing parameters of the device reveals that using this approach can lower the required SCLK rate from 120 MHz to approximately 70 MHz (while maintaining a sampling rate of 2 MSPS).

Should the SDO pin(s) switch during the critical zone to cause an unacceptable level of SNR degradation, the host needs to control the transfer of bits during the frame by:

- Activating the CSn and SCLK when the transfer begins, allowing SDO bits to be received from the ADC.
- Freezing the SCLK when the ADC is in the critical red zone ensures that the ADC freezes its SDO line.
- Resuming the SCLK after the red zone to allow the rest of the bits to be received, thereby completing the transfer.

These steps can be carried out using the standard SPI protocol, but there are some complexities.

4.1 Latency 1 Data Transfer Complexity When Using Standard SPI

While the Latency 1 approach can dramatically reduce SCLK rates, it ends up unduly complicating the design of the host that controls the SPI transfer. The digital designer must carefully study the data sheet parameters of the ADC to ensure that the logic of the host controls the SCLK exactly as needed. Additionally, the logic has to account for the number of bits received pre and post the red zone. If the application has a varying SCLK frequency, the host design must account for changes in SCLK freezing windows, as well as the pre-, or post-bit division.

One final issue with this approach is that the complexity and control required at the host now ends up limiting the system designer to choose an FPGA as the host, as microcontrollers and their SPI modules cannot handle such interrupted transfer protocols. These limitations often can render the solution infeasible, as reducing SCLK rates has to weigh against increased design complexity and overall project cost and effort.

4.2 Latency 1 Source-Synchronous Mode

multiSPI technology introduces a novel solution that allows the system designer all the benefits of clock rate reduction of Latency 1, but with none of the complexity explained in the previous section. This solution moves the complexity of ensuring a freeze of digital activity during the critical red zone from the host to the ADC. The multiSPI digital interface has been designed to continuously track the internal conversion status of the ADC, thereby knowing the exact boundaries of the red zone. This self-tracking mechanism enables the ADC to cease its own transmission during the critical zone, while the host does not need to comprehend this detail at all.

The host needs to carry out the following steps:

1. Set up the ADC in source-synchronous mode
2. Assert CSn to begin frame transfer
3. Supply SCLK, if the ADC is configured in SCLK reflect mode for generating the STRB

The host does not need to freeze SCLK during the red zones. The ADC will generate a STRB pulse and drive the bits on the SDO only outside the red zone. These signals will freeze (cease toggling) within the critical area, which minimizes SNR degradation.

Deactivate SCLK and CSn once the desired number of bits are captured in the receive buffer.

The multiSPI digital interface generates *only* as many STRB pulses as the number of data bits to be transferred. In the case where the ADC is configured in SCLK reflect mode, this further simplifies the host design as it does not need to shut off its SCLK exactly after having received all the bits. Extra SCLK pulses will not result in extra STRB cycles, nor will erroneous bits be captured into the receive buffer of the host. For example, in the ADS9110, the multiSPI digital interface will generate only 18 STRB pulses, if it is configured to output 18 bits of data.

This behavior is illustrated in Figure 10, which shows a close-up version of Figure 9 showing one frame only. The data transferred in this frame is from the previous conversion (Latency 1). The SCLK generated by the host is shown to toggle, even in the red zone, because there is no requirement on the host to freeze it. However, the ADC freezes all activity on its STRB and SDO pins during the critical silent zone.

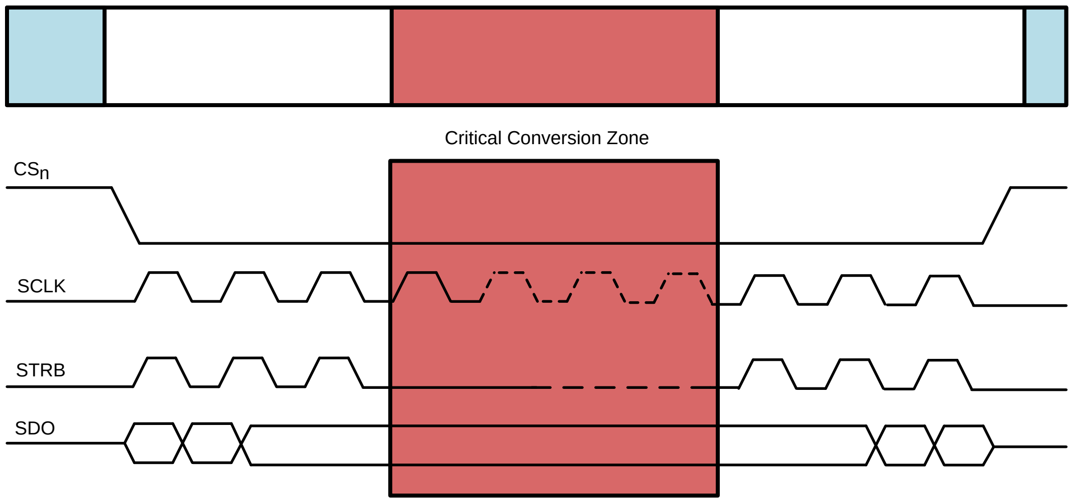

Figure 10. Closer Look at Latency 1 Operation Through Critical Silent Zones

Figure 10 shows that SCLK remains active through the red-zone. Usually this is acceptable because the ADC ensures that the SCLK does not propagate into the device during the red zone. Hence, the noise-coupling due to this signal remaining active is minimal. However, if the designer wishes to eliminate noise-coupling, the designer can use the internal mode to keep the SCLK off and have the ADC generate its STRB using its own internal clock.

When transferring data using Latency 1, the CSn should be kept active for as long as it takes to receive all the bits (and related STRB edges) from the ADC. Clearly, some frame time is lost due to a complete absence of transfers in the red zone. Furthermore, as the SCLK is completely unrelated to the internal oscillator of the ADC, an additional cycle may be used up by the ADC to detect the red zone boundaries and correlate them to the edges of the SCLK. Keeping these factors in mind, use Equation 5 to determine the duration of the CSn to properly transmit all of the bits from the ADC to the host:

$$T_{DATA\_TRANSFER} = N \times T_{SCLK} + T_{SU\_CSn\_SCLK} + T_{D\_CLK\_DO} + T_{RED\_ZONE} + 2 \times T_{SCLK}$$

where

- N = Number of bits to be transmitted (5)

Equation 5 is the same as Equation 3, which involved source synchronous transfer with no latency, except that the red zone width and the two-cycle margin are added.

The additional two-cycle margin can be eliminated by using the internal source STRB generation feature of the ADC. The ADC does not have to spend extra cycles to determine the clock edge relationships between the transfer and the red zone. The reason is that both the data transfer and internal conversion, which determines the red zone boundaries, use the internal clock of the ADC.

Equation 6 is the same as Equation 4, with the red zone width added in. The final takeaway from Figure 10 is that it illustrates that the entire time period of a conversion frame (barring the red and blue zones) is available for the data transfer. This feature enables the case where SCLK and STRB speed can be very slow and yet there be enough time to transmit the desired number of bits.

$$T_{DATA\_TRANSFER} = N \times T_{STRB} + T_{RED\_ZONE} + T_{D\_CSn\_STRB}$$

where

- N = Number of bits to be transmitted (6)

To sum it all, the Latency 1 source-synchronous mode offers a number of benefits, while adding minimal complexity to the logic in the host. Note that the multiSPI digital interface also allows the designer the flexibility to allow the SDO switching in the critical zones. This will improve data transfer times (if the final SNR of the system is within acceptable bounds).

4.3 The Best of Both Worlds

At this point, a system engineer may be tempted to ask, “Is it possible to combine both the high-frequency benefits of source-synchronous mode transfers and the frequency-reducing benefits of Latency 1 source-synchronous mode to get the best of both worlds?” The answer is yes. The system designer may choose to do any of the following:

1. Use the source-synchronous mode to enable a very high SCLK rate
2. Use Latency 1 to enable a very low SCLK rate, where data transfer continues even within the critical zone
3. Use Latency 1 to enable a medium SCLK rate, where data transfer freezes within the critical zone
4. Combine the two modes to achieve an ideal mix for the needs of the application

It should be noted that the multiSPI digital interface was designed to be independent of the conversion phases within the ADC. If the various modes of operation are combined as listed in the preceding points, the data exported by the ADC depends only on when the transfer is first initiated, and will reflect the last successfully-completed conversion.

5 Reducing Clock Rates Even Further

The multiSPI technology includes additional features for the system designer to take advantage of to further reduce the clock rate.

5.1 Multiple SDO lines

More SDO data lines mean less clock cycles to transmit the required number of bits; hence, shorter data transfer times. The multiSPI digital interface can be configured to export data on one, two, or four SDO lines. Selecting an SDO width of two reduces the required SCLK frequency by a factor of two, and selecting a width of four reduces the frequency by a factor of four. For example, if a simple SPI protocol requires a very high SCLK speed of 120 MHz for an 18-bit, 2-MSPS ADC, the same throughput can be achieved with an SCLK of 30 MHz, if the SDO width is set to four. The penalty is that extra pins are used at the host end, and the board will need additional traces.

As the width of the SDO is increased from one to two or four, the multiple data bits are transmitted simultaneously in each cycle. Figure 11 illustrates an example of how data bits are transmitted over four SDO lines.

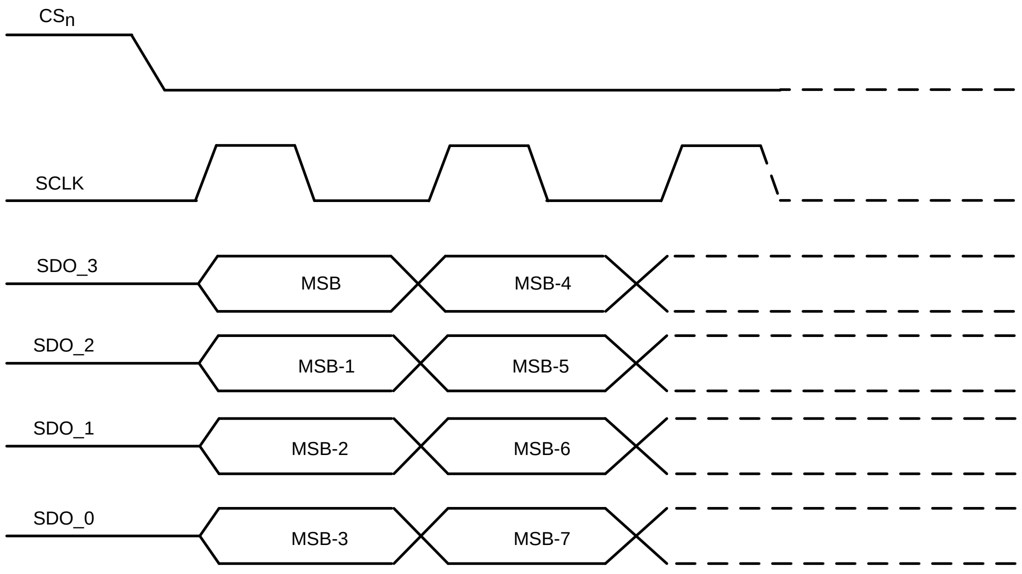

Figure 11. SPI Transmission With Four SDO Lines

5.2 DDR Signaling

The multiSPI digital interface source-synchronous mode of transfer supports double data rate (DDR) signaling, which allows data bits to be updated on both edges of the clock. This allows the overall clock rate to be decreased by a factor of two without compromising the data throughput.

For example, an SPI that would originally need an 80-MHz clock would achieve the same data throughput as with a 20-MHz clock with multiSPI digital interface configured to operate in source-synchronous mode with two SDOs transmitting data in DDR mode. The transfer of bits in this configuration is illustrated in Figure 12.

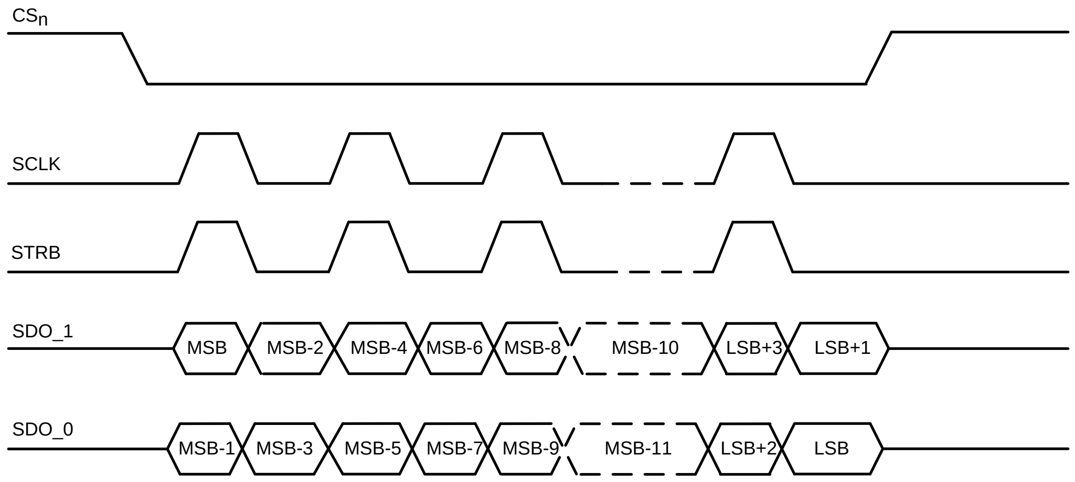

Figure 12. Data Transfer With Two SDOs and DDR

The DDR mode has been architected to connect seamlessly with FPGAs, as most current generation FPGAs support DDR structures within their IO modules.

6 Using the multiSPI With Legacy SPIs

While multiSPI technology introduces many innovative high-performance features, this digital interface is aware that legacy SPI modes remain a popular data transfer option for a lot of SAR ADC-based applications. This is especially true for microcontroller-based hosts. For all such cases, the multiSPI digital interface continues to support the following legacy SPI protocols:

- SPI_xy (polarity = x, phase = y): SPI_00, SPI_01, SPI10, SPI_11 are all supported
- Polarity: indicates the value that the SCLK is held at when the frame is inactive
- Phase:
  - when 0, the first data bit is launched on the falling edge of CSn
  - when 1, the first data bit is launched on the first edge of SCLK after CSn transitions to active

multiSPI technology allows SPI protocols to be combined with multiple SDO widths (1, 2, or 4) to help reduce the SCLK rate. Because the DDR option is not part of the legacy SPI protocol, it is not supported in this mode.

7 Combining multiSPI Technology Features

The multiSPI digital interface offers flexibility so you can decide the final combination of features you may want to use. This section gives you an idea of how various features may be used together to find the best solution for a particular application. Table 1 is not an exhaustive list, but it gives you an idea of the flexibility and versatility of this digital interface.

Table 1. Implementing multiSPI Technology Features Based on Application Needs

Throughput Requirement	Low Latency Required?	Number of Pins or Traces is a Constraint?	High-Performance Host an Option?	Suggested Mode
Low	Yes	Yes	Does not matter	Standard SPI protocol at low speeds.
Low or Medium	Yes	Yes (isolation channels to be minimized)	Yes	Use source-synchronous mode to meet throughput despite high delay of isolators.
High	Yes	Yes	Yes	Use source-synchronous mode to achieve up to 100 MHz data transfer.
High	Yes	No	No	Standard SPI protocol with multiple SDO pins
High	No	Yes	No	Use source-synchronous mode with Latency 1 to enable lower STRB (clock) rates; enable the device to freeze its outputs in the critical window; choose the internal clock feature if SCLK cannot be left free-running through the critical zones.
High	Yes	No	Yes	Use source synchronous with two or four SDO lines to complete data transmission quickly and allow for a large acquisition time for the next sample.

8 Simplifying System Connectivity

Having established how multiSPI technology can enable high-throughput solutions, we now take it one step further. These multiSPI digital interface features further simplify system design and debug, helping you to drive down overall project design cycles. Each feature adds value to the project design effort, without impacting complexity or cost.

8.1 Feed-Through Shift-Mode

multiSPI technology supports a feed-through mode, which allows data values fed into the SDI pin of the ADC to eventually feed out of the SDO pin (see Figure 13).

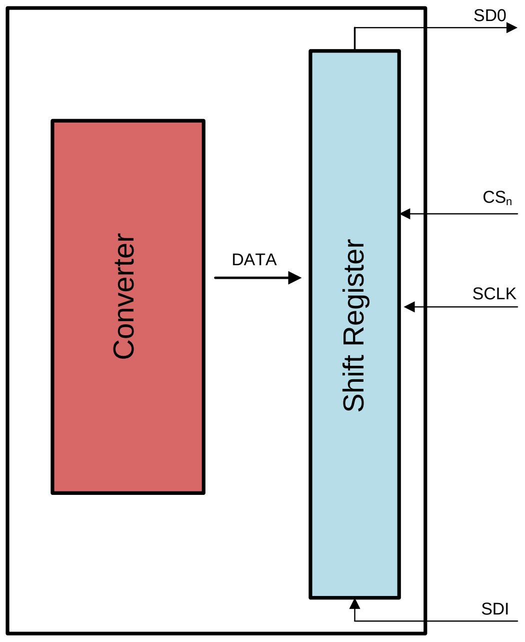

Figure 13. multiSPI Digital Interface SDI to SDO Feed-Through Shift-Mode Feature

When the data transfer is initiated using CSn, the result of the data conversion of the ADC is captured in the shift register (blue) and shifted out on the SDO. Concurrently, the SDI feeds into the shift register, allowing a register read or write command to be simultaneously input into the ADC. If the CSn is kept active and SCLK continues to pulse after all the data bits have been shifted out, then after a certain defined number of clock cycles (20 in the case of the ADS9110), the value originally shifted into the SDI now starts shifting out on the SDO. This architecture simplifies a number of things for the system designer.

The first advantage of this feed-through feature is that it decouples the timing of the SPI shift operation from the rest of the digital logic of the ADC. Previously, debugging the basic data transfer between an ADC and a host involved three steps:

1. First the host loads a write command into the ADC, which writes into one of the ADC registers.
2. Next, the host reads back the register value to check if it receives what is expected.
3. Finally, the host reads the conversion data from the ADC. If the register and conversion data results are found to be accurate, you can assume that the data communication between the host and ADC is robust.

If there is a data comparison failure, the system engineer cannot be sure if there is a timing problem of one or more of the CSn, SDI, SCLK, or SDO pins. It also may be possible that he data is being shifted into the ADC correctly, but that the command bit sequence is incorrect, or if the ADC itself is misinterpreting the command. These variables can lead to long debug cycle times.

The feed-through approach allows for a simple ‘data flush’ test to eliminate pin timing issues. This can be done through the following steps:

1. The host activates CSn and flushes a long stream of zeros through the SDI to SDO path.
2. It then feeds in a known data pattern into SDI.
3. The same data pattern should appear on the SDO after a delay of 20 clock cycles.

This sequence automatically does a sanity check of all timing related to the CSn, SCLK, SDI, and SDO pins at the targeted system speed. The system designer can then start to access registers or reading out conversion data.

The feed-through option is available only when the multiSPI digital interface is operating in one of the standard SPI modes with the SDO width configured to 1. This is because these are the only modes when both the SDI and SDO follow identical bit-timing rules. For each bit shifted into the SDI, one bit is shifted out from the SDI and all shifts are based off the same clock (SCLK). In all other enhanced multiSPI digital-interface modes (source synchronous, multiple SDO lines, or DDR), the rate at which bits are shifted into the SDI is either different from the rate of export at the SDO, or the SDI works on SCLK, while the SDO updates on STRB (two different clocks). In such cases, feed-through mode is not supported.

When planning to use an enhanced multiSPI digital interface mode, use the feed-through mode in the initial system bring-up and debug stages to ensure that the host can access and configure the ADC correctly (before the enhanced mode is configured).

8.2 Eliminating the Extra DAISY Pin

Feed-through mode simplifies the daisy-chain implementation as multiple multiSPI digital interface-based devices can be connected in a simple daisy chain (see Figure 14). Each device will exhibit a total delay of 20 clock cycles through it, causing the total transmission length to be 60 cycles so the data of all three devices can be shifted out. Similarly, the 60-cycle frame can be used to shift individual commands into each device. This solution eliminates the extra DAISY pin available on typical ADCs, as the daisy operation is now a part of the normal SPI data shift protocol.

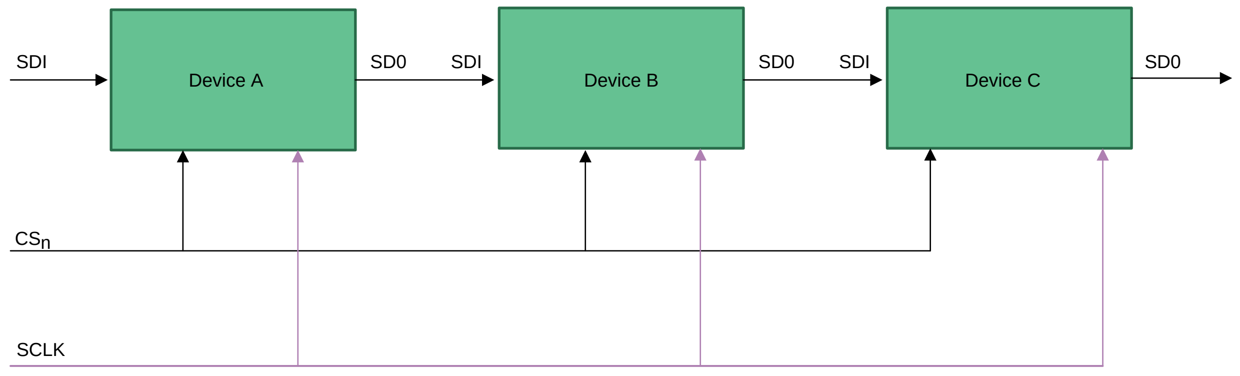

Figure 14. Daisy Chain of multiSPI Digital Interface-Based Devices

8.3 *Improving Data Integrity Using Parity Bits*

In certain applications, there may be a need to add an extra check for data integrity of the transmissions, as bits may be corrupted by system noise. The multiSPI technology allows the user to append up to two parity bits to the output data of the device. These bits can be evaluated at the host-end to decide if the transmission has proceeded without errors.

Figure 15 illustrates the data packet and two parity placement schemes with P1 and P2 being the two parity bits added to the conversion data.

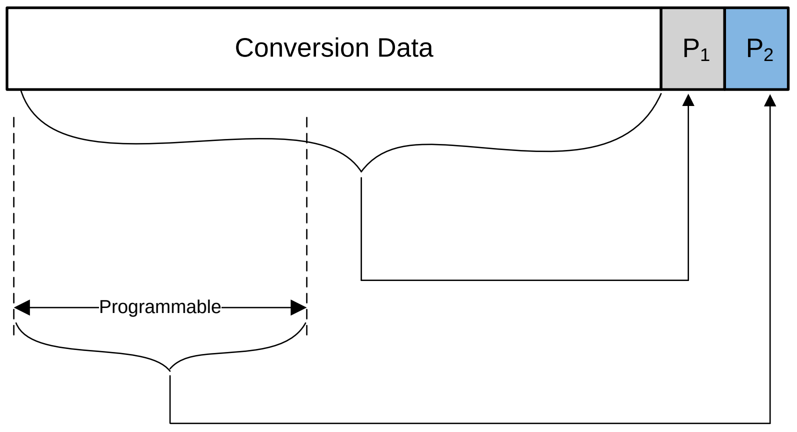

Figure 15. Parity Bits Computation Based on the Conversion Data

The functions of the parity bits are:

- The first parity bit (P1) reflects an EVEN parity for the entire 18-bit conversion result generated by the ADC.
- The second parity bit (P2) implements a novel floating parity option, and reflects an EVEN parity for a programmable number of most significant bits (MSBs). This allows the application to receive a packet where the overall parity is corrupted, but it may choose to retain the data if the desired number of MSBs still reflects accurate parity.

Parity checks can only detect an odd number of bit flips (corruptions), as an even number of flips will still result in a correct parity.

The ADS9110 implements both P1 and P2 parity bits, but other multiSPI digital interface-based devices may implement just P1.

As with all other enhanced multiSPI digital interface features, the parity option is disabled by default, and is enabled only if the system engineer requires the feature. Enabling the parity feature increases the total number of bits to be transmitted per frame and, hence, can decrease the overall frame rate.

8.4 Simplifying Debug of ADC-Host Data Transfer

In the past, system engineers have tested the host ADC read-back mechanism by holding the analog input of the ADC to a known DC value and carrying out reads to see if the data received is within the range of codes expected for that value. While this may be an acceptable approach, there is still a level of non-determinism involved in the process. The multiSPI technology eliminates this problem by offering modes wherein the ADC outputs the following pre-defined data patterns: all 0s and 1s, repeating a pattern of 10, repeating a pattern of 1100.

These data patterns can be used to debug any mode in which the multiSPI digital interface has been configured. The system designer should use this mode of debug to carry out a sanity check of the data transmission between the ADC and host before proceeding to actual application debug.

8.5 Self-Protection From Truncated Short Frames

multiSPI technology adds protection against erroneous configuration of ADCs due to truncated (short) frames. In such cases, the ADC devices can misbehave, because they may interpret a part of the aborted frame received on the SDI pin as a valid write command that ends up configuring internal registers. The multiSPI digital interface ensures that if a frame lasts less than 20 SCLK cycles, then upon the de-activation of CSn the ADC rejects any data received on the SDI and treats the received bits as a NOP command. This feature saves a significant amount of debug time as often an erroneous configuration can result in the host and the ADC going completely out of sync with each other. The effect of which may show up much later in the runtime of the application.

The self-protection mechanism is not supported in devices with daisy-chained configurations, such as the ADS9110. This is because a daisy chain will result in an effective shift length that is larger than 20 bits. If we refer to the example in Figure 14, the total shift length of three daisy-chained devices will be 60 cycles. If a frame aborts after a number of cycles greater than 20 but less than 60, the three devices will attempt to interpret the data received on their respective SDIs.

8.6 Monitoring Command Processing

multiSPI technology offers visibility into the command processing status of the ADC through the RDY signal. Once the CSn pin deactivates (transitions from zero to one to signal the end of the frame), the RDY signal can behave in one of the following ways:

1. If the command shifting into the SDI during the frame is valid, the RDY signal is held to zero until the register read or write is complete. After this, the RDY signal transitions back to one, which indicates that the device has completed processing the command and is ready for the next frame.
2. If the command shifted in was a truncated frame (or NOP), the RDY transitions from zero to one immediately (within 10 ns) after the CSn transitions from zero to one. If the host has not shifted in a valid command, yet the RDY is found to remain low for more than 30 ns after the CSn transitions to one, it is an indicator to the system that the ADC has interpreted a valid command, which may need further debugging.
3. The host software (SW) can use the RDY pin as a GPIO to monitor and ensure that prior commands have been recognized and executed by the device before it initiates a new set of operations.

If the host has executed a register write that can change an analog parameter of the device, then the RDY transitioning to one indicates that the write has taken effect. If an analog parameter has been modified, the host must also wait for any settling time involved (as specified in the device data sheet).

8.7 Device Register Read or Write During Standby Modes

multiSPI technology offers another innovative feature not yet available in most ADCs. The multiSPI digital interface remains active through standby modes, enabling the application to access registers, even if the ADC is in a low-power state. This is particularly useful if the host needs to monitor certain registers in the device, irrespective of whether it is active or in standby. This also may be especially helpful in scenarios where the ADC includes logic for detecting alarms or threshold crossings, and this logic is kept operational while the rest of the ADC is in standby mode.

9 Conclusion

The intent of this document is to give the user a good understanding of the many features and advantages of the multiSPI digital interface available initially on the ADS9110, and on all forthcoming TI SAR ADCs. multiSPI technology has been architected to allow TI ADCs to integrate seamlessly into a wide range of applications and targeted hosts, while simplifying system design, reducing system costs and minimizing the time and effort required for system debug and bring-up.

TI’s multiSPI digital interface is a significant advancement for engineers designing with SAR ADCs. We invite you to post any questions you might have about the technology to the *Precision Data Converters Forum* in the TI E2E™ Community.

10 Resources

- Watch a video overview on the multiSPI digital interface.
- Download data sheets for SAR ADCs that include multiSPI technology:
  - ADS9110*18-Bit, 2-MSPS, 15-mW, SAR ADC with iSPI™ Interface* (SBAS629)
  - ADS8681 *16-Bit, 1-MSPS, Single-Supply, SAR ADC Data Acquisition System with Programmable, Bipolar Input Ranges* (SBAS633)
  - ADS9120 *16-Bit, 2.5-MSPS, 15.5-mW, SAR ADC with multiSPI™ Interface* (SBAS710)
  - ADS8900B *20-Bit, High-Speed SAR ADCs With Integrated Reference Buffer, Integrated LDO, and multiSPI™ Digital Interface*
- Learn more about TI’s SAR ADC portfolio and find additional technical resources.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		